Resolving zero divisors using Hensel lifting.

John Kluesner
Department of Mathematics
Simon Fraser University
Burnaby, Canada
Email: jkluesner4112@gmail.com

Michael Monagan
Department of Mathematics
Simon Fraser University
Burnaby, Canada
Email: mmonagan@sfu.ca

***Abstract*—Algorithms which compute modulo triangular sets must respect zero divisors. We present Hensel lifting as a tool for resolving them. We give an application: a modular algorithm for computing gcds of univariate polynomials with coefficients modulo a radical triangular set over the rational numbers. We have implemented our algorithm using Maple's RECDEN package. We compare our implementation with the procedure RegularGcd in the RegularChains package.**

***Keywords*-Modular algorithms, Triangular sets, Radical ideals, Hensel lifting, GCD.**

I. Introduction

Suppose that we seek to find the greatest common divisor g 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 [12] and improved by Encarnacion [7]. Their solution first found a primitive element and then applied an algorithm for one extension. Monagan and van Hoeij [10] improved the multiple extension case by circumventing the primitive element.

The computational model for an algebraic number field is the quotient ring $\mathbb{Q}[z_1, \ldots, z_n]/T$ 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, requested by Daniel Lazard at ISSAC 2002, is to consider the same problem when each t_i is possibly reducible in which case $\mathbb{Q}[z_1, \ldots, z_n]/T$ may contain 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 $T = \{z_1^2+1, z_2^2+1\}$. Notice that $z_1 - z_2$ and $z_1 + z_2$ are zero-divisors modulo T. Consider computing the gcd of $a = x^4 + (z_1 + 18z_2)x^3 + (3z_1 - z_2)x^2 + 324x + 323$, and $b = x^3 + (z_1 + 18z_2)x^2 + (-19z_2 + 2z_1)x + 324$ 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, the division $b \div r_1$ can be performed giving the remainder

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

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. A correct approach would be to split T into triangular sets $\{z_1^2+1, z_2 - z_1\}$ and $\{z_1^2+1, z_2 + z_1\}$ using the factorization $z_2^2 + 1 = (z_2 - z_1)(z_2 + z_1) \pmod{z_1^2 + 1}$. 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 [11] and regular-gcd in [13]. We follow this trend with our definition componentwise-gcd in section 4.

Now, consider trying to compute $\gcd(a, b)$ using a modular 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 [18], [16]. 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 lifts a zero-divisor modulo a prime 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 $T \bmod p$ 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 [11].

In section 3, we present how to use Hensel lifting to resolve zero-divisors. 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 follows the Hensel construction, but the presence of zero-divisors demands a careful implementation.

In section 4, we present an application of 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 gcd algorithm. Our algorithm is best seen as a generalization of Monagan and van Hoeij's modular gcd algorithm over number fields [10]. We give pseudo-code for the modular gcd algorithm and all necessary sub-procedures.

In section 5 we give the complexity of our algorithm. We include a new practical method for multiplying in R. In section 6, 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 [13].

II. Triangular Sets

A. Notation and Definitions

We begin with some notation. All computations will be done in the ring $k[z_1, \ldots, z_n]$ where k is a field. We will use the ordering $z_i < z_{i+1}$. 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 $\text{mdeg}(f) = \deg_{\text{mvar}(f)}(f)$.

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 $k[z_1, \ldots, z_n]$ satisfying

(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 set $\{z_1^3 + 4z_1, z_2^2 + (z_1+1)z_2 + 4\}$ is 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 $\deg_{z_1}(t_2) = 4 > \text{mdeg}(t_1)$.

We will let $R = k[z_1, \ldots, z_n]/T$ throughout this paper. Since R is a finite-dimensional k-algebra, all nonzero elements are either zero-divisors or units.

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 [5].

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

Example 2. It is possible for a monic polynomial to factor as two polynomials with zero-divisors as leading coefficients. Consider the triangular set $T = \{z_1^4 + 3z_1^2 + 2, z_2^3 - z_2\}$. There are some obvious zero-divisors modulo T which come from the factorizations $t_1 = (z_1^2+1)(z_1^2+2)$ and $t_2 = (z_2-1)(z_2+1)z_2$, but a not so obvious one is

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

Factoring $z_2^3 - z_2 = (z_2^2 - 1)z_2$ is nicer because it creates a splitting $\langle T \rangle = \langle t_1, z_2^2 - 1 \rangle \cap \langle t_1, z_2 \rangle$ of triangular sets where the factorization in (1) would not. This greatly enhances the complexity of handling zero-divisors. Equation (1) also shows that the degree formula for the product of two polynomials does not hold.

Example 3. Another difficulty is that denominators in a monic factor f of $a \in R[x]$ may not appear in the denominator of a. For instance, let $T = \{z_1^2 - 5\}$. Then, $x^2 + x - 1 = (x - \frac{1}{2}z_1 + \frac{1}{2})(x + \frac{1}{2}z_1 + \frac{1}{2})$. The source of the denominator of f is the defect d of R. It is known that the discriminant Δ of t_1 is a multiple of d, usually, much larger than d, see [7]. Thus we could try to recover Δf with Chinese remaindering then make this result monic. Instead we use rational number reconstruction which circumvents this difficulty.

For clarity and conciseness, it's important to explicitly state that $g = \gcd(a, b)$ if (i) $g \mid a$ and $g \mid b$, and (ii) any common divisor of a and b divides g.

B. 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 triangular sets. One could prove this by using the associated primes of T as done in Proposition 4.7 of [11]. The structure theorem gives many powerful corollaries.

Theorem 1. Let $T \subset k[z_1, \ldots, z_n]$ 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 radical.

Corollary 2. Let $T \subset k[z_1, \ldots, z_n]$ be a radical 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 since being a gcd is an invariant of an isomorphism. ∎

Corollary 3 (Extended Euclidean Representation)**.** Let $T \subset k[z_1, \ldots, z_n]$ be a radical triangular set and $R = k[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$ with $g = \gcd(a, b)$. Then, there exist $A, B \in R[x]$ such that $aA + bB = g$.

Proof: Note that $R[x] \cong \prod F_i[x]$ where F_i is a field. 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 3 works even if running the EA on a and b encounters a zero-divisor. This shows it's more powerful than the extended Euclidean algorithm.

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 $T \bmod p \subset \mathbb{Z}_p[z_1, \ldots, z_n]$ remains radical.

Example 4. The triangular set $\{z_1^2 - 3\}$ is radical over $\mathbb{Q}$. All primes besides 2 and 3 are radical since the discriminant of $z_1^2 - 3$ is 12.

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 [6]. We use the following lemma which is a restatement of Corollary 7.3 of [11]. It also serves as the main idea of our algorithm for testing if a prime is radical; see IsRadicalPrime below.

Lemma 4. Let $T \subset k[z_1, \ldots, z_n]$ be a triangular set. Then T is radical if and only if $\gcd(t_i, t_i') = 1 \pmod{T_{i-1}}$ for all i.

Theorem 5. Let $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ be a radical triangular set. All but finitely many primes are radical primes.

Proof: By Lemma 4, $\gcd(t_i, t_i') = 1$. By the extended Euclidean representation (Corollary 3), there exist polynomials $A_i, B_i \in (\mathbb{Q}[z_1, \ldots, z_{i-1}]/T_{i-1})[z_i]$ where $A_it_i + B_it_i' = 1 \pmod{T_{i-1}}$. 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 $A_it_i + B_it_i' = 1 \pmod{T_{i-1}, p}$. This implies 1 is a $\gcd(t_i, t_i') \bmod p$ and so T remains radical modulo p by Lemma 4. 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 4 which relies on a gcd computation modulo p, which, if computed by the EA, may encounter a zero-divisor. If this happens we output the zero-divisor.

Algorithm 1: IsRadicalPrime

```
Input : A radical triangular set $T$ of $\mathbb{Q}[z_1, \ldots, z_n]$
        and a prime number $p$ where $p \nmid \mathrm{den}(t_i)$ for
        $t_i \in T$.
Output: A boolean indicating if $T$ remains radical
        modulo $p$, or a zero-divisor.
1 for $i = 1, \ldots, n$ do
2     $dt := \frac{\partial}{\partial z_i} T[i]$;
3     $g := \gcd(T[i], dt)$ over $\mathbb{Z}_p[z_1, \ldots, z_i]/T_{i-1}$;
4     if $g =$ ["zerodivisor", $u$] then
5         return ["zerodivisor", $u$]
6     else if $g \neq 1$ then return False;
7 end
8 return True;
```

III. 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 [9] from $k[x]$ to $R[x]$. The proof is the same since a, b are monic.

Lemma 6. Let $T \subset \mathbb{Z}_p[z_1, \ldots, z_n]$ be a radical triangular set and $R = \mathbb{Z}_p[z_1, \ldots, z_n]/T$. Let $a, b \in R[x]$ be nonzero and monic. Then, there exist unique $\sigma, \tau \in R[x]$ such that $a\sigma + b\tau = c$, $\deg(\sigma) < \deg(b)$ for any $c \in R[x]$.

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 $t_n = uv \pmod{T_{n-1}}$ by the division algorithm.

The next proposition shows that lifting is possible. It can be proven using the Hensel construction as described in chapter 6 of [9].

Proposition 7. Let $T \subset \mathbb{Q}[z_1, \ldots, z_n]$ be a radical triangular set and p a radical prime. Suppose $t_n \equiv u_0v_0 \pmod{T_{n-1}, p}$ where u_0 and v_0 are monic. Then, there exist unique monic $u_j, v_j \in \mathbb{Z}_{p^j}[z_1, \ldots, z_n]$ such that $t_n \equiv u_jv_j \pmod{T_{n-1}, p^j}$ where $u_j \equiv u_0 \mod \pmod{T_{n-1}, p}$ and $v_j \equiv v_0 \mod \pmod{T_{n-1}, p}$ for all $j \geq 0$.

The algorithm HenselLift is a formal presentation of the Hensel construction. The input is $f \in R[x]$ and $u_0, v_0 \in R/\langle p\rangle[x]$ 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 failure. A crucial part of the Hensel construction is solving the diophantine equation $\sigma u_0 + \tau v_0 = c$ in $R/\langle p\rangle[x]$. This is done using the extended Euclidean algorithm (EEA) and Lemma 6. It's possible that a zero-divisor is encountered in this process and we account for it in lines 2-4.

Algorithm 2: HenselLift

```
Input : A radical triangular set $T$ over $\mathbb{Q}$, a radical
        prime $p$, $f \in R[x]$, $a_0, b_0$ over $\mathbb{Z}_p$, and a
        bound $B$. Assume $f \equiv a_0 b_0 \pmod{p}$ and
        $\gcd(a_0, b_0) = 1$.
Output: Either $a, b \in R[x]$ 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.
1  Solve $sa_0 + tb_0 = 1$ using EEA for $s, t$ over $\mathbb{Z}_p$;
2  if a zero-divisor $w$ is encountered then
3      return ["zerodivisor", $w$];
4  end
5  Initialize $u := a_0, v := b_0$ as polynomials in $R$;
6  for $i = 1, 2, \ldots$ do
7      Apply rational reconstruction mod $p^i$ to $u$;
8      if RR succeeded with output $a$ then
9          if $a \mid f$ then return $a$, $f/a$;
10     end
11     if $p^i > 2B$ then return FAIL;
12     Compute $e := f - uv$ in $R[x]$;
13     Set $c := (e/p^i) \mod p$ ;
14     Solve $\sigma a_0 + \tau b_0 = c$ for $\sigma, \tau \in R/\langle p\rangle[x]$ using
         $sa_0 + tb_0 = 1$;
15     Lift $\sigma$ and $\tau$ to $R$;
16     Update $u := u + \tau p^i$ and $v := v + \sigma p^i$;
17 end
```

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 [8]). For the case $n = 1$ Weinberger and Rothschild [19] 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 factor 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 reconstruction.

The prime application of Hensel lifting will be to resolve zero-divisors. This is the goal of the algorithm HandleZeroDivisorHensel. It assumes a zero-divisor has been encountered mod a prime p 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, it recursively calls itself. If the Hensel lifting fails because a bound is reached, it outputs FAIL so the algorithm using it picks 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.

Algorithm 3: HandleZeroDivisorHensel

```
Input : A radical triangular set $T$ of $\mathbb{Z}_p[z_1, \ldots, z_n]$
        and a zero-divisor $u_0$ modulo $T$ with
        $\text{mvar}(u) = z_n$.
Output: FAIL or a factorization of some $t_k \in T$.
1  Set $v_0 :=$ Quotient($t_n, u_0$) $\pmod{T_{n-1}, p}$;
2  if $v_0 =$ ["zerodivisor", $w$] then
3      return HandleZeroDivisorHensel($w$);
4  end
5  if $B$ is unassigned then
6      set $B := 2^{60}$ as a global variable;
7  else set $B := B^2$;
8  Set $u, v :=$ HenselLift($t_n, u_0, v_0, B$);
9  if $u =$ ["zerodivisor", $w$] then
10     return HandleZeroDivisorHensel($w$)
11 else if $u =$ FAIL then return FAIL;
12 else return the factorization $f = uv \pmod{T}$;
```

We'd like to make it clear that this is not the first case of using p-adic lifting techniques on triangular sets. Lifting the triangular decomposition of a regular chain has been used by Dahan, Maza, Schost, Wu, Xie in [6].

IV. 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, we just return both gcds along with their associated triangular sets and refer to the output as a component-wise gcd, or c-gcd. This approach is similar to Hubert's in [11] which she calls a pseudo-gcd.

Definition. Let R be a commutative ring with unity such that $R \cong \prod_{i=1}^{r} R_i$ and $\pi_i \colon R \to R_i$ be the natural projections. A *component-wise gcd* of $a, b \in R[x]$ is a tuple $(g_1, \ldots, g_r) \prod_{i=1}^{r} R_i$ 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 $R = \mathbb{Q}[z_1, \ldots, z_n]/T$

with T a radical triangular set. As with all modular gcd algorithms, it's possible that a prime is bad or unlucky, but we prove there are only finitely many such primes.

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 $g = \text{c-gcd}(a, b)$. A prime number p is *unlucky* if $g \not\equiv \text{c-gcd}(a, b) \pmod{T, p}$. 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 mod p.

Theorem 8. 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 $g = \text{c-gcd}(a, b)$. Only finitely many primes are unlucky.

Proof: Let $R[x] \cong \prod R_i[x]$ where $g \mapsto (g_i)$ so $g_i = \gcd(a_i, b_i)$ over R_i. Let $a \mapsto (a_i)$ and $b \mapsto (b_i)$. Suppose $g_i = \gcd(a_i, b_i)$ is monic without loss of generality. Let $\overline{a}_i$ and $\overline{b}_i$ satisfy $a_i = g_i\overline{a}_i$ 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 $fg_iq = g_i$ and so $(fq - 1)g_i = 0$. 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 3), there exists $A_i, B_i \in R_i[x]$ where $\overline{a}_iA_i + \overline{b}_iB_i = 1$.

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 may reduce

$$\overline{a}_iA_i + \overline{b}_iB_i = 1 \pmod{p}, \tag{2}$$

$$a_i = g_i\overline{a}_i \pmod{p}, \qquad b_i = g_i\overline{b}_i \pmod{p}. \tag{3}$$

We will now show that $g_i = \gcd(a_i, b_i) \pmod{p}$. By (3), 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 (2) through by g_i gives $a_iA_i + b_iB_i = g_i \pmod{p}$. 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. ∎

The crux of ModularC-GCD is an algorithm to compute $\gcd(a, b)$ for $a, b \in \mathbb{Z}_p[z_1, \ldots, z_n]/T[x]$. We'll be using MonicEuclideanC-GCD, 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 an extended version of EuclideanC-GCD(u, t_k) modulo T_{k-1} where $\text{mvar}(u) = z_k$. 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. Second, Lemma 10 justifies the use of the modular gcd heuristics used in lines 21-25 by proving a uniqueness criterion for monic gcds. After that, we prove Lemma 11, a statement about the

Algorithm 4: ModularC-GCD

```
   Input : A radical triangular set $T$ over $\mathbb{Q}$ and
           $a, b \in R[x]$ where $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ and
           $\deg(a) \geq \deg(b) \geq 0$.
   Output: c-gcd$(a, b)$
1  Initialize $dg := \deg(b)$, $M = 1$;
2  Main Loop: Pick a prime $p$ that isn't bad;
3  $N :=$ isRadicalPrime$(T, p)$;
4  if $N =$ ["zerodivisor", $u$] then
5      $K :=$ HandleZeroDivisorHensel$(u)$;
6      if $K =$ FAIL then go to Main Loop;
7      else if $K$ gives $t_k = wv \pmod{T_{k-1}}$ then
8          Create triangular sets $T^w$ and $T^v$ where $t_k$ is
             replaced by $w$ and $v$;
9          return ModularC-GCD$(a, b) \pmod{T^w}$,
           ModularC-GCD$(a, b) \pmod{T^v}$;
10     end
11 else if $N =$ False then go to Main Loop;
12 Set $g := \gcd(a, b) \pmod{T, p}$;
13 if $g =$ ["zerodivisor", $u$] then
14     $K :=$ HandleZeroDivisorHensel$(u)$;
15     if $K =$ FAIL then go to Main Loop;
16     else if $K$ gives $t_k = wv \pmod{T_{k-1}}$ then
17         Create triangular sets $T^w$ and $T^v$ where $t_k$ is
             replaced by $w$ and $v$;
18         return ModularC-GCD$(a, b) \pmod{T^w}$,
           ModularC-GCD$(a, b) \pmod{T^v}$;
19     end
20 else
21     if $\deg(g) = dg$ then
22         Use CRT to combine $g$ with other gcds (if any)
             into $G$ and set $M := M \times p$;
23     else if $\deg(g) < dg$ then
24         Set $G := g$, $dg = \deg(g)$, and $M := p$;
25     else if $\deg(g) > dg$ then go to Main Loop;
26     $h :=$ RationalReconstruction$(G \pmod{M})$;
27     if $h \neq$ FAIL and $h \mid a$ and $h \mid b$ then
28         return $h$
29     end
30     Go to Main Loop;
31 end
```

Algorithm 5: MonicEuclideanC-GCD

```
Input  : A radical triangular set $T$ over $\mathbb{Z}_p$ and
         $a, b \in R[x]$ where $R = k[z_1, \dots, z_n]/T$ and
         $\deg(a) \geq \deg(b) \geq 0$.
Output: Either $\gcd(a, b)$ or a zero-divisor.
1 Initialize $r_0 := a$, $r_1 := b$ and $i := 1$;
2 while $r_i \neq 0$ do
3     Compute $s := \mathrm{lc}(r_i)^{-1} \pmod{T_{n-1}}$;
4     if $s =$["zerodivisor", $u$] then
5         return ["zerodivisor", $u$]
6     else $r_i := s \times r_i$;
7     Set $r_{i+1} :=$ Rem($r_{i-1}$, $r_i$) and $i = i + 1$;
8 end
9 return $r_{i-1}$
```

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 of localization, the formal process of introducing denominators in a ring; see chapter 1 of 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 $R = \mathbb{Z}[z_1, \dots, z_n]/T$, it's required that any prime dividing any den(t_i) must be included in S for R_S to be a ring. We will also use the iterated resultant:

The iterated resultant of $f \in R$ with respect to T is

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

One important property is that if $f, T \in R'[x] \subset R[x]$ where R' is a subring, then there exist $A, B_1, \dots, B_n \in R'[x]$ where $Af + B_1 t_1 + \dots + B_n t_n = \text{iterres}(f, T)$. This follows from the same proof as given in Theorem 7.1 of [9].

Proposition 9. Let $R = \mathbb{Q}[z_1, \dots, z_n]/T$ where T is a radical triangular set and $a, b \in R[x]$. A finite number of zero-divisors are encountered in ModularC-GCD(a, b).

Proof: We use induction on the degree of the extension $\delta = d_1 d_2 \cdots d_n$ where $d_i = \text{mdeg}(t_i)$. If $\delta = 1$, then $R \cong \mathbb{Q}$ 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 EA 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 8, this happens for only a finite amount of primes. Thus, a finite number of zero-divisors are encountered. ■

Lemma 10. Let $R = k[z_1, \dots, z_n]/T$ where T is a radical triangular set and let $a, b \in R[x]$. If $g = \gcd(a, b)$ is monic, g is the unique monic $\gcd(a, b)$.

Proof: Let h be a $\gcd(a, b)$. Then, $h \mid g$ and $g \mid h$ so there exist $u, v \in R[x]$ where $hu = g$ and $gv = h$. This gives $g(uv - 1) = 0$, but g can not be a zero-divisor as it is monic. Therefore, v is a unit and so $\deg_x(v) = 0$ because T is radical. Thus, $\deg_x(h) = \deg_x(g)$. In particular, if h were monic, then $v = 1$ and so $g = h$. ■

Lemma 11. Let T be a radical triangular set over $\mathbb{Q}$ and $R = \mathbb{Q}[z_1, \dots, z_n]/T$. Let $F = \mathbb{Z}[z_1, \dots, z_n]$. Suppose $f, u \in R[x]$ are monic such that $u \mid f$. Let

$$S = \{\text{prime numbers } p \in \mathbb{Z} : p \text{ isn't radical or } p \mid \text{den}(f)\}.$$

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 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 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 [7] for instance), den(u_i) consists of primes dividing $\Delta(a_i)$ or 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 particular, $u_i \in F_S[x]$.

The last concern is if combining $(u_1, \dots, u_s) \mapsto u$ 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 $r = \text{res}_{z_1}(a_1, a_2)$. Note that there are $A, B \in F_S$ where $Aa_1 + Ba_2 = r$. Any prime $p \mid r$ forces $\gcd(a_1, a_2) \neq 1 \pmod{p}$ and so t_1 wouldn't be square-free; this shows $p \in S$ and $\frac{A}{r}, \frac{B}{r} \in F_S$. Let $v = (\frac{A}{r})a_1 u_2 + (\frac{B}{r})a_2 u_1$. Note that $v \mod a_1 = u_1$ and v

$\bmod a_2 = u_2$. Since the CRT gives an isomorphism, $u = v$ and indeed $u \in F_S[x]$. This completes the base case.

For the inductive step, decompose T into its triangular decomposition in 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 $\prod_i \mathbb{Q}[z_1]/a_i$ with $a_i \in F_S$.
2) We can factor the image $t_2^{(i)}$ of t_2 over $\mathbb{Q}[z_1]/a_i$ into $t_2^{(i)} = b_1^{(i)} b_2^{(i)} \cdots b_{s_2}^{(i)}$. 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 den(a_i).
3) By the induction hypothesis, any prime p dividing den$(b_j^{(i)})$ is either not a radical prime of the triangular set $\{a_i\}$ or comes from den$(t_2^{(i)})$. If $\{a_i\}$ is not radical modulo p, then neither is $\{t_1\}$.
4) This decomposes $\mathbb{Q}[z_1, z_2]/T_2$ into a product of fields $\prod_{i,j} \mathbb{Q}[z_1, z_2]/\langle a_i, b_j^{(i)} \rangle$ where $a_i, b_j^{(i)} \in F_S$.
5) Repeat to decompose $\mathbb{Q}[z_1, \ldots, z_n]/T$ into a product of fields $\prod \mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ where $T^{(i)} \subset F_S$ using the induction hypothesis.

Let $f^{(i)} = f \bmod T^{(i)}$ and similarly $u^{(i)} = u \bmod T^{(i)}$. By the construction above, these reductions only introduce primes in S into denominators.

Note that $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ is an algebraic number field. Using the theorem of the primitive element, one can write $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)} = \mathbb{Q}(\alpha)$ for some $\alpha \in \mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$. Further, we may assume $\alpha = \lambda_1 z_1 + \cdots + \lambda_n z_n$ for integers λ_i using the same proof as in page 119 of Theorem 5.4.1 in [4] (since the only property used of the base field was that it is infinite). Next, we claim the minimal polynomial $m_{\alpha,\mathbb{Q}}(z)$ can be computed as $m(z) :=$ iterres$(z - \alpha, T^{(i)})$. To verify, note that $m(\alpha) = 0$ and $\deg_z($iterres$(z - \alpha, T^{(i)}))$ is bounded above by the degree of $T^{(i)}$. The fields $\mathbb{Q}(\alpha)$ and $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ are equal and so have the same degree over $\mathbb{Q}$ which implies $\deg(m_{\alpha,\mathbb{Q}})$ equals the degree of $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$. Because both are monic, uniqueness of the minimal polynomial gives $m(z) = m_{\alpha,\mathbb{Q}}(z)$. This result was used in the single extension case by Trager in [17]. Also, iterres$(z - \alpha, T^{(i)})$ is computed without introducing any prime into denominators other than those in the defining polynomials of $T^{(i)}$. We now prove that there are mappings $\mathbb{Q}(\alpha) \longleftrightarrow \mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}$ that only introduce primes that are contained in S. First, $\alpha \mapsto \lambda_1 z_1 + \cdots + \lambda_n z_n$ gives one such map. Conversely, note that $t_1^{(i)}$ certainly has a root in $\mathbb{Q}(\alpha)$. This implies $(z_1 - b_1(\alpha)) \mid t_1^{(i)}$. In particular, the base case guarantees $z_1 - b_1(\alpha)$ only has primes in the denominator that are contained in S. We will use $z_1 \mapsto b_1(\alpha)$. For $t_2^{(i)}$, first substitute $z_1 = b_1(\alpha)$ and then use the same idea as with $t_1^{(i)}$ to get a map $z_2 \mapsto b_2(\alpha)$. Repeat to get maps for each z_i that is a function of α and does not introduce any primes into denominators besides those in S. This shows that only primes in S are introduced into the denominator of $u^{(i)}$ when being mapped between the rings $\mathbb{Q}(\alpha)[x]$ and $\mathbb{Q}[z_1, \ldots, z_n]/T^{(i)}[x]$. Now, the proof for the field case in the base case shows that $u^{(i)} \in F/T^{(i)}[x]$.

Of course den$(u^{(i)}) \neq$ 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 den(f) or nonradical. This will follow from using iterated resultants similarly to the resultants in the base case noting that any prime dividing $r =$ iterres(res$(t_n^{(i)}, t_n^{(j)}), T_{n-1}^{(i)})$ is in S, and $r \equiv A t_n^{(i)} + B t_n^{(j)} \pmod{T_{n-1}^{(i)}}$. ■

Theorem 12. Let $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ where T is a radical triangular set and let $a, b \in R[x]$. The output of ModularC-GCD(a, b) is a c-gcd(a, 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 ModularC-GCD with T and let $g = \gcd(a, b) \pmod{T}$ over $\mathbb{Q}$.

First, we may assume b is monic. If lc$_x(b)$ is a unit, multiply through by it's inverse. This doesn't change $\gcd(a, b)$. If 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 27, it follows that $h \mid g$ and hence $\deg(h) \leq \deg(g)$ since h is monic. Suppose lc(g) is invertible. If so, make g monic without loss of generality. Let p be a prime used to compute h. Since g and b are monic and $g | b$, any prime appearing in den(g) is either nonradical or a divisor of den(b) by Lemma 11. 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 gcd of a and b.

Assume lc(g) is a zero-divisor and mvar(lc$(g)) = z_n$. Inspect lc$_{z_n}$(lc(g)); if this is a unit, make it monic. If it's a zero-divisor, inspect lc$_{z_{n-1}}$(lc$_{z_n}(g)$)). Continue until $u =$ lc$_{z_{k+1}}(\cdots($lc$_{z_n}($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 11 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 $g_u = g \bmod T^{(u)}$ and $g_v = g \bmod T^{(v)}$ 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 $\text{lc}_{z_{k+1}}(\cdots(\text{lc}_{z_n}(\text{lc}_x(g))\cdots) = 1$. Reinspect $w = \text{lc}_{z_{k+2}}(\cdots(\text{lc}_{z_n}(\text{lc}_x(g_v)))\cdots)$. 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 $\text{lc}_{z_{k+1}}(\cdots(\text{lc}_{z_n}(\text{lc}_x(g_u))\cdots)$ has changed; if it's invertible, multiply through by its inverse until a monic zero-divisor is found in the leading coefficient chain. We will wind up in the situation with a monic factorization of t_j that is reducible mod p.

This process must terminate with a splitting where the image of g is monic since $\text{lc}_x(g)$ has finite degree in each variable. We have already shown that the image of h would be a gcd in this case. ■

V. Complexity of Algorithm ModularC-GCD

Let $R = k[z_1, \ldots, z_n]/T$ where k is a field. Our gcd algorithm will do many multiplications in R. Let $a, b \in R$ be reduced modulo T and let $M(n)$ be the number of field multiplications (fmuls) in k to multiply $a \times b$ in R. The obvious approach is to first multiply in $k[z_1, \ldots, z_n]$ then reduce mod T. We describe the classical approach outlined by Li et. al. in [14]. Let $d_i = \text{mdeg}(t_i)$ and let $\delta = \prod_{i=1}^n d_i$ be the degree of T. First view a and b as polynomials in z_n with coefficients modulo T_{n-1}. Multiplying $a(z_n) \times b(z_n) \mod T_{n-1}$ involves recursively multiplying all of coefficients of $a(z_n)$ by those of $b(z_n)$ and reducing them modulo T_{n-1}. There are $\le d_n^2$ such products and we get a polynomial $c(z_n)$ of degree $\le 2(d_n - 1)$ in z_n. Next we divide $c(z_n)$ by $t_n(z_n)$. Using the high-school division algorithm this does $\le \deg c - d_n + 1$ iterations for a total of $d_n(d_n - 1)$ recursive multiplications modulo T_{n-1}. We have

$$M(n) \le (d_n^2 + d_n(d_n - 1))M(n-1), \text{ and } M(0) = 1.$$

Solving this recurrence we obtain $M(n) \in O(2^n\delta^2)$. The factor of 2^n is significant. For n quadratic extensions we have $M(n) \in O(\delta^3)$. In [14] Li et. al. developed an FFT based method for the case $k = \mathbb{Z}_p$. Their method is $M(n) \in O(4^n\delta\log(\delta)\log\log\delta)$. We modify the division of $c(z_n)$ by $t_n(z_n)$ modulo T_{n-1} to obtain a simple method which is faster for d_i up to about 16.

Theorem 13. Let $M(n)$ be the number of field multiplications in k for computing $a \times b \mod T$. Let $\delta_0 = 1, \delta_1 = d_1$, $\delta_2 = d_1 d_2$, ..., $\delta_n = d_1 d_2 \ldots d_n$. Then

$$M(n) \le \delta_n^2 + \sum_{k=1}^{n} \delta_k^2 \frac{d_k - 1}{d_k} \prod_{j=k+1}^{n} (2d_j - 1)$$

which is exact in the dense case. Moreover, $M(n) \le 3\delta^2$.

Proof: Let $D(n)$ be the number of fmuls needed to reduce a polynomial of degree $2(d_n-1)$ by T_n. It is assumed $D(n)$ works by first reducing by t_1 then by $t_2 \mod T_1$, etc. Multiplying ab takes δ_n^2 multiplications before reducing mod T. Thus $M(n) = \delta_n^2 + D(n)$. Let us divide $c = ab$ by $t_n \mod T_{n-1}$. Let $c = \sum_{i=0}^{2(d_n-1)} c_i z_n^i$ and $t_n = z_n^{d_n} + \sum_{i=0}^{d_n-1} p_i z_n^i$. We can compute the quotient $q = \sum_{i=0}^{d_n-2} q_i z_n^i$ and remainder $r = \sum_{i=0}^{d_n-1} r_i z_n^i$ via the linear system $r = c - t_n q$ as follows:

$$\begin{aligned}
q_{d_n-2} &= c_{2d_n-2}, \\
q_{d_n-3} &= c_{2d_n-3} - q_{d_n-2}p_{d_n-1}, \\
q_{d_n-4} &= c_{2d_n-4} - q_{d_n-3}p_{d_n-1} - q_{d_n-2}p_{d_n-2}, \\
&\vdots \\
q_0 &= c_{d_n} - q_1 p_{d_n-1} - \cdots - q_{d_n-2}p_2, \\
r_{d_n-1} &= c_{d_{n-1}} - q_0 p_{d_n-1} - q_1 d_{n-2} - \cdots - q_{d_n-2}p_1, \\
r_{d_n-2} &= c_{d_{n-2}} - q_0 p_{d_n-2} - q_1 d_{n-3} - \cdots - q_{d_n-3}p_0, \\
&\vdots \\
r_1 &= c_1 - q_0 p_1 - q_1 p_0, \\
r_0 &= c_0 - q_0 p_0.
\end{aligned}$$

The key idea is to compute the entire right-hand-side before reduction by T_{n-1}. This reduces the total number of reductions from quadratic in d_n to linear in d_n. The total cost of the reductions is $(2d_n - 1)D(n-1)$ fmuls. We note that the idea was used in [15] for reducing the number of integer divisions by m when multiplying two polynomials in $\mathbb{Z}_m[x]$.

Now multiplying each $q_i p_j$ takes δ_{n-1}^2 fmuls. There are $0 + 1 + 2 + \cdots + d_n - 2$ in the top $d_n - 1$ rows and $1 + 2 + \cdots + (d_n - 1) + (d_n - 1)$ in the bottom d_n rows for a total of $d_n(d_n - 1)$. Therefore

$$D(n) = (2d_n - 1)D(n-1) + d_n(d_n - 1)\delta_{n-1}^2.$$

If $n = 0$ it takes 0 multiplications to reduce so we have $D(0) = 0$. Solving the recurrence leads to the first result. To obtain $M(n) \le 3\delta^2$ we proceed by induction on n. For $n = 0$ we have $D(0) = 0 \le 2 = 2\delta_0$. For the induction step we have

$$\begin{aligned}
D(n) &= (2d_n - 1)D(n-1) + d_n(d_n - 1)\delta_{n-1}^2 \\
&\le (2d_n - 1)2\delta_{n-1}^2 + d_n(d_n - 1)\delta_{n-1}^2 \\
&= 4d_n\delta_{n-1}^2 - 2\delta_{n-1}^2 + d_n^2\delta_{n-1}^2 - d_n\delta_{n-1}^2 \\
&= (3d_n - 2)\delta_{n-1}^2 + \delta_n^2.
\end{aligned}$$

Note that $3d_n - 2 \le d_n^2$ which follows from $d_n^2 - 3d_n + 2 = (d_n - 2)(d_n - 1) \ge 0$ for all $d_n \ge 1$ thus $D(n) \le d_n^2\delta_{n-1}^2 + \delta_n^2 = 2\delta_n^2$. Finally $M(n) = \delta_n^2 + D(n) \le 3\delta_n^2 = 3\delta^2$. ■

Let $a, b \in R[x]$ and let p be a prime. Let $d_a = \deg_x a$, $d_b = \deg_x b$ with $d_a \ge d_b$ and let $d_g = \deg_x g$. The number of multiplications in R mod p that Algorithm MonicEuclideanC-GCD does is $\le (d_a - d_b + 2)(d_g + d_b)$ for the first division and $\le \sum_{i=d_g}^{d_g+d_b-1} 2i = d_b(d_b + 2d_g - 1)$

for the remaining divisions. Thus if Algorithm ModularC-GCD uses M primes of constant size the total cost of the monic Euclidean algorithm is $O(Md_ad_b)$ multiplications in R modulo p. The trial divisions $h|a$ and $h|b$ cost d_ad_g and d_bd_g multiplications in R respectively. Note, if the rational reconstruction algorithm in [16] is used, then with high probability $h = g$ so the trial divisions will be done once. Summarizing, the expected number of multiplications Algorithm ModularC-GCD does is $O(Md_ad_b\delta^2)$ in $\mathbb{Z}_p$ and $O(d_g(d_a + d_b)\delta^2)$ in $\mathbb{Q}$.

VI. Comparison with `RegularGcd`

We have implemented `ModularC-GCD` as presented above using Maple's RECDEN package, which uses a recursive dense data structure for polynomials with extensions; see [10] for details. The Maple code for our software as well as several examples with their output can be found 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 [13]) which is in the `RegularChains` package of Maple. `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` doesn't return reduced output. The command `NormalForm` is used to compute the reduced version. `ModularC-GCD` uses the CRT and RR 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 5. We'd like to illustrate the differences with an example provided by an anonymous referee of an earlier version of this paper. Let

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

When we run our algorithm to compute c-gcd(a, b), it returns

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

The same example using `RegularGcd` returns

$$\begin{aligned} 360z + 1800 &\quad (\bmod\ y+2, x),\\ 62208 &\quad (\bmod\ y-1, x),\\ z^2 + z &\quad (\bmod\ y, x-1),\\ 360z - 360 &\quad (\bmod\ y-2, x-1),\\ z^2 - 5z + 4 &\quad (\bmod\ y, x+1),\\ -360z + 720 &\quad (\bmod\ y+1, x+1) \end{aligned}$$

after using the `NormalForm` command. The outputs were not reduced prior to using `NormalForm`. In general, the output of our algorithm deals with smaller numbers. This is an advantage for the user. Note that `RegularGcd` may output a smaller decomposition than `ModularC-GCD` as well, it depends on the input.

Finally, we'd like to conclude with some timing tests which show the power of using a modular gcd algorithm that recovers the monic c-gcd 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.

extension	ModularC-GCD		`RegularGcd`	
degrees	time	#primes	time	#terms
[2, 2]	0.029	3	0.346	720
[3, 3]	0.184	17	4.433	2645
[2, 2, 2]	0.218	9	29.357	8640
[4, 4]	0.512	33	40.705	5780
[2, 2, 2, 2]	1.403	33	758.942	103680
[3, 3, 3]	2.755	65	1307.46	60835
[4, 2, 4]	1.695	33	86.088	19860
[64]	6.738	65	160.021	3470
[8, 8]	13.321	129	5251.05	30420
[4, 4, 4]	17.065	129	22591.4	196520

Table I
The first column is the degree of the extensions, the second is the CPU time it took to compute c-gcd of the inputs for ModularC-GCD, the third is the number of primes needed to recover g, the fourth is the CPU time it took for `RegularGcd` to do the same computation, the fifth is the number of terms in the unnormalized gcd output by `RegularGcd`. All times are in seconds.

In the previous dataset, g isn't created monic in x, but ModularC-GCD computes the monic $\gcd(A, B)$. Since $\mathrm{lc}(g)$ 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 II below g is a monic degree 4 polynomial with a and b still of degree 6 and 5. 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`	
degrees	time	#primes	time	#terms
[2, 2]	0.02	2	0.329	715
[3, 3]	0.048	2	4.412	2630
[2, 2, 2]	0.05	2	31.766	8465
[4, 4]	0.077	2	36.854	5750
[2, 2, 2, 2]	0.117	2	431.368	99670
[3, 3, 3]	0.222	2	1615.28	57645
[4, 2, 4]	0.05	2	71.351	16230
[64]	0.304	2	98.354	3450
[8, 8]	0.482	2	5979.51	29505
[4, 4, 4]	0.525	2	4751.04	192825

Table II
THE COLUMNS ARE THE SAME AS FOR TABLE I

Let $d_a = \deg_x a$, $d_b = \deg_x b$ and $d_g = \deg_x g$. In Table III below we increased d_a and d_b from 6 and 5 in Table I to 9 and 8 leaving the d_g at 4. By increasing d_b we increase 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. Increasing d_a, d_b, d_g from $6, 5, 4$ in Table I 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 I and Table III the reader can see that the increase in ModularC-GCD is less than a factor of 2.

extension	ModularC-GCD		`RegularGcd`	
degrees	time	#primes	time	#terms
[2, 2]	0.043	5	1.912	1620
[3, 3]	0.214	17	34.513	6125
[2, 2, 2]	0.287	9	173.53	29160
[4, 4]	0.638	33	245.789	13520
[2, 2, 2, 2]	2.05	33	3528.41	524880
[3, 3, 3]	3.35	33	11924.0	214375
[4, 2, 4]	2.399	33	869.116	68940
[64]	10.097	65	658.518	5360
[8, 8]	21.890	129	38554.9	72000
[4, 4, 4]	37.007	129	> 50000	–

Table III
THE COLUMNS ARE THE SAME AS FOR TABLE I

VII. CONCLUSION

A modular gcd algorithm is always preferable to a non-modular gcd algorithm that computes over $\mathbb{Q}$ or $\mathbb{Z}$. This is evident from our data. What we have shown is that it's possible to extend the modular algorithm of Monagan and van Hoeij from [10] to work over $R = \mathbb{Q}[z_1, \ldots, z_n]/T$ where R is not a field and the Euclidean algorithm may encounter a zero divisor. We use Hensel lifting to recover the zero-divisor over $\mathbb{Q}$ then split the triangular set T so that the modular algorithm can continue.

REFERENCES

[1] John Abbott. Fault-tolerant modular reconstruction of rational numbers. J. Symb. Comp., **80**: 707 – 718, 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. *Galois Theory*. Wiley-Interscience, 2004.

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

[6] X. Dahan, M. Moreno Maza, E. Schost, W. Wu and Y. Xie. Lifting Techniques for Triangular Decompositions. Proceedings of ISSAC'05, ACM Press, 108 – 115, 2005.

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

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

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

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

[11] E. Hubert. Notes on Triangular Sets and Triangulation-Decomposition Algorithms I: Polynomial Systems. Springer-Verlag LNCS **2630**, pp. 1 – 39. 2003.

[12] L. Langemyr, S. McCallum. The Computation of Polynomial GCDs over an Algebraic Number Field, J. Symb. Comp. **8**: 429 – 448, 1989.

[13] Xin Li, Marc Moreno Maza, and Wei Pan. Computations Modulo Regular Chains. Proceedings of ISSAC '09, pp. 239–246, 2009.

[14] Xin Li, Marc Moreno Maza, and Eric Schost. Fast Arithmetic for Triangular Sets: from Theory to Practice. *J. Symb. Comp.*, **44**(7): 891–907, 2009.

[15] M. B. Monagan. In-place arithmetic for polynomials over $\mathbf{Z}_n$. *Proceedings of DISCO '92*, Springer-Verlag LNCS, **721**, pp. 22–34, 1993.

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

[17] Barry Trager. Algebraic Factoring and Rational Function Integration. *Proceedings of SYMSAC '76*, ACM Press, 219–226, 1976.

[18] 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.

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