A Modular Algorithm for Computing Greatest Common Right Divisors of Ore Polynomials

Ziming Li

Mathematics-Mechanization Research Center
Institute of Systems Science
Beijing (100084), China
zmli@mmrc.iss.ac.cn
http://mmrc.iss.ac.cn/zmli

István Nemes

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria
Istvan.Nemes@risc.uni-linz.ac.at

Abstract

This paper presents a modular algorithm for computing the greatest common right divisor (gcrd) of two univariate Ore polynomials over $\mathbf{Z}[t]$. The subresultants of Ore polynomials are used to compute the evaluation homomorphic images of the gcrd. Rational number and rational function reconstructions are used to recover coefficients. The experimental results illustrate that the present algorithm is markedly superior to the Euclidean algorithm and the subresultant algorithm for Ore polynomials.

1 Introduction

Ore polynomials establish a general mathematical setting to describe linear operational polynomials, for example, linear differential, difference, and q-difference polynomials. Recent years have seen a rapid development of the algorithms for the manipulation of the functions that are annihilated by linear operational polynomials [1, 2, 12, 15]. This development motivates us to design an efficient algorithm for computing the gcrd of two Ore polynomials over $\mathbf{Z}[t]$. The gcrd-calculation plays an important role in the computation of linear operational polynomials. For instance, if L_1 and L_2 are two linear differential operators, then their gcrd corresponds to the intersection of the solution spaces of L_1 and L_2. To represent the sum of the two solution spaces, one needs the least common left multiple of L_1 and L_2, which is expressible as a determinant with entries being the derivatives of coefficients of L_1 and L_2, as long as the gcrd is obtained [8]. The greatest common *left* divisor of L_1 and L_2 can be obtained from the gcrd of their adjoint operators.

Non-modular gcrd algorithms such as: the Euclidean algorithm and subresultant algorithm, cause severe intermediate expression swell, as seen in the case for bivariate commutative polynomials. We will extend the techniques used in modular gcd algorithms as much as we can (see [3, 6]). Two new problems that cannot be tackled by the classical techniques, are that

- evaluation mappings are *not* Ore ring homomorphisms
- the normalization of leading coefficients is different from that in the algebraic modular algorithm.

The first problem will be solved by the subresultant theory for Ore polynomials; the second one by rational number and rational function reconstructions. To the authors' knowledge the present algorithm is the first modular algorithm for computing gcrds. The non-modular algorithms are the Euclidean algorithm [11] and the subresultant algorithm [8]. Grigor'ev [7] presents a method for computing the gcrds for several linear differential operators by Gaussian elimination.

This paper is organized as follows. In Section 2, we review some basic results from Ore polynomial rings and specify the notation that will be used later. The outline and detailed description of the modular method are given in Sections 3 and 4, respectively. Some experimental results are given in Section 5.

2 Preliminaries

This section has two parts: Section 2.1 concerns Ore polynomial rings and notation, Section 2.2 concerns modular and evaluation mappings. The reader is referred to [11, 1, 2] for more details about Ore rings.

2.1 Ore polynomial rings

Let $\mathcal{R}$ be a commutative domain and σ an injective endomorphism of $\mathcal{R}$, which is called *conjugate operator* by Ore. An endomorphism δ of the additive group $\mathcal{R}$ is called a *pseudo-derivation* with respect to σ if

$$\delta(ab) = \sigma(a)\delta(b) + \delta(a)b, \quad \text{for all } a, b \in \mathcal{R}. \tag{1}$$

The (non-commutative) multiplication in $\mathcal{R}[X]$ is defined by the commutation rule

$$Xa = \sigma(a)X + \delta(a), \quad \text{for all } a \in \mathcal{R}. \tag{2}$$

The triple $(\mathcal{R}[X], \sigma, \delta)$ is called an *Ore polynomial ring*. For $A, B \in \mathcal{R}[X]$, the product of A and B is denoted by AB and the degree of AB is equal to the sum of the degrees of A and B. The conjugate operator σ and pseudo-derivation δ can be uniquely extended to the quotient field of $\mathcal{R}$ by letting $\sigma(a/b) = \sigma(a)/\sigma(b)$ and $\delta(a/b) = (b\delta(a) - a\delta(b))/(\sigma(b)b)$, for $a, b \in \mathcal{R}$ with $b \neq 0$.

For $A, C \in \mathcal{R}[X]$, we say that C is a right factor of A if there exist non-zero elements $r \in \mathcal{R}$ and $B \in \mathcal{R}[X]$ such that $rA = BC$. A common right factor of A and B, with the highest degree, is called a *gcrd* of A and B.

Example 1 Let the identity mapping and differential operator $\frac{d}{dt}$ be the conjugate operator and pseudo-derivation on $\mathbf{Z}[t]$, respectively. Let X be an indeterminate over $\mathbf{Z}[t]$. Then the Ore polynomial ring $\mathbf{Z}[t][X]$ is $\mathbf{Z}[t]$-isomorphic to the ring of linear differential operators over $\mathbf{Z}[t]$. If

$$A = X^4 \quad \text{and} \quad B = (t^2 - t)X^3 - 3tX + 6,$$

then $(tX - 2)$ is a gcrd of A and B. Observe that

$$t^2 A = (tX^3 - X^2)(tX - 2) \text{ and } B = ((t-1)X^2 - 3)(tX - 2).$$

Hence the product of two primitive polynomials is not necessarily primitive, and there does not exist an Ore polynomial F in $\mathbf{Z}[t][X]$ such that $A = F(tX - 2)$.

Example 2 Let E be the endomorphism of $\mathbf{Z}[t]$ over $\mathbf{Z}$ that sends t to $(t + 1)$. Let E and the null mapping be the conjugate operator and pseudo-derivation on $\mathbf{Z}[t]$, respectively. Then the Ore polynomial ring $\mathbf{Z}[t][X]$ is $\mathbf{Z}[t]$-isomorphic to the ring of linear shift operators over $\mathbf{Z}[t]$. If

$$A = t(t+1)X^2 - 2t(t+2)X + (t+1)(t+2)$$

and

$$B = (t-1)X^2 - (3t-2)X + 2t,$$

then $tX - (t + 1)$ is the primitive gcrd of A and B w.r.t. X.

We shall work in Ore polynomial rings whose ground domains are univariate polynomial rings over the integers or over a finite field. Throughout the paper, p is a prime, and $\mathbf{Z}_p$ is the Galois field with p elements. For an indeterminate t, $\mathbf{Z}[t]$ and $\mathbf{Z}_p[t]$ are the rings of usual commutative polynomials in t over $\mathbf{Z}$ and $\mathbf{Z}_p$, respectively. Let X be a new indeterminate. We regard $\mathbf{Z}[t][X]$ ($\mathbf{Z}_p[t][X]$) as a left $\mathbf{Z}[t]$-module ($\mathbf{Z}_p[t]$-module). For non-zero F in $\mathbf{Z}[t][X]$ or $\mathbf{Z}_p[t][X]$, the leading coefficient of F in X is denoted by $\mathrm{lc}(F)$, the leading coefficient of $\mathrm{lc}(F)$ in t is called the *head coefficient* of F and denoted by $\mathrm{hc}(F)$, the degree of F in X is denoted by $\deg F$, and the degree of F in t by $\deg_t F$.

From now on, we assume that the triple $(\mathbf{Z}[t][X], \sigma, \delta)$ is an Ore polynomial ring over $\mathbf{Z}[t]$. For brevity we denote this triple by $\mathbf{Z}[t][X]$. If $A, B \in \mathbf{Z}[t][X]$, then the *normalized* gcrd of A and B is the gcrd of A and B that is in $\mathbf{Z}[t][X]$ and primitive with respect to X, and has positive head coefficient. If A and B are in the Ore polynomial ring $\mathbf{Z}_p[t][X]$, then the *normalized* gcrd of A and B is the gcrd of A and B that is in $\mathbf{Z}_p[t][X]$ and primitive with respect to X, and has head coefficient 1. The normalized gcrd of A and B, where A and B are either in $\mathbf{Z}[t][X]$ or $\mathbf{Z}_p[t][X]$, is denoted by $\mathrm{gcrd}(A, B)$.

2.2 Modular and evaluation mappings

A modular mapping $\phi_p : \mathbf{Z}[t][X] \longrightarrow \mathbf{Z}_p[t][X]$ is a left-module homomorphism (over $\mathbf{Z}[t]$) defined for p by

$$\phi_p(A) = A \bmod p, \quad \text{for } A \in \mathbf{Z}[t][X].$$

An evaluation mapping $\psi_k : \mathbf{Z}_p[t][X] \longrightarrow \mathbf{Z}_p[X]$ is a left-module homomorphism (over $\mathbf{Z}_p[t]$) defined for $k \in \mathbf{Z}_p$ by

$$\psi_k(A(t, X)) = A(k, X), \quad \text{for } A \in \mathbf{Z}_p[t][X].$$

Note that ϕ_p and ψ_k are also ring homomorphisms from $\mathbf{Z}[t]$ to $\mathbf{Z}_p[t]$ and from $\mathbf{Z}_p[t]$ to $\mathbf{Z}_p$, respectively. Thus, these two module homomorphisms are well defined. Unlike the usual commutative case it is not obvious that ϕ_p and ψ_k are Ore ring homomorphisms.

Lemma 2.1 If p is not a divisor of $\mathrm{hc}(\sigma(t))$, then the triple $(\mathbf{Z}_p[t][X], \sigma_p, \delta_p)$ is an Ore polynomial ring and ϕ_p is a ring homomorphism, where σ_p and δ_p are defined by the respective rules:

$$\sigma_p(\phi_p(f)) = \phi_p(\sigma(f)) \text{ and } \delta_p(\phi_p(f)) = \phi_p(\delta(f)), \ f \in \mathbf{Z}[t].$$

Proof The mappings σ_p and δ_p are well-defined because σ and δ are endomorphisms of the additive group $\mathbf{Z}[t]$.

First, we show that σ_p is a monomorphism of $\mathbf{Z}_p[t]$. Since $\sigma(m) = m$, for $m \in \mathbf{Z}$, we have $\deg_t \sigma(t) > 0$. Let

$$f = f_n t^n + f_{n-1} t^{n-1} + \cdots + f_0 \in \mathbf{Z}[t].$$

If $\sigma_p(\phi_p(f)) = 0$, then the definition of σ_p implies

$$\phi_p\left(f_n \sigma(t)^n + f_{n-1}\sigma(t)^{n-1} + \cdots + f_0\right) = 0.$$

Since $\phi_p(\mathrm{hc}(\sigma(t))) \neq 0$, $\phi_p(\sigma(t))$ is of positive degree in t. Hence, $\phi_p(f_i) = 0$ for $i = 0, \ldots, n$, thus, $\phi_p(f) = 0$.

Applying ϕ_p to (1), we see that δ_p is a pseudo-derivation with respect to σ_p, hence, that $(\mathbf{Z}_p[t][X], \sigma_p, \delta_p)$ is an Ore polynomial ring.

To show that ϕ_p is a ring homomorphism, we need only to show that $\phi_p(Xa) = X\phi_p(a)$, for $a \in \mathbf{Z}[t]$. Applying ϕ_p to (2) yields this assertion. □

Lemma 2.1 says that modular homomorphisms are ring homomorphisms except for a finite number of primes. What about evaluation homomorphisms ? The next lemma asserts that evaluation homomorphisms are usually *not* ring homomorphisms.

Lemma 2.2 If $(\mathbf{Z}_p[X], \sigma, \delta)$ is an Ore polynomial ring, then the multiplication in $(\mathbf{Z}_p[X], \sigma, \delta)$ is commutative.

Proof Note that $\mathbf{Z}_p$ is generated by 1 as an additive group. Hence σ is the identity mapping of $\mathbf{Z}_p$, and, moreover, δ is the null mapping of $\mathbf{Z}_p$ because $\delta(1) = 0$ by (1). Thus, the multiplicative rule (2) becomes $Xa = aX$, which defines the usual commutative multiplication. □

Let $\mathbf{Z}_p[t][X]$ be the ring of differential operators over $\mathbf{Z}_p[t]$ and k be in $\mathbf{Z}_p$. Then (2) implies $\psi_k(Xt) = kX + 1$. On the other hand, Lemma 2.2 implies $\psi_k(X)\psi_k(t) = kX$. Hence, ψ_k is not a ring homomorphism.

3 Outline of the modular method

Basically we have three algorithms:

- GCRD_m: This algorithm reduces the gcrd problem in $\mathbf{Z}[t][X]$ to a series of gcrd problems in $\mathbf{Z}_p[t][X]$, for several primes p, by applying modular homomorphisms.
- GCRD_p: This algorithm reduces the gcrd problem in $\mathbf{Z}_p[t][X]$ to a series of problems of finding evaluation homomorphic images of the monic associate of the sought-after gcrd.
- GCRD_e: This algorithm computes the evaluation homomorphic images of the monic associate of the gcrd of two given polynomials in $\mathbf{Z}_p[t][X]$.

The idea of GCRD_m is as follows. For $A, B \in \mathbf{Z}[t][X]$, we choose several "good" primes p successively, and invoke GCRD_p to compute the gcrd of $\phi_p(A)$ and $\phi_p(B)$ for these primes. Determine "lucky" gcrds and combine them by the Chinese remainder algorithm (CRA). Use rational number reconstruction to recover the rational coefficients of the combined image, say H. When two successive rational number

reconstructions yield the same result we attempt a trial-division of both A and B by H, using more primes if the division is not exact. We use rational number reconstruction because known bounds for the head coefficient of $\gcd(A, B)$ are loose. A similar situation also occurs in the gcd computation over algebraic number fields [5].

The idea of GCRD_p is similar. For given A_p and B_p in $\mathbf{Z}_p[t][X]$. We choose several "good" evaluation points k successively, and invoke GCRD_e to compute the monic associate of $\psi_k(\gcd(A_p, B_p))$. Notice that this monic associate is essentially different from the monic associate of $\gcd(\psi_k(A_p), \psi_k(B_p))$, because ψ_k is not a ring homomorphism by Lemma 2.2. The combining process consists of interpolation and rational function reconstruction. The termination of GCRD_p is again determined by a trial division.

To outline the idea of the algorithm GCRD_e, let us recall the usual commutative case. Assume that $\mathbf{Z}_p[t][X]$ is the usual commutative polynomial ring. Then the diagram

$$\begin{array}{ccccc} \mathbf{Z}_p[t][X] & \times & \mathbf{Z}_p[t][X] & \xrightarrow{\gcd} & \mathbf{Z}_p[t][X] \\ & \downarrow \psi_k & & & \downarrow \psi_k \\ \mathbf{Z}_p[X] & \times & \mathbf{Z}_p[X] & \xrightarrow{\gcd} & \mathbf{Z}_p[X] \end{array}$$

commutes unless k is a root of some polynomial (relative to A_p and B_p) in $\mathbf{Z}_p[t]$. The commutativity allows us to compute $\psi_k(\gcd(A_p, B_p))$ by Euclid's algorithm in $\mathbf{Z}_p[X]$.

If $\mathbf{Z}_p[t][X]$ is an Ore polynomial ring, then the foregoing diagram is usually not commutative when the mapping gcd is replaced by gcrd, because ψ_k is usually not a ring homomorphism. Thus, Euclid's algorithm in $\mathbf{Z}_p[X]$ will not produce what we desire.

To overcome this difficulty, we return to the commutative case. A careful observation of the classical method reveals that one may obtain $\psi_k(\gcd(A_p, B_p))$ without using the property that ψ_k is a ring homomorphism. The idea goes as follows. We form the Sylvester matrix M of A_p and B_p, apply ψ_k to the entries of M to get the matrix M_k, and then compute the rank of M_k using Gaussian elimination. Set $d = \deg A_p + \deg B_p - \operatorname{rank}(M_k)$. Then d is the degree of $\gcd(A_p, B_p)$ unless k is a root of some known polynomial (see [6, Theorem 7.2]). We then form the determinant polynomial for the dth subresultant of A_p and B_p, apply ψ_k to its entries to get the determinant polynomial S_d, and finally expand S_d, which is $\mathbf{Z}_p$-linearly dependent on $\psi_k(\gcd(A_p, B_p))$ by the algebraic subresultant theory. In this approach the multiplication in $\mathbf{Z}_p[X]$ is not used. This idea can be generalized to Ore polynomial rings.

4 Detailed description of the modular method

This section has five parts: in Section 4.1 we define the notion of subresultants for Ore polynomials; Sections 4.2, 4.4 and 4.5 are devoted to describing the algorithms GCRD_e, GCRD_p, GCRD_m, respectively. In Section 4.3 we briefly review rational number and function reconstructions.

4.1 Subresultants for Ore polynomials

In this section we introduce the notion of subresultants for Ore polynomials, and prove some properties that are used later. More details about the subresultant theory for Ore polynomials are presented in [8]. Throughout this section we let $\mathcal{R}[X]$ be an Ore polynomial ring with the conjugate operator σ and pseudo-derivation δ.

Let $\mathcal{A} : A_1, A_2, \ldots, A_m$ be a sequence in $\mathcal{R}[X]$ and n the maximum of the degrees of the A_i's. The matrix associated with $\mathcal{A}$, $\operatorname{mat}(\mathcal{A})$, is the $m \times (n+1)$ matrix whose entry in the ith row and jth column is the coefficient of X^{n+1-j} in A_i. If $m \le n+1$, then the determinant polynomial of $\mathcal{A}$, $\operatorname{detpol}(\mathcal{A})$, is $\operatorname{detpol}(\operatorname{mat}(\mathcal{A}))$ (see [9, 10]).

Definition 4.1 Let A and B be in $\mathcal{R}[X]$ with respective degrees m and n, where $m \ge n$. The nth subresultant of A and B is defined to be B. For $0 \le j \le n-1$, the jth subresultant of A and B, $\operatorname{sres}_j(A, B)$, is defined to be the determinant polynomial of the sequence

$$X^{n-j-1}A, \ldots, XA, A, X^{m-j-1}B, \ldots, XB, B.$$

Example 3 Let A, B be the same as those in Example 1. Then $\operatorname{sres}_1(A, B)$ is $\operatorname{detpol}(XA, A, X^2B, XB, B)$ equal to

$$\operatorname{detpol}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ t^2-t & 4t-2 & -3t+2 & 0 & 0 & 0 \\ 0 & t^2-t & 2t-1 & -3t & 3 & 0 \\ 0 & 0 & t^2-t & 0 & -3t & 6 \end{pmatrix}.$$

The following lemma can be seen as an extension of the Leibniz rule in calculus.

Lemma 4.1 If $r \in \mathcal{R}$ and $A \in \mathcal{R}[X]$, then the polynomial $(X^i(rA) - \sigma^i(r)X^iA)$ is an $\mathcal{R}$-linear combination of $X^{i-1}A$, $X^{i-2}A$, ..., A, for $i \in \mathbf{Z}^+$

Proof If $i = 1$, then $X(rA) - \sigma(r)XA = \delta(r)A$ by (2). The lemma follows by induction on i. □

In the rest of this section let A and B be in $\mathcal{R}[X]$ with respective positive degrees m and n, and $m \ge n$. For $r \in \mathcal{R}$ and $i \in \mathbf{Z}^+$, the *σ-factorial of r with order i* is the product of r, $\sigma(r)$, ..., $\sigma^{i-1}(r)$, and is denoted by $r^{[i]}$. Note that $\operatorname{lc}(XB) = \sigma(\operatorname{lc}(B))$ by (2). Hence it is easy to prove by induction that $\prod_{i=0}^{m-n} \operatorname{lc}(X^iB) = \operatorname{lc}(B)^{[m-n+1]}$.

Definition 4.2 The (right) *pseudo-remainder* of A and B, $\operatorname{prem}(A, B)$, is defined to be R with $\deg R < n$ such that

$$\operatorname{lc}(B)^{[m-n+1]}A = QB + R. \tag{3}$$

where $Q \in \mathcal{R}[X]$ with $\deg Q = m - n$.

Lemma 4.2 $(X^i\operatorname{prem}(A, B) - \operatorname{prem}(X^iA, X^iB))$ is an $\mathcal{R}$-linear combination of $X^{i-1}A$, ..., A, $X^{m-n+i}B$, ..., B, for all $i \in \mathbf{Z}^+$.

Proof Set $b = \operatorname{lc}(B)^{[m-n+1]}$. By (3) we get

$$X^ibA = X^iQB + X^i\operatorname{prem}(A, B)$$

and, since $\operatorname{lc}(X^iB) = \sigma^i(\operatorname{lc}(B))$, for all $i \in \mathbf{N}$,

$$\sigma^i(b)X^iA = Q_iX^iB + \operatorname{prem}(X^iA, X^iB),$$

where $Q, Q_i \in \mathcal{R}[X]$ both with degree $m - n$. Lemma 4.1 implies that $X^ibA - \sigma^i(b)X^iA$ is an $\mathcal{R}$-linear combination of $X^{i-1}A$, ..., A. The lemma is proved by subtracting the above two equations. □

The following two propositions form the basis for our modular method. In their proofs we use the properties of determinant polynomials given in [10, pp. 241–243]. Except Theorem 7.5.1, all the assertions in Section 7.5 in [10] hold for Ore polynomials. In linear differential case these two propositions are implicitly stated in [7].

Proposition 4.3 If d is the degree of the gcrds of A and B, then the dth subresultant of A and B is a gcrd of A and B.

Proof Let $S_d = \mathrm{sres}_d(A,B)$ and $R = \mathrm{prem}(A,B)$.

The proof will be done by induction on n. If $n = 1$, then either $d = 1$ or $d = 0$. If $d = 1$, then B is a gcrd of A and B, and B is $\mathrm{sres}_n(A,B)$ by Definition 4.1. If $d = 0$, then $\mathrm{sres}_0(A,B) = \mathrm{detpol}(A, X^{m-1}B, \ldots, B)$. It follows from (3) that

$$\mathrm{lc}(B)^{[m]}\mathrm{sres}_0(A,B) = (-1)^m \mathrm{lc}(B)^{[m]} R.$$

Since $R \neq 0$, $\mathrm{sres}_0(A,B)$ is non-zero, and it is clearly a gcrd of A and B.

Assume that Proposition 4.3 holds for $i < n$. If $d = n$, then B is a gcrd of A and B. Hence, we may assume that $d < n$, i.e. $R \neq 0$. Let $l = \deg R$ and $b = \mathrm{lc}(B)^{[m-n+1]}$. Then $l \geq d$. By the definition of subresultants we have

$$S_d = \mathrm{detpol}(X^{n-d-1}A, \ldots, A,\ X^{m-d-1}B, \ldots, B).$$

It follows from (3) that

$$\sigma^{n-d-1}(b) S_d = \mathrm{detpol}(R_{n-d-1}, X^{n-d-2}A, \ldots, A,\ X^{m-d-1}B, \ldots, B),$$

where $R_{n-d-1} = \mathrm{prem}(X^{n-d-1}A, X^{n-d-1}B)$.

By Lemma 4.2, R_{n-d-1} can be replaced by $X^{n-d-1}R$. Hence,

$$\sigma^{n-d-1}(b) S_d = \mathrm{detpol}(X^{n-d-1}R, X^{n-d-2}A, \ldots, A,\ X^{m-d-1}B, \ldots, B).$$

In the same way we replace $X^i A$ by $X^i R$, for $i = n-d-2$, $n-d-3, \ldots, 0$. Thus,

$$rS_d = \mathrm{detpol}(X^{n-d-1}R, \ldots, R,\ X^{m-d-1}B, \ldots, B),$$

for some non-zero $r \in \mathcal{R}$. Therefore, there exists a non-zero $h \in \mathcal{R}$ such that

$$rS_d = h\,\mathrm{detpol}(X^{l-d-1}B, \ldots, B, X^{n-d-1}R, \ldots, R), \quad (4)$$

The right-hand side of (4) is equal to $h\,\mathrm{sres}_d(B,R)$. Therefore, $rS_d = h\,\mathrm{sres}_d(B,R)$. The proposition then follows from the induction hypothesis. □

Proposition 4.4 Let M be the matrix associated with the sequence $X^{n-1}A, \ldots, XA, A, X^{m-1}B, \ldots, XB, B$. If the gcrds of A and B have degree d, then $\mathrm{rank}(M)$ is equal to $(m+n-d)$.

Proof Since S_d is nonzero, the rows of M represented by $X^{n-d-1}A, \ldots, A, X^{m-d-1}B, \ldots, B$ are $\mathcal{R}$-linearly independent. Hence, the rows of M represented by $X^{n-d-1}A, \ldots, A, X^{m-1}B, \ldots, X^{m-d-1}B, \ldots, B$ are $\mathcal{R}$-linearly independent. Thus, we conclude that $\mathrm{rank}(M) \geq m+n-d$.

There are non-zero $u, v \in \mathcal{R}$ and $U, V \in \mathcal{R}[X]$ such that $uA = US_d$ and $vB = VS_d$, because S_d is a gcrd of A and B by Proposition 4.3. Hence all the $X^i A$, for $0 \leq i \leq n-1$, and $X^j B$, for $0 \leq j \leq m-1$, are $\mathcal{R}$-linear combinations of $X^{m+n-d-1}S_d, \ldots, S_d$. Therefore, $\mathrm{rank}(M) \leq m+n-d$. □

Example 4 Let A, B be the same as those in Example 1, and $M = \mathrm{mat}(X^2A, XA, A, X^3B, X^2B, XB, B)$, that is,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t^2-t & 6t-3 & -3t+6 & -3 & 0 & 0 & 0 \\ 0 & t^2-t & 4t-2 & -3t+2 & 0 & 0 & 0 \\ 0 & 0 & t^2-t & 2t-1 & -3t & 3 & 0 \\ 0 & 0 & 0 & t^2-t & 0 & -3t & 6 \end{pmatrix}.$$

The rank of M is equal to 6. It follows from Proposition 4.4 that $\mathrm{gcrd}(A,B)$ is of degree 1 and then from Proposition 4.3 that $\mathrm{gcrd}(A,B)$ is $\mathbf{Z}[t]$-linearly dependent on $\mathrm{sres}_1(A,B)$, which is equal to $-9t(3t-2)(tX-2)$.

4.2 The algorithm GCRD_e

In this section let $(\mathbf{Z}_p[t][X], \sigma_p, \delta_p)$ be an Ore polynomial ring. Fix an element k of $\mathbf{Z}_p$ and the evaluation mapping ψ_k. Assume $A,\ B \in \mathbf{Z}_p[t][X]$, with $\deg A = m$ and $\deg B = n$, and $m \geq n \geq 1$. Let

$$M = \mathrm{mat}(X^{n-1}A, \ldots, XA,\ A,\ X^{m-1}B, \ldots, XB,\ B).$$

We show how to use the arithmetic in $\mathbf{Z}_p$ to compute the monic associate of $\psi_k(\mathrm{gcrd}(A,B))$.

Lemma 4.5 Let $G = \mathrm{gcrd}(A,B)$ with $\deg G = d$, and let $S_d = \mathrm{sres}_d(A,B)$. If $\psi_k(\mathrm{lc}(S_d))$ is nonzero, then

$$\psi_k(G)/\psi_k(\mathrm{lc}(G)) = \psi_k(S_d)/\psi_k(\mathrm{lc}(S_d)).$$

Proof By Proposition 4.3 there exists a non-zero element r of $\mathbf{Z}_p[t]$ such that $rG = S_d$. Since $\psi_k(\mathrm{lc}(S_d))$ is nonzero, $\psi_k(\mathrm{lc}(G))$ is also nonzero. Applying ψ_k to

$$G/\mathrm{lc}(G) = S_d/\mathrm{lc}(S_d)$$

yields the lemma. □

Definition 4.3 The evaluation point k is *unlucky* for A and B if either $\psi_k(\sigma_p^i(\mathrm{lc}(B))) = 0$, for some i with $0 \leq i \leq m-1$, or $\psi_k(\mathrm{lc}(\mathrm{sres}_d(A,B))) = 0$, where $d = \deg \mathrm{gcrd}(A,B)$.

Now, we present GCRD_e for computing the evaluation homomorphic images of the monic associate of $\mathrm{gcrd}(A,B)$ when evaluation points are not unlucky.

algorithm GCRD_e

Input: A prime p, a residue $k \in \mathbf{Z}_p$, and $A, B \in \mathbf{Z}_p[t][X]$ with $\deg A \geq \deg B \geq 1$.

Output: $g \in \mathbf{Z}_p[X]$. If k is not unlucky, g is the monic associate of $\psi_k(\mathrm{gcrd}(A,B))$, otherwise g is 0 or of degree greater than d.

1. $m \leftarrow \deg A$; $n \leftarrow \deg B$;
2. **if** $\psi_k\left(\prod_{i=0}^{m-1} \sigma_p^i(\mathrm{lc}(B))\right) = 0$ **then return**(0);
3. $M_k \leftarrow$ the image of M under ψ_k;
4. $r \leftarrow \mathrm{rank}(M_k)$;
5. **if** $r = m+n$ **then return**(1);
6. $S \leftarrow$ the determinant formula of $(\mathrm{sres}_{m+n-r}(A,B))$;
7. $g \leftarrow$ the expansion of $\psi_k(S)$;
8. **if** $\deg g = m+n-r > 0$ **then** $g \leftarrow g/\mathrm{lc}(g)$;
9. **else** $g \leftarrow 0$;
10. **return**(g);

The following example shows how GCRD_e works.

Example 5 We regard A and B in Example 1 as polynomials in $\mathbf{Z}_{11}[t][X]$. Let us use GCRD_e to compute evaluation homomorphic images of $\mathrm{gcrd}(A,B)$. The evaluation points 0 and 1 are unlucky, because the leading coefficient of B is $t(t-1)$. For $k = 2$, the matrix M given in Example 4 is mapped by ϕ_2 to M_2 whose entries are those of M evaluated at 2. The rank of M_2 is 6, and hence we get an upper bound for the degree of $\mathrm{gcrd}(A,B)$ to be $\deg(A) + \deg(B) - 6 = 1$ by Proposition 4.4. In line 6, S, the determinant formula of $\mathrm{sres}_1(A,B)$, is formed as given in Example 3. The entries of the determinant polynomial $\psi_2(S)$ in line 7 are those of S evaluated at 2. Expanding $\psi_2(S)$ yields $g = 10X + 1$ in

line 7. Since $\deg g = 1$, g is normalized to be $X + 10$ in line 8.

If $k = 8$, then in line 4 we get the rank of M_8, $r = 6$, but g computed by the commands in lines 6 and 7 is zero. Hence 8 is also an unlucky point.

Proposition 4.6 The algorithm GCRD_e is correct.

Proof Let $\deg \gcrd(A, B) = d$ and $S_d = \mathrm{sres}_d(A, B)$. We may exclude the case when $\psi_k(\sigma_p^i(\mathrm{lc}(B))) = 0$, for some i with $0 \le i < m$. Thus, $\deg \psi_k(X^i B) = n + i$, for $0 \le i < m$, since $\mathrm{lc}(X^i B) = \sigma_p^i(\mathrm{lc}(B))$. Let r be obtained from line 4. Note that $r \le \mathrm{rank}(M)$, since M_k is the evaluation homomorphic image of M.

If $r = m+n$, then $\mathrm{rank}(M) = m+n$, consequently, $d = 0$ by Proposition 4.4. Therefore, it is correct that GCRD_e returns 1 in line 5.

Suppose that $r < m + n$. Let $l = m + n - r$. Then Proposition 4.4 implies that $l \ge d$. Note that M_k in line 3 is the matrix associated with the sequence

$$\begin{array}{llll} \psi_k(X^{n-1}A), & \ldots, & \psi_k(X^{n-d-1}A), & \ldots, \quad \psi_k(A), \\ \psi_k(X^{m-1}B), & \ldots, & \psi_k(X^{m-d-1}B), & \ldots, \quad \psi_k(B). \end{array}$$

If k is not unlucky, then the members of the sequence

$$\begin{array}{llll} & & \psi_k(X^{n-d-1}A), & \ldots, \quad \psi_k(A), \\ \psi_k(X^{m-1}B), & \ldots, & \psi_k(X^{m-d-1}B), & \ldots, \quad \psi_k(B) \end{array}$$

are $\mathbf{Z}_p$-linearly independent, because $\psi_k(\mathrm{lc}(S_d)) \neq 0$. Consequently, we have $r = m + n - d$. Hence, the polynomial g obtained from lines 6 and 7 is $\psi_k(S_d)$ of degree d. By Lemma 4.5 GCRD_e is correct.

Now, assume that $\psi_k(\mathrm{lc}(S_d))) = 0$. Then we have either $l > d$ or $l = d$ and $\deg \psi_k(S_d) < d$. In the former case the polynomial g obtained from lines 6 and 7 has degree either greater than d or less than l, so GCRD_e returns either 0 or a polynomial with degree greater than d. In the latter case GCRD_e returns 0, according to line 9. □

To analyze GCRD_e for linear differential operators over $\mathbf{Z}_p[t]$, we count the number of word operations needed, and we assume that arithmetic operations in $\mathbf{Z}_p$ can be performed in unit time. By the complexity of an algorithm we mean the worse-case complexity.

Proposition 4.7 If

$$m = \max(\deg A, \deg B), \text{ and } m_t = \max(\deg_t A, \deg_t B),$$

then the complexity of GCRD_e is $O(m_t m^2 + m^3)$.

Proof Since σ is the identity mapping, GCRD_e takes $O(m_t)$ in line 2. There are two calculations in line 3, namely, constructing M by differentiation and computing M_k by Horner's evaluation, each of which takes $O(m_t m^2)$. The time for computing the rank of M_k in line 4 is bounded by $O(m^3)$, and so is the time for expanding $\psi_k(S)$ in line 7. The time for other calculations can certainly be neglected. □

In GCRD_e, the homomorphic image of the $(m+n-r)$th subresultant of A and B is computed by expanding its *determinant formula* (not by the subresultant algorithm). It can also be read off from the Gaussian elimination for computing the rank of M, provided the pivot rows are chosen properly. The interested reader is referred to [8] for a more sophisticated and efficient version of GCRD_e.

4.3 Rational Number and Function Reconstructions

To use interpolation to combine the evaluation homomorphic images of the monic gcrd of two Ore polynomials, say A and B, in $\mathbf{Z}_p[t][X]$, we need to know a multiplicative bound for the denominators of the monic gcrd of A and B. In the algebraic case, such a bound is $\gcd(\mathrm{lc}(A), \mathrm{lc}(B))$. However, there are counterexamples showing that neither the gcd nor the lcm of $\mathrm{lc}(A)$ and $\mathrm{lc}(B)$ is the desired multiplicative bound. A multiplicative bound is the leading coefficient of the dth subresultant of A and B if $\gcrd(A, B)$ has degree d. Unfortunately, this multiplicative bound is loose. Inspired by the work in [5], we use rational function reconstruction to combine the evaluation homomorphic images of $\gcrd(A, B)$. A similar problem arises when A and B are in $\mathbf{Z}[t][X]$. So, the rational number reconstruction is also needed.

We will not go into the details about rational number and rational function reconstructions. The reader may find relevant materials about rational number reconstruction in [13, 14, 4], and those about rational function reconstruction in [8], as well as the Maple function `Ratrecon`.

Applying rational number (function) reconstruction to the coefficients of a polynomial, one may easily get:

algorithm COEFF_n

Input: A modulus $m \in \mathbf{Z}^+$ and a residue $R \in \mathbf{Z}_m[t][X]$.
Output: $A \in \mathbf{Q}[t][X]$, such that $A \equiv R \bmod m$ and the denominators and numerators of the rational coefficients in A range from $-\sqrt{m/2}$ to $\sqrt{m/2}$ if such a polynomial exists. Otherwise, `NIL` is returned.

algorithm COEFF_f

Input: A modulus $M \in \mathbf{Z}_p[t]$ and a residue $R \in \mathbf{Z}_p[t][X]$.
Output: $A \in \mathbf{Z}_p(t)[X]$, such that $A \equiv R \bmod M$ and the denominators and numerators of the rational function coefficients in A have degrees less than $(\deg_t M)/2$ if such a polynomial exists. Otherwise, `NIL` is returned.

In these two algorithms we use the least non-negative representation for integral residues. We also remark that the solution satisfying the specification of COEFF_n (COEFF_f) is unique if existent.

4.4 The algorithm GCRD_p

Let $(\mathbf{Z}_p[t][X], \sigma_p, \delta_p)$ be an Ore polynomial ring. We present the modular algorithm GCRD_p for computing gcrds in this ring. First, we reduce the gcrd problem in $\mathbf{Z}_p[t][X]$ to a series of problem of computing the evaluation homomorphic images in $\mathbf{Z}_p[X]$, which will be later solved by the algorithm GCRD_e. The "lucky" evaluation homomorphic images are combined by Newton's interpolation and COEFF_f. The termination of GCRD_p is determined by trial division. It is a rare, though possible case that there are not enough lucky evaluation points in $\mathbf{Z}_p$. If this happens, GCRD_p reports *failure*.

algorithm GCRD_p

Input: A prime p, and $A, B \in \mathbf{Z}_p[t][X]$ with $\deg A \ge \deg B \ge 1$.
Output: C, where $C = \gcrd(A, B)$.

```
[initialize the modulus, residue, and degree ]
1.  k ← 0;
2.  repeat
3.          if k = p then report failure;
4.          R_k ← GRCD_e(p, k, A, B); k ← k + 1;
5.  until R_k ≠ 0
6.  d_k ← deg R_k;
7.  if d_k = 0 then return(1);
8.  M ← t − k; R ← R_k; d ← d_k; C ← 0;
[main loop]
9.  while true do {
10.         repeat
11.                 if k = p then report failure;
12.                 R_k ← GRCD_e(p, k, A, B); k ← k + 1;
13.         until R_k ≠ 0
14.         d_k ← deg R_k;
            [ test for unlucky evaluation homomorphism ]
15.         if d_k < d then goto line 7;
16.         if d_k = d then {
              [ combine ]
17.           R ← Interpolation(R, M, R_k, t − k);
18.           M ← (t − k)M;
19.           C' ← COEFF_f(M, R);
20.           if C ≠ 0 and C = C' then
                [ trial division ]
21.             if A ≡ 0 mod C and B ≡ 0 mod C then
22.               C ← the numerator of C;
23.               return(C);
24.           C ← C'; } }
```

Example 6 Let A and B be the same as those in Example 5. We compute $\gcd(A, B)$ by the algorithm GCRD_p. The loop from line 2 to 5 discards the evaluation points 0 and 1 and finds $R_2 = X + 10$, as computed in Example 5. Hence M, R, and d are initialized to be $t-2$, $X+10$, and 1, respectively. The while-loop yields:

k	R_k	R	C'
3	$X+3$	$X+4t+2$	NIL
4	$X+5$	$X+10t^2+9t+7$	$X+9/t$
5	$X+4$	$X+t+6t^2+9t^3$	$X+9/t$

COEFF_f yields the same result in the second and third iterations. We then invoke the trial division in line 21 which affirms $\gcd(A, B) = tX + 9$.

Proposition 4.8 The algorithm GCRD_p is correct.

Proof Let $G = \gcd(A, B)$ with $\deg_t G = d_t$. If there are less than $(2d_t + 2)$ lucky points in $\mathbf{Z}_p$, then GCRD_p reports failure. Assume that there are more than $(2d_t + 1)$ lucky points in $\mathbf{Z}_p$. Then the tentative degree d in GCRD_p will be eventually equal to $\deg G$, because, for each unlucky point, GCRD_e returns either 0 or a polynomial in $\mathbf{Z}_p[X]$, whose degree is greater than $\deg G$. The unlucky evaluation points can be detected in line 15 as soon as a lucky one is encountered. So, we may suppose that d is equal to $\deg G$. Then each R_k entering Newton's interpolation in line 17 is equal to $\psi_k(G/\mathrm{lc}(G))$ by Proposition 4.5. Hence, the congruence $R \equiv G/\mathrm{lc}(G) \bmod M$ holds in GCRD_p . Since the solution to the rational function reconstruction problem is unique, COEFF_f in line 19 recovers $G/\mathrm{lc}(G)$ when $\deg_t M$ exceeds $2d_t$. COEFF_f produces $G/\mathrm{lc}(G)$ again when the next lucky evaluation point is encountered. Then the condition $C = C'$ in line 20 is satisfied. Therefore, GCRD_p returns G after a trial division. □

The next lemma ensures that GCRD_p does not report failure if p is "sufficiently" large.

Lemma 4.9 If $A, B \in \mathbf{Z}_p[t][X]$, $\deg A = m$, $\deg B = n$, and $m \geq n$, then there are at most

$$\deg_t \left(\prod_{i=0}^{m-1} \sigma_p^i(\mathrm{lc}(B)) \right) + m \deg_t B + n \deg_t A \qquad (5)$$

unlucky evaluation points for A and B.

Proof If k is unlucky for A and B, then k is a root of the polynomial $\left(\prod_{i=0}^{m-1} \sigma_p^i(\mathrm{lc}(B))\right) \mathrm{lc}(\mathrm{sres}_d(A, B))$, where d is the degree of $\gcd(A, B)$. The integer (5) is a degree bound for this polynomial. □

In analyzing the computing time of GCRD_p for linear differential operators over $\mathbf{Z}_p[t]$, we assume that no unlucky evaluation points occur, and that the verification of the gcrd, by means of the trial divisions, is successful on the first try. We let m be the maximum of degrees of A and B in X, d the degree of $\gcd(A, B)$, d_t the maximum of degrees of A, B and $\gcd(A, B)$ in t, and $d_t > 0$.

Proposition 4.10 If d is equal to 0, then the complexity of GCRD_p is $O(d_t m^2 + m^3)$. Otherwise it is

$$O(d_t^2 m^2 + d_t m^3 + d d_t^3) + T_p(m, d, d_t),$$

where the function $T_p(m, d, d_t)$ is the complexity of trial division.

Proof If d is equal to 0, then we need only perform GCRD_e once. The proposition then follows from Proposition 4.7.

If $d > 0$, then we compute $(2d_t + 2)$ monic homomorphic images of $\gcd(A, B)$ at a cost $O(d_t(d_t m^2 + m^3))$ by Proposition 4.7. COEFF_f is applied $(2d_t + 1)$ times to $(d + 1)$ sets of inputs with degree less than $(2d_t + 2)$; the total cost is $O(d d_t^3)$. This cost dominates the costs of interpolation in line 17 and simplification in line 22. □

Performing the trial division in GCRD_p is equivalent to deciding if $\mathrm{sres}_{d-1}(A, C)$ and $\mathrm{sres}_{d-1}(B, C)$ are zero. Using evaluation homomorphisms we can obtain a modular algorithm for the trial division with complexity $O(m^3 d_t^2 + m^4 d_t)$, which dominates other costs of GCRD_p. However, we separate the cost $T_p(m, d, m_t)$ from others because the trial division takes little time in practice.

4.5 The algorithm GCRD_m

In this section, we let $A, B \in \mathbf{Z}[t][X]$ with $\deg A = m$ and $\deg B = n$. Let $m \geq n \geq 1$ and $G = \gcd(A, B)$. Using modular homomorphisms we transform the problem of computing G to a series of the problems of computing the monic associates of the modular homomorphic images of G. First, we define unlucky primes.

Definition 4.4 A prime p is *unlucky* for A and B if one of the following holds:

1. p is a divisor of $\mathrm{hc}(\sigma(t))\mathrm{lc}(A)\mathrm{lc}(B)$;
2. p is a divisor of $\mathrm{lc}(\mathrm{sres}_l(A, B))$, where $l = \deg G$;
3. p is a divisor of $\mathrm{hc}(G)$;
4. $\phi_p(G)$ is not primitive with respect to X.

Lemma 4.11 If p is not unlucky and $\mathbf{Z}_p[t][X]$ is the Ore polynomial ring as defined in Lemma 2.1, then

$$\gcd(\phi_p(A), \phi_p(B)) = \phi_p(G)/\phi_p(\mathrm{hc}(G)). \qquad (6)$$

Proof Let $\deg G = l$. Since $\deg A = \deg \phi_p(A)$ and $\deg B = \deg \phi_p(B)$, $\mathrm{sres}_l(\phi_p(A), \phi_p(B)) = \phi_p(\mathrm{sres}_l(A, B)) \neq 0$. So, the degree of $\gcrd(\phi_p(A), \phi_p(B))$ is not greater than l, since every common right factor of A and B must be a right factor of their subresultants. On the other hand, Lemma 2.1 implies that $\phi_p(G)$ is a common right factor of $\phi_p(A)$ and $\phi_p(B)$. Thus, $\phi_p(G)$ is a gcrd of $\phi_p(A)$ and $\phi_p(B)$ since $\deg \phi_p(G) = l$. Hence, (6) holds because $\phi_p(G)$ is primitive with respect to X. □

Clearly, there are only finitely many unlucky primes for A and B. For each lucky prime p, $\phi_p(G)/\mathrm{hc}(\phi_p(G))$ can be constructed by $\gcrd(\phi_p(A), \phi_p(B))$. These considerations lead to the algorithm GCRD_m.

algorithm GCRD_m

Input: $A, B \in \mathbf{Z}[t][X]$;
Output: C, where $C = \gcrd(A, B)$.

```
[ initialize ]
1.  if deg(A) ≥ deg(B) then { A_1 ← A; A_2 ← B; }
2.  else { A_1 ← B; A_2 ← A; }
3.  A_1 ← the primitive part of A_1 w.r.t. X;
4.  A_2 ← the primitive part of A_2 w.r.t. X;
5.  b ← hc(A_1)hc(A_2)hc(σ(t));
[ initialize the modulus, residue, and degrees ]
6.  p ← a large prime not dividing b;
7.  R_p ← GCRD_p(p, φ_p(A_1), φ_p(A_2));
8.  d_p ← deg R_p; d_{t,p} ← deg_t R_p;
9.  if d_p = 0 then return(1);
10. L ← p; R ← R_p; d ← d_p; d_t ← d_{t,p}; C ← 0
[main loop]
11. while true do {
12.       p ← a new large prime not dividing b
13.       R_p ← GCRD_p(p, φ_p(A_1), φ_p(A_2))
14.       d_p ← deg R_p; d_{t,p} ← deg_t R_p;
          [ test for unlucky primes ]
15.       if d_p < d then goto line 9;
16.       if d_p = d and d_{t,p} > d_t then goto line 10;
          [ combine ]
17.       if d_p = d and d_{t,p} = d_t then {
18.         R ← CRA(R, L, R_p, p);
19.         L ← pL;
20.         C' ← COEFF_n(L, R);
21.         if C ≠ 0 and C = C' then
              [ trial division ]
22.           if A_1 ≡ 0 mod C and A_2 ≡ 0 mod C then
23.             C ← the numerator of C;
24.             return(C);
25.       C ← C'; } }
```

Remark 1 By a "large prime" p we mean that p is so large that GCRD_p does not report failure. By Lemma 4.9 it is always possible to choose such p.

Example 7 Let A and B be the same as those in Example 5. We compute $\gcrd(A, B)$ by GCRD_m. We begin with $p = 11$. As given in Example 6, m, R are initialized to be 11, $tX + 9$, respectively. The while-loop yields:

p	R_p	R	C'
13	$tX + 11$	$tX + 141$	$tX - 2$
17	$tX + 15$	$tX + 2429$	$tX - 2$

COEFF_n yields the same result in the first and second iterations. Then the trial division in line 21 affirms $\gcrd(A, B) = tX - 2$, as stated in Example 1.

Proposition 4.12 The algorithm GCRD_m is correct.

Proof As b is assigned to be $\mathrm{hc}(A_1)\mathrm{hc}(A_2)\mathrm{hc}(\sigma(t))$ in line 5, GCRD_p can only result R_p in lines 7 and 13 such that either $\deg R_p > \deg G$ or $\deg_t R_p < \deg_t G$ if p is unlucky. The unlucky primes can be detected in lines 15 and 16 as soon as a lucky prime is encountered. Since there are only a finite number of unlucky primes, we may further assume that $d = \deg G$ and $d_t = \deg_t G$. Accordingly, the polynomial R in line 18 satisfies $R \equiv G/\mathrm{hc}(G) \bmod L$. Then the polynomial C' computed by COEFF_n in line 20 is equal to $G/\mathrm{hc}(G)$ as soon as $\sqrt{L/2}$ exceeds the maximum of the absolute values of the integral coefficients of G. Thus, GCRD_m returns G. □

The advantages of GCRD_m are clear. The problem of finding $\gcrd(A, B)$ is mapped to a domain in which the arithmetic does not cause any intermediate swelling. In addition, GCRD_m can recognize the case that $\gcrd(A, B) = 1$ as soon as a lucky prime is encountered.

We analyze GCRD_m for linear differential operators over $\mathbf{Z}$ under the similar assumptions made in the previous sections. These estimates will involve two additional parameters: S, the maximum of the absolute values of the integral coefficients A and B; and s, that of the integral coefficients of G. To make things simple we assume that A and B are primitive with respect to X, that m, d and d_t are the same as those in Proposition 4.10, and that the primes used in GCRD_m are lucky and of length one.

Proposition 4.13 If d is equal to 0, then the complexity of GCRD_m is $O(d_t m \log S + d_t m^2 + m^3)$. Otherwise it is

$$O((d_t m \log S + d_t^2 m^2 + d_t m^3 + d d_t^3) \log s + d d_t \log^3 s) + T_p(m, d, d_t) O(\log s) + T(S, s, m, d, d_t)$$

where $T_p(m, d, d_t)$ is the same as that in Proposition 4.10 and $T(S, s, m, d, d_t)$ is the complexity of the trial division in GCRD_m.

Proof If $d = 0$ then we need only one prime and one evaluation point. Thus, GCRD_m returns 1 in

$$O(d_t m \log S + d_t m^2 + m^3)$$

word operations, where the first term receives contribution from computing $\phi_p(A)$ and $\phi_p(B)$, and the others from computing $\gcrd(\phi_p(A), \phi_p(B))$ (see Proposition 4.7).

If $d > 0$ then we compute $O(\log s)$ modular gcrds in

$$O((d_t m \log S + d_t^2 m^2 + d_t m^3 + d d_t^3 + T_p(m, d, d_t)) \log s)$$

word operations by Proposition 4.10. COEFF_n is applied $O(\log s)$ times to $O(d d_t)$ sets of inputs of length $O(\log s)$; the total cost for COEFF_n is $O(d d_t \log^3 s)$. This cost also dominates the costs of Chinese remainder algorithm in line 18 and the simplification in line 23. □

The trial division in GCRD_m can be realized by deciding if $\mathrm{sres}_{d-1}(A, C)$ and $\mathrm{sres}_{d-1}(B, C)$ are zero. Using a naive modular method we can see that

$$T(S, s, m, d, d_t) = T_p(m, d, d_t) O(\log(m! S s^{m-d-1})),$$

which dominates other costs in GCRD_m. However, we prefer to separate the cost $T(S, s, m, d, d_t)$ from others because the trial division takes little time in practice, as shown in the next section.

As a crudification of this result, let $D = \max(m, d, d_t)$ and $L = \max(S, s)$, and neglect the cost of trial divisions. Then the complexity of GCRD_m is $O(D^4 \log L + D^2 \log^3 L)$.

5 Experimental Results

This section presents experimental results to compare the algorithm GCRD_m, subresultant algorithm, and primitive Euclidean algorithm. We implemented in *Maple V* (Release 3) these three algorithms for the linear differential operators and linear shift operators with coefficients in $\mathbf{Z}[t]$.

The first suite was generated as follows. We used the Maple function `randpoly` to generate pairs of bivariate polynomials in $\mathbf{Z}[t, X]$ with total degree n and $n-1$, where $n = 5$, 10, and 15. These polynomials have five terms with coefficients ranging from -99 to 99. We then regarded these polynomials as differential operators and shift operators over $\mathbf{Z}[t]$, respectively, and computed the gcrd of each pair. The timings are summarized in Figure 1, in which the column labeled n gives the total degrees of the polynomials; the columns labeled DM, DS, DPE, give the respective computing times for GCRD_m, subresultant algorithm, and primitive Euclidean algorithm whose inputs are differential operators; similarly, the columns labeled SM, SS, SPE, give the respective computing times for GCRD_m, subresultant algorithm, and primitive Euclidean algorithm whose inputs are shift operators. All the entries are Maple CPU time and given in seconds.

n	DM	DS	DPE	SM	SS	SPE
5	0.20	0.27	0.19	0.17	0.25	0.21
10	0.99	38.86	39.71	0.59	42.73	40.71
15	1.65	301.25	374.00	0.77	436.47	485.91

Figure 1: Computing times for the first suite

The timings of Figure 1 shows that GCRD_m is considerably faster than the non-modular ones when the input polynomials are of total degree more than eight. This is not a surprise since two random polynomials usually do not have a nontrivial gcrd. In practice, GCRD_m can decide if two polynomials are relatively prime by one or two primes.

To construct the second suite, we used `randpoly` to generate three polynomials, say A, B, and C, with respective total degrees $n-2$, $n-3$, and 2, where $n = 5$, 10, and 15. The number of terms and length of coefficients are the same as those in the first suite. We took the differential (shift) products AC and BC as the input polynomials. Thus, the gcrd of each pair of the input polynomials was usually nontrivial. The timings are summarized in Figure 2, A dash (–) indicates that our implementation of the primitive Euclidean algorithm took more than 3 hours without any output. This could happen because it took very long time to compute the primitive part of a polynomial in $\mathbf{Z}[t][X]$ when the content had large integral coefficients. In these examples the trial division in GCRD_m took less than one percent of the total computing time.

n	DM	DS	DPE	SM	SS	SPE
5	2.26	0.25	0.15	1.29	0.30	0.15
10	9.91	64.25	16.72	3.74	57.66	18.67
15	27.23	1348.83	–	6.46	1999.64	–

Figure 2: Computing times for the second suite

Again, the timings in Figure 2 indicate that GCRD_m is more efficient than the non-modular ones. We also remark that the subresultant algorithm may be slower than the primitive Euclidean algorithm when the input polynomials have a non-trivial gcrd. This is because the primitive Euclidean algorithm removes more extraneous factors after each division when the gcrd is not monic (see [8, §2.2]).

References

[1] M. Bronstein and M. Petkovšek. On Ore Rings, Linear Operators and Factorization. *Programming and Comput. Software*, **20**, pp. 14–26, 1994.

[2] M. Bronstein and M. Petkovšek. An introduction to pseudo-linear algebra, *Theoretical Computer Science*, **157**, pp. 3 – 33, 1996.

[3] W. S. Brown. On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *JACM*, **18**, pp. 478-504, 1971.

[4] G. E. Collins and M. J. Encarnación. Efficient Rational Number Reconstruction. *Journal of Symbolic Computation* **20**, pp. 299–313, 1995.

[5] M. J. Encarnación. Computing GCDs of Polynomials over Algebraic Number Fields. *Journal of Symbolic Computation* **20**, pp. 287-297, 1995.

[6] K. O. Geddes, S. R. Czapor and G. Labahn. *Algorithms for Computer Algebra.* Kluwer Academic Publishers, 1992.

[7] D. Yu. Grigor'ev. Complexity of Factoring and Calculating the GCD of Linear Ordinary Differential Operators. *Journal of Symbolic Computation*, **10**, pp. 7-37, 1990.

[8] Z. Li. *A Subresultant Theory for Ore Polynomials and its Applications.* PhD Thesis, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, A-4040, Austria, 1996.

[9] R. Loos. Generalized Polynomial Remainder Sequences. *Computer Algebra, Symbolic and Algebraic Computation*, B. Buchberger, G. E. Collins and R. Loos (eds.), Springer-Verlag, Wien-New York, pp. 115-137, 1982.

[10] B. Mishra. *Algorithmic Algebra.* Texts and Monographs in Computer Science, D. Gries and F.B. Schneider (eds.) Springer-Verlag. 1993.

[11] O. Ore. Theory of Non-Commutative Polynomials. *Annals of Math*, **34**, pp. 480-508, 1933.

[12] B. Salvy and P. Zimmermann. Gfun: A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable. *ACM Transactions on Mathematical Software*, **20**, pp. 163 – 177, 1994.

[13] P. S. Wang. A p-adic Algorithm for Univariate Partial Fractions. In *Proceedings of the 1981 Symposium on Symbolic and Algebraic Computation*, pp. 212-217. ACM Press, 1981.

[14] P. S. Wang, M. J. T. Guy and J. H. Davenport. p-adic Reconstruction of Rational Numbers. *SIGSAM Bulletin*, **16**, pp. 2-3, 1982.

[15] H. S. Wilf and D. Zeilberger. An Algorithmic Proof of Theory for Hypergeometric (Ordinary and "q") Multisum / Integral Identities. *Inventiones Mathematicae*, 108, pp. 575–633, 1992.