Polynomial greatest common divisor

From Wikipedia, the free encyclopedia

In algebra, the **greatest common divisor** (frequently abbreviated as GCD) of two polynomials is a polynomial, of the highest possible degree, that is a factor of both the two original polynomials. This concept is analogous to the greatest common divisor of two integers.

In the important case of univariate polynomials over a field the polynomial GCD may be computed, like for the integer GCD, by Euclid's algorithm using long division. The main difference lies in the fact that there is no natural total order on the polynomials. Therefore, "greatest" is meant for the relation of divisibility. As this relation is only a preorder, the polynomial GCD is defined only up to the multiplication by an invertible constant.

The similarity between the integer GCD and the polynomial GCD allows us to extend to univariate polynomials all the properties that may be deduced from Euclid's algorithm and Euclidean division. Moreover, the polynomial GCD has specific properties that make it a fundamental notion in various areas of algebra. Typically, the roots of the GCD of two polynomials are the common roots of the two polynomials, and this allows to get information on the roots without computing them. For example the multiple roots of a polynomial are the roots of the GCD of the polynomial and its derivative, and further GCD computations allow to compute the square-free factorization of the polynomial, which provides polynomials whose roots are the roots of a given multiplicity.

The greatest common divisor may be defined and exists, more generally, for multivariate polynomials over a field or the ring of integers, and also over a unique factorization domain. There exist algorithms to compute them as soon as one has a GCD algorithm in the ring of coefficients. These algorithms proceed by a recursion on the number of variables to reduce the problem to a variant of Euclid's algorithm. They are a fundamental tool in computer algebra, because computer algebra systems use them systematically to simplify fractions. Conversely, most of the modern theory of polynomial GCD has been developed to satisfy the need of efficiency of computer algebra systems.

Contents

General definition

Let p and q be polynomials with coefficients in an integral domain F, typically a field or the integers. A **greatest common divisor** of p and q is a polynomial d that divides p and q and such that every common divisor of p and q also divides d. Every pair of polynomials has a GCD if and only if F is a unique factorization domain.

If F is a field and p and q are not both zero, d is a greatest common divisor if and only if it divides both p and q and it has the greatest degree among the polynomials having this property. If $p = q = 0$, the GCD is 0. However, some authors consider that it is not defined in this case.

The greatest common divisor of p and q is usually denoted "gcd(p, q)".

The greatest common divisor is not unique: if d is a GCD of p and q, then the polynomial f is another GCD if and only if there is an invertible element u of F such that

$$f = ud$$

and

$$d = u^{-1}f.$$

In other words, the GCD is unique up to the multiplication by an invertible constant.

In the case of the integers, this indetermination has been settled by choosing, as the GCD, the unique one which is positive (there is another one, which is its opposite). With this convention, the GCD of two integers is also the greatest (for the usual ordering) common divisor. When one want to settle this indetermination in the polynomial case, one lacks of a natural total order. Therefore, one chooses once for all a particular GCD that is then called *the* greatest common divisor. For univariate polynomials over a field, this is usually the unique GCD which is monic (that is has 1 as coefficient of the highest degree). In more general cases, there is no general convention and above indetermination is usually kept. Therefore equalities like d = gcd(p, q) or gcd(p, q) = gcd(r, s) are usual abuses of notation which should be read "d is a GCD of p and q" and "p, q has the same set of GCD as r, s". In particular, gcd(p, q) = 1 means that the invertible constants are the only common divisors, and thus that p and q are coprime.

Properties

- As stated above, the GCD of two polynomials exists if the coefficients belong either to a field, the ring of the integers or more generally to a unique factorization domain.
- If c is any common divisor of p and q, then c divides their GCD.
- $\gcd(p,q) = \gcd(q,p)$.
- $\gcd(p,q) = \gcd(q,p+rq)$ for any polynomial r. This property is at the basis of the proof of Euclid's algorithm.
- For any invertible element k of the ring of the coefficients, $\gcd(p,q) = \gcd(p,kq)$.
- Hence $\gcd(p,q) = \gcd(a_1p + b_1q, a_2p + b_2q)$ for any scalars a_1, b_1, a_2, b_2 such that $a_1b_2 - a_2b_1$ is invertible.
- If $\gcd(p,r) = 1$, then $\gcd(p,q) = \gcd(p,qr)$.
- If $\gcd(q,r) = 1$, then $\gcd(p,qr) = \gcd(p,q)\ \gcd(p,r)$.
- For two univariate polynomials p and q over a field, there exist polynomials a and b, such that $\gcd(p,q) = ap + bq$ and $\gcd(p,q)$ divides every such linear combination of p and q (Bézout's identity).
- The greatest common divisor of three or more polynomials may be defined similarly as for two polynomials. It may be computed recursively from GCD's of two polynomials by the identities:

  $$\gcd(p,q,r) = \gcd(p,\gcd(q,r)),$$

  and

  $$\gcd(p_1,p_2,\ldots,p_n) = \gcd(p_1,\gcd(p_2,\ldots,p_n)).$$

GCD by hand computation

There are several ways to find the greatest common divisor of two polynomials. Two of them are:

1. *Factorization*, in which one finds the factors of each expression, then selects the set of common factors held by all from

within each set of factors. This method may be useful only in very simple cases, as, like for the integers, factoring is usually much more difficult than computing the greatest common divisor. Moreover, there are fields of coefficient for which there is no factorization algorithm, while Euclidean algorithm always exists.

2. The *Euclidean algorithm*, which can be used to find the GCD of two polynomials in the same manner as for two numbers.

Factoring

To find the GCD of two polynomials using factoring, simply factor the two polynomials completely. Then, take the product of all common factors. At this stage, we do not necessarily have a monic polynomial, so finally multiply this by a constant to make it a monic polynomial. This will be the GCD of the two polynomials as it includes all common divisors and is monic.

Example one: Find the GCD of $x^2 + 7x + 6$ and $x^2 - 5x - 6$.

$x^2 + 7x + 6 = (x + 1)(x + 6)$

$x^2 - 5x - 6 = (x + 1)(x - 6)$

Thus, their GCD is $x + 1$.

Euclidean algorithm

Factoring polynomials can be difficult, especially if the polynomials have large degree. The Euclidean algorithm is a method which works for any pair of polynomials. It makes repeated use of polynomial long division or synthetic division. When using this algorithm on two numbers, the size of the numbers decreases at each stage. With polynomials, the degree of the polynomials decreases at each stage. The last nonzero remainder, made monic if necessary, is the GCD of the two polynomials.

More specifically, assume we wish to find the gcd of two polynomials $a(x)$ and $b(x)$, where we can suppose

$$\deg(b(x)) \leq \deg(a(x)) .$$

We can find two polynomials $q(x)$ and $r(x)$ which satisfy (see Polynomial long division)

- $a(x) = q_0(x)b(x) + r_0(x)$
- $\deg(r_0(x)) < \deg(b(x))$

The polynomial $q_0(x)$ is called the quotient and $r_0(x)$ is the remainder. Notice that a polynomial $g(x)$ divides $a(x)$ and $b(x)$ if and only if it divides $b(x)$ and $r_0(x)$. We deduce

$$\gcd(a(x), b(x)) = \gcd(b(x), r_0(x)) .$$

Then set

$$a_1(x) = b(x), b_1(x) = r_0(x) .$$

Repeat the Polynomial long division to get new polynomials $q_1(x)$, $r_1(x)$, $a_2(x)$, $b_2(x)$ and so on. At each stage we have

$$\deg(a_{k+1}) + \deg(b_{k+1}) < \deg(a_k) + \deg(b_k) ,$$

so the sequence will eventually reach a point at which

$$b_N(x) = 0$$

and we will have found our GCD:

$$\gcd(a, b) = \gcd(a_1, b_1) = \cdots = \gcd(a_N, 0) = a_N .$$

Example: Find the GCD of $x^2 + 7x + 6$ and $x^2 - 5x - 6$.

$x^2 + 7x + 6 = (x^2 - 5x - 6)(1) + (x + 1)(12)$

$x^2 - 5x - 6 = (x + 1)(x - 6) + 0$

Since $x + 1$ is the last nonzero remainder, the GCD of these polynomials is $x + 1$.

This method works only if one may test the equality to zero of the elements of the field of the coefficients, so one needs a description of the coefficients as elements of some finitely generated field, for which the generators and relations are known exactly. If the coefficients are floating point numbers, known only approximately, then one uses completely different techniques, usually based on SVD.

This induces a new difficulty: For all these fields except the finite ones, the coefficients are fractions. If the fractions are not simplified during the computation, the size of the coefficients grows exponentially during the computation, which makes it impossible except for very small degrees. On the other hand, it is highly time consuming to simplify the fractions immediately. Therefore two different alternative methods have been introduced (see below):

- Pseudo-remainder sequences, especially subresultant sequences.
- Modular GCD algorithm using modular arithmetic

Univariate polynomials with coefficients in a field

The case of univariate polynomials over a field is specially important for several reasons. Firstly, it is the most elementary case and therefore appear in most first courses in algebra. Secondly, it is very similar to the case of the integers, and this analogy is the source of the notion of Euclidean domain. A third reason is that the theory and the algorithms for the multivariate case and for coefficients in a unique factorization domain are strongly based on this particular case. Last but not least, polynomial GCD algorithms and derived algorithms allow one to get useful information on the roots of a polynomial, without computing them.

Euclidean division

Euclidean division of polynomials, which is used in Euclid's algorithm for computing GCDs, is very similar to Euclidean division of integers. Its existence is based on the following theorem: Given two univariate polynomials a and $b \neq 0$ defined over a field, there exist two polynomials q (the *quotient*) and r (the *remainder*) which satisfy

$$a = bq + r$$

and

$$\deg(r) < \deg(b),$$

where "deg(...)" denotes the degree and the degree of 0 is defined as negative. Moreover q and r are uniquely defined by these relations.

The difference from Euclidean division of the integers is that, for the integers, the degree is replaced by the absolute value, and that to have uniqueness one has to suppose that r is non-negative. The rings for which such a theorem exists are called Euclidean domains.

Like for the integers, the Euclidean division of the polynomials may be computed by the long division algorithm. This algorithm is usually presented for paper-and-pencil computation, but it works well on computers, when formalized as follows (note that the names of the variables correspond exactly to the regions of the paper sheet in a pencil-and-paper computation of long division). In the following computation "deg" stands for the degree of its argument (with the convention deg(0)<0), and "lc" stands for the leading coefficient, the coefficient of the highest degree of the variable.

Euclidean division
Input: a and $b \neq 0$ two polynomials in the variable x;
Output: q, the quotient and r, the remainder;
Begin
$q := 0; \quad r := a;$
$d := \deg(b); \quad c := \operatorname{lc}(b);$
while $\deg(r) \geq d$ ***do***

$$s := \frac{\operatorname{lc}(r)}{c} x^{\deg(r)-d};$$
$$q := q + s;$$
$$r := r - sb;$$

end do;
return (q, r);
end.

The proof of the validity of this algorithm relies on the fact that during the whole "while" loop, we have $a = bq + r$ and $\deg(r)$ is a non-negative integer that decreases at each iteration. Thus the proof of the validity of this algorithm also proves the validity of Euclidean division.

Euclid's algorithm

As for the integers, Euclidean division allows us to define Euclid's algorithm for computing GCDs.

Starting from two polynomials a and b, Euclid's algorithm consists of recursively replacing the pair (a, b) by $(b, \text{rem}(a, b))$ (where "rem(a, b)" denotes the remainder of the Euclidean division, computed by the algorithm of the preceding section), until $b = 0$. The GCD is the last non zero remainder.

Euclid's algorithm may be formalized in the recursive programming style as:

$$\gcd(a, b) := \text{if } b = 0 \text{ then } a \text{ else } \gcd(b, \text{rem}(a, b)).$$

In the imperative programming style, the same algorithm becomes, giving a name to each intermediate remainder:

$r_0 := a; \quad r_1 := b; \quad i := 1;$
while $r_i \neq 0$ ***do***

$$r_{i+1} := \text{rem}(r_{i-1}, r_i); \quad i := i + 1;$$

end do;
return (r_{i-1}).

The sequence of the degrees of the r_i is strictly decreasing. Thus after, at most, $\deg(b)$ steps, one get a null remainder, say r_k. As (a, b) and $(b, \text{rem}(a,b))$ have the same divisors, the set of the common divisors is not changed by Euclid's algorithm and thus all pairs (r_i, r_{i+1}) have the same set of common divisors. The common divisors of a and b are thus the common divisors of r_{k-1} and 0. Thus r_{k-1} is a GCD of a and b. This not only proves that Euclid's algorithm computes GCDs, but also proves that GCDs exist.

Bézout's identity and extended GCD algorithm

Bézout's identity is a GCD related theorem, initially proved for the integers, which is valid for every principal ideal domain. In the case of the univariate polynomials over a field, it may be stated as follows.

If g is the greatest common divisor of two polynomials a and b, then there are two polynomials u and v such that

$$au + bv = g \quad \text{(Bézout's identity)}$$

and

$$\deg(u) < \deg(b) - \deg(g), \quad \deg(v) < \deg(a) - \deg(g).$$

The interest of this result in the case of the polynomials is that there is an efficient algorithm to compute the polynomials *u* and *v*, This algorithm differs from Euclid's algorithm by a few more computation done at each iteration of the loop. It is therefore called **extended GCD algorithm**. Another difference with Euclid's algorithm is that it uses the quotient, denoted "quo", of the Euclidean division instead of the remainder. This algorithm works as follows.

Extended GCD algorithm
Input: a, b,univariate polynomials
Output:

g,the GCD of a and b
u, v, such that

$$au + bv = g,$$
$$\deg(u) < \deg(b) - \deg(g)$$
$$\deg(v) < \deg(a) - \deg(g)$$

a_1, b_1, such that

$$a = ga_1$$
$$b = gb_1$$

Begin

$$r_0 := a; \quad r_1 := b;$$
$$s_0 := 1; \quad s_1 := 0;$$
$$t_0 := 0; \quad t_1 := 1;$$
$$i := 1;$$

while $r_i \neq 0$ ***do***

$$q := \operatorname{quo}(r_{i-1}, r_i);$$
$$r_{i+1} := r_{i-1} - qr_i;$$
$$s_{i+1} := s_{i-1} - qs_i;$$
$$t_{i+1} := t_{i-1} - qt_i;$$
$$i := i + 1;$$

end do;

$$g := r_{i-1};$$
$$u := s_{i-1}; \quad v := t_{i-1};$$
$$a_1 := (-1)^{i-1}t_i; \quad b_1 := (-1)^i s_i;$$

end.

The proof that the algorithm satisfies its output specification relies on the fact that, for every *i* we have

$$r_i = as_i + bt_i$$
$$s_i t_{i+1} - t_i s_{i+1} = s_i t_{i-1} - t_i s_{i-1},$$

the latter equality implying

$$s_i t_{i+1} - t_i s_{i+1} = (-1)^i.$$

The assertion on the degrees follows from the fact that, at every iteration, the degrees of s_i and t_i increase at most as the degree of r_i decreases.

An interesting feature of this algorithm is that, when the coefficients of Bezout's identity are needed, one gets for free the quotient of the input polynomials by their GCD.

Arithmetic of algebraic extensions

An important application of extended GCD algorithm is that it allows to compute division in algebraic field extensions.

Let L an algebraic extension of a field K, generated by an element whose minimal polynomial f has degree n. The elements of L are usually represented by univariate polynomials over K of degree less than n.

The addition in L is simply the addition of polynomials:

$$a +_L b = a +_{K[X]} b.$$

The multiplication in L is the multiplication of polynomials followed by the division by f:

$$a \cdot_L b = \operatorname{rem}(a._{K[X]} b), f).$$

The inverse of a non zero element a of L is the coefficient u in Bézout's identity $au + fv = 1$, which may be computed by extended GCD algorithm. (the GCD is 1 because the minimal polynomial f is irreducible). The degrees inequality in the specification of extended GCD algorithm shows that a further division by f is not needed to get $\deg(u) < \deg(f)$.

Subresultants

In the case of univariate polynomials, there is a strong relationship between greatest common divisors and resultants. In fact the resultant of two polynomials P, Q is a polynomial function of the coefficients of P and Q which has the value zero if and only if the GCD of P and Q is not constant.

The subresultants theory is a generalization of this property that allows to characterize generically the GCD of two polynomials, and the resultant is the 0-th subresultant polynomial.[1]

The i-th *subresultant polynomial* $S_i(P, Q)$ of two polynomials P and Q is a polynomial of degree at most i whose coefficients are polynomial functions of the coefficients of P and Q, and the i-th *principal subresultant coefficient* $s_i(P, Q)$ is the coefficient of degree i of $S_i(P, Q)$. They have the property that the GCD of P and Q has a degree d if and only if

$$s_0(P,Q) = \cdots = s_{d-1}(P,Q) = 0 \ , s_d(P,Q) \neq 0.$$

In this case, $S_d(P, Q)$ is a GCD of P and Q and

$$S_0(P,Q) = \cdots = S_{d-1}(P,Q) = 0.$$

Every coefficient of the subresultant polynomials is defined as the determinant of a submatrix of the Sylvester matrix of P and Q. This implies that that subresultants "specialize" well. More precisely, subresultants are defined for polynomials over any commutative ring R, and have the following property.

Let ϕ be a ring homomorphism of R into another commutative ring S. It extends to another homomorphism, denoted also ϕ between the polynomials rings over R and S. Then, if P and Q are univariate polynomials with coefficients in R such that

$$\deg(P) = \deg(\varphi(P))$$

and

$$\deg(Q) = \deg(\varphi(Q)),$$

then the subresultant polynomials and the principal subresultant coefficients of $\phi(P)$ and $\phi(Q)$ are the image by ϕ of those of P and Q.

The subresultants have two important properties whch make them fundamental for the computation on computers of the GCD of two polynomials with integer coefficients. Firstly, their definition through determinants allows to bound, through Hadamard inequality, the size of the coefficients of the GCD. Secondly, this bound and the property of good specialization allow to

compute the GCD of two polynomials with integer coefficients through modular computation and Chinese remainder theorem (see below).

Technical definition

Let

$$P = p_0 + p_1X + \cdots + p_mX^m, \quad Q = q_0 + q_1X + \cdots + q_nX^n.$$

be two univariate polynomials with coefficients in a field K. Let us denote by $\mathcal{P}_i$ the K vector space of dimension i the polynomials of degree less than i. For non negative integer i such that $i \leq m$ and $i \leq n$, let

$$\varphi_i : \mathcal{P}_{n-i} \times \mathcal{P}_{m-i} \to \mathcal{P}_{m+n-i}$$

be the linear map such that

$$\varphi_i(A, B) = AP + BQ.$$

The resultant of P and Q is the determinant of the Sylvester matrix, which is the (square) matrix of φ_0 on the bases of the powers of X. Similarly, the i-subresultant polynomial is defined in term of determinants of submatrices of the matrix of φ_i.

Let us describe these matrices more precisely;

Let $p_i = 0$ for $i < 0$ or $i > m$, and $q_i = 0$ for $i < 0$ or $i > n$. The Sylvester matrix is the $(m + n) \times (m + n)$-matrix such that the coefficient of the i-th row and the j-th column is p_{m+j-i} for $j \leq n$ and q_{j-i} for $j > n$:[2]

$$S = \begin{pmatrix} p_m & 0 & \cdots & 0 & q_n & 0 & \cdots & 0 \\ p_{m-1} & p_m & \cdots & 0 & q_{n-1} & q_n & \cdots & 0 \\ p_{m-2} & p_{m-1} & \ddots & 0 & q_{n-2} & q_{n-1} & \ddots & 0 \\ \vdots & \vdots & \ddots & p_m & \vdots & \vdots & \ddots & q_n \\ \vdots & \vdots & \cdots & p_{m-1} & \vdots & \vdots & \cdots & q_{n-1} \\ p_0 & p_1 & \cdots & \vdots & q_0 & q_1 & \cdots & \vdots \\ 0 & p_0 & \ddots & \vdots & 0 & q_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & p_1 & \vdots & \vdots & \ddots & q_1 \\ 0 & 0 & \cdots & p_0 & 0 & 0 & \cdots & q_0 \end{pmatrix}.$$

The matrix T_i of φ_i is the $(m + n - i) \times (m + n - 2i)$-submatrix of S which is obtained by removing the last i rows of zeros in the submatrix of the columns 1 to n-i and n+1 to m+n-i of S (that is removing i columns in each block and the i last rows of zeros). The *principal subresultant coefficient* s_i is the determinant of the $m + n - 2i$ first rows of T_i.

Let V_i be the $(m + n - 2i) \times (m + n - i)$ matrix defined as follows. First we add $(i + 1)$ columns of zeros to the right of the $(m + n - 2i - 1) \times (m + n - 2i - 1)$ identity matrix. Then we border the bottom of the resulting matrix by a row consisting in $(m + n - i - 1)$ zeros followed by $X^i, X^{i-1}, \ldots, X, 1$:

$$V_i = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & X^i & X^{i-1} & \cdots & 1 \end{pmatrix}.$$

With this notation, the i-th *subresultant polynomial* is the determinant of the matrix product V_iT_i. Its coefficient of degree j is the determinant of the square submatrix of T_i consisting in its $m + n - 2i - 1$ first rows and the $(m + n - i - j)$-th row.

Sketch of the proof

It is not obvious that, as defined, the subresultants have the desired properties. In fact the proof is rather simple if the properties of linear algebra and those of polynomials are put together.

As defined, the columns of the matrix T_i are the vectors of the coefficients of some polynomials belonging to the image of φ_i. The definition of the i-th subresultant polynomial S_i shows that the vector of its coefficients is a linear combination of these column vectors, and thus that S_i belongs to the image of φ_i.

If the degree of the GCD is greater than i, then Bézout's identity shows that every non zero polynomial in the image of φ_i has a degree larger than i. This implies that S_i=0.

If, on the other hand, the degree of the GCD is i, then Bézout's identity again allows to prove that the multiples of the GCD that have a degree lower than $m + n - i$ are in the image of φ_i. The vector space of these multiples has the dimension $m + n - 2i$ and has a base of polynomials of pairwise different degrees, not smaller than i. This implies that the submatrix of the $m + n - 2i$ first rows of the column echelon form of T_i is the identity matrix and thus that s_i is not 0. Thus S_i is a polynomial in the image of φ_i, which is a multiple of the GCD and has the same degree. It is thus a greatest common divisor.

GCD and root finding

Square-free factorization

Main article: Square-free factorization

Most root-finding algorithms behave badly with polynomials that have multiple roots. It is therefore useful to detect and remove them before calling a root-finding algorithm. A GCD computation allows to detect the existence of multiple roots, because the multiple roots of a polynomial are the roots of the GCD of the polynomial and its derivative.

After computing the GCD of the polynomial and its derivative, further GCD computations provide the complete *square-free factorization* of the polynomial, which is a factorization

$$f = \prod_{i=1}^{\deg(f)} f_i^i$$

where, for each i, the polynomial f_i either is 1 if f does not have any root of multiplicity i or is a square-free polynomial (that is a polynomial without multiple root) whose roots are exactly the roots of multiplicity i of f (see Yun's algorithm).

Thus the square-free factorization reduces root finding of a polynomial with multiple roots to root finding of several square-free polynomials of lower degree. The square-free factorization is also the first step in most polynomial factorization algorithms.

Sturm sequence

Main article: Sturm sequence

The *Sturm sequence* of a polynomial with real coefficients is the sequence of the remainders provided by a variant of Euclid's algorithm applied to the polynomial and its derivative. For getting the Sturm sequence, one simply replaces the instruction

$$r_{i+1} := \operatorname{rem}(r_{i-1}, r_i)$$

of Euclid's algorithm by

$$r_{i+1} := -\operatorname{rem}(r_{i-1}, r_i).$$

Let $V(a)$ be the number of changes of signs in the sequence, when evaluated at a point a. Sturm's theorem asserts that $V(a)$-$V(b)$ is the number of real roots of the polynomial in the interval $[a,b]$. Thus the Sturm sequence allows to compute the number of real roots in a given interval. By subdividing the interval until every subinterval contains at most one root, this provides an algorithm that locates the real roots in intervals of arbitrary small length.

GCD over a ring and over its field of fractions

In this section, we consider polynomials over a unique factorization domain R, typically the ring of the integers, and over its field of fractions F, typically the field of the rational numbers, and we denote $R[X]$ and $F[X]$ the rings of polynomials in a set of variables over these rings.

Primitive part–content factorization

See also: Content (algebra) and Gauss's lemma (polynomial)

The *content* of a polynomial $p \in R[X]$, denoted "cont(p)", is the GCD of its coefficients. A polynomial $q \in F[X]$ may be written

$$q = \frac{p}{c}$$

where $p \in R[X]$ and $c \in R$: it suffices to take for c a multiple of all denominators of the coefficients of q (for example their product) and $p = cq$. The *content* of q is defined as:

$$\operatorname{cont}(q) = \frac{\operatorname{cont}(p)}{c}.$$

In both cases, the content is defined up to the multiplication by a unit of R.

The *primitive part* of a polynomial in $R[X]$ or $F[X]$ is defined by

$$\operatorname{primpart}(p) = \frac{p}{\operatorname{cont}(p)}.$$

In both cases, it is a polynomial in $R[X]$ that is *primitive*, which means that 1 is a GCD of its coefficients.

Thus every polynomial in $R[X]$ or $F[X]$ may be factorized as

$$p = \operatorname{cont}(p) \operatorname{primpart}(p),$$

and this factorization is unique up to the multiplication of the content by a unit of R and of the primitive part by the inverse of this unit.

Gauss's lemma implies that the product of two primitive polynomials is primitive. It follows that

$$\operatorname{primpart}(pq) = \operatorname{primpart}(p) \operatorname{primpart}(q)$$

and

$$\operatorname{cont}(pq) = \operatorname{cont}(p) \operatorname{cont}(q).$$

Relation between the GCD over R and over F

The relations of the preceding section implies a strong relation between the GCD's in $R[X]$ and in $F[X]$. In order to avoid ambiguities, the notation "*gcd*" will be indexed, in the following, by the ring in which the GCD is computed.

If q_1 and q_2 belong to $F[X]$, then

$$\operatorname{primpart}(\gcd_{F[X]}(q_1, q_2)) = \gcd_{R[X]}(\operatorname{primpart}(q_1), \operatorname{primpart}(q_2)).$$

If p_1 and p_2 belong to $R[X]$, then

$$\gcd_{R[X]}(p_1, p_2) = \gcd_R(\operatorname{cont}(p_1), \operatorname{cont}(p_2)) \gcd_{R[X]}(\operatorname{primpart}(p_1), \operatorname{primpart}(p_2)),$$

and

$$\gcd_{R[X]}(\operatorname{primpart}(p_1), \operatorname{primpart}(p_2)) = \operatorname{primpart}(\gcd_{F[X]}(p_1, p_2)).$$

Thus the computation of polynomial GCD's is essentially the same problem over $F[X]$ and over $R[X]$.

For univariate polynomials over the rational numbers one may think that Euclid's algorithm is a convenient method for computing the GCD. However, it involves to simplify a large number of fractions of integers, and the resulting algorithm is not efficient. For this reason, methods have been designed to modify Euclid's algorithm for working only with polynomials over the

integers. They consist in replacing Euclidean division, which introduces fractions, by a so-called *pseudo-division*, and replacing the remainder sequence of Euclid's algorithm by so-called *pseudo-remainder sequences* (see below).

Proof that GCD exists for multivariate polynomials

In the previous section we have seen that the GCD of polynomials in $R[X]$ may be deduced from GCDs in R and in $F[X]$. A closer look on the proof shows that this allows us to prove the existence of GCDs in $R[X]$, if they exist in R and in $F[X]$. In particular, if GCDs exist in R, and if X is reduced to one variable, this proves that GCDs exist in $R[X]$ (Euclid's algorithm proves the existence of GCDs in $F[X]$).

A polynomial in n variables may be considered as a univariate polynomial over the ring of polynomials in $(n - 1)$ variables. Thus a recursion on the number of variables shows that if GCDs exists and may be computed in R, then they exist and may be computed in every multivariate polynomial ring over R. In particular, if R is either the ring of the integers or a field, then GCDs exist in $R[x_1,..., x_n]$, and what precedes provides an algorithm to compute them.

The proof that a polynomial ring over a unique factorization domain is also a unique factorization domain is similar, but it does not provide an algorithm, because there is no general algorithm to factor univariate polynomials over a field (there are examples of fields for which there does not exist any factorization algorithm for the univariate polynomials).

Pseudo-remainder sequences

In this section, we consider an integral domain Z (typically the ring **Z** of the integers) and its field of fractions Q (typically the field **Q** of the rational numbers). Given two polynomials A and B in the univariate polynomial ring $Z[X]$, the Euclidean division (over Q) of A by B provides a quotient and a remainder which may not belong to $Z[X]$.

For, if one applies Euclid's algorithm to

$$X^8 + X^6 - 3X^4 - 3X^3 + 8X^2 + 2X - 5$$

and

$$3X^6 + 5X^4 - 4X^2 - 9X + 21,$$

the successive remainders of Euclid's algorithm are

$$-\frac{5}{9}X^4 + \frac{1}{9}X^2 - \frac{1}{3},$$
$$-\frac{117}{25}X^2 - 9X + \frac{441}{25},$$
$$\frac{233150}{19773}X - \frac{102500}{6591},$$
$$-\frac{1288744821}{543589225}.$$

One sees that, despite the small degree and the small size of the coefficients of the input polynomials, one has to manipulate and simplify integer fractions of rather large size.

The *pseudo-division* has been introduced to allow a variant of Euclid's algorithm for which all remainders belong to $Z[X]$.

If $\deg(A) = a$ and $\deg(B) = b$ and $a \geq b$, the **pseudo-remainder** of the pseudo-division of A by B, denoted by $\operatorname{prem}(A,B)$ is

$$\operatorname{prem}(A, B) = \operatorname{rem}(\operatorname{lc}(B)^{a-b+1} A, B),$$

where $\operatorname{lc}(B)$ is the leading coefficient of B (the coefficient of X^b).

The pseudo-remainder of the pseudo-division of two polynomials in $Z[X]$ belongs always to $Z[X]$.

A **pseudo-remainder sequence** is the sequence of the (pseudo) remainders r_i obtained by replacing the instruction

$$r_{i+1} := \operatorname{rem}(r_{i-1}, r_i)$$

of Euclid's algorithm by

$$r_{i+1} := \frac{\operatorname{prem}(r_{i-1}, r_i)}{\alpha},$$

where α is an element of Z that divides exactly every coefficient of the numerator. Different choices of α give different pseudo-remainder sequences, which are described in the next subsections.

As the common divisors of two polynomials are not changed if the polynomials are multiplied by invertible constants (in Q), the last non zero term in a pseudo-remainder sequence is a GCD (in $Q[X]$) of the input polynomials. Therefore pseudo-remainder sequences allows to compute GCD's in $Q[X]$ without introducing fractions in Q.

Trivial pseudo-remainder sequence

The simplest (to define) remainder sequence consists in taking always α=1. In practice, it is not interesting, as the size of the coefficients grow exponentially with the degree of the input polynomials. This appears clearly on the example of the preceding section, for which the successive pseudo-remainders are

$$\begin{aligned}
&-15\,X^4 + 3\,X^2 - 9,\\
&15795\,X^2 + 30375\,X - 59535,\\
&1254542875143750\,X - 1654608338437500,\\
&12593338795500743100931141992187500.
\end{aligned}$$

The number of digits of the coefficients of the successive remainders is more than doubled at each iteration of the algorithm. This is a typical behavior of the trivial pseudo-remainder sequences.

Primitive pseudo-remainder sequence

The *primitive pseudo-remainder sequence* consists in taking for α the content of the numerator. Thus all the r_i are primitive polynomials.

The primitive pseudo-remainder sequence is the pseudo-remainder sequence, which generates the smallest coefficients. However it requires to compute a number of GCD's in Z, and therefore is not sufficiently efficient to be used in practice, especially when Z is itself a polynomial ring.

With the same input as in the preceding sections, the successive remainders, after division by their content are

$$\begin{aligned}
&-5\,X^4 + X^2 - 3,\\
&13\,X^2 + 25\,X - 49,\\
&4663\,X - 6150,\\
&1.
\end{aligned}$$

The small size of the coefficients hides the fact that a number of integers GCD and divisions by the GCD have been computed.

Subresultant pseudo-remainder sequence

The subresultant pseudo-remainder sequence consists in choosing α is such a way that every r_i is a subresultant polynomial. Surprisingly, the computation of α is very easy (see below). On the other hand the proof of correctness of the algorithm is difficult, because it should take into account all the possibilities for the difference of degrees of two consecutive remainders.

The coefficients in the subresultant pseudo-remainder sequence are rarely much larger than those of the primitive pseudo-remainder sequence. As GCD computations in Z are not needed, the subresultant pseudo-remainder sequence is the pseudo-remainder sequence which gives the most efficient computation.

With the same input as in the preceding sections, the successive remainders are

$$\begin{aligned}
&-15\,X^4 + 3\,X^2 - 9,\\
&65\,X^2 + 125\,X - 245,\\
&9326\,X - 12300,\\
&260708.
\end{aligned}$$

The coefficients have a reasonable size. They are obtained without any GCD computation, only exact divisions. This makes this algorithm more efficient than that of primitive pseudo-remainder sequences.

The algorithm computing the subresultant pseudo-remainder sequence is given below. In this algorithm, the input (a, b) is a pair of polynomials in $Z[X]$. The r_i are the successive pseudo remainders in $Z[X]$, the variables i and d_i are non negative integers, and the Greek letters denote elements in Z. The functions deg() and rem() denote the degree of a polynomial and the remainder of the Euclidean division. In the algorithm, this remainder is always in $Z[X]$. Finally the divisions denoted / are always exact and have their result either in $Z[X]$ or in Z.

$$r_0 := a; \quad r_1 := b; \quad i := 1;$$
$$d_1 := \deg(r_0) - \deg(r_1); \quad \gamma_1 := \mathrm{lc}(r_1); \quad \beta_1 := (-1)^{d_1+1}; \quad \psi_1 = -1;$$
while $r_i \neq 0$ ***do***

$$r_{i+1} := \mathrm{rem}(\gamma^{d_i+1} r_{i-1}, r_i)/\beta_i;$$
$$i := i + 1;$$
$$d_i := \deg(r_{i-1}) - \deg(r_i); \quad \gamma_i := \mathrm{lc}(r_i); \quad \psi_i := (-\gamma_i)^{d_{i-1}} / \psi_{i-1}^{d_{i-1}-1}; \quad \beta_i := -\gamma_{i-1}\psi_i^{d_i}$$

end do.

This algorithm computes not only the greatest common divisor (the last non zero r_i), but also all the subresultant polynomials: The remainder r_i is the $(\deg(r_{i-1})-1)$-th subresultant polynomial. If $\deg(r_i)<\deg(r_{i-1})-1$, the $\deg(r_i)$-th subresultant polynomial is $\mathrm{lc}(r_i)^{\deg(r_{i-1})-\deg(r_i)-1} r_i$. All the other subresultant polynomials are null.

Modular GCD algorithm

If f and g are polynomials in $F[x]$ for some finitely generated field F, the Euclidean Algorithm is the most natural way to compute their GCD. However, modern computer algebra systems only use it if F is finite because of a phenomenon called intermediate expression swell. Although degrees keep decreasing during the Euclidean algorithm, if F is not finite then the bitsize of the polynomials can increase (sometimes dramatically) during the computations because repeated arithmetic operations in F tends to lead to larger expressions. For example, the addition of two rational numbers whose denominators are bounded by b leads to a rational number whose denominator is bounded by b^2, so in the worst case, the bitsize could nearly double with just one operation.

To expedite the computation, take a ring D for which f and g are in $D[x]$, and take an ideal I such that D/I is a finite ring. Then compute the GCD over this finite ring with the Euclidean Algorithm. Using reconstruction techniques (Chinese remainder theorem, rational reconstruction, etc) one can recover the GCD of f and g from its image modulo a number of ideals I. One can prove[3] that this works provided that one discards modular images with non-minimal degree, and avoids ideals I modulo which a leading coefficient vanishes.

Suppose $F = \mathbb{Q}(\sqrt{3}), D = \mathbb{Z}[\sqrt{3}], f = \sqrt{3}x^3 - 5x^2 + 4x + 9$ and $g = x^4 + 4x^2 + 3\sqrt{3}x - 6$. If we take $I = (2)$ then D/I is a finite ring (not a field since I is not maximal in D). The Euclidean algorithm applied to the images of f, g in $(D/I)[x]$ succeeds and returns 1. This implies that the GCD of f, g in $F[x]$ must be 1 as well. Note that this example could easily be handled by any method because the degrees were too small for expression swell to occur, but it illustrates that if two polynomials have GCD 1, then the modular algorithm is likely to terminate after a single ideal I.

See also

- List of polynomial topics
- Multivariate division algorithm

References

1. ^ Saugata Basu; Richard Pollack, Marie-Françoise Roy (2006). *Algorithms in real algebraic geometry, chapter 4.2* (http://perso.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted1.html). Springer-Verlag.

2. **^** Many author define the Sylvester matrix as the transpose of *S*. This breaks the usual convention for writing the matrix of a linear map.
3. **^** *M. van Hoeij, M.B. Monagan: Algorithms for polynomial GCD computation over algebraic function fields. ISSAC 2004: 297-304*

- Davenport, James H.; Siret, Yvon; Tournier, Èvelyne (1988). *Computer algebra: systems and algorithms for algebraic computation*. Translated from the French by A. Davenport and J.H. Davenport. Academic Press. ISBN 978-0-12-204230-0.
- Knuth, Donald E (1997). *Seminumerical Algorithms*. The Art of Computer Programming **2** (Third ed.). Reading, Massachusetts: Addison-Wesley. pp. 439–461, 678–691. ISBN 0-201-89684-2.
- Loos, Rudiger (1982), "Generalized polynomial remainder sequences", in B. Buchberger; R. Loos; G. Collins, *Computer Algebra*, Springer Verlag
- *S.M.M. Javadi, M.B. Monagan: A sparse modular GCD algorithm for polynomials over algebraic function fields. ISSAC 2007: 187-194*

Categories: Polynomials | Computer algebra
