GCD and Factorisation of multivariate polynomials

R Freund
(freund@mytum.de)

JASS 2007,
Course "The Power of Polynomials and How To Use Them"

28 March 2007

Abstract

Some widely known techniques can be used to factorise univariate polynomials over the domain of integers. However, finding algorithms which factorise univariate and multivariate polynomials over $\mathbb{Z}$ and other domains is a little trickier. Several factorisation algorithms first need GCDs of the polynomials. Computing GCDs of polynomials is also necessary for adding rational functions. Both problems are aided by algebraical concepts which allow us break a hard problem (a multivariate polynomial over the integers) up into several much easier problems (univariate polynomials over finite fields). This is done by applying the Chinese Remainder Algorithm and Hensel Liftings.

1 Introduction

First, let us introduce some basic definitions and arithmetic concepts. For a more verbose explanation see the first chapters of [1].

1.1 Euclidean Algorithm

Definition 1 *An Euclidean domain is an integral domain D with a valuation $v : D\backslash\{0\} \to \mathbb{N}$ with the following properties:*

1. *For all $a, b \in D\backslash\{0\}$, $v(ab) \geq v(a)$*
2. *For all $a, b \in D$ with $b \neq 0$ there exist elements $q, r \in D$ such that $a = bq + r$ where either $r = 0$ or $v(r) < v(b)$*

We let $quo(a, b) := q$ and $rem(a, b) := r$.

Example:

- $\mathbb{Z}$ is a Euclidean domain with the valuation $v(a) = |a|$
- Any field F is a Euclidean domain with the trivial valuation $v(a) = 1 \; \forall \; a \neq 0$
- If F is a field then $F[x]$ is a Euclidean domain with the valuation $v(a(x)) = \deg(a(x))$

We have that
Fields $\subset$ Euclidean domains $\subset$ UFDs $\subset$ integral domains.

In Euclidean domains, we can use the Euclidean algorithm to determine GCDs.

Algorithm: *Euclidean Algorithm*
Input: $a, b \in D$ where D is an Euclidean domain
Output: $GCD(a, b)$

$c \leftarrow n(a)$
$d \leftarrow n(b)$
<u>while</u> $d \neq 0$ <u>do</u> {
  $r \leftarrow rem(c, d)$
  $c \leftarrow d$
  $d \leftarrow r$
$g \leftarrow n(c)$
<u>return</u> g

1.2 Multivariate polynomials

Definition 2 *Let R be a ring. $R[x_1, \ldots, x_k] = R[\mathbf{x}]$ is the set of all multivariate polynomials over R. We write $a(\mathbf{x}) \in R[\mathbf{x}]$ as*

$$a(\mathbf{x}) = \sum_{\mathbf{e} \in \mathbb{N}^k} a_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$$

$\mathbf{e} = (e_1, e_2, \ldots, e_k)$ is the degree vector of a term $x_1^{e_1} x_2^{e_2} \cdots x_k^{e_k}$.

To work with multivariate polynomials, we need some basic arithmetic concepts such as an ordering.

Definition 3 *Let $d, e \in \mathbb{N}^k$ be two exponent vectors. Let $j < k$ be the smallest integer such that $d_j \neq e_j$. Define the lexicographical ordering as follows:*
$d < e$ if $d_j < e_j$
$d > e$ if $d_j > e_j$
The coefficient of the first term of a lexicographically ordered polynomial is called leading coefficient and denoted by lcoeff($a(\mathbf{x})$).

Example: The following polynomial $\in \mathbb{Z}[x, y, z]$ is arranged in lexicographically decreasing order:
$A(x) = 2x^3y^3z^7 + 3x^3y^2z^8 - 5x^2y^7 + z^{10}$

Definition 4
- *The degree vector $\delta(A(\mathbf{x}))$ of a multivariate polynomial is the exponent vector of its leading term.*
- *The total degree of a term with $\mathbf{e} = (e_1, e_2, \ldots, e_k)$ is $\sum_{i=1}^{k} e_i$.*
- *The total degree of a multivariate polynomial is the maximum of all total degrees of its terms.*

In principle, the Euclidean Algorithm can be modified to work in polynomial rings. However, problematic is the growth of remainders. Consider the following example:

Example: Let $A(x), B(x) \in \mathbb{Z}[x]$ be defined as

$$A(x) = x^8 + x^6 - 3x^4 - 3x^3 + x^2 + 2x - 5$$

$$B(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21$$

Running the Euclidean Algorithm in Q yields the following remainder sequence:

$$\begin{aligned} R_2(x) &= -\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3} \\ R_3(x) &= -\frac{117}{25}x^2 - 9x + \frac{411}{25} \\ R_4(x) &= \frac{233150}{19773}x - \frac{102500}{6591} \\ R_5(x) &= -\frac{1288744821}{543589225} \end{aligned}$$

Since $R_5(x)$ is a unit in $\mathbb{Q}$, A and B are relatively prime.

2 GCD

We now explain different algorithms for computing the GCD of multivariate polynomials.

2.1 MGCD

The first algorithm we will look at is called *Modular GCD Algorithm* (**MGCD**). It uses ring homomorphisms to map polynomials from a domain D to a simpler, finite domain D'. Since information is "lost" when we map to finite fields, we need to do this several times, and use fields with different characteristics. It then solves for the GCD in the new domain - for example, by the Euclidean Algorithm. Since it can be shown that deg(GCD in D) $\leq$ deg(GCD in D') we have an upper bound for the degree of the GCD in D.

Multivariate polynomials are handled recursively by viewing $R[x_1, \ldots, x_k]$ as $R[x_1, \ldots, x_{k-1}][x_k]$. This is done by evaluating the polynomial at random points using evaluation homomorphism $\phi_{w-b} : R[w] \rightarrow R$. The GCDs are then mapped to the original domain with the Chinese Remainder Algorithm.

Example: Let $A(x), B(x) \in \mathbb{Z}[x]$ be given as

$$A(x) = x^4 + 25x^3 + 145x^2 - 171x - 360$$

$$B(x) = x^5 + 14x^4 + 15x^3 - x^2 - 14x - 15$$

We now perform several reductions (modulo 5, 7 and 11) and calculate GCDs in $\mathbb{Z}_5, \mathbb{Z}_7, \mathbb{Z}_{11}$, respectively.
In $\mathbb{Z}_5$ we reduct A and B to

$$\begin{aligned} A_5(x) &= x^4 - x \\ B_5(x) &= x^5 - x^4 - x^2 + x \end{aligned}$$

Calculating the GCD in $\mathbb{Z}_5$ yields $x^4 - x$.
In $\mathbb{Z}_7$ we reduct A and B to

$$\begin{aligned} A_7(x) &= x^4 - 3x^3 - 2x^2 - 3x - 3 \\ B_7(x) &= x^5 + x^3 - x^2 - 1 \end{aligned}$$

Calculating the GCD in $\mathbb{Z}_7$ yields $x^2 + 1$.
Since the result in $\mathbb{Z}_7$ gives a much better upper bound for the GCD's degree, we discard the result in $\mathbb{Z}_5$. Such homomorphisms are called *unlucky homomorphism*.
In $\mathbb{Z}_{11}$ we reduct A and B to

$$\begin{aligned} A_{11}(x) &= x^4 + 3x^3 + 2x^2 + 5x + 3 \\ B_{11}(x) &= x^5 + 3x^4 + 4x^3 - x^2 - 3x - 4 \end{aligned}$$

Calculating the GCD in $\mathbb{Z}_{11}$ yields $x^2 + 3x + 4$.
Since two reductions have the same degree, we assume both of them to be good. We thus have

$$GCD(A(x), B(x)) = x^2 + ax + b$$

Using the two GCDs we have

$$\begin{aligned} a &\equiv 0 \mod 7, a \equiv 3 \mod 11 \\ b &\equiv 1 \mod 7, b \equiv 4 \mod 11 \end{aligned}$$

The Chinese Remainder Algorithm can then compute the unique values $a = 14$, $b = 15$. Checking the candidate GCD by division shows that $x^2 + 14x + 15$ is indeed the desired GCD.

Algorithm: *Modular GCD algorithm MGCD*
Input: $A, B \in \mathbb{Z}[x_1, \ldots, x_k]$
Output: GCD of A, B

// Make A and B monic, compute coefficient bound
$a \leftarrow icont(A), b \leftarrow icont(B), A \leftarrow A/a, B \leftarrow B/b;$
$c \leftarrow igcd(a, b), g \leftarrow igcd(lcoeff(A), lcoeff(B))$

$(q, H) \leftarrow (0, 0)$, $n \leftarrow min(deg_k(A), deg_k(B))$
$limit \leftarrow 2^n\,|g|\,min(|A|_\infty, |B|_\infty)$
// Choose homomorphisms

while true do { $p \leftarrow New(LargePrime)$
  while $p \mid g$ do $p \leftarrow New(LargePrime)$
  $A_p \leftarrow A \mod p$; $B_p \leftarrow B \mod p$
  $g_p \leftarrow g \mod p$; $C_p \leftarrow PGCD(A_p, B_p, p)$; $m \leftarrow deg_k(C_p)$;

  // Normalise so that $g_p = lcoeff(C_p)$
  $C_p \leftarrow g_p lcoeff(C_p)^{-1} C_p$
// Test for unlucky homomorphisms
  if $m < n$ then {
    $(q, H) \leftarrow (p, C_p)$; $n \leftarrow m$ }
  elseif $m = n$ then {
// Test for completion, update coefficients of GCD

// candidate H via integer CRA
  for all coefficients h_i in H do {
    $h_i \leftarrow IntegerCRA([q, p], [h_i, (c_p)_i])$
    $q \leftarrow qp$ }
  if $q > limit$ then {
// Remove integer content of result, division check

    $C \leftarrow pp(H)$
    if $C \mid A$ and $C \mid B$ then
      return $c \cdot C$ }
  elseif $m = 0$ then
    return c
end

Luckily, it can be shown that unlucky homomorphisms are rare, and do therefore not impact the algorithm significantly.
Problematic for multivariate polynomials, however, is the fact that the number of domains which have to be used is exponential in the number of variables. This is very ineffective when the polynomials have a sparse rather than a dense structure. Whilst MGCD might therefore work well for univariate polynomials, for multivariate polynomials we need to look at other algorithms.

2.2 SparseMod

A probabilistic algorithm better suited for multivariate polynomials was developed by Zippel in 1979 in his PhD thesis ([2]). The algorithm is based on the observation, that evaluating a polynomial at a random point will almost never yield zero. It determines the GCD using probabilistic techniques, and by constructing alternating sequences of dense and sparse interpolations. A *dense interpolation* assumes that all coefficients of the polynomial of degree n are unknown. It therefore requires $n + 1$ evaluation points to determine the polynomial. We assume that the coefficients which are zero in one dense interpolation are probably zero in all cases.
In contrast, a *sparse interpolation* assumes only t unknown coefficients where $t << n$. It can be computed with $t + 1$ evaluation points.
We will illustrate this technique with an example from [3]

Example: Let $G, A, B \in \mathbb{Z}[x, y]$ be defined as follows:

$$\begin{aligned} G(x,y) &= x^2 + 3y^3x + 35 \\ A(x,y) &= (y+1)G = (y+1)x^2 + (3y^4 + 3y^3)x + 35y + 35 \\ B(x,y) &= (x+1)G = x^3 + (3y^3+1)x^2 + (3y^3+35)x + 35 \end{aligned}$$

We work in $\mathbb{Z}_{11}$ and compute the first GCD image using dense interpolation:

1. Compute degree bound on y: $d_y = \deg(GCD(A(1,y), B(1,y)) \mod 11) = 3$

2. Evaluate at four random points (this is the *dense interpolation*) for y and compute univariate GCD images in $\mathbb{Z}_{11}$ using Euclidean algorithm:

$$\begin{aligned} g_1 = GCD(A(x,2), B(x,2)) \mod 11 &= x^2 + 2x + 2 \\ g_2 = GCD(A(x,4), B(x,4)) \mod 11 &= x^2 + 5x + 2 \\ g_3 = GCD(A(x,5), B(x,5)) \mod 11 &= x^2 + x + 2 \\ g_4 = GCD(A(x,6), B(x,6)) \mod 11 &= x^2 + 10x + 2 \end{aligned}$$

3. Interpolate in y to construct the first bivariate image modulo 11:

$$G_1 = x^2 + 3y^3x + 2$$

Now we work in $\mathbb{Z}_{13}$ to compute a second GCD using sparse interpolation:

4. Assume the form of the GCD and substitute variables:

$$H = x^2 + \alpha y^3 x + \beta$$

5. Evaluate at $y = 8$ (*sparse interpolation*):

$$\begin{aligned} H(x,9) \mod 13 &= x^2 + 5\alpha x + \beta \\ GCD(A(x,8), B(x,8)) \mod 13 &= x^2 + 2x + 9 \end{aligned}$$

6. Equate the coefficients and solve in $\mathbb{Z}_{13}$

$$\alpha = 3, \beta = 9$$

7. Substitute this back in H to get second bivariate image:

$$G_2 = x^2 + 3y^3x + 9$$

8. Finally, apply CRA to G_1, G_2 to reconstruct coefficients in $\mathbb{Z}$:

$$x^2 + 3y^3x + 35$$

This is just an example illustrating how the algorithm works. For the complete algorithm and detailed discussion see chapter 7 of [1] and [2].

2.3 EZ-GCD

Another GCD algorithm better suited for multivariate polynomials is the *Extended Zassenhaus Algorithm* (**EZ-GCD**). It was developed by Moses and Yun in 1973 ([4]) and named after Zassenhaus to honour his work in number theory. Again, multivariate polynomials are reduced to a univariate representation, the GCD is then determined in a simpler domain. Hensel's Lemma is used repeatedly to lift the results up to the multivariate domain. As with the MGCD, relatively prime polynomials are discovered quickly.
Algorithm: *Extended Zassenhaus GCD algorithm EZ-GCD*
Input: $A, B \in \mathbb{Z}[x, y_1, \ldots, y_k]$
Output: $GCD(A, B)$

1. Viewing A, B as polynomials with coefficients from $\mathbb{Z}[y_1, \ldots, y_k]$ compute content, primitive part, lcoeff
2. Choose valid evaluation prime (p so that $lcoeff \neq 0 \mod p$)
3. Choose an evaluation point $\mathbf{b} = (b_1, \ldots, b_k), 0 \leq b_i < p$
4. Compute $A(\mathbf{b}) \mod p$ and $B(\mathbf{b}) \mod p$ and their GCD g_1. If $\deg(g_1) = 0$ then $GCD(A, B) = GCD(cont(A), cont(B))$.
5. To check the answer, we choose second prime and evaluation point and compute second GCD g_2. If their degrees are not equal, throw the result with bigger degree away. Repeat until we have two GCD candidates with equal degrees in $\mathbb{Z}_p[x]$
6. Construct candidates for cofactors
7. Use Hensel liftings to lift factorisations in $\mathbb{Z}_p[x]$ up to $\mathbb{Z}[x, y_1, \ldots, y_k]$
8. Check if result is indeed a GCD in $\mathbb{Z}[x, y_1, \ldots, y_k]$

Numerous improvements of the EZ-GCD Algorithm have been published, for some details see [1] and [5].

3 Factorisation

3.1 Introduction

Multivariate factorisation problems over $\mathbb{Z}$ can also be reduced to univariate problems modulo a prime. At first we look at an easy algorithm which calculates square-free factors of a polynomial. This algorithm can be applied by other factorisation algorithms, and the square-free decomposition used to find the "proper" factorisation.

Definition 5 *A primitive polynomial $a(x) \in R[x]$ is called square-free if it has no repeated factors.*

Definition 6 *The square-free factorisation of $a(x)$ is $a(x) = \prod_{i=1}^{k} a_i(x)^i$, where each $a_i(x)$ is square-free, and $GCD(a_i(x), a_j(x)) = 1$ for $i \neq j$.*

Theorem 1 *Let $a(x) \in R[x]$, R UFD with $char(R) = 0$.*
Then $a(x)$ has repeated factors iff $GCD(a(x), a'(x)) \neq 1$.

This result can be used to determine the square-free factorisation.
Let $a(x) = \prod_{i=1}^{k} a_i(x)^i$ be the square-free factorisation of a. Taking the derivative, we have

$$a'(x) = \sum_{i=1}^{k} a_1(x) \cdots a_{i-1}(x) \cdot i \cdot a_i(x)^{(i-1)} a_i'(x) a_{i+1}(x)^{i+1} \cdots a_k(x)^k$$

Let

$$c(x) := GCD(a, a') = \prod_{i=2}^{k} a_i(x)^{i-1}$$

and

$$w(x) := a(x)/c(x) = a_1(x)a_2(x)\cdots a_k(x)$$

Then $w(x)$ is the product of the square-free factors without their multiplicities. We now calculate

$$y(x) := GCD(c(x), w(x))$$

Note that $a_1(x) = w(x)/y(x)$ gives us the first factor.
We can now use these equations recursively to compute all square-free factors. This algorithm is called *SquareFree*.
It can be modified to work with polynomials over $GF(q)$.
A little better, however, is the algorithm *Yun's Square-Free Factorisation*. It calculates one additional differentiation, but manages to avoid the repeated GCD calculation of Square-Free.
Berlekamp's Algorithm factorises univariate polynomials over the Galois Field $GF(q)$.
Multivariate factorisation can, similar to GCDs, be accomplished by factorising univariate polynomials over a finite field and Hensel liftings. A discussion of this can be found in [1], chapter 8. A paper which explores multivariate factorisation in greater detail is [6].

References

[1] K Geddes, S Czapor, G Labahn: *Algorithms for Computer Algebra, 1992*

[2] R Zippel: *Probabilistic Algorithms for Sparse Polynomials, 1979*

[3] J de Kleine, M Monagan, A Wittkopf: *The Non-Monic Case in the Sparse Modular GCD Algorithm, 2005*

[4] J Moses, D Yun: *The EZ GCD Algorithm, 1973*

[5] P Wang: *The EEZ-GCD Algorithm, 1980*

[6] D Musser: *Multivariate Polynomial Factorization, 1975*

[7] E Berlekamp: *Factoring Polynomials Over Finite Fields, 1967*