Weighted Greatest Common Divisors

Bachelor Thesis Mathematics

University of Groningen

R. Modderman

June 2019

Abstract

In this thesis, the notion of *weighted greatest common divisor* (wgcd) as defined for the integers in [6, Section 2] will be discussed. We will redefine the notion of wgcd for general integral domains, and we will use the concept of squarefree factorizations to prove existence and uniqueness (up to unit multiplication) for unique factorization domains. We will use Yun's squarefree factorization algorithm as given in [9] to write algorithms for computing wgcd's of polynomials over some fields, of which some will be implemented in the computer algebra system `PARI/GP` [7]. We will also extend the idea of factoring the usual gcd to compute wgcd's as described in [2, Lemma 3] to polynomials over any field and compare the corresponding algorithm to those based on Yun's algorithm.

First supervisor: dr. J.S. Müller
Second supervisor: prof. dr. J. Top

Contents

1 Introduction

A fundamental problem in modular arithmetic is to find the *greatest common divisor* (gcd) of $n \geq 2$ integers $x_1, \ldots, x_n \in \mathbf{Z}$ (not all zero). Formally, the gcd of the tuple $x = (x_1, \ldots, x_n)$ is defined by

$$\gcd(x) := \max\{d \in \mathbf{Z} : d \,|\, x_i \text{ for all } i = 1, \ldots, n\}. \tag{1.1}$$

In [6, Section 2], the notion of gcd is generalized to *weighted greatest common divisor* (wgcd). Alongside the integers $x_1, \ldots, x_n$, positive integers $w_1, \ldots, w_n \in \mathbf{Z}_{>0}$ (which are called *weights*) are chosen. Then, the wgcd of the tuple $x = (x_1, \ldots, x_n)$ with respect to the tuple of weights $w = (w_1, \ldots, w_n)$, denoted by $\gcd_w(x)$, is defined by

$$\underset{w}{\gcd}(x) := \max\left\{d \in \mathbf{Z} : d^{w_i} \,|\, x_i \text{ for all } i = 1, \ldots, n\right\}. \tag{1.2}$$

If all weights are 1, then $\gcd_w(x)$ is just the usual gcd.

For the tuple $(384, 900) = \left(2^7 \cdot 3, 2^2 \cdot 3^2 \cdot 5^2\right)$, for example, we have

$$\begin{aligned}
\gcd(384, 900) = \underset{(1,1)}{\gcd}(384, 900) &= 12,\\
\underset{(1,2)}{\gcd}(384, 900) &= 6,\\
\underset{(2,2)}{\gcd}(384, 900) &= 2,\\
\underset{(2,3)}{\gcd}(384, 900) &= 1.
\end{aligned}$$

1.1 The Euclidean algorithm

There exists an efficient algorithm to compute the usual gcd (Euclid's algorithm, as written down by Euclid in his *Elements* around 300 B.C.) which is based on the rule that for two integers $x_1, x_2 \in \mathbf{Z}$ with $x_1 \neq 0$ we have

$$\gcd(x_1, x_2) = \begin{cases} \gcd(x_2, x_1 \bmod x_2), & x_2 \neq 0,\\ x_1, & x_2 = 0. \end{cases} \tag{1.3}$$

For *weighted* gcd's, however, no efficient algorithm is known. We ask ourselves if we can design an efficient algorithm. In [2], however, it is argued that there are strong indications that there does not exist an efficient algorithm for computing wgcd's. In [2, Section 2], it is shown that $\gcd_w(x)$ always divides the usual gcd which is used to construct an algorithm in which wgcd's are computed by means of factoring the gcd, presented in [2, Lemma 3]. This algorithm uses the concept of prime factoring for which no efficient algorithm is known.

1.2 Formulating the goals

In this thesis, we will generalize the notion of wgcd to a notion of wgcd for unique factorization domains (UFDs), for which it is shown that one does not need full factorizations to compute wgcd's but that one only needs *squarefree* factorizations. For $\mathbf{Z}$, no efficient squarefree factorization algorithms have been found, but for $K[X]$ (the polynomial ring over K) with K any field, efficient squarefree factorization algorithms are known. We will use these to design algorithms for computing wgcd's in $K[X]$.

1.3 Conventions

Throughout this paper, we reserve the symbol n for the number of numbers (or objects) one takes gcd's or wgcd's from. We *always* take $n \geq 2$. Furthermore, empty products are defined to be $1 \in R$, where R is the ring where the computations take place.

If S is a set, then $|S|$ denotes the cardinality of the set S (if S is finite, then $|S|$ is the number of elements of S).

Throughout this paper, we use the symbol $\sim$ to indicate a very specific equivalence relation: being associated in a ring. More about being associated (specifically, the definition of associatives) can be found in definition 2.1.

The symbols $\mathbf{C}, \mathbf{R}, \mathbf{Q}, \mathbf{Z}$ are reserved to indicate the sets of complex, real, rational numbers, and integers, respectively. The finite field of p elements with p prime is denoted by $\mathbf{F}_p$.

For pseudo-code, we use the bullet notation to make clear whether we speak of scalars, vectors, or higher-dimensional tensors. For example, x would be a scalar, $v_\bullet$ a vector, and $T_{\bullet\bullet}$ a matrix. Another comment regarding pseudo-code: whenever a `return` is executed, the function in question terminates after giving the output, independent of whatever instructions are given after the `return` statement. If a main function uses auxiliary functions, then these auxiliary functions are always given *before* the main function is given.

2 Preliminaries

The preliminaries for this thesis can be divided into two components: ring theory (the basics on the one hand, a little on squarefree factorizations on the other hand), and complexity theory of algorithms. Let us revise both concisely.

2.1 Some concepts in ring theory

This part gives a brief overview of the most important concepts in ring theory relevant to this paper. For more background information on rings, integral domains, polynomial rings, and unique factorization domains, one can consult [5].

Definition 2.1. *Let R be an integral domain. Then, two elements $a, b \in R$ are said to be* associated *to one another* (*or simply to be each other's* associates) *if there exists $u \in R^\times$ such that $a = ub$. The existence of such unit u is denoted by $a \sim b$.*

Exercise for the reader: prove that, in any integral domain R, $\sim$ is an equivalence relation.

2.1.1 Polynomial rings

Definition 2.2. *Let R be a ring, and $h \in R[X]$. We define the* derivative *of h, denoted by h', as follows. If h is constant (i.e., if $\deg(h) \leq 0$), then $h' = 0$. Otherwise, we can write $h = \prod_{j=0}^{m} a_j X^j$ with $a_0, \ldots, a_m \in R$, $m > 0$, and $a_m \neq 0$ and define*

$$h' = \sum_{j=1}^{m} j a_j X^{j-1}$$

Lemma 2.3. *For rings, we have the sum and product rules. More specifically, if R is a ring and $g, h \in R[X]$, then $(g + h)' = g' + h'$ and $(gh)' + g'h + gh'$.*

Proof. The first fact will be left as an exercise to the reader. Now, note that, by the sum rule and the usual distributive laws for the ring $R[X]$, it is sufficient to prove that the product rule holds for the case $g = aX^m$ and $h = bX^k$, for some $a, b \in R$ and $m, k \in \mathbf{Z}_{\geq 0}$ with m and k both nonzero. We have

$$\begin{aligned}
(gh)' &= (abX^{m+k})' \\
&= (m+k)abX^{m+k-1} \\
&= mabX^{m+k-1} + kabX^{m+k-1} \\
&= maX^{m-1}bX^k + aX^m kbX^{k-1} \\
&= g'h + gh'.
\end{aligned}$$

□

Corollary 2.4. *Let R be a ring, $h \in R[X]$, and $l \in \mathbf{Z}_{>0}$. Then, the power rule*

$$(h^l)' = lh^{l-1} \cdot h'$$

holds.

Proof. The result will follow (using induction) by the product rule as in lemma 2.3. □

Proposition 2.5. *If R is an integral domain, then so is $R[X]$.*

Proof. This is [5, Proposition 17.2]. □

2.1.2 Fields

Definition 2.6. *An integral domain R is called a* field *if $R^\times = R\backslash\{0\}$. When speaking of a general field, from now on we use the symbol K instead of R.*

As such, $\mathbf{Z}$ is *not* a field, as $\mathbf{Z}^\times = \{\pm 1\}$. The rationals, however, *are* a field: whenever $a/b \in \mathbf{Q}$ ($a, b \in \mathbf{Z}$, $a, b \neq 0$), we have

$$(a/b) \cdot \underbrace{(b/a)}_{\in \mathbf{Q}} = 1$$

proving that $\mathbf{Q}^\times = \mathbf{Q}\backslash\{0\}$.

Lemma 2.7. *If K is a field, then either* $\mathrm{char}(K) = 0$ *or* $\mathrm{char}(K)$ *is a prime number.*

Proof. The situation that $\mathrm{char}(K) = 0$ occurs: take $K = \mathbf{Q}$, for example. Now, suppose that $\mathrm{char}(K) > 0$ is not a prime. Then, $\mathrm{char}(K) = m_1 m_2$ for some positive integers $m_1, m_2 > 1$. It follows that $(m_1 \cdot 1)(m_2 \cdot 1) = 0$. Since $m_1, m_2 < m_1 m_2$ we have $m_1 \cdot 1 \neq 0$ and hence $m_1 \cdot 1$ cannot be a unit, contradicting the fact that K is a field. Conclude that $\mathrm{char}(K)$ must be prime if $\mathrm{char}(K) > 0$. □

Finally, let us introduce the notion of *perfect field*.

Definition 2.8. *A field K is called* perfect *if every irreducible polynomial over K has distinct roots.*

Theorem 2.9. *A field K is perfect if and only if either*

- $\mathrm{char}(K) = 0$, *or*
- $\mathrm{char}(K) = p$ *and every element of K has a p-th root.*

Proof. This is [4, Theorem 2]. □

Corollary 2.10. *Finite fields are perfect.*

Proof. This is [4, Corollary 3]. □

Example 2.11. *An example of an imperfect field of characteristic p is the field $\mathbf{F}_p(t)$ (i.e., the field of fractions of the polynomial ring $\mathbf{F}_p[t]$ in the variable t; or, equivalently, the field of rational functions in the variable t with coefficients in $\mathbf{F}_p$). For example, $t \in \mathbf{F}_p(t)$ but no rational function $f \in \mathbf{F}_p(t)$ satisfies $f(t)^p = t$.*

Lemma 2.12. *Let K be a perfect field of characteristic p, and let $f \in K[X]$ be a non-constant polynomial. If $f' = 0$, then there exists a unique $h \in K[X]$ such that $h(X)^p = f(X)$.*

Proof. Let us write a constructive proof. First, as $f' = 0$ and $p = 0 \in K$ in characteristic p, there is no other possibility for f to be of the form

$$f = \sum_{j=0}^{m} a_j X^{pj} \tag{2.1}$$

with $m \geq 1$ and $a_m \neq 0$. Since K is perfect, there exist elements c_j such that $c_j^p = a_j$ for all $j = 0, \ldots, m$. Then, we can define the polynomial $h \in K[X]$ by

$$h = \sum_{j=0}^{m} c_j X^j.$$

Consider, for an arbitrary commutative ring R of characteristic p with p a prime number, the Frobenius homomorphism $F : R \to R$ defined by $F(x) = x^p$ for all $x \in R$. As this map is a ring homomorphism and $K[X]$ is a commutative ring of characteristic p, we have

$$(g_1 + \ldots + g_r)^p = g_1^p + \ldots + g_r^p$$

for *any* tuple of elements $(g_1, \ldots, g_r)$ from $K[X]$, as can be shown by induction on r. It follows that

$$\begin{aligned} h(X)^p &= (c_0 + c_1 X + \ldots + c_m X^m)^p \\ &= c_0^p + c_1^p X^p + \ldots + c_m^p X^{mp} \\ &= a_0 + a_1 X^p + \ldots + a_m X^{mp} \\ &= f(X). \end{aligned}$$

The fact that this $h \in K[X]$ is unique follows from the fact that the Frobenius homomorphism on any integral domain is injective (so if another $g \in K[X]$ would satisfy $g(X)^p = f(X)$, then $g(X)^p = h(X)^p$ and we could conclude that $g(X) = h(X)$ because $K[X]$ is an integral domain). □

Corollary 2.13. *If $K = \mathbf{F}_p$, then h is given by*

$$h = \sum_{j=0}^{m} a_j X^j.$$

Proof. This follows from Fermat's little theorem which states that $x^p = x$ for all $x \in \mathbf{F}_p$, so in particular $a_j^p = a_j$ for all $j = 0, \ldots, m$. □

2.1.3 Squarefree factorizations

Before we can define the notion of squarefree factorizations, we first have to make clear what is being understood under an element of a UFD to be squarefree.

Definition 2.14. *Let R be a UFD. Then a nonzero $a \in R$ is said to be squarefree if, whenever $b \in R$ satisfies $b^2 \mid a$, we have $b \in R^\times$.*

Definition 2.15. *Let R be a UFD, and let $a \in R$ be a nonzero non-unit. Then,*

$$a = u \prod_{j=1}^{s} b_j^j$$

is a squarefree factorization of a if $u \in R^\times$, and $b_j \in R$ for all j are squarefree and mutually coprime.

Let us now prove that squarefree factorizations in UFDs exist and are unique.

Theorem 2.16. *Let R be a UFD, and let $a \in R$ be a nonzero non-unit. Then, there exists a squarefree factorization*

$$u \prod_{j=1}^{s} b_j^j = a \tag{2.2}$$

of a. Furthermore, this factorization is unique in the following sense: whenever

$$a = v \prod_{j=1}^{s'} c_j^j \tag{2.3}$$

for some $v \in R^\times$ and some squarefree and mutually coprime elements $c_1, \ldots, c_{s'}$, then we have $s = s'$ and $b_j \sim c_j$ for all $j = 1, \ldots, s$.

Proof. Since R is a UFD, we have access to a prime factorization

$$a = u \prod_{j=1}^{t} p_j^{\alpha_j} \tag{2.4}$$

where $u \in R^\times$ and all $p_j \in R$ are distinct irreducible elements from R and $\alpha_j \geq 1$ for all j. Now, we take $b_j = \prod\{p_k : \alpha_k = j\}$ for all $j = 1, \ldots, s$ where $s = \max(\alpha_1, \ldots, \alpha_t)$. Clearly, (2.2) holds. Furthermore, the sets $\{p_k : \alpha_k = j\}$ are pairwise disjoint for $j = 1, \ldots, s$ meaning that $b_1, \ldots, b_s$ are mutually coprime (if two such sets corresponding to two elements of $\{b_1, \ldots, b_s\}$ are both empty, then both the elements are units hence also coprime).

Consider (2.2), (2.3) and (2.4). Note that, to prove uniqueness, it is enough to show that $s = \max\{\alpha_1, \ldots, \alpha_t\}$ and that

$$b_j \sim \prod\{p_k : \alpha_k = j\}$$

are associates, for all $j = 1, \ldots, s$. Clearly, $s \leq \max\{\alpha_1, \ldots, a_t\}$ (otherwise, not all b_j can be composed from the irreducible elements $p_1, \ldots, p_t$). On the other hand, if $s < \max(\alpha_1, \ldots, \alpha_t)$ then not all b_j can be squarefree so $s = \max(\alpha_1, \ldots, \alpha_t)$. Since $b_1, \ldots, b_s$ are mutually coprime and squarefree, it follows that there exist pairwise disjoint sets $\mathcal{I}_1, \ldots, \mathcal{I}_s \subset \{1, \ldots, t\}$ (possibly not all nonempty) such that

$$b_j \sim \prod_{k \in \mathcal{I}_j} p_k \text{ for all } j = 1, \ldots, s$$

where empty products are 1. From (2.2) and (2.4) combined, then, it follows that we must have $\mathcal{I}_j = \{k : \alpha_k = j\}$ for all j. The result follows. □

2.2 Algorithmic complexity theory

In general, an (algebraic) algorithm expects an input, and gives an output. The time it takes for an algorithm to output a result depends on the *input size* and is called the *time complexity* (or simply *complexity*) of the algorithm.

For instance, the time complexity of the Euclidean algorithm is logarithmic in the input size. Considering (1.3), for two positive integers $x_1, x_2 \in \mathbf{Z}_{>0}$ it can be shown that, if after two steps we obtain the pair (y_1, y_2), we have

$$x_1 + x_2 \geq \frac{4}{3} \cdot (y_1 + y_2)$$

meaning that only $\log(N)/\log(4/3)$ (where $N = x_1 + x_2$ is the input size) steps are required to reach termination, meaning that the time complexity of the Euclidean algorithm is in the *complexity class* $\mathcal{O}(\log(N))$. For more background on complexity theory, we refer to [1].

3 Wgcd's in unique factorization domains

In this section, we will generalize the notion of wgcd to a notion of wgcd for general integral domains and prove some elementary results. Most importantly, we will prove that in any UFD wgcd's exist and are unique up to unit multiplication. The proof for existence will be executed three times, each with a different constructive character.

3.1 Defining wgcd's for integral domains

Definition 3.1. *Let R be an integral domain, and let $x = (x_1, \ldots, x_n)$ be a nonzero tuple of elements $x_1, \ldots, x_n \in R$. Let $w = (w_1, \ldots, w_n)$ with $w_1, \ldots, w_n \in \mathbf{Z}_{>0}$ be a tuple of weights. Then, $g \in R$ is called a* weighted greatest common divisor (wgcd) *of x with respect to w if*

- *$g^{w_i} \mid x_i$ for all $i = 1, \ldots, n$, and*
- *whenever $d \in R$ satisfies $d^{w_i} \mid x_i$ for all $i = 1, \ldots, n$ we have $d \mid g$.*

At this stage, it is desirable to prove that $\gcd_w(x)$ as defined in (1.2) is a wgcd of x with respect to w in the sense of the above definition for $R = \mathbf{Z}$. It will be sufficient to prove the following lemma.

Lemma 3.2. *Consider* (1.2)*. For all $d \in \mathbf{Z}$, it holds that*

$$d \mid \underset{w}{\gcd}(x) \Leftrightarrow d^{w_i} \mid x_i \text{ for all } i = 1, \ldots, n.$$

Proof. The left-to-right direction is obvious, as the right-hand side of the above holds for $d = \gcd_w(x)$. Now, assume that $d \in \mathbf{Z}$ satisfies $d^{w_i} \mid x_i$ for all $i = 1, \ldots, n$. Without loss of generality, we can assume that $d > 0$. Write $g = \gcd_w(x)$. We can write down prime factorizations of d and g as follows,

$$d = \prod_{j=1}^{t} p_j^{\alpha_j},\ g = \prod_{j=1}^{t} p_j^{\beta_j}$$

with $p_1, \ldots, p_t \in \mathbf{Z}$ prime numbers and $\alpha_j, \beta_j \geq 0$ for all $j = 1, \ldots, t$. Since $d^{w_i} \mid x_i$ and $g^{w_i} \mid x_i$ for all i, we know that $\left(p_j^{\alpha_j}\right)^{w_i} \mid x_i$ and $\left(p_j^{\beta_j}\right)^{w_i} \mid x_i$ for all j, i. It follows that

$$\left(\prod_{j=1}^{t} p_j^{\max(\alpha_j, \beta_j)}\right)^{w_i}$$

divides x_i for all $i = 1, \ldots, n$ and hence that $(\mathrm{lcm}(d, g))^{w_i} \mid x_i$ for all i. From (1.2), it follows that $\mathrm{lcm}(d, g) \leq g$. Since $\mathrm{lcm}(d, g)$ is a multiple of g as well (a common multiple of d and g), we have $\mathrm{lcm}(d, g) \geq g$ and hence $\mathrm{lcm}(d, g) = g$. Conclude that $d \mid g$. □

Now we would like to show that, in the sense of definition 3.1, a wgcd for fixed x and w is unique up to unit multiplication.

Proposition 3.3. *In the notation of definition 3.1, suppose that g_1 and g_2 are both wgcd's of x with respect to w. Then, $g_1 \sim g_2$.*

Proof. By definition, we have $g_1 \mid g_2$ and $g_2 \mid g_1$. Therefore, there exist $a, b \in R$ such that $g_1 = ag_2$ and $g_2 = bg_1$. It follows that $g_1 = abg_1$ and $g_2 = abg_2$. If $g_1 = g_2$ then clearly $g_1 \sim g_2$. If $g_1 \neq g_2$, then one of them must be nonzero and hence $ab = 1$. Since R is a domain, it follows that $a \in R^\times$ so (as $g_1 = ag_2$) again $g_1 \sim g_2$. □

The above results enables us to introduce the following convention.

Convention 3.4. *From now on, for every integral domain $R \neq \mathbf{Z}$, every nonzero n-tuple $x = (x_1, \ldots, x_n) \in R^n$, and every set of weights w, we let* $\gcd_w(x)$ *denote* any *wgcd of x with respect to w. As such, if $y \in R$ is a wgcd of x with respect to w, then we will say that* $\gcd_w(x) \sim y$ *and* not $\gcd_w(x) = y$ unless $\gcd_w(x)$ *and y are exactly the same element.*

Let us now prove a final result.

3.2 Existence of wgcd's in unique factorization domains

Let R be a UFD, let $x = (x_1, \ldots, x_n)$ be a nonzero tuple of elements from R, and let $w = (w_1, \ldots, w_n)$ (with $w_1, \ldots, w_n \in \mathbf{Z}_{>0}$) be a tuple of weights. We will prove that $\gcd_w(x)$ exists, in three constructive fashions:

1. by using the prime factorizations of $x_1, \ldots, x_n$, which will be a generalization of [2, Proposition 1];
2. by using the prime factorization of $\gcd(x)$, which will be a generalization of [2, Lemma 3];
3. by using the *squarefree factorizations* of $x_1, \ldots, x_n$, where the result of theorem 2.16 is used.

Before stating the three theorems, let $\mathcal{I}$ be the set of indices for which the elements are nonzero. More specifically, we set $\mathcal{I} := \{i \in \{1, \ldots, n\} : x_i \neq 0\}$. Note that, as x was taken to be a nonzero tuple, we have $\mathcal{I} \neq \varnothing$.

Definition 3.5. *Let S be a UFD, and $a, p \in S$. Assume moreover that $a \neq 0$, and that p is irreducible in S. Then, we define the* valuation *of a at p, denoted by* $\mathrm{val}(a, p)$*, by*

$$\mathrm{val}(a, p) := \max\{m \in \mathbf{Z}_{\geq 0} : p^m \mid a\}.$$

Note that, as S is a UFD, valuations always exist (as a nonnegative integer).

Theorem 3.6. *In UFDs, wgcd's of nonzero tuples always exist and are unique up to unit multiplication.*

Proof. First, we write the prime factorizations of the components:

$$x_i \sim \prod_{j=1}^{t} p_j^{\alpha_{ij}} \text{ for all } i \in \mathcal{I},$$

where $p_1, \ldots, p_t \in R$ are irreducible and $\alpha_{ij} \in \mathbf{Z}_{\geq 0}$ for all j, i.

- Construction 1. For all $j = 1, \ldots, t$, let $\alpha_j := \min_{i \in \mathcal{I}} \lfloor \alpha_{ij}/w_i \rfloor$. Then, we construct $g \in R$ by

$$g := \prod_{j=1}^{t} p_j^{\alpha_j}.$$

We note that, for all $1 \leq j \leq t$ and all $i \in \mathcal{I}$, we have

$$\begin{aligned} w_i \alpha_j &= w_i \cdot \min_{k \in \mathcal{I}} \lfloor \alpha_{kj}/w_k \rfloor \\ &\leq w_i \cdot \lfloor \alpha_{ij}/w_i \rfloor \\ &\leq w_i \cdot (\alpha_{ij}/w_i) \\ &= \alpha_{ij} \end{aligned}$$

and hence $g^{w_i} \mid x_i$ for all $i \in \mathcal{I}$. Since every element of R divides $0 \in R$, we also have $g^{w_i} \mid x_i$ for all $i = 1, \ldots, n$ and hence g satisfies the first part of definition 3.1. Now, assume that an element $d \in R$ satisfies $d^{w_i} \mid x_i$ for all $i = 1, \ldots, n$. Clearly, d must be of the form

$$d \sim \prod_{j=1}^{t} p_j^{\delta_j}$$

for some exponents $\delta_1, \ldots, \delta_t \in \mathbf{Z}_{\geq 0}$. Since $p_1, \ldots, p_t$ are irreducible in R, it follows that we must have $w_i \delta_j \leq a_{ij}$ for all $1 \leq j \leq t$ and all $i \in \mathcal{I}$. Clearly, $\delta_j \leq a_{ij}/w_i$ for all j, i. Since $\delta_1, \ldots, \delta_t$ are integers and $\lfloor r \rfloor \leq r < \lfloor r \rfloor + 1$ for any $r \in \mathbf{R}$, we see that $\delta_j \leq \lfloor a_{ij}/w_i \rfloor$ for all j, i. By how $\alpha_1, \ldots, \alpha_t$ are defined, it follows that $\delta_j \leq \alpha_j$ for all j. We see that

$$\begin{aligned} g &= \prod_{j=1}^{t} p_j^{\alpha_j} \\ &\sim \underbrace{\prod_{j=1}^{t} p_j^{\alpha_j - \delta_j}}_{\in R} \cdot \underbrace{\prod_{j=1}^{t} p_j^{\delta_j}}_{d} \end{aligned}$$

and it follows that $d \mid g$. We conclude that g satisfies the second part of definition 3.1 as well. It follows that $\gcd_w(x) \sim g$, hence $\gcd_w(x)$ exists.

- Construction 2. As it is widely known that gcd's exist in UFDs, we can write

$$\gcd(x) \sim \prod_{j=1}^{t} p_j^{\gamma_j}$$

for some $\gamma_1, \ldots, \gamma_t \in \mathbf{Z}_{\geq 0}$ (in particular, we have $\gamma_j = \min_{i \in \mathcal{I}} \alpha_{ij}$). Defining

$$\beta_j := \min_{i \in \mathcal{I}} \lfloor \mathrm{val}(x_i, p_j)/w_i \rfloor$$

for all $j = 1, \ldots, t$, then, leads to the claim that

$$\gcd_w(x) \sim \prod_{j=1}^{t} p_j^{\beta_j}.$$

By proposition 3.3, it is sufficient to show that $\beta_j = \alpha_j$ for all $j = 1, \ldots, t$, but this follows immediately from the observation that $\mathrm{val}(x_i, p_j) = \alpha_{ij}$ for all j, i.

- Construction 3. For all nonzero x_i, we can write down the squarefree factorization. More specifically, we have

$$x_i \sim \prod_{j=1}^{s_i} b_{ij}^j \text{ for all } i \in \mathcal{I},$$

with b_{ij} squarefree for all (i, j) and $b_{i1}, \ldots, b_{is_i}$ mutually coprime for all $i \in \mathcal{I}$. Then, the claim is that

$$\gcd_w(x) \sim \gcd_{i \in \mathcal{I}} \left(\prod_{j=1}^{s_i} b_{ij}^{\lfloor j/w_i \rfloor} \right).$$

By theorem 2.16, we have $b_{ij} \sim \prod \{p_k : \alpha_{ik} = j\}$ for all (i, j). It follows that, for arbitrary $i \in \mathcal{I}$, we have

$$\begin{aligned} \prod_{j=1}^{s_i} b_{ij}^{\lfloor j/w_i \rfloor} &\sim \prod_{j=1}^{s_i} \left(\prod \{p_k : \alpha_{ik} = j\} \right)^{\lfloor j/w_i \rfloor} \\ &= \prod_{j=1}^{t} p_j^{\lfloor \alpha_{ij}/w_i \rfloor}. \end{aligned}$$

It follows that

$$\gcd_{i \in \mathcal{I}} \left(\prod_{j=1}^{s_i} b_{ij}^{\lfloor j/w_i \rfloor} \right) \sim \prod_{j=1}^{t} p_j^{\varepsilon_j}$$

where $\varepsilon_j = \min_{i \in \mathcal{I}} \lfloor \alpha_{ij}/w_i \rfloor$ for all $j = 1, \ldots, t$. But we have

$$\gcd_w(x) \sim \prod_{j=1}^{t} p_j^{\varepsilon_j},$$

because $\varepsilon_j = \alpha_j$ for all $j = 1, \ldots, t$, where $\alpha_1, \ldots, \alpha_t$ are defined under construction 1. The result follows.

As a final note, by proposition 3.3, any wgcd of x with respect to w is unique up to unit multiplication. $\square$

Now, let us show that wgcd's cannot always be extracted from the *squarefree* factorization of the gcd. In other words, it is not always possible to find exponents $\alpha_j \leq j$ for all $j = 1, \ldots, t$ such that

$$\operatorname*{gcd}_w(x) \sim \prod_{j=1}^{s} b_j^{\alpha_j}$$

if the product

$$\prod_{j=1}^{s} b_j^j$$

is the squarefree factorization of $\gcd(x)$. This phenomenon is illustrated below.

Example 3.7. *For the integers, consider the tuple* $x = (2250, 360)$. *Then,* $\gcd(x) = 90$, *and the squarefree factorization of* 90 *is* $90 = 10 \cdot 3^2$. *We have* $\gcd_{(1,3)}(2250, 360) = 2$, *and clearly* 2 *is not of the form* $10^{\alpha_1} \cdot 3^{\alpha_2}$ *for some exponents* $\alpha_1, \alpha_2 \in \mathbf{Z}_{\geq 0}$.

Now, let us prove an analogy of recursion that holds in any integral domain.

Proposition 3.8. *Let* R *be an integral domain, and let* $x = (x_1, \ldots, x_n) \in R^n$ *be a nonzero tuple and* $w = (w_1, \ldots, w_n) \in (\mathbf{Z}_{>0})^n$ *be a tuple of weights. Assume that* $n \geq 3$. *If* $x_2, \ldots, x_n$ *are not all zero, then we have*

$$\operatorname*{gcd}_w(x) \sim \operatorname*{gcd}_{(w_1,1)} \left(x_1, \operatorname*{gcd}_{(w_2,\ldots,w_n)} (x_2, \ldots, x_n) \right).$$

Proof. Write $g \sim \gcd_w(x)$, $g_{\geq 2} \sim \gcd_{(w_2,\ldots,w_n)}(x_2, \ldots, x_n)$ and $h \sim \gcd_{(w_1,1)}(x_1, g_{\geq 2})$. Take $d \in R$. By definition 3.1, we then have

$$\begin{aligned} d \,|\, g &\Leftrightarrow d^{w_1} \,|\, x_1,\ d^{w_2} \,|\, x_2,\ \ldots,\ d^{w_n} \,|\, x_n \\ &\Leftrightarrow d^{w_1} \,|\, x_1,\ d \,|\, g_{\geq 2} \\ &\Leftrightarrow d \,|\, h. \end{aligned}$$

$\square$

4 Wgcd's in polynomial rings over fields

In this section, we let K be an arbitrary field. In abstract algebra, it is well known that the polynomial ring in one variable over K (denoted by $K[X]$) is a UFD (this follows from [5, Proposition 17.2], as fields are UFDs). This enables us to look at wgcd's in the UFD $K[X]$ as well.

4.1 Sharpening the definition of wgcd's for polynomial rings over fields

We have seen that, following proposition 3.3, wgcd's are not strictly unique: given x and w (with R any domain) there are exactly $|R^\times|$ distinct wgcd's in R of x with respect to w. In $\mathbf{Z}$, to avoid having distinct wgcd's, we have used the total order on $\mathbf{R}$ by stating that we look for the *maximum* of a set hence always look for a positive integer. In general, there is no method to eliminate the possibility of having multiple (associated) wgcd's. For $K[X]$, however, there is: by restricting to monic polynomials.

Definition 4.1. *Let $f = (f_1, \ldots, f_n)$ be a nonzero tuple of polynomials $f_1, \ldots, f_n$ from $K[X]$. Let $w = (w_1, \ldots, w_n)$ with $w_1, \ldots, w_n \in \mathbf{Z}_{>0}$ be a tuple of weights. Then,* $\gcd_w(f)$ *is defined to be the unique monic* wgcd *of f with respect to w, where* wgcd *is as in definition 3.1.*

At this point, we have to justify that $\gcd_w(f)$ is uniquely defined. This is not hard to see, as $(K[X])^\times$ consists precisely of all nonzero constant polynomials from $K[X]$.

4.2 Squarefreeness in polynomial rings over fields

Now, for $\mathbf{Z}$ there is no efficient algorithm for computing squarefree factorizations, as there is no efficient method to check whether an integer is squarefree. For the case of polynomials over fields, however, there is, as presented below.

Lemma 4.2. *Let $h \in K[X]$ be a non-constant polynomial. If* $\gcd(h, h') = 1$*, then h is squarefree. If* $\mathrm{char}(K) = 0$*, then the converse is also true.*

Proof. If h is not squarefree, then there exist $h_1, h_2 \in K[X]$ with $\deg(h_1) \geq 1$ and $h = h_1^2 h_2$. From lemma 2.3 and corollary 2.4, it follows that

$$\begin{aligned}(h_1^2 h_2)' &= ((h_1^2)h_2)' \\ &= (h_1^2)' h_2 + (h_1)^2 h_2' \\ &= 2h_1 h_1' h_2 + h_1^2 h_2' \\ &= h_1 \cdot (2h_1' h_2 + h_1 h_2').\end{aligned}$$

It follows that $h_1 \mid h$ and $h_1 \mid h'$, hence $h_1 \mid \gcd(h, h')$. Conclude that $\gcd(h, h') \neq 1$. Now, assume that $\mathrm{char}(K) = 0$ and that h is squarefree. Then, there exist mutually

coprime irreducible polynomials $p_1, \ldots, p_t \in K[X]$ such that

$$h = \prod_{j=1}^{t} p_j.$$

By using the product rule for polynomials, we have

$$h' = \sum_{j=1}^{t} p_j' \prod_{k \neq j} p_k. \tag{4.1}$$

Now, suppose that $\gcd(h, h') \neq 1$. Since $K[X]$ is a UFD, then, there exists an irreducible $p \in K[X]$ such that $p \,|\, h$ and $p \,|\, h'$. It follows that there exists $j_0 \in \{1, \ldots, t\}$ such that $p = p_{j_0}$. By (4.1), we have $p_{j_0} \,|\, p_{j_0}' \prod_{j \neq j_0} p_j$ and hence $p_{j_0} \,|\, p_{j_0}'$ which is possible only if $p_{j_0}' = 0$. But since $\mathrm{char}(K) = 0$, we must have that $\deg(p_{j_0}) \leq 0$ which contradicts the fact that $p_{j_0} \in K[X]$ is irreducible. The result follows. □

4.3 Designing algorithms for polynomial rings over fields

At this stage, we have the tools to design multiple algorithms for $K[X]$. One of them will be fully based on construction 2 from theorem 3.6, whereas a second one will be based on construction 3 from theorem 3.6 where we will use Yun's squarefree factorization algorithm as described in [9] to compute the necessary squarefree factorizations. We will also exploit lemma 2.12 to deal with the case of K having prime characteristic, because non-constant polynomials over fields of prime characteristic can have zero derivative (e.g., $X^2 + 1 \in \mathbf{F}_2[X]$ is not constant but $(X^2 + 1)' = 2X = 0$).

5 All algorithms

Before describing all algorithms in detail, let us present an overview of all algorithms present in this paper. Note that, for polynomials over fields while using squarefree factorizations, the case that the field has prime characteristic and is not perfect is not covered.

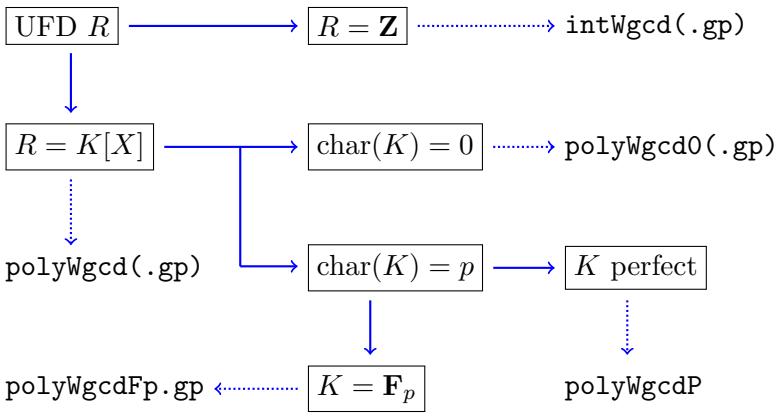

Figure 1: overview of all algorithms

If an algorithm is described in pseudo-code *and* is implemented in `PARI/GP`, then this will be denoted by `(.gp)`. If no pseudo-code is written but code in `PARI/GP` *is* written, then this will be denoted by `.gp`. No extension indicates that the algorithm is described in pseudo-code only. In this section, we will only provide pseudo-code, where all implementations written in `GP` script are presented as appendices.

5.1 For integers: `intWgcd`

Below, we use construction 2 from theorem 3.6 to construct an algorithm for $\mathbf{Z}$.

Algorithm 5.1 `intWgcd`: calculating in $\mathbf{Z}$ using construction 2 from theorem 3.6

algorithm `intWgcd` $(x_\bullet, w_\bullet)$
  input nonzero vector $x_\bullet$ of integers, weights $w_\bullet \in (\mathbf{Z}_{>0})^n$
  output $\gcd_{w_\bullet}(x_\bullet)$
  $\mathcal{I} \leftarrow \{i \in \{1, \ldots, n\} : x_i \neq 0\}$
  compute prime factorization $\prod_{j=1}^{t} p_j^{\gamma_j}$ of $\gcd(x_\bullet)$
  for $j = 1, \ldots, t$ **do**
    $\beta_j \leftarrow \min_{i \in \mathcal{I}} \lfloor \mathrm{val}(x_i, p_j)/w_i \rfloor$
  return $\left(\prod_{j=1}^{t} p_j^{\beta_j}\right)$

5.2 For polynomial rings over fields: `polyWgcd0`

Now, we use construction 3 from theorem 3.6 and Yun's squarefree factorization algorithm [9] to design an algorithm for polynomials over fields of characteristic zero. The auxiliary function `wPoly` is almost a strict copy of Yun's algorithm, where we use the fact that $e_j = \prod_{k=j}^{s} g_k$ for all $j = 1, \ldots, s$ if $\prod_{j=1}^{s} g_j^j$ is a squarefree factorization of the polynomial in question.

Algorithm 5.2 `polyWgcd0`: calculating $\gcd_w(f)$ for $\mathrm{char}(K) = 0$

---

algorithm `wPoly` (K, f, m)
  input field K with $\mathrm{char}(K) = 0$, nonzero polynomial $f \in K[X]$, integer $m \in \mathbf{Z}_{>1}$
  output the unique monic polynomial $g \in K[X]$ such that
    (1) $g^m \mid f$, and
    (2) whenever $h \in K[X]$ satisfies $h^m \mid f$, then $h \mid g$
  $d \leftarrow \gcd(f, f')$
  $e_1 \leftarrow f/d$
  $b_1 \leftarrow f'/d - e_1'$
  $i \leftarrow 1$
  while $e_i \neq 1$ **do**
    $g_i \leftarrow \gcd(e_i, b_i)$
    $e_{i+1} \leftarrow e_i/g_i$
    $b_{i+1} \leftarrow b_i/g_i - e_{i+1}'$
    $i \leftarrow i + 1$
  return $\left(\prod_{j=1}^{\lfloor (i-1)/m \rfloor} e_{j \cdot m}\right)$

algorithm `polyWgcd0` $(K, f_\bullet, w_\bullet)$
  input field K with $\mathrm{char}(K) = 0$, nonzero vector of polynomials $f_\bullet \in (K[X])^n$,
    weights $w_\bullet \in (\mathbf{Z}_{>0})^n$
  output $\gcd_{w_\bullet}(f_\bullet)$
  $\mathcal{I} \leftarrow \{i \in \{1, \ldots, n\} : f_i \neq 0\}$
  for $i \in \mathcal{I}$ **do**
    $g_i \leftarrow [\texttt{wPoly}(K, f_i, w_i)$ **if** $w_i > 1$; f_i **if** $w_i = 1]$
  return $\gcd(g_\bullet)$

5.3 For polynomial rings over fields: `polyWgcd`

Next, we present an algorithm based on construction 2 from theorem 3.6, which works for polynomials over *any* field.

Algorithm 5.3 `polyWgcd`: calculating $\gcd_w(f)$ over any field by factoring $\gcd(f)$

---

algorithm `polyWgcd` $(K, f_\bullet, w_\bullet)$
  input field K, nonzero vector of polynomials $f_\bullet \in (K[X])^n$, weights $w_\bullet \in (\mathbf{Z}_{>0})^n$
  output $\gcd_{w_\bullet}(f_\bullet)$
  $g \leftarrow \gcd(f_\bullet)$
  compute prime factorization $\prod_{j=1}^{t} p_j^{\gamma_j}$ of g
  $\mathcal{I} \leftarrow \{i \in \{1, \ldots, n\} : f_i \neq 0\}$
  for $j = 1, \ldots, t$ **do**
    $\beta_j \leftarrow \min_{i \in \mathcal{I}} \lfloor \mathrm{val}(f_i, p_j)/w_i \rfloor$
  return $\left(\prod_{j=1}^{t} p_j^{\beta_j}\right)$

5.4 For polynomial rings over fields: `polyWgcdP`

Finally, an algorithm what works for any perfect field of characteristic p. This algorithm is based on Yun's squarefree factorization algorithm (the Tobey-Horowitz variant, in particular, and follows [8, Chapter 5, Section 1.5]) and takes into account that non-constant polynomials can have vanishing derivatives in characteristic p. We use lemma 2.12 and its constructive proof. Note that, by corollary 2.13, this algorithm can be simplified significantly to the case of $K = \mathbf{F}_p$ because the p-th root of $x \in \mathbf{F}_p$ is always x itself (then, the auxiliary function `PthPolyRoot` would trivially find p-th roots of the coefficients of the polynomial in question).

Algorithm 5.4 `polyWgcdP`: calculating $\gcd_w(f)$ for perfect K with $\mathrm{char}(K) = p$

algorithm `wPoly` (K, f, m, k)
  input perfect field K with $\mathrm{char}(K) = p$, nonzero polynomial $f \in K[X]$
    with $f' \neq 0$, integer $m \in \mathbf{Z}_{>1}$, integer $k \in \mathbf{Z}_{\geq 0}$
  output the unique monic polynomial $g \in K[X]$ such that
    (1) $g^m \mid f^{p^k}$, and
    (2) whenever $h \in K[X]$ satisfies $h^m \mid f^{p^k}$, then $h \mid g$
  $d_1 \leftarrow \gcd(f, f')$
  $e_1 \leftarrow f/d_1$
  $i \leftarrow 1$
  while $d_i \neq 1$ **do**
    $d_{i+1} \leftarrow \gcd(d_i, d_i')$
    $e_{i+1} \leftarrow d_i/d_{i+1}$
    $g_i \leftarrow e_i/e_{i+1}$
    $i \leftarrow i + 1$
  return $\left(\prod_{j=1}^{i-1} g_j^{\lfloor j \cdot p^k/m \rfloor}\right)$

algorithm `PthPolyRoot` (K, f)
  input perfect field K of characteristic p, polynomial $f \in K[X]$ with $f' = 0$
  output polynomial $h \in K[X]$ such that $h(X)^p = f(X)$
  find coefficients $a_0, \ldots, a_m$ of f being of the form (2.1)
  find $c_0, \ldots, c_m \in K$ such that $c_j^p = a_j$ for all $j = 0, \ldots, m$
  return $\left(\sum_{j=0}^{m} c_j X^j\right)$

algorithm `polyWgcdP` $(K, f_\bullet, w_\bullet)$
  input perfect field K with $\mathrm{char}(K) = p$, nonzero vector of polynomials
    $f_\bullet \in (K[X])^n$, weights $w_\bullet \in (\mathbf{Z}_{>0})^n$
  output $\gcd_{w_\bullet}(f_\bullet)$
  $\mathcal{I} \leftarrow \{i \in \{1, \ldots, n\} : f_i \neq 0\}$
  for $i \in \mathcal{I}$ **do**
    $k \leftarrow 0$
    $h \leftarrow f_i$
    while $h' = 0$ **do**
      $h \leftarrow$ `PthPolyRoot`(K, h)
      $k \leftarrow k + 1$
    $g_i \leftarrow$ [`wPoly`(K, h, w_i, k) **if** $w_i > 1$; f_i **if** $w_i = 1$]
  return $\gcd(g_\bullet)$

6 Analyzing the GP scripts in PARI

Let us shortly analyze the four GP scripts we have written.

6.1 Example input

Let us briefly describe some example inputs to the four GP scripts as described in the appendices to show how the scripts should be called properly.

For `intWgcd.gp`, the input

```
intWgcd([1440,700], [3,2])
```

(this is [2, Example 1]) leads to the output `2`.

For `polyWgcd0.gp` and `polyWgcd.gp`, the respective inputs

```
polyWgcd0([X^4-2*X^2+1, X-1], [2,1])
 polyWgcd([X^4-2*X^2+1, X-1], [2,1])
```

lead to the output `X-1`.

For `polyWgcdFp.gp`, the input

```
polyWgcdFp(2, [Mod(1,2)*X^4+Mod(1,2),Mod(1,2)*X^2+Mod(1,2)], [2, 1])
```

leads to the output `Mod(1,2)*X^2+Mod(1,2)`.

6.2 Speed analysis

For some test sets, let us compare `polyWgcd.gp` to `polyWgcd0.gp` for the case $K = \mathbf{Q}$ and `polyWgcd.gp` to `polyWgcdFp.gp` for the case $K = \mathbf{F}_p$.

6.2.1 Test 1: `polyWgcd.gp` vs. `polyWgcd0.gp`, comparing running times

For the first test, we compare the speeds of the algorithms `polyWgcd.gp` (T_1) and `polyWgcd0.gp` (T_2) by giving the algorithms the exact same input, where polynomials are constructed by means of generating some irreducible polynomials and then multiplying those.

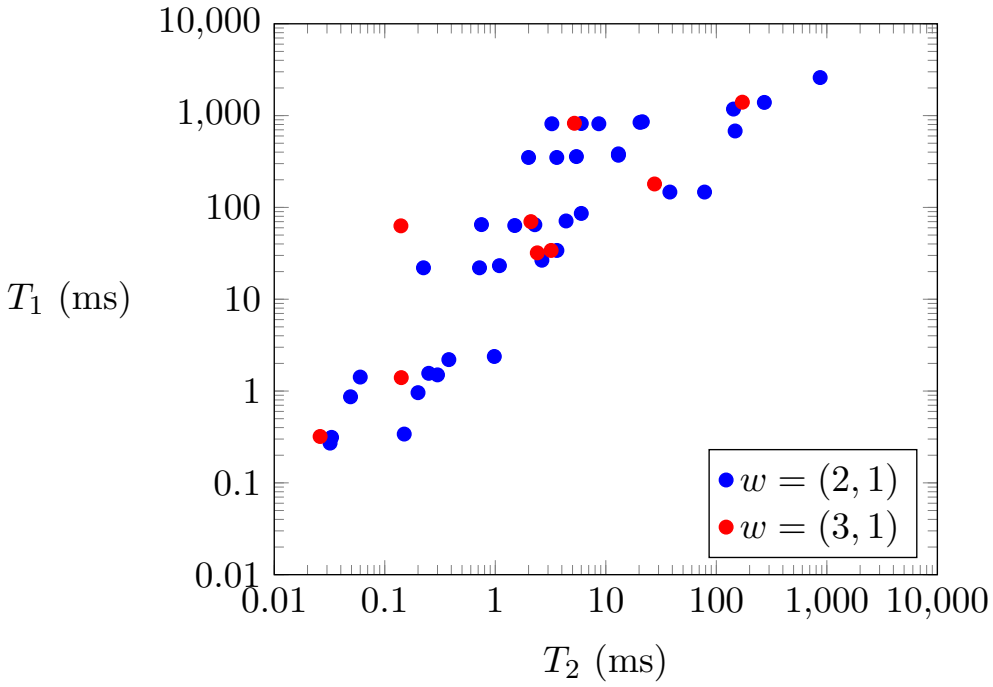

Figure 2: $K = \mathbf{Q}$, running time comparison

For the following tests, we use only the tuple of weights $w = (2, 1)$.

6.2.2 Test 2: `polyWgcd.gp` and `polyWgcd0.gp`, by relative 'compositeness'

For the second test, we compare the speeds of the algorithms `polyWgcd.gp` and `polyWgcd0.gp` by comparing their times (T) to the parameter λ, which is defined to be a measure for a pair of polynomials to share factors. For a tuple f, the parameter λ is defined by $\lambda := \deg(\gcd(f))/\max_i \deg(f_i)$. We use *another* test set (but generated by the same method as in test 1).

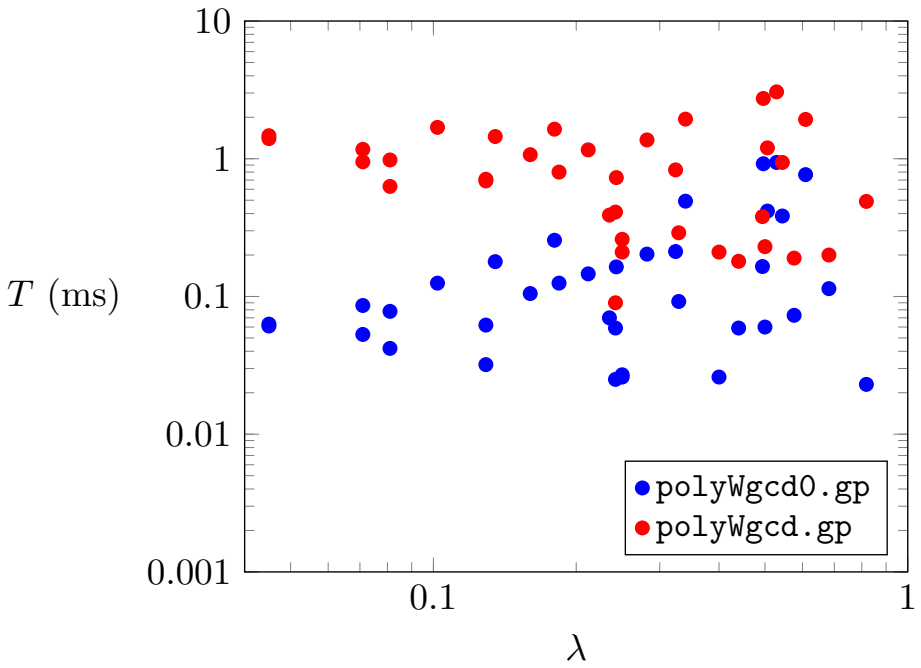

Figure 3: $K = \mathbf{Q}$, by 'relative compositeness'

6.2.3 Test 3: `polyWgcd.gp` vs. `polyWgcdFp.gp`, comparing running times

For the third test, we compare the speeds of the algorithms `polyWgcd.gp` and `polyWgcdFp.gp` for the case $K = \mathbf{F}_p$, for some values of p.

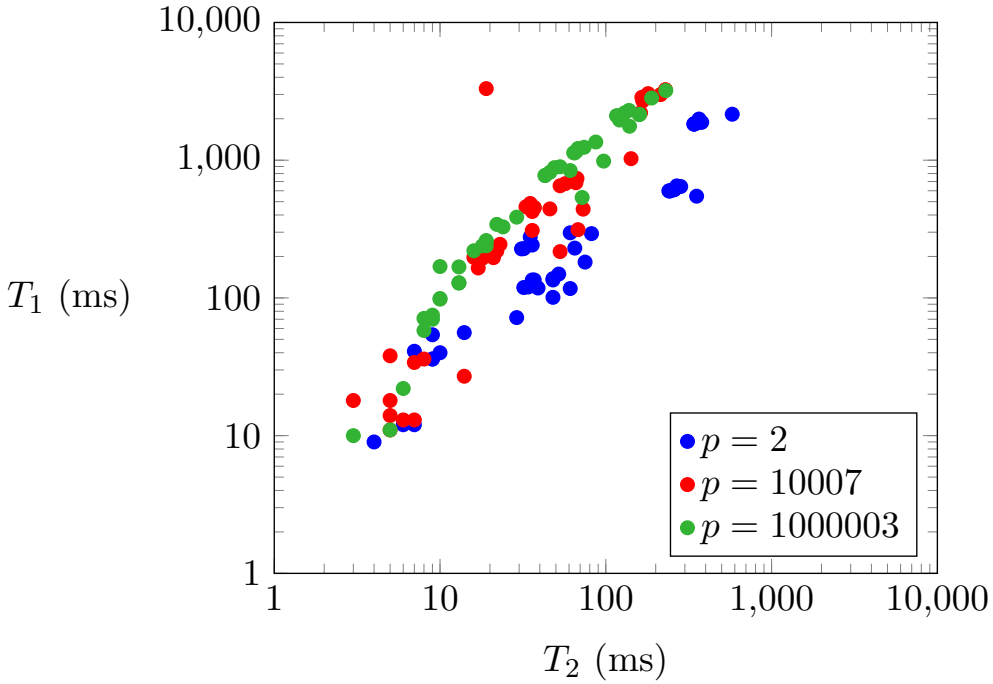

Figure 4: $K = \mathbf{F}_p$, running time comparison

7 Discussion

7.1 An application: weighted projective spaces

Given a field K, the *projective space* of dimension n over K is defined to be the set of equivalence classes

$$\mathbf{P}^n(K) := (K^{n+1} \backslash \{\mathbf{0}\}) / \sim$$

where $\mathbf{0} \in K^{n+1}$ is the zero vector and two tuples

$$(x_0, \ldots, x_n),\ (y_0, \ldots, y_n) \in K^{n+1}$$

are said to be *equivalent* (denoted by $(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$) if there exists $\lambda \in K$ such that $(x_0, \ldots, x_n) = (\lambda y_0, \ldots, \lambda y_n)$. It is not hard to show that $\sim$ is indeed an equivalence relation.

For the case $K = \mathbf{Q}$, it is commonly asked whether we can easily find integer representatives of such equivalence classes. In $\mathbf{P}^1(\mathbf{Q})$, we have $(1/4, 1/6) \sim (3, 2)$. This integer representative could have been computed via

$$(3, 2) = \frac{4 \cdot 6}{\gcd(4, 6)} \cdot (1/4, 1/6)$$

and now the gcd appears. In $\mathbf{P}^1(\mathbf{Q})$ in general, one has

$$(s_1/t_1, s_2/t_2) \sim \frac{t_1 \cdot t_2}{\gcd(t_1, t_2)}(s_1, s_2).$$

The notion of projective space can be generalized to that of a *weighted* projective space, where weights $w_0, \ldots, w_n \in \mathbf{Z}_{>0}$ are chosen. The corresponding equivalence relation is given by

$$(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n) \Leftrightarrow \exists \lambda \in K : (x_0, \ldots, x_n) = (\lambda^{w_0} y_0, \ldots, \lambda^{w_n} y_n)$$

As such, computing representatives of equivalence classes of weighted projective spaces over $\mathbf{Q}$ involves computing wgcd's of integers. In general, computing representatives of equivalence classes of weighted projective spaces over $Q(R)$ (the field of fractions of any integral domain R) involves computing wgcd's of elements of R, and if R is a UFD then this is theoretically possible by definition 3.1 and theorem 3.6.

7.2 Suggestions for further research

A natural suggestion would be to improve algorithms `polyWgcd0` and `polyWgcdFp`, since the figures in section 6 show that `polyWgcdFp` is faster by a constant factor rather than asymptotically faster. Another suggestion involving running time would be to investigate the time complexity of the theoretical algorithms as described in section 5. For this, [3] would be a very providing read.

From a theoretical point of view, one might be interested in studying wgcd's in other UFDs than $\mathbf{Z}$ and $K[X]$.

7.3 Conclusion

As mentioned in [2], computing wgcd's of integers seems to be almost as hard as computing prime factorizations of integers. By using the concept of squarefree factorizations, we have shown that wgcd's can be computed by computing the squarefree factorizations of the components (following construction 3 from theorem 3.6), and we used an efficient squarefree factorization algorithm for polynomials over fields [9] to design algorithms to compute wgcd's of polynomials over fields, after generalizing the concept of wgcd's to that as in definition 3.1.

Appendices

A Implementation of algorithm 5.1 in PARI/GP

Below, the implementation in PARI/GP of algorithm 5.1 is given.

```
intWgcd (xVec, wVec) = {
  n = length(xVec);
  g = gcd(xVec);
  gMat = factor(g);
  r = matsize(gMat)[1];
  betaVec = [];
  indVec = [];
  for (i = 1, n, if(xVec[i] != 0, indVec = concat(indVec, i)));
  indLength = length(indVec);
  for (j = 1, r,
    vVec = [];
    for(i = 1, indLength,
      k = indVec[i];
      vVec =  concat(vVec, valuation(xVec[k], gMat[j, 1]) / wVec[k]);
    );
    betaVec = concat(betaVec, floor(vecmin(vVec)));
  );
  prod(j = 1, r, gMat[j, 1] ^ betaVec[j]);
}
```

B Implementation of algorithm 5.2 in PARI/GP

Below, the implementation in PARI/GP of algorithm 5.2 is given.

```
wPoly (f, m) = {
  if (m > 1,
    d = gcd(f, f');
    e = [f/d];
    b = [f'/d - e[1]'];
    g = [];
    i = 1;
    while (e[i] != 1,
      g = concat(g, gcd(e[i], b[i]));
      e = concat(e, e[i] / g[i]);
      b = concat(b, b[i] / g[i] - e[i+1]');
      i++;
    );
  );
  if (m == 1, f, prod(j = 1, floor(i / m), e[j * m]));
}

polyWgcd0 (fVec, wVec) = {
  gVec = [];
  for (i = 1, length(fVec),
    gVec = concat(gVec, if(fVec[i] != 0, wPoly(fVec[i], wVec[i]), 0))
  );
  gcd(gVec);
}
```

C Implementation of algorithm 5.3 in PARI/GP

Below, the implementation in PARI/GP of algorithm 5.3 is given.

```
polyWgcd (fVec, wVec) = {
  n = length(fVec);
  g = gcd(fVec);
  gMat = factor(g);
  r = matsize(gMat)[1];
  betaVec = [];
  indVec = [];
  for (i = 1, n, if(fVec[i] != 0, indVec = concat(indVec, i)));
  indLength = length(indVec);
  for (j = 1, r,
    vVec = [];
    for(i = 1, indLength,
      k = indVec[i];
      vVec =  concat(vVec, valuation(fVec[k], gMat[j, 1]) / wVec[k]);
    );
    betaVec = concat(betaVec, floor(vecmin(vVec)));
  );
  prod(j = 1, r, gMat[j, 1] ^ betaVec[j]);
}
```

D The algorithm polyWgcdFp in PARI/GP

Below, the implementation in PARI/GP of polyWgcdFp is given.

```
polyWgcdFp (p, fVec, wVec) = {
  hVec = [];
  for (i = 1, length(fVec),
    hVec = concat(hVec, fVec[i]);
    if (fVec[i] != 0,
      factMat = factormodSQF(fVec[i], p);
      hVec =  concat(hVec, prod(j = 1, matsize(factMat)[1], \
          factMat[j, 1] ^ floor(factMat[j, 2] / wVec[i])));
    );
  );
  gcd(hVec);
}
```

References

[1] S. Arora and B. Barak. *Computational Complexity: a Modern Approach*. Cambridge University Press, 2019.

[2] L. Beshaj, J. Gutierrez, and T. Shaska. Weighted greatest common divisors and weighted heights. *arXiv:1902.06563v2*, 2019. arXiv: arXiv:1902.06563v2, `https://arxiv.org/pdf/1902.06563.pdf`.

[3] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[4] K. Conrad. Perfect fields. Available from `https://kconrad.math.uconn.edu/blurbs/galoistheory/perfect.pdf`.

[5] T.W. Judson and R.A. Beezer. *Abstract Algebra*. Orthogonal Publishing L3C, 2008.

[6] J. Mandili and T. Shaska. Computing heights on weighted projective spaces. *Communications in Contemporary Mathematics*, 734:149–160, 2019.

[7] The PARI Group, Univ. Bordeaux. *PARI/GP version* `2.11.0`, 2018. Available from `http://pari.math.u-bordeaux.fr/`.

[8] F.J. Wright. *Mathematics and Algorithms for Computer Algebra, Part 1, Chapter 5*. CBPF, Rio de Janeiro, 1992. Available from `https://people.eecs.berkeley.edu/~fateman/282/F%20Wright%20notes/week6.pdf`.

[9] David Y.Y. Yun. On square-free decomposition algorithms. In *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation*, SYMSAC '76, pages 26–35, New York, NY, USA, 1976. ACM.