2. Greatest common divisors of polynomials

The Euclidean algorithm (Eukleídes, ca. 300 BC) is sometimes described as the oldest non-trivial algorithm in Mathematics. It was originally formulated by Euclid for positiv integers, but it can be applied to so-called Euclidean domains, as we shall see. In this chapter we are concerned with the Euclidean algorithm in commutative rings of polynomials.

Proofs of theorems in this chapter can be found in [Winkler 1996], Chapter 4.

2.1. The Euclidean algorithm

Definition 2.1.1. Let I be an integral domain, $a(x), b(x) \in I[x]$. A polynomial $g(x) \in I[x]$ is **a greatest common divisor (gcd)** of a and b, iff

(i) g divides (evenly) both a and b (i.e. g is **a common divisor** of a and b) and

(ii) every other common divisor h of a and b divides g. □

The greatest common divisor is only determined up to multiplication by constants; i.e., in a polynomial domain $K[x]$, K a field, the gcd is only determined up to multiplication by non-zero field elements.

Theorem 2.1.2. *The (extended) Euclidean algorithm GCD_EUCLID computes the gcd $g(x)$ of polynomials $a(x), b(x)$ over a field K, and the Bézout cofactors $s(x), t(x)$ s.t. $g = s \cdot a + t \cdot b$:*

GCD_EUCLID
for given non-zero polynomials $a, b \in K[x]$,
the greatest common divisior g and the Bézout cofactors s, t are computed
(1) $(r_0, r_1, s_0, s_1, t_0, t_1) := (a, b, 1, 0, 0, 1)$;
$i := 1$;
(2) **while** $r_i \neq 0$ **do**
$q_i :=$ quotient of r_{i-1} on division by r_i;
$(r_{i+1}, s_{i+1}, t_{i+1}) := (r_{i-1}, s_{i-1}, t_{i-1}) - q_i \cdot (r_i, s_i, t_i)$;
$i := i + 1$
endwhile ;
(3) $(g, s, t) := (r_{i-1}, s_{i-1}, t_{i-1})$; **return**

Example 2.1.3. We consider univariate polynomials over $\mathbb{Q}$. Starting from the polynomials

$$\begin{aligned} a = r_0 &= x^4 + x^3 + 2x^2 + x - 1, \\ b = r_1 &= 2x^3 + 3x^2 + 3x + 2, \end{aligned}$$

the Euclidean algorithm generates the sequence of remainders

$$\begin{aligned} r_2 &= \tfrac{5}{4}x^2 + \tfrac{3}{4}x - \tfrac{1}{2}, \\ r_3 &= \tfrac{68}{25}x + \tfrac{68}{25}, \\ r_4 &= 0. \end{aligned}$$

So $x + 1$ is a gcd of a and b. □

Now let us investigate the computation of greatest common divisors of polynomials with coefficients in a unique factorization domain (ufd), for instance in $\mathbb{Z}$. Throughout this section we let I be a unique factorization domain and K the quotient field of I.

Definition 2.1.4. A univariate polynomial $f(x)$ over the ufd I is *primitive* iff there is no prime in I which divides all the coefficients in $f(x)$. □

Theorem 2.1.5. (Gauss' Lemma) *Let f, g be primitive polynomials over the ufd I. Then also $f \cdot g$ is primitive.*

Proof: Let $f(x) = \sum_{i=0}^{m} a_i x^i$, $g(x) = \sum_{i=0}^{n} b_i x^i$. For an arbitrary prime p in I, let j and k be the minimal indices such that p does not divide a_j and b_k, respectively. Then p does not divide the coefficient of x^{j+k} in $f \cdot g$. □

Gcd's and factorization are basically the same over I and over K.

Corollary. *Let $f_1, f_2 \in I[x]$ be primitive. Let $\tilde{g} = \gcd(f_1, f_2)$ in $K[x]$. Then there is $a \in K^*$ s.t. $a \cdot \tilde{g} = \gcd(f_1, f_2)$ in $I[x]$.*

Proof: Take a s.t. $g = a \cdot \tilde{g}$ is primitive in $I[x]$. Then

$$f_1 = a_1 \cdot h_1 \cdot g \ , \quad f_2 = a_2 \cdot h_2 \cdot g$$

for some $a_1, a_2 \in K$ and h_1, h_2 primitive in $I[x]$. By Gauss' Lemma $h_1 g$ and $h_2 g$ are primitive, and so are f_1, f_2. So a_1, a_2 have to be units in I. □

Definition 2.1.6. Up to multiplication by units we can decompose every polynomial $a(x) \in I[x]$ uniquely into

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

where $\mathrm{cont}(a) \in I$ and $\mathrm{pp}(a)$ is a primitive polynomial in $I[x]$. $\mathrm{cont}(a)$ is the *content* of $a(x)$, $\mathrm{pp}(a)$ is the *primitive part* of $a(x)$. □

Definition 2.1.7. Two non-zero polynomials $a(x), b(x) \in I[x]$ are *similar* iff there are *similarity coefficients* $\alpha, \beta \in I^*$ such that $\alpha \cdot a(x) = \beta \cdot b(x)$. In this case we write $a(x) \simeq b(x)$. Obviously $a(x) \simeq b(x)$ if and only if $\mathrm{pp}(a) = \mathrm{pp}(b)$. $\simeq$ is an equivalence relation preserving the degree.
In $I[x]$ we might not be able to divide polynomials $a(x), b(x)$ with quotient and remainder; the problem is that the leading coefficients might not be divisible. But we can certainly divide $\mathrm{lc}(b)^{m-n+1} \cdot a(x)$ by $b(x)$, where $m = \deg(a), n = \deg(b)$. The resulting quotient and remainder are called *pseudo-quotient* and *pseudo-remainder*, written as $\mathrm{pquot}(a, b)$ and $\mathrm{prem}(a, b)$. □

Definition 2.1.8. Let k be a natural number greater than 1, and $f_1, f_2, \ldots, f_{k+1}$ polynomials in $I[x]$.

Then $f_1, f_2, \ldots, f_{k+1}$ is a *polynomial remainder sequence (prs)* iff

- $\deg(f_1) \geq \deg(f_2)$,
- $f_i \neq 0$ for $1 \leq i \leq k$ and $f_{k+1} = 0$,
- $f_i \simeq \text{prem}(f_{i-2}, f_{i-1})$ for $3 \leq i \leq k+1$. □

Lemma 2.1.9. *Let* $a, b, a', b' \in I[x]^*$, $\deg(a) \geq \deg(b)$, *and* $r \simeq \text{prem}(a, b)$.

(a) If $a \simeq a'$ *and* $b \simeq b'$ *then* $\text{prem}(a, b) \simeq \text{prem}(a', b')$.

(b) $\gcd(a, b) \simeq \gcd(b, r)$.

Proof: Let $\alpha a = \alpha' a'$, $\beta b = \beta' b'$, and $m = \deg(a), n = \deg(b)$. By Lemma 2.2.4 in [Winkler 1996] we get

$$\begin{aligned} \beta^{m-n+1}\alpha\text{prem}(a, b) &= \text{prem}(\alpha a, \beta b) \\ &= \text{prem}(\alpha' a', \beta' b') = (\beta')^{m-n+1}\alpha'\text{prem}(a', b'). \end{aligned}$$ □

Therefore, if $f_1, f_2, \ldots, f_k, 0$ is a prs, then

$$\gcd(f_1, f_2) \simeq \gcd(f_2, f_3) \simeq \ldots \simeq \gcd(f_{k-1}, f_k) \simeq f_k.$$

If f_1 and f_2 are primitive, then by Gauss' Lemma also their gcd must be primitive, i.e. $\gcd(f_1, f_2) = \text{pp}(f_k)$. So the gcd of polynomials over the ufd I can be computed by the algorithm GCD_PRS.

These considerations lead to the following algorithm for computing the gcd of polynomials.

GCD_PRS (computation of gcd by prs)
for given non-zero polynomials $a, b \in I[x]$,
their greatest common divisor $g = \gcd(a, b)$ is computed
(1) **if** $\deg(a) \geq \deg(b)$
then $f_1 := \text{pp}(a)$**;** $f_2 := \text{pp}(b)$
else $f_1 := \text{pp}(b)$**;** $f_2 := \text{pp}(a)$**;**
(2) $d := \gcd(\text{cont}(a), \text{cont}(b))$**;**
(3) compute $f_3, \ldots, f_k, f_{k+1} = 0$ such that $f_1, f_2, \ldots, f_k, 0$ is a prs**;**
(4) $g := d \cdot \text{pp}(f_k)$**; return** g □

Actually GCD_PRS is a family of algorithms, depending on how exactly we choose the elements of the prs in step (3). Starting from primitive polynomials f_1, f_2, there are various possibilities for this choice.

In the so-called *generalized Euclidean algorithm* we simply set

$$f_i := \text{prem}(f_{i-2}, f_{i-1}) \quad \text{for} \quad 3 \leq i \leq k+1.$$

This choice, however, leads to an enormous blow-up of coefficients, as can be seen in the following example.

Example 2.1.10. We consider polynomials over $\mathbb{Z}$. Starting from the primitive polynomials

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

the generalized Euclidean algorithm generates the prs

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

So the gcd of f_1 and f_2 is the primitive part of f_6, i.e. 1. □

Although the inputs and the output of the algorithm may have extremely short coefficients, the coefficients in the intermediate results may be enormous. In particular, for univariate polynomials over $\mathbb{Z}$ the length of the coefficients grows exponentially at each step (see (Knuth 1981), Section 4.6.1). This effect of intermediate coefficient growth is even more dramatic in the case of multivariate polynomials.

Another possible choice for computing the prs in GCD_PRS is to shorten the coefficients as much as possible, i.e. always eliminate the content of the intermediate results.

$$f_i := \mathrm{pp}(\mathrm{prem}(f_{i-2}, f_{i-1})).$$

We call such a prs a *primitive prs*.

Example 2.1.10.(continued) The primitive prs starting from f_1, f_2 is

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

Keeping the coefficients always in the shortest form carries a high price. For every intermediate result we have to determine its content, which means doing a lot of gcd computations in the coefficient domain.

The goal, therefore, is to keep the coefficients as short as possible without actually having to compute a lot of gcd's in the coefficient domain. So we set

$$\beta_i f_i := \mathrm{prem}(f_{i-2}, f_{i-1}),$$

where β_i, a factor of $\mathrm{cont}(\mathrm{prem}(f_{i-2}, f_{i-1}))$ needs to be determined. The best algorithm of this form known is Collins' *subresultant prs algorithm* (Collins 1967), (Brown,Traub 1971).

So, as we have seen, the algorithm GCD_EUCLID works for integers and for univariate polynomials over a field. But what do we actually need for executing this algorithm? This question leads to the notion of a Euclidean domain.

Definition 2.1.11. A *Euclidean domain (ED) E* is an integral domain together with a *degree function* $\deg : E^* \longrightarrow \mathbb{N}$, such that

(i) $\deg(a \cdot b) \geq \deg(a)$ for all $a, b \in E^*$,

(ii) (division property) for all $a, b \in E$, $b \neq 0$, there exists a *quotient* q and a *remainder* r in D such that $a = q \cdot b + r$ and ($r = 0$ or $\deg(r) < \deg(b)$. □

Example 2.1.12. Examples of EDs are the integers $\mathbb{Z}$ with $\deg(a) = |a|$;
or a field K with $\deg(a) = 1$ for all $a \in K^*$,
or univariate polynomials over a field K, i.e. $K[x]$ with the usual degree function for polynomials.
But $K[x_1, \ldots, x_n]$, for $n > 1$, is not an ED;
and neither is $R[x]$, for a ring R which is not a field. □

Theorem 2.1.13. *A Euclidean domain E is a principal ideal domain (PID).*

Proof: Let I be an ideal in E. If $I = \langle 0 \rangle$, then I is generated by a single element.
Otherwise, let $a \in I^*$ s.t. $\deg(a)$ is minimal. For an arbitrary $b \in I$ we consider the remainder on division of b by a. So for some q we have

$$b = q \cdot a + r, \quad \text{where } (r = 0 \text{ or } \deg(r) < \deg(a)\,).$$

Obviously $r \in I$, so it cannot have degree less than a, and we must have $r = 0$. Thus, for all $b \in I$ we have $b \in \langle a \rangle$, and therefore $I = \langle a \rangle$. □

2.2. A modular gcd algorithm

For motivation let us once again look at the polynomials in Example 2.1.10,

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

If f_1 and f_2 have a common factor h, then for some q_1, q_2 we have

$$f_1 = q_1 \cdot h, \qquad f_2 = q_2 \cdot h \ . \tag{2.2.1}$$

These relations stay valid if we take every coefficient in (2.2.1) modulo 5. But modulo 5 we can compute the gcd of f_1 and f_2 in a very fast way, since all the coefficients that will ever appear are bounded by 5. In fact the gcd of f_1 and f_2 modulo 5 is 1. By comparing the degrees on both sides of the equations in (3.2.1) we see that also over the integers $\gcd(f_1, f_2) = 1$. In this section we want to generalize this approach and derive a modular algorithm for computing the gcd of polynomials over the integers.

Clearly the coefficients in the gcd can be bigger than the coefficients in the inputs:

$$\begin{aligned} a &= x^3 + x^2 - x - 1 &&= (x+1)^2(x-1), \\ b &= x^4 + x^3 + x + 1 &&= (x+1)^2(x^2 - x + 1), \\ \gcd(a, b) &= x^2 + 2x + 1 &&= (x+1)^2. \end{aligned}$$

So how big can the coefficients in the gcd be?

Theorem 2.2.1. (Landau–Mignotte–bound) *Let $a(x) = \sum_{i=0}^{m} a_i x^i$ and $b(x) = \sum_{i=0}^{n} b_i x^i$ be polynomials over $\mathbb{Z}$ ($a_m \neq 0 \neq b_n$) such that b divides a. Then*

$$\sum_{i=0}^{n} | b_i | \quad \leq \quad 2^n \left| \frac{b_n}{a_m} \right| \sqrt{\sum_{i=0}^{m} a_i^2}, \qquad \text{or} \qquad ||b||_1 \ \leq \ 2^n \cdot |b_n/a_m| \cdot ||a||_2. \qquad \square$$

Corollary. *Let $a(x) = \sum_{i=0}^{m} a_i x^i$ and $b(x) = \sum_{i=0}^{n} b_i x^i$ be polynomials over $\mathbb{Z}$ ($a_m \neq 0 \neq b_n$). Every coefficient of the gcd of a and b in $\mathbb{Z}[x]$ is bounded in absolute value by*

$$2^{\min(m,n)} \cdot \gcd(a_m, b_n) \cdot \min\Big(\frac{1}{| a_m |} ||a||_2, \frac{1}{| b_n |} ||b||_2 \Big). \qquad \square$$

The gcd of $a(x) \bmod p$ and $b(x) \bmod p$ may not be the modular image of the integer gcd of a and b. An example for this is $a(x) = x - 3, b(x) = x + 2$. The gcd over $\mathbb{Z}$ is 1, but modulo 5 a and b are equal and their gcd is $x + 2$. But fortunately these situations are rare.

So what we want from a prime p is the commutativity of the following diagram, where ϕ_p is the homomorphism from $\mathbb{Z}[x]$ to $\mathbb{Z}_p[x]$ defined as $\phi_p(f(x)) = f(x) \bmod p$.

$$\begin{array}{ccc} \mathbb{Z}[x] \times \mathbb{Z}[x] & \longrightarrow^{\phi_p} & \mathbb{Z}_p[x] \times \mathbb{Z}_p[x] \\ \text{gcd in } \mathbb{Z}[x] \downarrow & & \downarrow \text{ gcd in } \mathbb{Z}_p[x] \\ \mathbb{Z}[x] & \longrightarrow^{\phi_p} & \mathbb{Z}_p[x] \end{array}$$

This diagram commutes for all those primes p which do not divide a certain resultant. We will discuss resultants in the next chapter. Here it suffices to know that two polynomials have a non-trivial gcd if and only if their resultant is 0.

Lemma 2.2.2. *Let $a, b \in \mathbb{Z}[x]^*$, p a prime number not dividing the leading coefficients of both a and b. Let $a_{(p)}$ and $b_{(p)}$ be the images of a and b modulo p, respectively. Let $c = \gcd(a, b)$ over $\mathbb{Z}$.*

(a) $\deg(\gcd(a_{(p)}, b_{(p)})) \geq \deg(\gcd(a, b))$.

(b) *If p does not divide the resultant of a/c and b/c, then* $\gcd(a_{(p)}, b_{(p)}) = c \bmod p$.

Proof:
(a) $\gcd(a, b) \bmod p$ divides both $a_{(p)}$ and $b_{(p)}$, so it divides $\gcd(a_{(p)}, b_{(p)})$. Therefore $\deg(\gcd(a_{(p)}, b_{(p)})) \geq \deg(\gcd(a, b) \bmod p)$. But p does not divide the leading coefficient of $\gcd(a, b)$, so $\deg(\gcd(a, b) \bmod p) = \deg(\gcd(a, b))$.

(b) Let $c_{(p)} = c \bmod p$. a/c and b/c are relatively prime. $c_{(p)}$ is non-zero. So

$$\gcd(a_{(p)}, b_{(p)}) = c_{(p)} \cdot \gcd(a_{(p)}/c_{(p)}, b_{(p)}/c_{(p)}).$$

If $\gcd(a_{(p)}, b_{(p)}) \neq c_{(p)}$, then the gcd of the right hand side must be non-trivial. Therefore $\mathrm{res}(a_{(p)}/c_{(p)}, b_{(p)}/c_{(p)}) = 0$. The resultant, however, is a sum of products of coefficients, so p has to divide $\mathrm{res}(a/c, b/c)$. □

Of course, the gcd of polynomials over $\mathbb{Z}_p$ is determined only up to multiplication by non-zero constants. So by "$\gcd(a_{(p)}, b_{(p)}) = c \bmod p$" we actually mean "$c \bmod p$ is a gcd of $a_{(p)}, b_{(p)}$".

From Lemma 2.2.2 we know that there are only finitely many primes p which do not divide the leading coefficients of a and b but for which $\deg(\gcd(a_{(p)}, b_{(p)})) > \deg(\gcd(a, b))$. When these degrees are equal we call p a *lucky* prime.

In the sequel we describe a modular algorithm that chooses several primes, computes the gcd modulo these primes, and finally combines these modular gcd's by an application of the Chinese remainder algorithm. Since in $\mathbb{Z}_p[x]$ the gcd is defined only up to multiplication by constants, we are confronted with the so–called *leading coefficient problem*. The reason for this problem is that over the integers the gcd will, in general, have a leading coefficient different from 1, whereas over $\mathbb{Z}_p$ the leading coefficient can be chosen arbitrarily. So before we can apply the Chinese remainder algorithm we have to normalize the leading coefficient of $\gcd(a_{(p)}, b_{(p)})$. Let a_m, b_n be the leading coefficients of a and b, respectively. The leading coefficient of the gcd divides the gcd of a_m and b_n. Thus, for primitive polynomials we may normalize the leading coefficient of $\gcd(a_{(p)}, b_{(p)})$ to $\gcd(a_m, b_n) \bmod p$ and in the end take the primitive part of the result. These considerations lead to the following modular gcd algorithm.

GCD_MOD(modular gcd algorithm)
for given non-zero primitive polynomials $a, b \in \mathbb{Z}[x]^*$,
their greatest common divisor $g = \gcd(a, b)$ is computed.
Integers modulo m are represented as $\{k \mid -m/2 < k \le m/2\}$.

(1) $d := \gcd(\mathrm{lc}(a), \mathrm{lc}(b))$;
$M := 2 \cdot d \cdot (\mathrm{Landau} - \mathrm{Mignotte} - \mathrm{bound\ for\ } a, b)$;
[in fact any other bound for the size of the coefficients can be used]

(2) $p :=$ a new prime not dividing d;
$c_{(p)} := \gcd(a_{(p)}, b_{(p)})$; [with $\mathrm{lc}(c_{(p)}) = 1$]
$g_{(p)} := (d \bmod p) \cdot c_{(p)}$;

(3) **if** $\deg(g_{(p)}) = 0$ **then** $\{g := 1$; **return**$\}$;
$P := p$;
$g := g_{(p)}$;

(4) **while** $P \le M$ **do**
  $p :=$ a new prime not dividing d;
  $c_{(p)} := \gcd(a_{(p)}, b_{(b)})$; [with $\mathrm{lc}(c_{(p)}) = 1$]
  $g_{(p)} := (d \bmod p) \cdot c_{(p)}$;
  if $\deg(g_{(p)}) < \deg(g)$ **then goto** (3);
  if $\deg(g_{(p)}) = \deg(g)$
  then $g := \mathrm{CRA}(g, g_{(p)}, P, p)$;
    [actually CRA is applied to the coefficients of g and $g_{(p)}$]
    $P := P \cdot p$

(5) $g := \mathrm{pp}(g)$;
if $g \mid a$ and $g \mid b$ **then return** g ;
goto (2) □

In Step 4 we know the coefficients of a polynomial modulo P and p, and we want to know them modulo $P \cdot p$. So we have to solve a so-called **Chinese remainder problem (CRP) in** $\mathbb{Z}$:

given: $r_1, \ldots, r_n \in \mathbb{Z}$ (remainders)
$m_1, \ldots, m_n \in \mathbb{Z}^*$ (moduli), pairwise relatively prime

find: $r \in \mathbb{Z}$, such that $r \equiv r_i \bmod m_i$ for $1 \le i \le n$.

The following algorithm CRA (Chinese remainder algorithm) solves this problem. For details see [Winkler 1996].

CRA(Chinese remainder algorithm)
for given remainders r_1, r_2 and moduli m_1, m_2
a solution r of the corresponding CRP is computed

(1) $c := m_1^{-1} \bmod m_2$;
(2) $r_1' := r_1 \bmod m_1$;
(3) $\sigma := (r_2 - r_1')c \bmod m_2$;
(4) $r := r_1' + \sigma m_1$; **return** r □

Usually we do not need as many primes as the Landau–Mignotte–bound tells us for determining the integer coefficients of the gcd in GCD_MOD. Whenever g remains unchanged for a series of iterations through the **while**–loop, we might apply the test in step (5) and exit if the outcome is positive.

Example 2.2.3. We apply GCD_MOD for computing the gcd of

$$\begin{aligned} a &= 2x^6 - 13x^5 + 20x^4 + 12x^3 - 20x^2 - 15x - 18, \\ b &= 2x^6 + x^5 - 14x^4 - 11x^3 + 22x^2 + 28x + 8. \end{aligned}$$

$d = 2$. The bound in step (1) is

$$M = 2 \cdot 2 \cdot 2^6 \cdot 2 \cdot \min\Big(\frac{1}{2}\sqrt{1666}, \frac{1}{2}\sqrt{1654}\Big) \sim 10412.$$

As the first prime we choose $p = 5$. $g_{(5)} = (2 \bmod 5)(x^3 + x^2 + x + 1)$. So $P = 5$ and $g = 2x^3 + 2x^2 + 2x + 2$.
Now we choose $p = 7$. We get $g_{(7)} = 2x^4 + 3x^3 + 2x + 3$. Since the degree of $g_{(7)}$ is higher than the degree of the current g, the prime 7 is discarded.
Now we choose $p = 11$. We get $g_{(11)} = 2x^3 + 5x^2 - 3$. By an application of CRA_2 to the coefficients of g and $g_{(11)}$ modulo 5 and 11, respectively, we get $g = 2x^3 + 27x^2 + 22x - 3$. P is set to 55.
Now we choose $p = 13$. We get $g_{(13)} = 2x^2 - 2x - 4$. All previous results are discarded, we go back to step (3), and we set $P = 13, g := 2x^2 - 2x - 4$.
Now we choose $p = 17$. We get $g_{(17)} = 2x^2 - 2x - 4$. By an application of CRA_2 to the coefficients of g and $g_{(17)}$ modulo 13 and 17, respectively, we get $g = 2x^2 - 2x - 4$. P is set to 221.
In general we would have to continue choosing primes. But following the suggestion above, we apply the test in step (5) to our partial result and we see that $\mathrm{pp}(g)$ divides both a and b. Thus, we get $\gcd(a, b) = x^2 - x - 2$. □

Multivariate polynomials

We generalize the modular approach for univariate polynomials over $\mathbb{Z}$ to multivariate polynomials over $\mathbb{Z}$. So the inputs are elements of $\mathbb{Z}[x_1,\ldots,x_{n-1}][x_n]$, where the coefficients are in $\mathbb{Z}[x_1,\ldots,x_{n-1}]$ and the main variable is x_n. In this method we compute modulo irreducible polynomials $p(x)$ in $\mathbb{Z}[x_1,\ldots,x_{n-1}]$. In fact we use linear polynomials of the form $p(x) = x_{n-1} - r$ where $r \in \mathbb{Z}$. So reduction modulo $p(x)$ is simply evaluation at r.

For a polynomial $a \in \mathbb{Z}[x_1,\ldots,x_{n-2}][y][x]$ and $r \in \mathbb{Z}$ we let a_{y-r} stand for a mod $y-r$. Obviously the proof of Lemma 3.2.2 can be generalized to this situation.

Lemma 2.2.4. *Let $a, b \in \mathbb{Z}[x_1,\ldots,x_{n-2}][y][x]^*$ and $r \in \mathbb{Z}$ such that $y-r$ does not divide both $\mathrm{lc}_x(a)$ and $\mathrm{lc}_x(b)$. Let $c = \gcd(a,b)$.*

(a) $\deg_x(\gcd(a_{y-r}, b_{y-r})) \geq \deg_x(\gcd(a,b))$.

(b) *If $y-r \nmid \mathrm{res}_x(a/c, b/c)$ then $\gcd(a_{y-r}, b_{y-r}) = c_{y-r}$.* □

The analogue to the Landau-Mignotte bound is even easier to derive: let c be a factor of a in $\mathbb{Z}[x_1,\ldots,x_{n-2}][y][x]$. Then $\deg_y(c) \leq \deg_y(a)$. This leads to an algorithm GCD_MODm for modular computation of gcds of multivariate polynomials.

Example 2.2.5. We look at an example in $\mathbb{Z}[x,y]$. Let

$$\begin{aligned} a(x,y) &= 2x^2y^3 - xy^3 + x^3y^2 + 2x^4y - x^3y - 6xy + 3y + x^5 - 3x^2, \\ b(x,y) &= 2xy^3 - y^3 - x^2y^2 + xy^2 - x^3y + 4xy - 2y + 2x^2. \end{aligned}$$

We get as a degree bound for the gcd

$$M = 1 + \min(\deg_x(a), \deg_x(b)) = 4.$$

The algorithm proceeds as follows:

$r = 1:$ $\gcd(a_{x-1}, b_{x-1}) = y + 1$.

$r = 2:$ $\gcd(a_{x-2}, b_{x-2}) = 3y + 4$.
Now we use Newton interpolation to obtain $g = (2x-1)y + (3x-2)$.

$r = 3:$ $\gcd(a_{x-3}, b_{x-3}) = 5y + 9$. Now by Newton interpolation we obtain $g = (2x-1)y + x^2$ and this is the gcd (the algorithm would actually take another step). □

GCD_MODm (multivariate modular gcd algorithm)
for given non-zero polynomials $a, b \in \mathbb{Z}[x_1, \ldots, x_s][x_n]$ and $0 \leq s < n$
the greatest common divisor $g = \gcd(a, b)$ is computed by evaluation of x_s.

(0) **if** $s = 0$ **then** $g := \gcd(\text{cont}(a), \text{cont}(b))\text{GCD\_MOD}(\text{pp}(a), \text{pp}(b))$ **; return** g **;**

(1) $M := 1 + \min(\deg_{x_s}(a), \deg_{x_s}(b))$;
$a' := \text{pp}_{x_n}(a)$; $b' := \text{pp}_{x_n}(b)$;
$f := \text{GCD\_MODm}(\text{cont}_{x_n}(a), \text{cont}_{x_n}(b), s, s-1)$;
$d := \text{GCD\_MODm}(\text{lc}_{x_n}(a'), \text{lc}_{x_n}(b'), s, s-1)$;

(2) $r :=$ an integer s.t. $\deg_{x_n}(a'_{x_s-r}) = \deg_{x_n}(a')$ or $\deg_{x_n}(b'_{x_s-r}) = \deg_{x_n}(b')$;
$g'_{(r)} := \text{GCD\_MODm}(a'_{x_s-r}, b'_{x_s-r}, n, s-1)$;
$c := \text{lc}_{x_n}(g'_{(r)})$;
$g_{(r)} := (d_{x_s-r} \cdot g'_{(r)})/c$ (but if the division fails goto (2)) ;

(3) $m := 1$;
$g := g_{(r)}$;

(4) **while** $m \leq M$ **do**
  $r :=$ a new integer s.t. $\deg_{x_n}(a'_{x_s-r}) = \deg_{x_n}(a')$ or $\deg_{x_n}(b'_{x_s-r}) = \deg_{x_n}(b')$;
  $g'_{(r)} := \text{GCD\_MODm}(a'_{x_s-r}, b'_{x_s-r}, n, s-1)$;
  $c := \text{lc}_{x_n}(g'_{(r)})$;
  $g_{(r)} := (d_{x_s-r} \cdot g'_{(r)})/c$ (but if the division fails continue) ;
  if $\deg_{x_n}(g_{(r)}) < \deg_{x_n}(g)$ **then goto** (3);
  if $\deg_{x_n}(g_{(r)}) = \deg(g)$
  then incorporate $g_{(r)}$ into g by Newton interpolation ; $m := m + 1$;

(5) $g := f \cdot \text{pp}_{x_n}(g)$;
if $g \in \mathbb{Z}[x_1, \ldots, x_s][x_n]$ and $g \mid a$ and $g \mid b$ **then return** g **;**
goto (2) □

For computing the gcd of $a, b \in \mathbb{Z}[x_1, \ldots, x_n]$, the algorithm is initially called as $\text{GCD\_MODm}(a, b, n, n-1)$.

2.3. Squarefree factorization

Definition 2.3.1. A polynomial $a(x_1, \ldots, x_n)$ in $I[x_1, \ldots, x_n]$ (I a ufd) is *squarefree* iff every nontrivial factor $b(x_1, \ldots, x_n)$ of a (i.e. b not similar to a and not a constant) occurs with multiplicity exactly 1 in a. □

Theorem 2.3.2. *Let $a(x)$ be a nonzero polynomial in $K[x]$, where* $\text{char}(K) = 0$ *or $K = \mathbb{Z}_p$ for a prime p. Then $a(x)$ is squarefree if and only if* $\gcd(a(x), a'(x)) = 1$. *($a'(x)$ is the derivative of $a(x)$.)*

Proof: If $a(x)$ is not squarefree, i.e. for some non–constant $b(x)$ we have $a(x) = b(x)^2 \cdot c(x)$, then

$$a'(x) = 2b(x)b'(x)c(x) + b^2(x)c'(x).$$

So $a(x)$ and $a'(x)$ have a non–trivial gcd.
On the other hand, if $a(x)$ is squarefree, i.e.

$$a(x) = \prod_{i=1}^{n} a_i(x),$$

where the $a_i(x)$ are pairwise relatively prime irreducible polynomials, then

$$a'(x) = \sum_{i=1}^{n} \left(a_i'(x) \prod_{\substack{j=1 \\ j \neq i}}^{n} a_j(x) \right).$$

Now it is easy to see that none of the irreducible factors $a_i(x)$ is a divisor of $a'(x)$. $a_i(x)$ divides all the summands of $a'(x)$ except the i-th. This finishes the proof for characteristic 0. In $\mathbb{Z}_p[x]$, $a_i'(x)$ cannot vanish, for otherwise we could write $a_i(x) = b(x^p) = b(x)^p$ for some $b(x)$, and this would violate our assumption of squarefreeness. Thus, $\gcd(a(x), a'(x)) = 1$. □

The problem of squarefree factorization for $a(x) \in K[x]$ consists of determining the squarefree pairwise relatively prime polynomials $b_1(x), \ldots, b_s(x)$, such that

$$a(x) = \prod_{i=1}^{s} b_i(x)^i. \qquad (2.3.1)$$

The representation of a as in (2.3.1) is called the *squarefree factorization* of a.

In characteristic 0 (e.g. when $a(x) \in \mathbb{Z}[x]$), we can proceed as follows. We set $a_1(x) := a(x)$. We set

$$a_2(x) := \gcd(a_1, a_1') = \prod_{i=2}^{s} b_i(x)^{i-1}, \quad c_1(x) := a_1(x)/a_2(x) = \prod_{i=1}^{s} b_i(x).$$

$c_1(x)$ contains every squarefree factor exactly once. Now we set

$$a_3(x) := \gcd(a_2, a_2') = \prod_{i=3}^{s} b_i(x)^{i-2}, \quad c_2(x) := a_2(x)/a_3(x) = \prod_{i=2}^{s} b_i(x).$$

$c_2(x)$ contains every squarefree factor of muliplicity ≥ 2 exactly once. So

$$b_1(x) = c_1(x)/c_2(x).$$

$$a_4(x) := \gcd(a_3, a_3') = \prod_{i=4}^{s} b_i(x)^{i-3}, \quad c_3(x) := a_3(x)/a_4(x) = \prod_{i=3}^{s} b_i(x).$$

$$b_2(x) = c_2(x)/c_3(x).$$

Iteration this process until $c_{s+1}(x) = 1$, we ultimately get the desired squarefree factorization of $a(x)$.

SQFR_FACTOR
for a given non-zero primitive polynomial a in $\mathbb{Z}[x]$
the list of squarefree factors $[b_1(x), \ldots, b_s(x)]$ of a is computed.

(1) $a_1 := a$;
 $a_2 := \gcd(a_1, a_1')$;
 $c_1 := a_1/a_2$;
 $a_3 := \gcd(a_2, a_2')$;
 $c_2 := a_2/a_3$;
 $b_1 := c_1/c_2$;
 $i := 2$;
(2) **while** $c_i \neq 1$ **do**
  $a_{i+2} := \gcd(a_{i+1}, a_{i+1}')$;
  $c_{i+1} := a_{i+1}/a_{i+2}$;
  $b_i := c_i/c_{i+1}$;
  $i := i + 1$;
(3) **return** $[b_1, \ldots, b_{i-1}]$ □

If the polynomial $a(x)$ is in $\mathbb{Z}_p[x]$, the situation is slightly more complicated. First we determine

$$d(x) = \gcd(a(x), a'(x)).$$

If $d(x) = 1$, then $a(x)$ is squarefree and we can set $a_1(x) = a(x)$ and stop.
If $d(x) \neq 1$ and $d(x) \neq a(x)$, then $d(x)$ is a proper factor of $a(x)$ and we can carry out the process of squarefree factorization both for $d(x)$ and $a(x)/d(x)$.
Finally, if $d(x) = a(x)$, then we must have $a'(x) = 0$, i.e. $a(x)$ must contain only terms whose exponents are a multiple of p. So we can write $a(x) = b(x^p) = b(x)^p$ for some $b(x)$, and the problem is reduced to the squarefree factorization of $b(x)$.

All this development can be carried over to the multivariate case rather easily. Proposition 12 in Chapter 4.2 of [Cox,Little,O'Shea 1997] [1] leads to the following theorem.

Theorem 2.3.3. *Let* $a(x_1, \ldots, x_n) \in K[x_1, \ldots, x_n]$ *and* $\mathrm{char}(K) = 0$. *Then* a *is squarefree if and only if* $\gcd(a, \partial a/\partial x_1, \ldots, \partial a/\partial x_n) = 1$. □

[1] D.Cox, J.Little, D.O'Shea, *Ideals, Varieties, and Algorithms*, 2nd edition, Springer-Verlag (1997)