Resolving zero-divisors using Hensel lifting

John Kluesner and Michael Monagan

Department of Mathematics, Simon Fraser University
Burnaby, British Columbia, V5A-1S6, Canada
jkluesne@sfu.ca mmonagan@sfu.ca

Abstract

Algorithms which compute modulo triangular sets must respect the presence of zero-divisors. We present Hensel lifting as a tool for dealing with them. We give an application: a modular algorithm for computing GCDs of univariate polynomials with coefficients modulo a radical triangular set over $\mathbb{Q}$. Our modular algorithm naturally generalizes previous work from algebraic number theory. We have implemented our algorithm using Maple's RECDEN package. We compare our implementation with the procedure `RegularGcd` in the `RegularChains` package.

1 Introduction

Suppose that we seek to find the greatest common divisor of two polynomials $a, b \in \mathbb{Q}(\alpha_1, \ldots, \alpha_n)[x]$ where α_i are algebraic numbers. This problem was first solved using a modular algorithm by Langemyr and McCallum [11] and improved by Encarnacion [6]. Their solution first found a primitive element and then applied an algorithm for one extension. Monagan and van Hoeij [9] improved the multiple extension case by circumventing the primitive element.

The computational model used for an algebraic number field is $\mathbb{Q}[z_1, \ldots, z_n]/T$ where $T = \langle t_1(z_1), t_2(z_1, z_2), \ldots, t_n(z_1, \ldots, z_n)\rangle$ and each t_i is the minimal polynomial of α_i, hence irreducible, over $\mathbb{Q}(\alpha_1, \ldots, \alpha_{i-1})$. A natural generalization is to consider the same problem when each t_i is possibly reducible in which case $\mathbb{Q}[z_1, \ldots, z_n]/T$ has zero divisors.

The generators of T form what is known as a triangular set. Let $R = \mathbb{Q}[z_1, \ldots, z_n]/T$. This paper proposes a new algorithm for computing $\gcd(a, b)$ with $a, b \in R[x]$. The backbone of it is the Euclidean algorithm. However, the EA can't always be used in this ring. For example, suppose $R = \mathbb{Q}[z_1, z_2]/T$ and $T = \langle z_1^2 + 1, z_2^2 + 1\rangle$. Notice that $z_1^2 - z_2^2 = 0$ in R hence $z_1 - z_2$ and $z_1 + z_2$ are zero-divisors in R. Consider computing the gcd of

$$a = x^4 + (z_1 + 18\, z_2)\, x^3 + (-z_2 + 3\, z_1)\, x^2 + 324\, x + 323$$
$$b = x^3 + (z_1 + 18\, z_2)\, x^2 + (-19\, z_2 + 2\, z_1)\, x + 324$$

using the Euclidean algorithm. The remainder of $a \div b$ is

$$r_1 = (z_1 + 18z_2)x^2 + 323.$$

Since $z_1 + 18z_2$ is a unit, a division can be performed; dividing b by r_1 gives

$$r_2 = (z_1 - z_2)x + 1.$$

The next step in the Euclidean algorithm would be to invert $z_1 - z_2$, but it's a zero-divisor, so it cannot continue. The correct approach would be to factor $z_2^2 + 1 = (z_2 - z_1)(z_2 + z_1) \pmod{z_1^2 + 1}$ to split the triangular set T into $\{z_1^2 + 1, z_2 - z_1\}$ and $\{z_1^2 + 1, z_2 + z_1\}$. After that, finish the EA modulo each of these new triangular sets. It's possible to combine the results using the Chinese remainder theorem, but that is costly so it is common practice to instead return the output of the EA along with the associated triangular set. For example, see the definition of pseudo-gcd in [10] and regular-gcd in [12]. We follow this trend with our definition componentwise-gcd in section 4.

Now, consider trying to compute $\gcd(a, b)$ above using a modular GCD algorithm. One would expect to hit the modular image of the same zero-divisor at each prime and hence one could combine them using Chinese remaindering and rational reconstruction. For instance, the EA modulo 13 will terminate with the zero-divisor $z_1 + 12z_2 \pmod{13}$ as expected. However, running the EA modulo 17 terminates earlier because $\mathrm{lc}(r_1) = z_1 + 18z_2 \equiv z_1 + z_2 \pmod{17}$ is a zero-divisor. This presents a problem: $z_1 + z_2 \pmod{17}$ and $z_1 + 12z_2 \pmod{13}$ will never combine into a zero-divisor no matter how many more primes are chosen.

To circumvent, our algorithm finds a monic zero-divisor and lifts it using Hensel lifting to a zero-divisor over $\mathbb{Q}$. Our technique handles both the expected zero-divisors (such as $z_1 + 12z_2 \pmod{13}$ in the above example) and the unexpected zero-divisors (such as $z_1 + z_2 \pmod{17}$). A different approach that we tried is Abbott's fault tolerant rational reconstruction as described in [1]; although this is effective, we prefer Hensel lifting as it enables us to split the triangular set immediately thus saving work.

In section 2, we review important properties of triangular sets, such as being radical. If T is a radical triangular set over $\mathbb{Q}$, reduction modulo p doesn't always result in a radical triangular set. We prove that if T is radical over $\mathbb{Q}$, then $T \mod p$ is radical for all but finitely many primes. We give an algorithm for determining if a prime p enjoys this property, which is based on a corollary from Hubert [10].

In section 3, we present how to use Hensel lifting to solve the zero-divisor problem. We prove a variant of Hensel's lemma that's applicable to our ring and give explicit pseudo-code for a Hensel lifting algorithm. The algorithm is chiefly the Hensel construction, but the presence of zero-divisors demands a careful implementation.

In section 4, we give an application of the Hensel lifting to a modular gcd algorithm. Here, we define componentwise-gcds and prove they exist when T is a radical triangular set. We handle bad and unlucky primes, as par for the course with any modular algorithm. Our algorithm is best seen as a generalization of Monagan and van Hoeij's modular gcd algorithm over number fields [9]. We give pseudo-code for the modular gcd algorithm and all necessary sub-procedures. A second application which we are currently exploring is the inversion problem, that is, given $u \in \mathbb{Q}[z_1, \ldots, z_n]/T$, determine if u is invertible and if so compute u^{-1}.

In section 5, we discuss our implementation of the previously described algorithms in Maple using Monagan and van Hoeij's RECDEN package which uses a recursive dense data structure for polynomials and algebraic extensions. We compare it with the `RegularGcd` procedure in Maple's `RegularChains` package, which uses the subresultant algorithm of Li, Maza, and Pan as described in [12]. This comparison includes examples and time tests.

2 Triangular Sets

2.1 Notation and Definitions

We begin with some notation. All computations will be done in the ring $k[z_1, \ldots, z_n]$ endowed with the monomial ordering $z_i < z_{i+1}$ and k a field. Let $f \in k[z_1, \ldots, z_n]$ be non-constant. The *main* *variable* $\text{mvar}(f)$ of f is the largest variable with nonzero degree in f, and the *main degree* of f is $\text{mdeg}(f) = \deg_{\text{mvar}(f)}(f)$.

As noted in the introduction, triangular sets will be of key interest in this paper. Further, they are to be viewed as a generalization of an algebraic number field with multiple extensions. For this reason, we impose extra structure than is standard:

Definition. A *triangular set* T is a set of non-constant polynomials in $k[z_1, \ldots, z_n]$ with distinct main variables. Further:

(i) $|T| = n$,
(ii) $T = \{t_1, \ldots, t_n\}$ where $\text{mvar}(t_i) = z_i$,
(iii) t_i is monic with respect to z_i, and
(iv) $\deg_{z_j}(t_i) < \text{mdeg}(t_j)$ for $j < i$.

The degree of T is $\prod_{i=1}^{n} \text{mdeg}(t_i)$. Also, $T = \emptyset$ is a triangular set.

Condition (i) states there are no unused variables. This is equivalent to T being zero-dimensional. Condition (ii) gives a standard notation that will be used throughout this paper. Conditions (iii) and (iv) relates the definition to that of minimal polynomials. Condition (iv) is commonly referred to as a reduced triangular set as seen in [2]. The degree of T is akin to the degree of an extension.

Example 1. The polynomials $\{z_1^3 + 4z_1, z_2^2 + (z_1 + 1)z_2 + 4\}$ form a triangular set. However, $\{z_2^2 + (z_1 + 1)z_2 + 4\}$ wouldn't since there's no polynomial with z_1 as a main variable. Also, $\{t_1 = z_1^3 + 4z_1,\ t_2 = z_2^2 + z_1^4 z_2 + 3\}$ isn't because $\deg_{z_1}(t_2) = 4 > \text{mdeg}(t_1)$.

A zero-divisor $u \in k[z_1, \ldots, z_n]$ modulo T is a polynomial such that $u \notin \langle T\rangle$ and there is a polynomial $v \notin \langle T\rangle$ where $uv \in \langle T\rangle$. Since R is a finite-dimensional k-algebra, all nonzero elements are either zero-divisors or units modulo $\langle T\rangle$.

Given a triangular set T, we define $T_i = \{t_1, \ldots, t_i\}$ and $T_0 = \emptyset$. For example, let $T = \{z_1^3 + 1,\ z_2^3 + 2,\ z_3^3 + 3\}$. Then, $T_3 = T$, $T_2 = \{z_1^3 + 1, z_2^3 + 2\}$, $T_1 = \{z_1^3 + 1\}$. In general, since any triangular set T forms a Grobner basis with respect to the lex monomial ordering, it follows that $k[z_1, \ldots, z_i] \cap \langle T\rangle = \langle T_i\rangle$ when $\langle T_i\rangle$ is viewed as an ideal of $k[z_1, \ldots, z_i]$; this is a standard result of elimination theory, see Cox, Little, O'Shea [4].

The presence of zero-divisors presents many unforeseen difficulties that the following examples illustrate.

Example 2. It's possible for a monic polynomial to factor as two polynomials with zero-divisors as leading coefficients. For example, consider the triangular set $T = \{(z_1^2 + 2)(z_1^2 + 1), z_2^3 - z_2\}$. Observe that when working modulo $(z_1^2 + 2)(z_1^2 + 1)$,

$$z_2^3 - z_2 = \left((z_1^2 + 2)z_2^2 - 1\right)\left((z_1^2 + 1)z_2^3 + z_2\right).$$

Of course, a nicer factorization may exist, like $z_2^3 - z_2 = (z_2^2 - 1)z_2$, but it's not clear if this always occurs or how to compute it. This greatly enhances the complexity of handling zero-divisors. The above equation also shows that the degree formula for the product of two polynomials doesn't hold in this setting.

Example 3. Another difficulty is that denominators in the factors of a polynomial $a(x) \in R[x]$ may not appear in the denominators of $a(x)$. Weinberger and Rothschild give the following example in [15]. Let $t_1(z_1) = z_1^6 + 3z_1^5 + 6z_1^4 + z_1^3 - 3z_1^2 + 12z_1 + 16$ which is irreducible over $\mathbb{Q}$. The polynomial

$$f = x - \tfrac{4}{3} - \tfrac{11}{12}z_1 + \tfrac{7}{12}z_1^2 - \tfrac{1}{6}z_1^3 - \tfrac{1}{12}z_1^4 - \tfrac{1}{12}z_1^5$$

is a factor of $a(x) = x^3 - 3$ in $R[x]$. The denominator of any factor of $a(x)$ ($\text{denom}(f) = 12$ in this example) must divide the defect d of the field R. It is known that the discriminant Δ of $t_1(z_1)$

is a multiple of d, usually, much larger than d. Thus we could try to recover Δf with Chinese remaindering then make this result monic. Although one could try to generalize the discriminant to the case $n > 1$, using rational number reconstruction circumvents this difficulty and also allows us to recover g without using a lot more primes than necessary.

Lastly, since there is no standard definition of $\gcd(a, b)$ for $a, b \in R[x]$ where R is a commutative ring unless R is a unique factorization domain, we'd like to make it explicit that $g = \gcd(a, b)$ if (i) $g \mid a$ and $g \mid b$, and (ii) any common divisor of a and b is a divisor of g.

2.2 Radical Triangular Sets

An ideal $I \subset k[x_1, \ldots, x_n]$ is radical if $f^m \in I$ implies $f \in I$. To start, we give a structure theorem for radical and zero-dimensional triangular sets. One could prove this more generally by using the associated primes of T as done in Proposition 4.7 of [10]. The structure theorem gives many powerful corollaries.

Theorem 1. Let $T \subseteq k[z_1, \ldots, z_n]$ be a triangular set. Then, $k[z_1, \ldots, z_n]/T$ is isomorphic to a direct product of fields if and only if T is zero-dimensional and radical.

Corollary 1. Let $T \subset k[z_1, \ldots, z_n]$ be a radical, zero-dimensional triangular set and $R = k[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$. Then a greatest common divisor of a and b exists.

Proof. This follows straightforwardly using the CRT and Theorem 1. □

Corollary 2 (Extended Euclidean Representation)**.** Let $T \subset k[z_1, \ldots, z_n]$ be a radical, zero-dimensional triangular set and $R = k[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$ with $g = \gcd(a, b)$. Then, there exists polynomial $A, B \in R[x]$ such that $aA + bB = g$.

Proof. Note that $R \cong \prod F_i$ where F_i is a field, and we can extend this to $R[x] \cong \prod F_i[x]$. Let $a \mapsto (a_i)_i$ and $b \mapsto (b_i)_i$. Define $h_i = \gcd(a_i, b_i)$ in $F_i[x]$. By the extended Euclidean algorithm, there exists $A_i, B_i \in F_i[x]$ such that $a_iA_i + b_iB_i = h_i$. Let $h \mapsto (h_i)_i$ and $A \mapsto (A_i)_i$ and $B \mapsto (B_i)_i$. Clearly, $aA + bB = h$ in $R[x]$. Since $h \mid g$, we can multiply through by the quotient to write g as a linear combination of a and b. □

It should be noted that Corollary 2 works even if running the Euclidean algorithm on a and b encounters a zero-divisor. This shows it's more powerful than the extended Euclidean algorithm. Further, it also applies to the case where $\mathrm{lc}(g)$ is a zero-divisor.

We next turn our attention to working modulo primes.

Definition. Let $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ be a radical triangular set. A prime number p is a radical prime if p doesn't appear as a denominator of any of the polynomials in T, and if $T \mod p \subset \mathbb{Z}_p[z_1, \ldots, z_n]$ remains radical.

Example 4. The triangular set $\{z_1^2 - 3\}$ is radical over $\mathbb{Q}$. Since the discriminant of $z_1^2 - 3$ is 12, it follows that $2, 3$ aren't radical primes, but all other primes are.

If there were an infinite family of nonradical primes, it would present a problem for the algorithm. We prove this can't happen. This has also been proven with quantitative bounds in [5]. The following lemma is a restatement of Corollary 7.3 of [10]. It also serves as the main idea of our algorithm for testing if a prime is radical; see IsRadicalPrime below.

Lemma 1. Let $T \subset k[z_1, \ldots, z_n]$ be a zero-dimensional triangular set. Then T is radical if and only if $\gcd(t_i, t_i') = 1 \pmod{T_{i-1}}$ for all i.

Theorem 2. Let $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ be a radical, zero-dimensional triangular set. All but finitely many primes are radical primes.

Proof. By Lemma 1, $\gcd(t_i, t_i') = 1$. By the extended Euclidean representation (Corollary 2), there exist polynomials $A_i, B_i \in (\mathbb{Q}[z_1, \ldots, z_{i-1}]/T_{i-1})[z_i]$ where $A_i t_i + B_i t_i' = 1 \pmod{T_{i-1}}$. Take any prime p that doesn't divide the denominator of any A_i, B_i, t_i, t_i'. This means one can reduce this equation modulo p and so $A_i t_i + B_i t_i' \pmod{T_{i-1}, p}$. This implies $\gcd(t_i, t_i') = 1 \pmod{T_{i-1}, p}$ and so T remains radical modulo p by Lemma 1. There are only a finite amount of primes that divide the denominator of any of these polynomials. □

Lastly, we give an algorithm for testing if a prime p is radical. It may not always output `True` or `False` as it relies on Lemma 1 which relies on a gcd computation modulo p, which, if computed by the Euclidean algorithm, may encounter a zero-divisor. If this happens we output the zero-divisor. This case is caught later in the modular gcd algorithm, of which IsRadicalPrime is a subroutine.

Algorithm 1: IsRadicalPrime

Input : A zero-dimensional, radical triangular set $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ and a prime number p where $p \nmid \mathrm{den}(T)$.

Output: A boolean indicating if T remains radical modulo p, or a zero-divisor.

for $i = 1, \ldots, n$ **do**
  $dt := \frac{\partial}{\partial z_i} T[i]$;
  $g := \gcd(T[i], dt)$ over $\mathbb{Z}_p[z_1, \ldots, z_i]/T_{i-1}$;
  if $g =$ [“ZERODIVISOR”, u] **then return** [“ZERODIVISOR”, u];
  if $g \neq 1$ **then return** False;
end
return True;

3 Handling Zero-Divisors

We turn our attention to lifting a factorization $f = ab \pmod{T, p}$ for $a, b, f \in R[x]$. A general factorization will not be liftable; certain conditions are necessary for existence and uniqueness of each lifting step. For one, we will need $\gcd(a, b) = 1 \pmod{p}$ as is required in the case with no extensions to satisfy existence. Further, we will need both a and b to be monic to satisfy uniqueness. The following lemma gives a uniqueness criterion for the extended Euclidean representation. It generalizes Theorem 26 in Geddes, Czapor, Labahn [8] from $F[x]$ to $R[x]$. The proof follows the proof in [8] for $F[x]$. It works because b is monic.

Lemma 2. Let $T \subset k[z_1, \ldots, z_n]$ be a zero-dimensional triangular set and $R = k[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$ be nonzero and monic with polynomials A, B where $1 = Aa + Bb$. Then, for any polynomial $c \in R[x]$, there exist unique polynomials $\sigma, \tau \in R[x]$ such that

$$a\sigma + b\tau = c, \quad \deg(\sigma) < \deg(b).$$

Proof. Existence: Multiplying through $1 = Aa + Bb$ by c gives $a(cA) + b(cB) = c$. Dividing cA by b, which we can do since b is monic, gives $cA = qb + r$ with $r = 0$ or $\deg(r) < \deg(b)$. Define $\sigma = r$ and $\tau = cB + qa$. Observe that

$$a\sigma + b\tau = ar + b(cB + qa) = ar + bcB + abq = a(r + bq) + bcB = acA + bcB = c(aA + bB) = c$$

thus σ and τ satisfy the conditions of the Lemma. *Uniqueness*: Suppose both pairs σ_1, τ_1 and σ_2, τ_2 satisfy $a\sigma_i + b\tau_i = c$ with the desired degree constraint. This yields

$$(\sigma_1 - \sigma_2)a = b(\tau_2 - \tau_1).$$

Since $\gcd(a, b) = 1$, it follows that $b \mid \sigma_1 - \sigma_2$. However, since b is monic and $\deg(\sigma_1 - \sigma_2) < \deg(b)$, this is only possible if $\sigma_1 - \sigma_2 = 0$. Thus $0 = b(\tau_2 - \tau_1)$. Next, since b is not a zero-divisor (because it's monic), this can only happen if $\tau_2 - \tau_1 = 0$ as well. $\square$

We're particularly interested in trying to factor t_n modulo T_{n-1} because encountering a zero-divisor may lead to such a factorization; that is, if w is a zero-divisor with main variable z_n, we can write $u = \gcd(t_n, w)$ and then $t_n = uv \mod \langle T_{n-1}\rangle$ by the division algorithm. As long as T is radical, the next lemma shows we automatically get $\gcd(u, v) = 1$.

Lemma 3. Let $T \subset k[z_1, \ldots, z_n]$ be a radical, zero-dimensional triangular set. Suppose $t_n \equiv uv \pmod{T_{n-1}}$. Then, $\gcd(u, v) = 1 \pmod{T_{n-1}}$.

Proof. Let $u = \overline{u}g \pmod{T_{n-1}}$ and $v = \overline{v}g \pmod{T_{n-1}}$. Note that $t_n \equiv \overline{u}\overline{v}g^2 \pmod{T_{n-1}}$. This would imply $(\overline{u}\overline{v}g)^2 \equiv 0 \pmod{T}$; that is, $\overline{u}\overline{v}g$ is a nilpotent element. However, since nilpotent elements don't exist modulo a radical ideal, $\overline{u}\overline{v}g \equiv 0 \pmod{T}$. This would imply $\overline{u}\overline{v}g \equiv qt_n \pmod{T_{n-1}}$ for some polynomial q. Then,

$$(gq - 1)t_n \equiv gqt_n - t_n \equiv g\overline{u}\overline{v}g - t_n \equiv 0 \pmod{T_{n-1}}.$$

Since t_n is monic in z_n, it can't be a zero-divisor modulo T_{n-1}. Therefore, $gq - 1 \equiv 0 \pmod{T_{n-1}}$. Thus, g is a unit modulo T_{n-1} and so indeed $\gcd(u, v) = 1 \pmod{T_{n-1}}$. $\square$

Finally, the next proposition shows that lifting is possible. The proof given is simply the Hensel construction.

Proposition 1. Let $T \subset \mathbb{Z}_p[z_1, \ldots, z_n]$ be a zero-dimensional triangular set with p a prime number. Suppose $t_n \equiv u_0v_0 \pmod{T_{n-1}, p}$ where u_0 and v_0 are monic. Assume there are polynomials A, B where $1 = Au_0 + Bv_0$. Then, there exist unique monic polynomials u_k, v_k such that $t_n \equiv u_kv_k \pmod{T_{n-1}, p^k}$ and $u_k \equiv u_0 \mod \pmod{T_{n-1}, p}$ and $v_k \equiv v_0 \mod \pmod{T_{n-1}, p}$ for all $k \geq 1$.

Proof. (by induction on k): The base case is clear. For the inductive step, we want to be able to write $u_k = u_{k-1} + p^{k-1}a \pmod{T_{n-1}, p^k}$ and $v_k = v_{k-1} + p^{k-1}b \pmod{T_{n-1}, p^k}$ satisfying

$$t_n \equiv u_kv_k \pmod{T_{n-1}, p^k}.$$

Multiplying out u_k, v_k gives

$$t_n \equiv u_kv_k \equiv u_{k-1}v_{k-1} + p^{k-1}(av_{k-1} + bu_{k-1}) \pmod{T_{n-1}, p^k}.$$

Subtracting $u_{k-1}v_{k-1}$ on both sides and dividing through by p^{k-1} gives

$$\frac{t_n - u_{k-1}v_{k-1}}{p^{k-1}} \equiv av_0 + bu_0 \pmod{T_{n-1}, p}.$$

Let $c = \frac{t_n - u_{k-1}v_{k-1}}{p^{k-1}}$. By Lemma 2, there exists unique polynomials σ, τ such that $u_0\sigma + v_0\tau \equiv c \pmod{T_{n-1}, p}$ with $\deg(\sigma) < \deg(v_0)$ and $\deg(\tau) < \deg(u_0)$ since certainly $\deg(c) = \deg(t_n - u_{k-1}v_{k-1}) < \deg(t_n) = \deg(u_0) + \deg(v_0)$. Set $a = \tau$ and $b = \sigma$. Because of these degree constraints, $u_k = u_{k-1} + ap^{k-1}$ has the same leading coefficient as u_{k-1} and hence u_0; in particular u_k is monic. Similarly, v_k is monic as well. By uniqueness of σ and τ, we get uniqueness of u_k and v_k. $\square$

What follows is a formal presentation of the Hensel construction. The algorithm HenselLift takes input $u_0, v_0, f \in R/\langle p\rangle[x]$ where u_0, v_0 are monic and $f = u_0v_0 \pmod p$. It also requires a bound B that's used to notify termination of the Hensel construction and output FAIL. A crucial part of the Hensel construction is solving the diophantine equation $\sigma u_0 + \tau v_0 = c \pmod{T, p}$. This is done using the extended Euclidean algorithm and Lemma 2. It's possible that a zero-divisor is encountered in this process. This has to be accounted for. Therefore, we allow the HenselLift algorithm to also output ["ZERODIVISOR", u] if it encounters a zero-divisor $u \in R/\langle p\rangle$.

Algorithm 2: HenselLift

Input : A zero-dimensional triangular set $T \subset \mathbb{Q}[z_1, \ldots, z_n]$, a radical prime p, polynomials $f \in R[x]$ and $a_0, b_0 \in R/\langle p\rangle[x]$ where $R = \mathbb{Q}[z_1, \ldots, z_n]/T$, and a bound B. Further, assume $f \equiv a_0b_0 \pmod p$ and $\gcd(a_0, b_0) = 1$.

Output: Either polynomials $a, b \in R[x]$ where $f = ab$, FAIL if the bound B is reached, or ["ZERODIVISOR", w] if a zero-divisor $w \in R/\langle p\rangle$ is encountered.

Solve $sa_0 + tb_0 = 1$ using the Monic extended Euclidean algorithm for $s, t \in \mathbb{R}/\langle p\rangle[x]$;
if a zero-divisor w is encountered **then return** ["ZERODIVISOR", w];
Initialize $u = a_0, v = b_0$ and lift u and v from $R/\langle p\rangle$ to R;
for $i = 1, 2, \ldots$ **do**
  Apply rational reconstruction mod p^i to the coefficients of u;
  if rational reconstruction succeeded with output a and $a|f$ in $R[x]$ **then return** $(a, f/a)$;
  if $p^i > 2B$ **then return** FAIL;
  Compute $e := f - uv$ in $R[x]$;
  Set $c := (e/p^i) \mod p$;
  Solve $\sigma a_0 + \tau b_0 = c$ for $\sigma, \tau \in R/\langle p\rangle[x]$ using $sa_0 + tb_0 = 1$;
  Lift σ and τ from $R/\langle p\rangle$ to R and set $u := u + \tau p^i$ and $v := v + \sigma p^i$;
end

In general the input f will have fractions thus the error e in our Hensel lifting algorithm will also have fractions and hence it can never become 0. Note the size of the rational coefficients of e grow linearly with i as f is fixed and the magnitude of the integer coefficients in the product uv are bounded by $p^{2i}(1 + \deg u)$.

The standard implementation of Hensel lifting requires a bound on the coefficients of the factors of the polynomial $f \in R[x]$. For the base case $n = 0$ where $R[x] = \mathbb{Q}[x]$ one can use the Mignotte bound (see [7]). For the case $n = 1$ Weinberger and Rothschild [15] give a bound but note that it is large. We do not know of any bounds for the general case $n > 1$ and hypothesize that they would be bad. Therefore a more "engineering"-esque approach is needed. Since we do not know whether the input zero-divisor a_0 is the image of a monic factor of f, we repeat the Hensel lifting each time a zero-divisor is encountered in our modular GCD algorithm, first using a bound of 2^{60}, then 2^{120}, then 2^{240} and so on, until the coefficients of any monic factor of f can be recovered using rational number reconstruction.

The prime application of Hensel lifting will be as a solution to the zero-divisor problem. This is the goal of the HandleZeroDivisorHensel algorithm. The algorithm assumes a zero-divisor modulo a prime p has been encountered by another algorithm (such as our modular gcd algorithm). It attempts to lift this zero-divisor using HenselLift. If HenselLift encounters a new zero-divisor w, it recursively calls HandleZeroDivisorHensel(w). If the Hensel lifting fails (i.e., a bound is reached), it instructs the algorithm using it to pick a new prime. If the Hensel lifting succeeds in finding a factorization $t_n = uv \pmod{T_{n-1}}$ over $\mathbb{Q}$, then the algorithm using it works recursively on new triangular sets $T^{(u)}$ and $T^{(v)}$ where t_n is replaced by u and v, respectively.

Algorithm 3: HandleZeroDivisorHensel

Input : A zero-dimensional triangular set $T \subset k[z_1, \ldots, z_n]$ modulo a prime p and a zero-divisor $u_0 \in R$ where $R = k[z_1, \ldots, z_n]/T$. Assume $\text{mvar}(u) = n$.

Output: A message indicating the next steps that should be carried out, including any important parameters;

Set $v_0 :=$Quotient$(t_n, u_0) \pmod{T_{n-1}, p}$;
if $v_0 =$ [“ZERODIVISOR”, w] **then return** HandleZeroDivisorHensel(w);
if the global variable B is unassigned **then** set $B := 2^{60}$ **else** set $B := B^2$;
Set $u, v :=$HenselLift(t_n, u_0, v_0, B);
if $u =$ [“ZERODIVISOR”, w] **then**
  return HandleZeroDivisorHensel(w)
else if $u =$FAIL **then**
  return FAIL. This indicates that a new prime or bigger bound is needed;
else
  return u and v;
end

We'd like to make it clear that this is not the first case of using p-adic lifting techniques on triangular sets. In particular, lifting the triangular decomposition of a regular chain has been used by Dahan, Maza, Schost, Wu, Xie in [5].

4 A Modular Gcd Algorithm

The main content of this section is to fully present and show the correctness of our modular gcd algorithm. First, suppose a zero-divisor w over $\mathbb{Q}$ is found while running the modular algorithm. It will be used to factor $t_k = uv \pmod{T_{k-1}}$ where u and v are monic with main variable z_k. From here, the algorithm proceeds to split T into $T^{(u)}$ and $T^{(v)}$ where t_k is replaced with u in $T^{(u)}$ and v in $T^{(v)}$. Of course t_i is reduced for $i > k$ as well. The algorithm then continues recursively. Once the recursive calls are finished, we could use the CRT to combine gcds into a single gcd, but this would be very time consuming. Instead, it's better to just return both gcds along with their associated triangular sets. This approach is similar to Hubert's in [10] which she calls a pseudo-gcd. Here, we refer to this as a component-wise gcd, or c-gcd for short:

Definition. Let R be a commutative ring with unity such that $R \cong \prod_{i=1}^{r} R_i$ and $a, b \in R[x]$. Let $\pi_i \colon R \to R_i$ be the natural projections. A component-wise gcd of a and b is a tuple $(g_1, \ldots, g_r) \in \prod_{i=1}^{r} R_i[x]$ where each $g_i = \gcd(\pi_i(a), \pi_i(b))$ and $\text{lc}(g_i)$ is a unit.

The modular algorithm's goal will be to compute c-gcd(a, b) given $a, b \in R[x]$ where $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ and $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ is a radical triangular set. As with all modular gcd algorithms, it's possible that some primes are unlucky. We also prove this only happens for a finite number of cases.

Definition. Let $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ be a radical triangular set, and $R = \mathbb{Q}[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$ and $g =$ c-gcd(a, b). A prime number p is an *unlucky prime* if g doesn't remain a componentwise greatest common divisor of a and b modulo p. Additionally, a prime is *bad* if it divides any denominator in T, any denominator in a or b, or if $\text{lc}(a)$ or $\text{lc}(b)$ vanishes modulo p.

Theorem 3. Let $T \subset \mathbb{Q}[z_1, \dots, z_n]$ be a radical triangular set, and $R = \mathbb{Q}[z_1, \dots, z_n]/T$. Let $a, b \in R[x]$ and $g = \text{c-gcd}(a, b)$. Only finitely many primes are unlucky.

Proof. Let $R[x] \cong \prod R_i[x]$ where $g = (g_i)$ and $g = \gcd(a, b) \in R_i[x]$. Let $a \mapsto (a_i)$ and $b \mapsto (b_i)$. If $g_i = 0$, then $a_i = 0$ and $b_i = 0$ and no primes are unlucky since, $\gcd(0, 0) \equiv 0 \pmod p$. Suppose $g_i = \gcd(a_i, b_i)$ is nonzero and monic. Let $\overline{a}_i$ and $\overline{b}_i$ be the cofactors $a_i = g_i \overline{a}_i$ and $b_i = g_i \overline{b}_i$. I claim $\gcd(\overline{a}_i, \overline{b}_i) = 1$. To show this, consider a common divisor f of $\overline{a}_i$ and $\overline{b}_i$. Note that $fg_i \mid a_i$ and $fg_i \mid b_i$. Since $g_i = \gcd(a_i, b_i)$, it follows that $fg_i \mid g_i$; so there exists $q \in R_i[x]$ where $fg_iq = g_i$. Rewrite this equation as $(fq - 1)g_i = 0$. Well, g_i is monic in x, and so can't be a zero-divisor. This implies $fq - 1 = 0$ and so indeed f is a unit. Thus, $\gcd(\overline{a}_i, \overline{b}_i) = 1$. By the extended Euclidean representation (Corollary 2), there exists $A_i, B_i \in R_i[x]$ where $\overline{a}_i A_i + \overline{b}_i B_i = 1$.

Let p be a prime where p doesn't divide any of the denominators in $a_i, \overline{a}_i, A_i, b_i, \overline{b}_i, B_i, g_i$. Then, we can reduce the equations

$$\overline{a}_i A_i + \overline{b}_i B_i = 1 \pmod p, \tag{1}$$

$$a_i = g_i \overline{a}_i \pmod p, \qquad b_i = g_i \overline{b}_i \pmod p. \tag{2}$$

We will now show that $g_i = \gcd(a_i, b_i) \pmod p$. By (2), we get g_i is a common divisor of a_i and b_i modulo p. Consider a common divisor c of a_i and b_i modulo p. Multiplying equation (1) through by g_i gives $a_i A_i + b_i B_i = g_i \pmod p$. Clearly, $c \mid g_i$ modulo p. Thus, g_i is indeed a greatest common divisor of a_i and b_i modulo p. As there are finitely many primes that can divide the denominators of fractions in the polynomials $a_i, \overline{a}_i, A_i, b_i, \overline{b}_i, B_i, g_i$, there are indeed finitely many unlucky primes. □

Example 5. This example illustrates how the IsRadical function can run into a zero-divisor. Consider $T = \{z_1^2 - 1, z_2^3 + 9z_2^2 + \frac{3z_1+51}{2} z_2 - \frac{53z_1+3}{2}\}$. We will run the algorithm over $\mathbb{Q}$ to illustrate. First, it would determine that $T_1 = \{z_1^2 - 1\}$ is radical. Now, when it is running the Euclidean algorithm on $t_2 = z_2^3 + 9z_2^2 + \frac{3z_1+51}{2} z_2 - \frac{53z_1+3}{2}$ and $t_2' = 3z_2^2 + 18z_2 + \frac{3z_1+51}{2}$, the first remainder would be $(z_1 - 1)z_2 - 28z_1 - 27$. However, $z_1 - 1$ is a zero-divisor, so the algorithm would output ["ZERODIVISOR", $z_1 - 1$]. This same zero-divisor will show up for every odd prime (2 appears in the denominator of t_2 and so shouldn't be considered). This explains why we can't just simply pick a new prime in ModularC-GCD if IsRadical encounters a zero-divisor.

The crux of ModularC-GCD is an algorithm to compute $\gcd(a, b)$ for two polynomials $a, b \in (\mathbb{Z}_p[z_1, \dots, z_n]/T)[x]$. The algorithm we'll be using for this is MonicEuclideanC-GCD. It is a variant of the monic Euclidean algorithm. For computing inverses, the extended Euclidean algorithm can be used; modifying MonicEuclideanC-GCD to do this is straightforward.

A short discussion about the zero-divisors that may appear is warranted. To compute an inverse, the modular algorithm will be using the extended Euclidean algorithm. The first step would be to invert a leading coefficient u of some polynomial. This requires a recursive call to ExtendedEuclideanC-GCD$(u, t_k) \pmod{T_{k-1}}$ where $z_k = \text{mvar}(u)$. If u isn't monic, then it would again attempt to invert $\text{lc}(u)$. Because of the recursive nature, it will keep inverting leading coefficients until it succeeds or a monic zero-divisor is found. The main point is that we may assume that the zero-divisors encountered are monic.

Now that all algorithms have been given, we give a proof of correctness for ModularC-GCD. First, we show that a finite number of zero-divisors can be encountered. This ensures that the algorithm terminates. After that, we prove a lemma about the primes that may occur in a monic factorization modulo the triangular set; note this is nontrivial by example 3. This a key step in the proof that the returned value of ModularC-GCD is correct. The proof will require the concept

Algorithm 4: ModularC-GCD

Input : A zero-dimensional, radical triangular set $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ and two polynomials $a, b \in R[x]$ where $R = \mathbb{Q}[z_1, \ldots, z_n]/T$. Assume $\deg(a) \geq \deg(b) \geq 0$.

Output: A tuple consisting of comaximal triangular sets $T^{(i)}$ such that $T = \bigcap T^{(i)}$ and $g_i = \gcd(a, b) \mod \langle T^{(i)} \rangle$ where $g_i = 0$ or $\mathrm{lc}(g_i)$ is a unit.

Initialize $dg := \deg(b)$, $P = 1$;
Main Loop: Pick a prime p that isn't bad;
Test if p is a radical prime, $N :=$ isRadicalPrime(T, p);
if $N =$ [“ZERODIVISOR”, u] **then**
    $M :=$ HandleZeroDivisorHensel(u);
    if $M =$ FAIL **then** Pick a new prime, go to Main Loop;
    else if M is a factorization $t_k = wv \pmod{T_{k-1}}$ **then**
        Create triangular sets $T^{(w)}$ and $T^{(v)}$ where t_k is replaced by w and v, respectively;
        return ModularC-GCD$(a, b) \pmod{T^{(w)}}$, ModularC-GCD$(a, b) \pmod{T^{(v)}}$
    end
else if $B =$ False **then**
    Pick a new prime: Go to Main Loop;
end
Set $g := \gcd(a, b) \mod \langle T, p \rangle$ using algorithm MonicEuclideanC-GCD;
if $g =$ [“ZERODIVISOR”, u] **then**
    $M :=$ HandleZeroDivisorHensel(u);
    if $M =$ FAIL **then** Pick a new prime: Go to Main Loop;
    else if M is a factorization $t_k = wv \pmod{T_{k-1}}$ **then**
        Create triangular sets $T^{(w)}$ and $T^{(v)}$ where t_k is replaced by w and v, respectively;
        return ModularC-GCD$(a, b) \pmod{T^{(w)}}$, ModularC-GCD$(a, b) \pmod{T^{(v)}}$
    end
else `// The following treatment of unlucky primes and rational reconstruction`
    `// follows van Hoeij and Monagan [9] page 115.`
    if $\deg(g) = dg$ **then**
        The chosen prime seems to be lucky;
        Use CRT to combine g other gcds (if any), store the result in G and set $P := P \times p$;
    else if $\deg(g) > dg$ **then**
        The chosen prime was unlucky, discard g;
        Pick a new prime: Go to Main Loop;
    else if $\deg(g) < dg$ **then**
        All previous primes were unlucky, discard G;
        Set $G := g$ and $P := p$;
    end
    Set $h :=$ RationalReconstruction$(G \pmod{P})$;
    if $h \neq$ FAIL and $h \mid a$ and $h \mid b$ **then return** (T, h);
    Pick a new prime: Go to Main Loop;
end

Algorithm 5: MonicEuclideanC-GCD

Input : A zero-dimensional, radical triangular set $T \subset k[z_1, \ldots, z_n]$ and two polynomials $a, b \in R[x]$ where $R = k[z_1, \ldots, z_n]/T$. Assume $\deg(a) \geq \deg(b) \geq 0$.

Output: Either $\gcd(a, b) \pmod{T}$ or a zero-divisor.

Initialize $r_0 := a$, $r_1 := b$ and $i := 1$;
while $r_i \neq 0$ **do**
  Compute $s := \mathrm{lc}(r_i)^{-1} \pmod{T_{n-1}}$ using the EEA;
  if $s =$["ZERODIVISOR", u] **then return** ["ZERODIVISOR", u] **else** $r_i := s \times r_i$;
  Let r_{i+1} be the remainder of r_{i-1} divided by r_i;
  $i = i + 1$;
end
return r_{i-1}

of localization, the formal process of including denominators in a ring; see Bosch [3] for details. For notation purposes, we let S be a set of prime numbers and define R_S as the localization of R with respect to S. Note that when $R = \mathbb{Q}[z_1, \ldots, z_n]/T$, it's required that any prime dividing any $\mathrm{den}(t_i)$ must be included in S for R_S to be a ring. We will also need the concept of the iterated resultant. Let f be non-zero in $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ and let $\mathrm{res}_z(f(z), t(z))$ denote the Sylvester resultant of two polynomials $f(z)$ and $t(z)$. The iterated resultant of f with T is

$$\mathrm{iterres}(f, T) = \mathrm{iterres}(\mathrm{res}_{z_n}(f, t_n), T_{n-1}), \qquad \mathrm{iterres}(f, \{t_1\}) = \mathrm{res}_{z_1}(f, t_1).$$

One important property is that if $f, T \in R'[x] \subset R[x]$ where R' is a subring, then there exist $A, B_1, \ldots, B_n \in R'[x]$ where $Af + B_1 t_1 + \cdots + B_n t_n = \mathrm{iterres}(f, T)$. This follows from the same proof as given in Theorem 7.1 of [8]. Another important property is that $\mathrm{iterres}(f, T) = 0$ if and only if f is a zero-divisor, see [2].

Proposition 2. Let $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ where T is a radical zero-dimensional triangular set. Put $a, b \in R[x]$. A finite number of zero-divisors are encountered when running ModularC-GCD(a, b).

Proof. We use induction on the degree of the extension $\delta = d_1 \cdots d_n$ where $d_i = \mathrm{mdeg}(t_i)$. If $\delta = 1$, then $R = \mathbb{Q}$ so no zero-divisors occur.

First, there are a finite number of non-radical primes. So we may assume that T remains radical modulo any chosen prime. Second, consider (theoretically) running the monic Euclidean algorithm over $\mathbb{Q}$ where we split the triangular set if a zero-divisor is encountered. In this process, a finite number of primes divide either denominators or leading coefficients of the remainders appearing in the Euclidean algorithm; so we may assume the modular algorithm isn't choosing these primes without loss of generality.

Now, suppose a prime p is chosen by the modular algorithm and a zero-divisor u_p is encountered modulo p at some point of the algorithm. This implies $\gcd(u_p, t_k) \not\equiv 1 \pmod{T_{k-1}, p}$ for some $1 \leq k \leq n$. We may assume that $u_p = \gcd(u_p, t_k) \pmod{T_{k-1}, p}$ and that u_p is monic; this is because the monic Euclidean algorithm will only output such zero-divisors. If u_p lifts to a zero-divisor over $\mathbb{Q}$, the algorithm constructs two triangular sets, each with degree smaller than δ. So by induction, a finite number of zero-divisors occur in each recursive call. Now, suppose lifting fails. This implies there is some polynomial u over $\mathbb{Q}$ that reduces to u_p modulo p and appears in the theoretical run of the Euclidean algorithm over $\mathbb{Q}$. Note that $\gcd(u, t_k) = 1 \pmod{T_{k-1}}$ over $\mathbb{Q}$ since we're assuming the lifting failed. By Theorem 3, this happens for only a finite amount of primes. Thus, a finite number of zero-divisors are encountered. □

Lemma 4. Let T be a radical, zero-dimensional triangular set of $F = \mathbb{Z}[z_1, \ldots, z_n]$. Suppose $f, u \in R[x]$ are monic such that $u \mid f$. Let

$$S = \{\text{prime numbers } p \in \mathbb{Z} : p \text{ is a nonradical prime with respect to } T, \text{ or } p \mid \operatorname{den}(f)\}.$$

Then, $u \in F_S[x]/T$. In particular, the primes appearing in denominators of any monic factor of f are either nonradical primes or divisors of $\operatorname{den}(f)$.

Proof. Proceed by induction on n. Consider the base case $n = 1$. Let $t_1 = a_1 a_2 \cdots a_s$ be the factorization into monic irreducibles. Note that a_i, a_j are relatively prime since t_1 is square-free and $a_1, a_2 \in F_S$ by Gauss's lemma (since S contains any primes dividing $\operatorname{den}(t_1)$). Let $u_i = u \mod a_i$ and $f_i = f \mod a_i$. By known results from algebraic number theory (see Theorem 3.2 of [6] for instance), $\operatorname{den}(u_i)$ consists of primes dividing $\Delta(a_i)$ or $\operatorname{den}(f_i)$. Note that any prime $p \mid \Delta(a_i)$ would force a_i, and hence t_1, to not be square-free modulo p. This would imply p is nonradical and so is contained in S; in partiulcar, $u_i \in F_S[x]$.

The last concern is if combining $(u_1, u_2, \ldots, u_s) \mapsto u$ introduces another prime p into the denominator. We prove this can only happen if p is nonradical. It's sufficient to show that combining two extensions is enough since we can simply combine two at a time until the list is exhausted. Now, consider the resultant $r = \operatorname{res}_{z_1}(a_1, a_2)$. There are polynomials $A, B \in F_S$ where $Aa_1 + Ba_2 = r$. Note that any prime $p \mid r$ forces $\gcd(a_1, a_2) \not\equiv 1 \pmod p$ and so t_1 wouldn't be square-free; in particular, $A/r, B/r \in F_S$. Now, let $v = (A/r)a_1u_2 + (B/r)a_2u_1$. Note that $v \mod a_1 = (B/r)a_2u_1 = (1 - (A/r)a_1)u_1 = u_1$. Similarly, $v \mod a_2 = u_2$. Since the CRT gives an isomorphism, $u = v$ and indeed $u \in F_S[x]$. This completes the base case.

For the general case, we will generalize each step used in the base case. Instead of just factoring t_1, we decompose T as a product of comaximal triangular sets known as its triangular decomposition. In place of discriminants of polynomials, we use discriminants of algebraic number fields. Finally, for the combining, iterated resultants are used instead of resultants.

With that in mind, start by decomposing T into its triangular decomposition, which can be done in the the following way:

1. Factor $t_1 = a_1 a_2 \cdots a_{s_1}$ into relatively prime monic irreducibles over $\mathbb{Q}$ as in the base case. This gives $\mathbb{Q}[z_1]/T_1$ is isomorphic to the product of fields $\prod_i \mathbb{Q}[z_1]/a_i$. By Gauss's lemma, a prime dividing the $\operatorname{den}(a_i)$ must also divide $\operatorname{den}(f)$. In particular, $a_i \in F_S[x]/T$.

2. We can factor the image of $t_2^{(i)}$ over each $\mathbb{Q}[z_1]/a_i$ into monic relatively prime irreducibles $t_2^{(i)} = b_1^{(i)} b_2^{(i)} \cdots b_{s_2}^{(i)}$. Note that changing rings from $\mathbb{Q}[z_1]/t_1$ to $\mathbb{Q}[z_1]/a_i$ only involves division by a_i, and hence the only primes introduced into denominators can come from $\operatorname{den}(a_i)$.

3. By the induction hypothesis, any prime p dividing $\operatorname{den}(b_j^{(i)})$ is either not a radical prime of the triangular set $\{a_i\}$ or comes from $\operatorname{den}(t_2^{(i)})$. If $\{a_i\}$ isn't radical modulo p, then neither is $\{t_1\}$, clearly.

4. Use this to decompose $k[z_1, z_2]/T_2$ into fields $\mathbb{Q}[z_1, z_2]/\langle a_i, b_j^{(i)} \rangle$ where $a_i, b_j^{(i)} \in F_S[x]/T$.

5. Repeat to get $\mathbb{Q}[z_1, \ldots, z_n]/T \cong \prod \mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ where each $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ is a field and $T^{(i)} \subset F_S$ using the induction hypothesis.

Let $f^{(i)} = f \mod T^{(i)}$ and similarly $u^{(i)} = u \mod T^{(i)}$. Since $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ is an algebraic number field, any prime p occuring in $\operatorname{den}(u^{(i)})$ must either divide the discriminant $\Delta(\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)})$ or $\operatorname{den}(f^{(i)})$. This implies p must be nonradical with respect to $T^{(i)}$ or divide $\operatorname{den}(f^{(i)})$. (To be more explicit, one could write $\mathbb{Q}[z_1,\ldots,z_n]/T^{(i)} = \mathbb{Q}(\alpha)$ and note that $p \mid \Delta(\mathbb{Q}[z_1,\ldots,z_n]/T^{(i)})$ which divides the discriminant $\Delta(m_{\alpha,\mathbb{Q}})$ of the primitive minimal polynomial $m_{\alpha,\mathbb{Q}}$ of α. If $p \mid \Delta(m_{\alpha,\mathbb{Q}})$, then $m_{\alpha,\mathbb{Q}}(z)$ isn't square-free and so $\mathbb{Z}_p[z]/m_{\alpha,\mathbb{Q}}$ would contain a nilpotent element.)

Of course $\mathrm{den}(u^{(i)}) \neq \mathrm{den}(u)$. It remains to show that going from $\prod \mathbb{Q}[z_1,\ldots,z_n]/T^{(i)}$ to $\mathbb{Q}[z_1,\ldots,z_n]/T$ only introduces primes in the denominators that are divisors of $\mathrm{den}(f)$ or nonradical. This will follow from using iterated resultants similarly to the resultants in the base case. Suppose we are trying to combine $T^{(i)}$ and $T^{(j)}$ with all $t_k^{(i)} = t_k^{(j)}$ besides $t_n^{(i)} \neq t_n^{(j)}$. Now, perform the iterated resultant and write

$$r = \mathrm{iterres}(\mathrm{res}(t_n^{(i)}, t_n^{(j)}), T_{n-1}^{(i)}) = At_n^{(i)} + Bt_n^{(j)}$$

with $A, B \in F_S[x]$ since $t_n^{(i)}, t_n^{(j)} \in F_S[x]$ are by construction. Well, any prime p that divides r would have the property of $\gcd(t_n^{(i)}, t_n^{(j)}) \neq 1 \pmod{p}$. Hence t_n wouldn't be square-free and so T wouldn't be radical mod p. Thus, after recovering all splittings into the ring $\mathbb{Q}[z_1,\ldots,z_n][x]/T$, we indeed get $u \in F_S[x]$. □

Theorem 4. Let $R = \mathbb{Q}[z_1,\ldots,z_n]/T$ where T is a radical zero-dimensional triangular set and let $a, b \in R[x]$. The modular algorithm using Hensel lifting to handle zero-divisors outputs a correct c-gcd if run on a and b.

Proof. It is enough to prove this for a single component of the decomposition. For ease of notation, let T be the triangular set associated to this component and let h be the monic polynomial returned from the modular algorithm with T and let $g = \gcd(a,b) \pmod{T}$ over $\mathbb{Q}$.

First, we may assume that b is monic. If $\mathrm{lc}_x(b)$ is a unit, divide through by it's inverse and this doesn't change $\gcd(a,b)$. If $\mathrm{lc}_x(b)$ is a zero-divisor, the EA mod p would catch it and cause a splitting, contradicting that the EA mod p didn't encounter a zero-divisor in this component of the c-gcd. Since h passed the trial division in step 34, it follows that $h \mid g$ and hence $\deg(h) \leq \deg(g)$ since h is monic. Suppose $\mathrm{lc}(g)$ is invertible. If so, make g monic without loss of generality. Let p be a prime used to compute h. Since g is monic and divides b which is also monic, any prime appearing in $\mathrm{den}(g)$ is either nonradical or a divisor of $\mathrm{den}(b)$ by Lemma 4. In particular, since the prime p was used successfully to compute h, it can't occur in the denominator of g. So, we may reduce g modulo p. Let $\overline{f}$ denote the reduction of a polynomial $f \in R[x]$ mod p. Since $\overline{g} \mid \overline{a}$ and $\overline{g} \mid \overline{b}$, it follows that $\overline{g} \mid \overline{h}$ and so $\deg(g) \leq \deg(h)$. Since $h \mid g$, they have the same degree, and both are monic, it must be that $h = g$ and so indeed h is a greatest common divisor of a and b.

Suppose $\mathrm{lc}(g)$ was a zero-divisor and that $\mathrm{mvar}(\mathrm{lc}(g)) = z_n$ without loss. Inspect $\mathrm{lc}_{z_n}(\mathrm{lc}(g))$; if this is a unit, make it monic. If it's a zero-divisor, inspect $\mathrm{lc}_{z_{n-1}}(\mathrm{lc}_{z_n}(g)))$. Continue until $u = \mathrm{lc}_{z_{k+1}}(\cdots(\mathrm{lc}_{z_n}(\mathrm{lc}_x(g))\cdots)$ is a monic zero-divisor. Further, if $\gcd(u, t_k) \neq u$, then $u/\gcd(u,t_k)$ is a unit and so we can divide through by it to ensure $\gcd(u,t_k) = u$. Let $t_k = uv \pmod{T_{k-1}}$ be a monic factorization. Note that Lemma 4 guarantees that the same factorization $\overline{uv} = \overline{t_k} \pmod{T_{k-1}, p}$ occurs modulo p. Hence, we can split T into triagular sets $T^{(u)}$ and $T^{(v)}$ where t_k is replaced by u and v, respectively, and this same splitting occurs modulo p.

Let $g_u = g \mod T^{(u)}$ and $g_v = g \mod T^{(v)}$ and similarly for other relevant polynomials. It's straightforward to show that $\overline{h_u}$ is still a gcd of $\overline{a_u}$ and $\overline{b_u}$ and g_u for a_u and b_u. Now, we consider both triangular sets $T^{(u)}$ and $T^{(v)}$. First, in $T^{(v)}$, u is invertible otherwise T wouldn't be radical. So, multiply g_v by u^{-1} so that $\mathrm{lc}_{z_{k+1}}(\cdots(\mathrm{lc}_{z_n}(\mathrm{lc}_x(g))\cdots) = 1$. Reinspect $w = \mathrm{lc}_{z_{k+2}}(\cdots(\mathrm{lc}_{z_n}(\mathrm{lc}_x(g_v)))\cdots)$. If w isn't a zero-divisor, multiply through by it's inverse and repeat until a zero-divisor is encountered as a leading coefficient. Do the same computations to find another splitting and be in the same situation as that of u in T. Otherwise, in $T^{(u)}$, $u = 0$ and so $\mathrm{lc}_{z_{k+1}}(\cdots(\mathrm{lc}_{z_n}(\mathrm{lc}_x(g_u))\cdots)$ has changed; if it's invertible, multiply through by it's inverse until a monic zero-divisor is found in the leading coefficient chain. We again wind up in the situation with a monic factorization of t_j that is reducible modulo p.

The process described in the last paragraph must terminate with a splitting in which the image of g is monic since $\mathrm{lc}_x(g)$ has finite degree in each variable. We have already shown that the image of h would be an associate of the image in g in this case. Since being a gcd persists through isomorphisms, this gives indeed that h is a $\gcd(a, b)$ modulo T, as desired. □

5 Comparison with `RegularGcd`

We have implemented algorithm `ModularC-GCD` as presented above using Maple's RECDEN package which uses a recursive dense data structure for polynomials with extensions. Details can be found in Monagan and van Hoeij's paper [9]. The reader may find our Maple code for our software there together with several examples and their output at `http://www.cecm.sfu.ca/CAG/code/MODGCD`.

The remainder of this section will be used to compare our algorithm with the `RegularGcd` algorithm (see [12]) which is in the `RegularChains` package of Maple. Algorithm `RegularGcd` computes a subresultant polynomial remainder sequence and outputs the last non-zero element of the sequence. We highlight three differences between the output of `RegularGcd` and `ModularC-GCD`.

1. The algorithms may compute different triangular decompositions of the input triangular set.
2. `RegularGcd` returns the last non-zero subresultant but not reduced modulo T; it often returns a gcd g with $\deg_{z_i}(g) > \mathrm{mdeg}(t_i)$. To compute the reduced version, the procedure `NormalForm` is required. `ModularC-GCD` uses the CRT and rational reconstruction on images of the c-gcd modulo multiple primes, so it computes the reduced version of the c-gcd automatically.
3. `RegularGcd` computes gcds up to units, and for some inputs the units can be large. `ModularC-GCD` computes the monic gcd which may have large fractions.

Example 6. We'd like to illustrate the differences with an example provided by an anonymous referee of an earlier version of this paper. Let

$$\begin{aligned}
&T = \{x^3 - x,\;\; y^2 - \tfrac{3}{2}yx^2 - \tfrac{3}{2}yx + y + 2x^2 - 2\},\\
&a = z^2 - \tfrac{8}{3}zyx^2 + 3zyx - \tfrac{7}{3}zy - \tfrac{1}{3}zx^2 + 3zx - \tfrac{5}{3}z + \tfrac{25}{6}yx^2 - \tfrac{13}{2}yx + \tfrac{10}{3}y + \tfrac{16}{3}x^2 - 2x - \tfrac{10}{3},\\
&b = z^2 + \tfrac{29}{12}zyx^2 + \tfrac{7}{4}zyx - \tfrac{11}{3}zy - \tfrac{8}{3}zx^2 + 3zx + \tfrac{2}{3}z + \tfrac{67}{12}yx^2 - \tfrac{11}{4}yx - \tfrac{13}{3}y - \tfrac{13}{3}x^2 - 2x + \tfrac{19}{3}.
\end{aligned}$$

When we run our algorithm to compute $\text{c-gcd}(a, b) \pmod{T}$, it returns

$$\begin{aligned}
z^2 + (3x-2)z - 2x + 2 &\quad (\mathrm{mod}\ y, x^2 - 1),\\
z + \tfrac{1}{2}x - \tfrac{3}{2} &\quad (\mathrm{mod}\ y - \tfrac{3}{2}x - \tfrac{1}{2}, x^2 - 1),\\
z + 5 &\quad (\mathrm{mod}\ y + 2, x),\\
1 &\quad (\mathrm{mod}\ y - 1, x).
\end{aligned}$$

The same example using `RegularGcd` returns

$$\begin{aligned}
(-96\,y + 168)\,z - 552\,y + 696 &\quad (\mathrm{mod}\ y + 2, x),\\
154368\,y^3 - 117504\,y^2 - 559872\,y + 585216 &\quad (\mathrm{mod}\ y - 1, x),\\
z^2 + (\tfrac{2}{3} - \tfrac{8}{3}x^2 + 3x)z &\quad (\mathrm{mod}\ y, x - 1),\\
(366x^2 - 90x - 96)yz + (102x^2 + 270x - 552)y &\quad (\mathrm{mod}\ y - 2, x - 1),\\
z^2 + (\tfrac{2}{3} - \tfrac{8}{3}x^2 + 3x)z + \tfrac{19}{13} - \tfrac{13}{3}x^2 - 2x &\quad (\mathrm{mod}\ y, x + 1),\\
(366x^2 - 90x - 96)yz + (102x^2 + 270x - 552)y &\quad (\mathrm{mod}\ y + 1, x + 1).
\end{aligned}$$

As can be seen, our algorithm decomposes T into only 4 triangular sets while `RegularGcd` decomposes T into 6 and each component in our output is reduced while the output of `RegularGcd` isn't. Applying the `NormalForm` command to reduce the output of `RegularGcd` returns

$$\begin{aligned} 360z+1800 &\pmod{y+2, x}, & 62208 &\pmod{y-1, x}, \\ z^2+z &\pmod{y, x-1}, & 360z-360 &\pmod{y-2, x-1}, \\ z^2-5z+4 &\pmod{y, x+1}, & -360z+720 &\pmod{y+1, x+1}. \end{aligned}$$

Notice that it circumvents fractions. In general, the output of our algorithm deals with smaller numbers. This can certainly be seen as an advantage for a user.

Finally, we'd like to conclude with some timimg tests which show the power of using a modular GCD algorithm that recovers the monic c-gcd from images modulo primes using rational reconstruction. We first construct random triangular sets where each t_i is monic in z_i and dense in $z_1, \ldots, z_{i-1}$ with random two digit coefficients. We then generate $a, b, g \in R[x]$ with degrees $6, 5$, and 4, respectively. Then, compute c-gcd(A, B) where $A = ag$ and $B = bg$. Maple code for generating the test inputs is included on our website.

In the previous dataset, g isn't created as a monic polynomial in x, but ModularC-GCD computes the monic gcd(A, B). Since lc(g) is a random polynomial, its inverse in R will likely have very large rational coefficients, and so additional primes have to be used to recover the monic gcd. This brings us to an important advantage of our algorithm: it is output-sensitive. In Table 2 below g is a monic degree 4 polynomial with a and b still of degree 6 and 5. You'll notice that our algorithm finishes much faster than the earlier computation, while `RegularGcd` takes about the same amount of time. This happens because the coefficients of subresultants of A and B are always large no matter how small the coefficients of gcd(A, B) are.

extension		ModularC-GCD			`RegularGcd`		
n	degrees	time	divide	#primes	time real	cpu	#terms
1	[4]	0.013	0.006	3	0.064	0.064	170
2	[2, 2]	0.029	0.022	3	0.241	0.346	720
2	[3, 3]	0.184	0.138	17	1.73	4.433	2645
3	[2, 2, 2]	0.218	0.204	9	10.372	29.357	8640
2	[4, 4]	0.512	0.391	33	12.349	40.705	5780
4	[2, 2, 2, 2]	1.403	1.132	33	401.439	758.942	103680
3	[3, 3, 3]	2.755	1.893	65	413.54	1307.46	60835
3	[4, 2, 4]	1.695	1.233	33	39.327	86.088	19860
1	[64]	6.738	5.607	65	43.963	160.021	3470
2	[8, 8]	13.321	11.386	129	1437.76	5251.05	30420
3	[4, 4, 4]	17.065	14.093	129	7185.85	22591.4	196520

Table 1: The first column is the number of algebraic variables, the second is the degree of the extensions, the third is the CPU time it took to compute c-gcd of the inputs for ModularC-GCD, the fourth is the CPU time in ModularC-Gcd spent doing trial divisions over $\mathbb{Q}$, the fifth is the number of primes needed to recover g, the sixth is the real time it took for `RegularGcd` to do the same computation, the seventh is the total CPU time it took for `RegularGcd` and the last is the number of terms in the unnormalized gcd output by `RegularGcd`. All times are in seconds.

Let $d_a = \deg_x a$, $d_b = \deg_x b$ with $d_a \geq d_b$ and let $d_g = \deg_x g$. In Table 3 below we increased d_a and d_b from 6 and 5 in Table 1 to 9 and 8 leaving the degree of g at 4. By increasing d_b we increase

extension		ModularC-GCD			`RegularGcd`		
n	degrees	time	divide	#primes	time real	cpu	#terms
1	[4]	0.01	0.006	2	0.065	0.065	170
2	[2, 2]	0.02	0.016	2	0.238	0.329	715
2	[3, 3]	0.048	0.041	2	1.771	4.412	2630
3	[2, 2, 2]	0.05	0.041	2	11.293	31.766	8465
2	[4, 4]	0.077	0.068	2	11.521	36.854	5750
4	[2, 2, 2, 2]	0.117	0.097	2	321.859	431.368	99670
3	[3, 3, 3]	0.222	0.201	2	508.465	1615.28	57645
3	[4, 2, 4]	0.05	0.032	2	34.358	71.351	16230
1	[64]	0.304	0.282	2	27.55	98.354	3450
2	[8, 8]	0.482	0.455	2	1628.7	5979.51	29505
3	[4, 4, 4]	0.525	0.477	2	2989.18	4751.04	192825

The columns are the same as for Table 1

the number of steps in the Euclidean algorithm which causes an expression swell in `RegularGcd` in the size of the integer coefficients and the degree of each $z_1, \ldots, z_n$, that is, the expression swell is $(n+1)$ dimensional. The number of multiplications in R that the monic Euclidean algorithm does is at most $(d_a - d_b + 2)(d_g + d_b)$ for the first division and $\sum_{i=d_g}^{d_g+d_b-1} 2i = d_b(d_b + 2d_g - 1)$ for the remaining divisions. The trial divisions of A by g and B by g cost $d_a d_g$ and $d_b d_g$ multiplications in R respectively. Increasing d_a, d_b, d_g from $6, 5, 4$ in Table 1 to $9, 8, 4$ increases the number of multiplications in R in the monic Euclidean algorithm from 87 to 156 and from $24 + 20 = 44$ to $36 + 32 = 68$ for the trial divisions but the monic gcd remains unchanged. Comparing Table 1 and Table 3 the reader can see that the increase in ModularC-GCD is less than a factor of 2.

extension		ModularC-GCD			`RegularGcd`		
n	degrees	time	divide	#primes	time real	cpu	#terms
1	[4]	0.021	0.011	5	0.124	0.13	260
2	[2, 2]	0.043	0.031	5	0.968	1.912	1620
2	[3, 3]	0.214	0.163	17	10.517	34.513	6125
3	[2, 2, 2]	0.287	0.204	9	64.997	173.53	29160
2	[4, 4]	0.638	0.427	33	67.413	245.789	13520
4	[2, 2, 2, 2]	2.05	1.613	33	2725.13	3528.41	524880
3	[3, 3, 3]	3.35	2.731	33	3704.61	11924.0	214375
3	[4, 2, 4]	2.399	1.793	33	334.201	869.116	68940
1	[64]	10.097	8.584	65	171.726	658.518	5360
2	[8, 8]	21.890	18.086	129	10418.4	38554.9	72000
3	[4, 4, 4]	37.007	31.369	129	> 50000	–	–

The columns are the same as for Table 1

References

[1] John Abbott. Fault-tolerant modular reconstruction of rational numbers. Journal of Symbolic Computation Volume 80, pages $707 - 718$. May-June 2017.

[2] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. J. Symb. Comp., **28**: 105 – 124, 1999.

[3] S. Bosch. Algebraic Geometry and Commutative Algebra. Springer-Verlag London. 2013.

[4] D. Cox, J. Little, D. O'Shea. Ideals, Varieties and Algorithms. Springer-Verlag, 1991.

[5] X. Dahan, M. Moreno Maza, . Schost, W. Wu and Y. Xie. Lifting Techniques for Triangular Decompositions. Proceedings of ISSAC'05, Beijing, China, ACM Press, 2005.

[6] M. J. Encarnacion. Computing GCDs of Polynomials over Algebraic Number Fields, J. Symb. Comp. **20**: 299 – 313, 1995.

[7] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed., Cambridge University Press, 2013.

[8] K. O. Geddes, S. R. Czapor, and G. Labahn. Algorithms for Computer Algebra. Kluwer, 1992.

[9] Mark van Hoeij and Michael Monagan, A modular GCD algorithm over number fields presented with multiple extensions. Proceedings of ISSAC 02, ACM Press, pp. 109 – 116. 2002.

[10] E. Hubert. Notes on Triangular Sets and Triangulation-Decomposition Algorithms I: Polynomial Systems. In Symbolic and Numerical Scientific Computing edited by F. Winkler and U. Langer. Lecture Notes in Computer Science 2630, pp. 1 – 39. 2003.

[11] L. Langemyr, S. McCallum. The Computation of Polynomial GCDs over an Algebraic Number Field, J. Symbolic Computation 8, pp. 429 – 448. 1989.

[12] Xin Li, Marc Moreno Maza, and Wei Pan. Computations Modulo Regular Chains. Proceedings of ISSAC '09, pp. 239–246, 2009. See also https://arxiv.org/pdf/0903.3690 .

[13] M. B. Monagan. Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. *Proceedings of ISSAC '2004*, ACM Press, pp. 243–249, 2004.

[14] Paul S. Wang, M.J.T. Guy, and J.H. Davenport. P-adic reconstruction of rational numbers. ACM SIGSAM Bulletin **16**(2): 2–3, 1982.

[15] P.J. Weinberger and L.P. Rothschild. Factoring Polynomials over Algebraic Number Fields. *ACM Trans. on Math. Soft.* **2**(4): 335–350, 1976.