A Modular Algorithm to Compute the Resultant of Multivariate Polynomials over Algebraic Number Fields Presented with Multiple Extensions

Mahsa Ansari and Michael Monagan

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

Abstract. Let f_1 and f_2 be two multivariate polynomials over an algebraic number field $\mathbb{Q}(\alpha_1, \dots, \alpha_n)$. In this paper, we present MRES, a modular algorithm for computing the resultant of f_1 and f_2. To enhance the efficiency, our algorithm converts f_1 and f_2 to their corresponding polynomials over $\mathbb{Q}(\gamma)$ where γ is a primitive element of $\mathbb{Q}(\alpha_1, \dots, \alpha_n)$. This conversion is done modulo a prime to prevent the coefficient growth. Next, our algorithm employs evaluation and dense interpolation to reduce the problem to the computation of the resultant of two univariate polynomials where we apply the monic Euclidean algorithm. Employing the monic Euclidean algorithm, we present a new formula for computing the resultant of univariate polynomials. Finally, our modular algorithm applies the Chinese remaindering and the rational number reconstruction to recover the rational coefficients of the resultant.
We have implemented our algorithm in Maple. We include the expected time complexity of the algorithm, two benchmarks, and a partial failure probability analysis.

Keywords: Resultant. Modular Algorithms. Algebraic Number Fields. Primitive Elements.

1 Introduction

Computing the resultant of two polynomials plays a significant role across various areas of mathematics. Resultants appear as a subproblem in solving systems of multivariate polynomials, elimination theory [5] and factorization of polynomials over algebraic fields [10].

In this paper, we are interested in computing the resultant of two multivariate polynomials over an algebraic number field $\mathbb{Q}(\alpha_1, \dots, \alpha_n)$. In 1971, Collins [4] introduced a modular algorithm to compute the resultant of multivariate polynomials over $\mathbb{Z}$. In 2002, based on previous work by Encarnacion [6], Monagan and van Hoeij [11] designed a modular GCD algorithm for $\mathbb{Q}(\alpha_1, \dots, \alpha_n)[x]$. In 2023, Ansari and Monagan [1] designed a modular GCD algorithm that reduces the gcd problem over $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ to gcd calculation over $\mathbb{Q}(\gamma)$ where γ is a primitive element of $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$. Given $f_1, f_2 \in \mathbb{Q}(\alpha_1, \ldots, \alpha_n)[x_1, \ldots, x_k, y]$, we build upon [4,11] and [1] to compute $r = \mathrm{res}(f_1, f_2, y) \in \mathbb{Q}(\alpha_1, \ldots, \alpha_n)[x_1, \ldots, x_k]$.

1.1 Computing over $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$

Let $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ be our number field. Let $L_0 = \mathbb{Q}$ and $L_i = L_{i-1}[z_i]/\langle M_i(z_i)\rangle$ where $M_i(z_i)$ is the minimal polynomial of α_i over L_{i-1} for $1 \le i \le n$. Let $L = L_n$ and $d_i = \deg(M_i, z_i)$. The field L is isomorphic to $\mathbb{Q}[z_1, \ldots, z_n]/\langle M_1, \ldots, M_n\rangle$ and it can be specified as a $\mathbb{Q}$-vector space of dimension $d = \prod_{i=1}^n d_i$. Furthermore, $B_L = \{\prod_{i=1}^n (z_i)^{e_i} \mid 0 \le e_i < d_i\}$ is a basis of L. To compute in $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$, we use the fact that $\mathbb{Q}(\alpha_1, \ldots, \alpha_n) \cong L$. Thus, we just need to map elements from $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ to L and compute over L. In our algorithm, we suppose that we are given the minimal polynomials $M_1(z_1), \ldots, M_n(z_n)$ of the algebraic numbers $\alpha_1, \ldots, \alpha_n$.

Let $f = \sum_{e_i \in \mathbb{Z}^k_{\ge 0}} a_{e_i} X^{e_i} \in L[x_1, \ldots, x_k, y]$. Since B_L is a basis for L, we have $a_{e_i} = \sum_{j=1}^d C_{e_i j} b_j$ for $b_j \in B_L$ and $C_{e_i j} \in \mathbb{Q}$. We define the **coordinate vector** of f w.r.t. B_L as the vector of dimension d, denoted by $[f]_{B_L} = [v_1, \ldots, v_d]^T$, where $v_j = \sum_{e_i \in \mathbb{Z}^k_{\ge 0}} C_{e_i j} X^{e_i}$.

Example 1. Let $\mathbb{Q}(\sqrt{5}, \sqrt{11}) \cong L$ where $L = \mathbb{Q}[z_1, z_2]/\langle z_1^2 - 5, z_2^2 - 11\rangle$ with $\phi(\sqrt{5}) = z_1$ and $\phi(\sqrt{11}) = z_2$. Let $B_L = \{1, z_2, z_1, z_1 z_2\}$ be a basis for L. If $f = 3z_1 x + 2y + z_2 + 4z_1 z_2 \in L[x, y]$, then $[f]_{B_L} = [2y, 1, 3x, 4]^T$.

Our modular resultant algorithm, which we call MRES, incorporates a preprocessing step to remove fractions. It replaces the minimal polynomials $M_1(z_1)$, …, $M_n(z_n)$ and the input polynomials f_1, f_2 with their semi-associates, defined in Definition 1.

Definition 1. *Let $L_{\mathbb{Z}} = \mathbb{Z}[z_1, \ldots, z_n]$. For any $f \in L[x]$, the **denominator** of f, denoted by $den(f)$, is the smallest positive integer such that $den(f) f \in L_{\mathbb{Z}}[x]$. The **associate** of f is defined as $\tilde{f} = den(h)h$ where $h = monic(f)$. The **semi-associate** of f, denoted by $\check{f}$, is defined as rf, where r is the smallest positive rational number for which $den(rf) = 1$.*

Example 2. Consider L as described in Example 1 and let $f = \frac{7}{5} z_1 x + z_2 \in L[x]$. We have $den(f) = 5$, $\mathrm{monic}(f) = x + \frac{1}{7} z_1 z_2$, $\tilde{f} = 7x + z_1 z_2$, and $\check{f} = 7 z_1 x + 5 z_2$.

When $\mathrm{lc}(f_1)$ and $\mathrm{lc}(f_2)$ are complicated algebraic numbers, computing associates can be expensive. Instead, by employing semi-associates we can effectively remove fractions from the inputs. After eliminating fractions from the inputs, Algorithm MRES computes $\mathrm{res}(f_1, f_2)$ modulo a sequence of primes. Let p be a prime such that $p \nmid \prod_{i=1}^n \mathrm{lc}(\check{M}_i) \cdot \mathrm{lc}(\check{f}_1) \cdot \mathrm{lc}(\check{f}_2)$. Let $m_i(z_i) = \check{M}_i \bmod p$ for $1 \le i \le n$. Define $L_p = \mathbb{Z}_p[z_1, \ldots, z_n]/\langle m_1, \ldots, m_n\rangle$. The finite ring L_p has p^d elements which may include zero divisors. To reconstruct the rational coefficients of the potential resultant, MRES employs the Chinese remaindering (CRT) and rational number reconstruction (RNR) [12,8], respectively. Example 3 demonstrates how MRES manages zero-divisors in L_p and it emphasizes the rationale for employing a primitive element.

Example 3. Let $f_1 = x^3 + \frac{1}{5}yz_2 - z_1$ and $f_2 = z_2x + 4yz_1$ be two polynomials over $L = \mathbb{Q}[z_1, z_2]/\langle z_1^2 - 2, z_2^2 - 7\rangle$ and let $M_1(z_1) = z_1^2 - 2$ and $M_2(z_2) = z_2^2 - 7$. Assume that MRES chooses the prime $p_1 = 7$. Thus, $m_1 = M_1 \mod p_1 = z_1^2 + 5$, $m_2 = M_2 \mod p_1 = z_2^2$ and

$$L_{p_1} = \mathbb{Z}_7[z_1, z_2]/\langle\, z_1^2 + 5, z_2^2\,\rangle.$$

Next, MRES picks an evaluation point $y = \beta \in \mathbb{Z}_7$ and attempts to compute the $\operatorname{res}(f_1(x, \beta), f_2(x, \beta), x) \in L_{p_1}$ using the monic Euclidean algorithm (MEA) (see Theorem 4). However, the MEA fails since the $\operatorname{lc}(f_2(x, \beta)) = z_2$ is not invertible over L_{p_1}. Since MRES cannot identify whether this failure is due to the choice of the prime p_1 or the evaluation point β, it aborts the computation of $\operatorname{res}(f_1, f_2)$ modulo p_1 and tries another prime, say $p_2 = 3$. In this case, we have

$$L_{p_2} = \mathbb{Z}_3[z_1, z_2]/\langle\, z_1^2 + 1, z_2^2 + 2\,\rangle.$$

MRES picks $y = \beta \in \mathbb{Z}_3$ randomly and computes $\operatorname{res}(f_1(x, \beta), f_2(x, \beta)) \in L_{p_2}$ using the MEA. This time $\operatorname{lc}(f_2(x, \beta)) = z_2$ is a unit in L_{p_2} and the MEA succeeds and outputs $\operatorname{res}(f_1(x, \beta), f_2(x, \beta)) \in L_{p_2}$. MRES iterates this procedure for additional β values and primes and eventually recovers $\operatorname{res}(f_1, f_2, x) = 128z_1y^3 - \frac{49}{5}y + 7z_1z_2$ through polynomial interpolation for y, followed by CRT and RNR [12,8] to recover the coefficients $128, -\frac{49}{5}$ and 7.

The majority of computational tasks within MRES take place within the finite ring L_p. To speed up MRES, we employ a primitive element to speed up arithmetic operations within L_p. That is, instead of computing over a ring with multiple extensions, L_p, we do the computation over a quotient ring with a single extension. Furthermore, our Maple implementation of MRES utilizes 31-bit primes avoiding zero-divisors in L_p with high probability.

1.2 Organization of the Paper

Following a review of preliminaries in Section 2, we present an algorithm to compute the resultant of two univariate polynomials over L. In Section 3, we present our modular algorithm, MRES, and its subalgorithms. Some implementation details and two timing benchmarks are described in Section 4. We compute the expected time complexity of the MRES algorithm in Section 5. Finally, in Section 6, we study the failure probability of our MRES algorithm.

2 Preliminaries

2.1 Mapping $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ to a single extension $\mathbb{Q}(\gamma)$

We use Ansari and Monagan's method in [1] to identify a primitive element for $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ called γ and compute its minimal polynomial. To do so, Ansari and Monagan used Algorithm 1 over $\mathbb{F} = \mathbb{Z}_p$ where p is a 31-bit prime. Then, they construct the quotient ring $\bar{L}_p = \mathbb{Z}_p[z]/\langle M(z)\rangle$ where $M(z)$ is the characteristic polynomial of γ modulo p.

Algorithm 1: LAminpoly

Require: A list of the minimal polynomials $[m_1(z_1), \ldots, m_n(z_n)]$, the ground field $\mathbb{F} = \mathbb{Z}_p$ over which the computation is performed, and $\gamma = z_1 + C_1 z_2 + \ldots + C_{n-1} z_n$ where $0 \neq C_i \in \mathbb{Z}$ for $1 \leq i \leq n-1$

Ensure: Either a message FAIL or a polynomial $M(z) \in \mathbb{F}[z]$ such that $M(\gamma) = 0$, the matrix A and A^{-1} .

$B_{L_p} = \{ \prod_{i=1}^{n} (z_i)^{e_i} \; 0 \leq e_i < d_i \}$ s.t. $d_i = \deg(m_i(z_i))$ // A basis for L_p
$d = \prod_{i=1}^{n} d_i$
Initialize A to be a $d \times d$ zero matrix over $\mathbb{F}$.
$g_0 = 1$
for $i = 1$ to d **do**
    Set column i of A to be $[g_{i-1}]_{B_{L_p}}$
    $g_i = \gamma \cdot g_{i-1}$
end for
if $\det(A) = 0$ **then return**(FAIL) **end if**
Compute A^{-1} and set $q = A^{-1} \cdot (-[g_d]_{B_{L_p}})$
Construct the polynomial $M(z) = z^d + q_d z^{d-1} + \ldots + q_2 z + q_1$
return($M(z)$, A, A^{-1})

Example 4. Let $\mathbb{Q}(\sqrt{2}, \sqrt{3\sqrt{2}+1})$ and $M_1(z_1) = z_1^2 - 2$ be the minimal polynomial of $\alpha_1 = \sqrt{2}$ over $\mathbb{Q}$ and $M_2(z_2) = z_2^2 - 3z_1 + 1$ be the minimal polynomial of $\alpha_2 = \sqrt{3\sqrt{2}+1}$ over $\mathbb{Q}[z_1]/\langle z_1^2 - 2\rangle$. Thus

$$\mathbb{Q}(\sqrt{2}, \sqrt{3\sqrt{2}+1}) \cong L = \mathbb{Q}[z_1, z_2]/\langle z_1^2 - 2, z_2^2 - 3z_1 + 1\rangle.$$

Let us choose $p = 7$ so the ground field is $\mathbb{F} = \mathbb{Z}_7$. After reducing minimal polynomials modulo p, we have $L_7 = \mathbb{Z}_7[z_1, z_2]/\langle z_1^2 + 5, z_2^2 + 4z_1 + 1\rangle$. The dimension of L_7 as a $\mathbb{Q}-$vector space is 4 and $B_{L_p} = \{1, z_2, z_1, z_1 z_2\}$ is a basis for it. We aim to find a primitive element γ such that $\mathbb{Z}_7(\gamma) \cong L_7$, and compute its characteristic polynomial $M(z)$ so we can construct $\bar{L}_7 = \mathbb{Z}_7[z]/\langle M(z)\rangle$ such that $\bar{L}_7 \cong L_7$. Let us try $\gamma = 2z_1 + z_2$. We first construct the 4×4 matrix A whose i'th column in $[\gamma^i]_{B_{L_p}}$ for $0 \leq i \leq 3$. We obtain

$$A = \begin{bmatrix} 1 & 0 & 7 & 36 \\ 0 & 1 & 0 & 23 \\ 0 & 2 & 3 & 10 \\ 0 & 0 & 4 & 3 \end{bmatrix}.$$

Since $\det(A) = 153 \mod 7 \neq 0$, we consider $\gamma = 2z_1 + z_2$ as a primitive element of $\mathbb{Z}_7(\sqrt{2}, \sqrt{3\sqrt{2}+1})$. Since $153 = 3^2 \cdot 17$, if we had chosen $p = 3$ or $p = 17$, then $\det(A) = 0 \mod p$ and A would not be invertible. We call 3 and 17 det-bad primes and define them in Section 3.2. Computing $q = A^{-1} \cdot (-[\gamma^4]_{B_L})$, we construct the characteristic polynomial $M(z) = z^4 + z$ and finite ring $\bar{L}_7 = \mathbb{Z}_7[z]/\langle z^4 + z\rangle$.

If $\det(A) \neq 0$, we can define the isomorphism $\phi_\gamma : L_p \longrightarrow \bar{L}_p$. To do so, let B_{L_p} and $B_{\bar{L}_p}$ be bases for L_p and $\bar{L}_p$, respectively. Let $C : L_p \longrightarrow \mathbb{Z}_p^d$ be a bijection such that $C(a) = [a]_{B_{L_p}}$. Let $D : \bar{L}_p \longrightarrow \mathbb{Z}_p^d$ be another bijection such that $D(b) = [b]_{B_{\bar{L}_p}}$. Define $\phi_\gamma : L_p \longrightarrow \bar{L}_p$ with $\phi_\gamma(a) = D^{-1}(A^{-1} \cdot C(a))$. Furthermore, $\phi_\gamma^{-1} : \bar{L}_p \longrightarrow L_p$ is given by $\phi_\gamma^{-1}(b) = C^{-1}(A \cdot D(b))$.

Lemma 1. *(See Lemma 1 in [1]) If $\det(A) \neq 0$, then the mapping ϕ_γ defined above is a ring isomorphism.*

Isomorphism ϕ_γ induces the natural isomorphism $\phi_\gamma : L_p[x_1,\ldots,x_k,y] \longrightarrow \bar{L}_p[x_1,\ldots,x_k,y]$. Example 5 illustrates how ϕ_γ works.

Example 5. We continue Example 4, where $L_7 = \mathbb{Z}_7[z_1,z_2]/\langle z_1^2 - 2, z_2^2 - 3z_1 + 1\rangle$ and $\bar{L}_7 = \mathbb{Z}_7[z]/\langle z^4 + z\rangle$. Let $B_{\bar{L}_7} = \{1, z, z^2, z^3\}$ and $B_{L_7} = \{1, z_2, z_1, z_1z_2\}$ be bases for $\bar{L}_7$ and L_7, respectively. Let $f = x^2z_1z_2 + 2xy + z_2 \in L_7[x,y]$. We wish to compute $\phi_\gamma(f) \in \bar{L}_7[x,y]$. To do so, we first need to compute $C(f) = [f]_{B_{L_p}} = [2xy, 1, 0, x^2]^T$ which is the coordinate vector of f relative to B_{L_p}. Then, we have

$$b = A^{-1} \cdot C(f) = A^{-1}[f]_{B_{L_p}} = [3x^2 + 2xy + 1, 6x^2 + 3, 6x^2 + 6, 4x^2 + 6]^T$$

as the coordinate vector of $\phi_\gamma(f)$ relative to $B_{\bar{L}_p} = \{1, z, z^2, z^3\}$. Consequently,

$$\phi_\gamma(f) = (4x^2 + 6)z^3 + (6x^2 + 6)z^2 + (6x^2 + 3)z + 3x^2 + 2xy + 1 \in \bar{L}_7[x_1, x_2].$$

2.2 Resultants

Let R be a commutative ring with identity $1 \neq 0$.

Definition 2. *Let $f_1 = \sum_{j=0}^m a_j y^j$ and $f_2 = \sum_{j=0}^n b_j y^j$ be two non-zero polynomials where $m = \deg(f_1, y)$, $n = \deg(f_2, y)$, and $a_j, b_j \in R[x_1,\ldots,x_k]$. The Sylvester matrix of f_1 and f_2 w.r.t. the variable y is the $(m+n) \times (m+n)$ matrix*

$$\mathrm{sylv}(f_1, f_2, y) = \begin{bmatrix} a_m & \cdots & a_0 & & \\ & a_m & \cdots & a_0 & \\ & & \ddots & & \\ & & a_m & \cdots & a_0 \\ b_n & \cdots & b_0 & & \\ & & \ddots & & \\ & & b_n & \cdots & b_0 \end{bmatrix}$$

in which there are n rows of f_1 coefficients, m rows of f_2 coefficients, and all entries not shown are zero. The resultant of f_1 and f_2 w.r.t. the variable y is defined as

$$res(f_1, f_2, y) = \det(\text{sylv}(f_1, f_2, y))$$

which is a polynomial in $R[x_1, \ldots, x_k]$. If f_1, f_2 are univariate polynomials in $R[x]$, we write $\text{res}(f_1, f_2)$ *for* $\text{res}(f_1, f_2, x)$.

Theorem 1. *(See theorem 9.2 and 9.3 of [7]) Let $f_1, f_2 \in R[x]$ with $\deg f_1 = m > 0$ and $\deg f_2 = n > 0$. Let $c \in R$ and $\phi : R \to S$ be a ring homomorphism. Then*

(i) $\text{res}(c, f_2) = c^n$.
(ii) $\text{res}(f_1, f_1) = 0$.
(iii) $\text{res}(f_1, f_2) = (-1)^{nm}\text{res}(f_2, f_1)$.
(iv) $\text{res}(cf_1, f_2) = c^n\text{res}(f_1, f_2)$.
(v) If $\deg(\phi(f_1)) = m$ *and* $\deg(\phi(f_2)) = k$ *where* $0 \le k \le n$, *then* $\phi(\text{res}(f_1, f_2)) = (\phi(a_m))^{n-k}\text{res}(\phi(f_1), \phi(f_2))$.
(vi) If $\deg(\phi(f_1)) = m$ *and* $\deg(\phi(f_2)) = n$ *then* $\phi(\text{res}(f_1, f_2)) = \text{res}(\phi(f_1), \phi(f_2))$.

Theorem 2. *Let $f_1, f_2 \in R[x]$ and suppose $g = \gcd(f_1, f_2)$ exists. Then $\deg(g, x) > 0$ if and only if $\text{res}(f_1, f_2) = 0$.*

Proof. Corollary (Sylvester's Criterion) chapter 7 [7].

2.3 Computing Resultants of Univariate Polynomials

Let $f_1, f_2 \in R[x]$. In this section, we describe how $\text{res}(f_1, f_2) \in R$ can be computed using the Monic Euclidean Algorithm.

Definition 3. *Let $f \in R[x]$. If $f = 0$, we define $monic(f) = 0$. Otherwise, we define $monic(f) = \text{lc}(f)^{-1}f$, where $\text{lc}(f)$ is the leading coefficient of f. If $\text{lc}(f)$ is not unit in R, then $monic(f) =$ "$failed$". We say f is monic if $f = monic(f)$.*

Our modular resultant algorithm attempts to compute the resultant of two univariate polynomials over $\bar{L}_p$, a finite ring. In $\bar{L}_p$, elements are either zero, units, or zero-divisors. Thus, $monic(f) =$ "$failed$" means that the algorithm encountered a zero-divisor. Let $f_1, f_2 \in R[x]$ such that $0 \le \deg(f_2) \le \deg(f_1)$. Algorithm 2, the Monic Euclidean Algorithm takes f_1 and f_2 as its inputs and returns either a message "FAIL" or the monic gcd of f_1 and f_2.

Definition 4. *Given $f_1, f_2 \in R[x]$ with $\deg(f_2) \le \deg(f_1)$, assume that the Monic Euclidean Algorithm (MEA) does not fail for f_1 and f_2 and terminates after l iterations. We define the Monic Polynomial Remainder Sequence, m.p.r.s., generated by polynomials f_1 and f_2 as the sequence $r_1, r_2, \ldots, r_l, r_{l+1}$ obtained from the execution of the Monic Euclidean Algorithm such that $r_1 = f_1$, $r_2 = f_2$, $r_3 = r_1 - M_2 q_3$, and $r_{i+2} = M_i - M_{i+1} q_{i+1}$ with $M_i = monic(r_i)$ and $\deg(r_{i+1}) < \deg(r_i)$ for $2 \le i \le l-1$ and $r_{l+1} = 0$.*

Algorithm 2: Monic Euclidean Algorithm

Require: $f_1, f_2 \in R[x]$ such that $0 \leq \deg(f_2) \leq \deg(f_1)$ and R is a commutative ring with identity $1 \neq 0$.
Ensure: Either the monic $\gcd(f_1, f_2)$ or FAIL.

$r_1, r_2 = f_1, f_2$
$M_1, i = r_1, 2$
while $r_i \neq 0$ **do**
  $M_i = monic(r_i)$
  if $M_i = failed$ **then return**(*FAIL*) `// The algorithm encountered a zero-divisor.`
  Set r_{i+1} to be the remainder of M_{i-1} divided by M_i
  Set $i = i + 1$
end while
$l = i - 1$
return(M_l)

Remark 1. The remainders appearing in m.p.r.s. are not monic polynomials. We call them Monic Polynomial Remainder Sequence since they are obtained from the MEA.

Theorem 3. *Let $f_1, f_2 \in R[x]$ such that $\mathrm{lc}(f_2)$ is a unit and $f_1 = f_2 q + r$ where $r, q \in R[x]$ and $\deg(r) < \deg(f_2)$ or $r = 0$. Let $n_1 = \deg(f_1)$, $n_2 = \deg(f_2)$, and $n_r = \deg(r)$. Then*

$$\mathrm{res}(f_2, f_1) = \mathrm{lc}(f_2)^{n_1 - n_r} \mathrm{res}(f_2, r)$$

Proof. [5], section 3.5, exercise 16 part b.

Theorem 4. *(m.p.r.s.)*
Suppose that $f_1, f_2 \in \bar{L}_p[x]$ and the Monic Euclidean Algorithm does not fail for f_1 and f_2. Let $r_1, r_2, \ldots, r_l, r_{l+1}$ be the m.p.r.s. generated by f_1 and f_2 where $r_{l+1} = 0$. Let $n_i = \deg(r_i)$ for $1 \leq i \leq l$. If $\deg(r_l) > 0$, then $\mathrm{res}(f_1, f_2) = 0$. Otherwise, we have

$$\mathrm{res}(f_1, f_2) = (-1)^v \left(\prod_{i=2}^{l-1} \mathrm{lc}(r_i)^{n_{i-1}}\right) \mathrm{lc}(r_l)^{n_{l-1}}$$

where $v = \sum_{i=1}^{l-2} n_i n_{i+1}$.

Proof. If $\deg(r_l) \neq 0$, then the monic $\gcd(f_1, f_2) \neq 1$. Applying Theorem 2, we have $\mathrm{res}(f_1, f_2) = 0$. On the other hand, in the first step of Algorithm 2, we have $M_1 = M_2 q_3 + r_3$ where $M_1 = f_1$, $M_2 = monic(f_2)$ and $\deg(r_3) < \deg(M_2)$. According to Theorem 3, since $\mathrm{lc}(M_2) = 1$, we have $\mathrm{res}(M_2, M_1) = \mathrm{res}(M_2, r_3)$. We have,

$$\begin{aligned}
\mathrm{res}(M_2, M_1) &= \mathrm{res}(\mathrm{lc}(f_2)^{-1} f_2, f_1) \\
&= (\mathrm{lc}(f_2)^{-1})^{n_1} \mathrm{res}(f_2, f_1) \\
&= (-1)^{n_1 n_2} (\mathrm{lc}(f_2)^{-1})^{n_1} \mathrm{res}(f_1, f_2)
\end{aligned}$$

Thus, $\mathrm{res}(M_2, r_3) = (-1)^{n_1 n_2} (\mathrm{lc}(f_2)^{-1})^{n_1} \mathrm{res}(f_1, f_2)$. Continuing this process, in the i-th step of the MEA, where $M_i = M_{i+1} q_{i+2} + r_{i+2}$, we have $\mathrm{res}(M_i, r_{i+1}) = (-1)^v (\prod_{j=2}^{i} (\mathrm{lc}(r_j)^{-1})^{n_{j-1}}) \mathrm{res}(f_1, f_2)$ where $v = \sum_{j=2}^{i} n_{j-1} n_j$. Moreover, in the last step of the MEA, since $r_l = c \in \bar{L}_p$, we have $\mathrm{res}(M_{l-1}, r_l) = r_l^{n_{l-1}} = \mathrm{lc}(r_l)^{n_{l-1}}$ which implies the result.

Applying Theorem 4, we can modify the MEA to compute the resultant of two univariate polynomials $f_1, f_2 \in R[x]$ in Algorithm 3. This algorithm is used in the base case of Algorithm 4, line 1.

Algorithm 3: URES

Require: $f_1, f_2 \in R[x]$ such that $0 \le \deg(f_2) \le \deg(f_1)$ where R is a commutative ring with identity $1 \neq 0$.
Ensure: Either $\mathrm{res}(f_1, f_2)$ or FAIL.

$r_1 = f_1$, $r_2 = f_2$, $i = 2$
$M_1 = r_1$, $R = 1$, $v = 0$
$n_1 = \deg(f_1)$,$n_2 = \deg(f_2)$
while $r_i \neq 0$ **do**
  $M_i = monic(r_i)$
  if $M_i = failed$ **return** *(FAIL)*// The algorithm encounters a zero-divisor.
  Set r_{i+1} to be the remainder of M_{i-1} divided by M_i
  Set $n_{i+1} = \deg(r_{i+1})$
  if $n_{i+1} < 0$ and $n_i \neq 0$ **then return**(0) // If $\gcd(f_1, f_2)$ is not a constant, then $\mathrm{res}(f_1, f_2) = 0$
  Set $R = R \cdot \mathrm{lc}(r_i)^{n_{i-1}}$
  Set $v = v + n_i n_{i-1}$
  Set $i = i + 1$
end while
$R = (-1)^v R$
return(R)

Example 6. Let $f_1, f_2 \bar{L}_3[x]$ where $\bar{L}_3 = \mathbb{Z}_3[z]/\langle z^2 - 2 \rangle[x]$. On input of $f_1 = x^3 + (2z)x + 1$ and $f_2 = 2x^2 + xz$ Algorithm 3 returns $R = z + 1$ the resultant of f_1 and f_2. Table 1 shows the intermediate values.

Table 1. Example 6

Dividend	Divisor	Remainder	R
M_1	$M_2 = \mathrm{monic}(f_2) = x^2 + 2xz$	$r_3 = (2z + 2)x + 1$	$R = 2$
M_2	$M_3 = \mathrm{monic}(r_3) = x + 2z + 1$	$r_4 = 2z + 1$	$R = z$
M_3	$M_4 = \mathrm{monic}(r_4) = 1$	$r_5 = 0$	$R = z + 1$

3 The Modular Resultant Algorithm

Let $f_1, f_2 \in L[x_1, \ldots, x_k, y]$. In this section, we present a modular algorithm to compute $r = \text{res}(f_1, f_2, y)$. We can present $f_1 = \sum_{i=0}^{m} a_i y^i \in L[x_1, \ldots, x_k][y]$ and $f_2 = \sum_{i=0}^{n} b_i y^i \in L[x_1, \ldots, x_k][y]$, where $\deg(f_1, y) = m$, $\deg(f_2, y) = n$, and $a_i, b_i \in L[x_1, \ldots, x_k]$. In general, modular algorithms use two fundamental homomorphisms, the modular and evaluation homomorphisms. The modular homomorphism, $\phi_p : \mathbb{Z} \longrightarrow \mathbb{Z}_p$, maps integers into their remainders modulo p. We choose p to be a prime so $\mathbb{Z}_p$ is a finite field. This homomorphism is used to prevent the growth of integer coefficients of algebraic numbers in MEA. Let $R = \bar{L}_p[x_k]$ and $R' = \bar{L}_p$. We define the evaluation homomorphism $\phi_{x_k=\beta}$: $R[x_1, \ldots, x_{k-1}, y] \longrightarrow R'[x_1, \ldots, x_{k-1}, y]$ such that $\phi_{x_k=\beta}(f) = f(\beta)$.

Our modular resultant algorithm, MRES, first computes the resultant modulo a sequence of primes. For each prime, MRES calls PRES which uses the evaluation homomorphism and interpolation to calculate the resultant of f_1 and f_2 over $\bar{L}_p$. Subsequently, MRES employs CRT and RNR to reconstruct the rational coefficients of the resultant. However, the successful reconstruction of the resultant is not guaranteed for all primes and evaluation points. In Section 3.1 and Section 3.2 we identify problematic evaluation points and primes, respectively.

3.1 Algorithm PRES

Let p be a large prime. To compute $\text{res}(f_1, f_2, y)$ for $f_1, f_2 \in \bar{L}_p[x_1, \ldots, x_k][y]$, Algorithm PRES, Algorithm 4, uses evaluation and dense interpolation as in [2]. PRES is recursive. If $f_1, f_2 \in \bar{L}_p[y]$, PRES computes $\text{res}(f_1, f_2) \in \bar{L}_p$ using Theorem 4. Otherwise, PRES chooses $\beta \in \mathbb{Z}_p$ randomly and in Step 9 reduces f_1 and f_2 to polynomials in $\bar{L}_p[x_1, \ldots, x_{k-1}][y]$ by evaluating them at $x_k = \beta$. To apply Theorem 1 (vi) we need that the leading coefficients of f_1 and f_2 do not vanish at $x_k = \beta$. Subsequently, Algorithm PRES recursively computes

$$R_\beta = \text{res}(f_1(x_1, \ldots, x_{k-1}, \beta, y), f_2(x_1, \ldots, x_{k-1}, \beta, y)) \in \bar{L}_p[x_1, x_2, \ldots, x_{k-1}].$$

Next, Algorithm PRES interpolates x_k in $\text{res}(f_1, f_2, y)$. Let $m = \deg(f_1, y)$, $n = \deg(f_2, y)$, $d_1 = \deg(f_1, x_k)$ and $d_2 = \deg(f_2, x_k)$. From Sylvester's matrix we have $\deg(\text{res}(f_1, f_2, x_k) \leq nd_1 + md_2$ thus we need at most $nd_1 + md_2 + 1$ evaluation points.

We emphasize that not all choices for β lead to a successful computation of the resultant mod p. Definition 5 classifies the problematic evaluation points.

Definition 5. *Let $f_1, f_2 \in \bar{L}_p[x_1, \ldots, x_k, y]$. Assume that $\text{res}(f_1, f_2, y)$ exists. Let $\beta \in \mathbb{Z}_p^k$ and let $x_k = \beta_k, x_{k-1} = \beta_{k-1}, \ldots, x_1 = \beta_1$ be an evaluation point. We identify three types of evaluation points as follows:*

- ***Lc-bad Evaluation Points:*** *Let $f_1, f_2 \in \bar{L}_p[x_1, \ldots, x_k][y]$. We call β an lc-bad evaluation point if $\text{lc}(f_1)(\beta) = 0$ or $\text{lc}(f_2)(\beta) = 0$.*

Algorithm 4: PRES

Require: $f_1, f_2 \in \bar{L}_p[x_1, \ldots, x_k][y]$
Ensure: $\mathrm{res}(f_1, f_2, y) \in \bar{L}_p[x_1, \ldots, x_k]$ or FAIL
1: **if** $k = 0$ **return**($\mathrm{URES}(f_1, f_2)$) // $f_1, f_2 \in \bar{L}_p[y]$
2: $(m, n) = \deg(f_1, y), \deg(f_2, y)$
3: $B = n \deg(f_1, x_k) + m \deg(f_2, x_k)$
4: **for** $j = 0$ to B **do**
5: Pick a new evaluation point β at random from $\mathbb{Z}_p$ such that β is not lc-bad
6: $F_{1\beta} = f_1(x_k = \beta)$ and $F_{2\beta} = f_2(x_k = \beta)$
7: $R_\beta = PRES(F_{1\beta}, F_{2\beta}, y)$
8: **if** $R_\beta = FAIL$ **then return**(FAIL) **end if**
9: **if** $j = 0$ **then**
10: $(R, prod) = (R_\beta, (x - \beta))$ `// First iteration`
11: **else**
12: // Interpolate x_k in the resultant, R, incrementally
13: $V_\beta = prod(x_k = \beta)^{-1} \cdot (R_\beta - R(x_k = \beta))$
14: $R = R + V_\beta \cdot prod$
15: $prod = prod \cdot (x_k - \beta)$
16: **end if**
17: **end for**
18: **return**(R)

– ***Zero-Divisor Evaluation Points:*** *If β is not lc-bad we call β a zero-divisor evaluation point if Algorithm 3 when called by Algorithm PRES in step 3 tries to invert a zero-divisor in $\bar{L}_p$.*
– ***Good Evaluation Points:*** *If β is neither lc-bad nor a zero-divisor evaluation point call β a good evaluation point.*

Example 7. Let $f_1 = (x+1)y^3 + xz$ and $f_2 = (x+z)y + zx$ be two polynomials in $\bar{L}_7[x][y]$ where $\bar{L}_7 = \mathbb{Z}_7[z]/\langle z^2 \rangle$. The evaluation point $x = 6$ is an lc-bad evaluation point since $\mathrm{lc}(f_1)(6) = 0 \mod 7$. If we choose $x = 0$, then $f_1(0, y) = y^3$ and $f_2(0, y) = yz$. Since $\mathrm{lc}(f_2)(0) = z$ is not invertible over $\bar{L}_7$, Algorithm 3 fails to compute the resultant of $f_1(0, y)$ and $f_2(0, y)$ which implies that $x = 0$ is a zero-divisor evaluation point.

3.2 Algorithm MRES

Algorithm MRES, presented as Algorithm 5, computes the resultant of two polynomials $f_1, f_2 \in L[x_1, \ldots, x_k, y]$. MRES first replaces f_1, f_2 with their semi-associates. After applying ϕ_p to map the coefficients in L to L_p, MRES uses ϕ_γ to convert the polynomials over L_p to their corresponding polynomials over $\bar{L}_p$. Subsequently, MRES calls PRES to compute $\mathrm{res}(f_1, f_2, y) \in \bar{L}_p[x_1, \ldots, x_k]$. Let R_p be the output of PRES. If $R_p = FAIL$ then when PRES called URES in Step 2, a zero divisor was encountered. MRES chooses a new prime. In step 12, MRES converts $R_p \in \bar{L}_p[x_1, \ldots, x_k]$ to its corresponding polynomial over L_p. Employing, CRT and RNR, MRES algorithm tries to reconstruct rational coefficients of R_p. If RNR does not fail and the current result of RNR, denoted by H, is equal to the previous result of RNR, then MRES returns H as the $\mathrm{res}(f_1, f_2, y)$. Therefore MRES is a Monte Carlo algorithm. It can output an incorrect answer with low probability.

Algorithm 5: MRES

Require: $f, g \in L[x_1, \ldots, x_k, y]$ and $\mathbb{P}$ a large set of primes.
Ensure: $\mathrm{res}(f, g, y) \in L[x_1, \ldots, x_k]$.

1: $presult = 0$
2: $M = 1$
3: $f, g = \check{f}, \check{g}$ `// Clear fractions`
4: **while** true **do**
5: Choose a new prime p from $\mathbb{P}$ at random that is not lc-bad.
6: Choose $C_1, \ldots, C_{n-1}$ from $[1, p)$ at random and set $\gamma = z_1 + \sum_{i=2}^{n} C_{i-1} z_i$
7: Call Algorithm 1 with inputs $[\phi_p(\check{M}_1), \ldots, \phi_p(\check{M}_n)]$, $\mathbb{Z}_p$ and $\phi_p(\gamma)$ to compute $M(z)$, A, and A^{-1} `// check if` p `is a det-bad prime`
8: **if** Algorithm 1 returns FAIL **then** Go back to step 5 **end if**
9: $R_p = PRES(\phi_\gamma(\phi_p(\check{f}_1)), \phi_\gamma(\phi_p(\check{f}_2)), y) \in \bar{L}_p[x_1, \ldots, x_k]$
10: **if** R_p= FAIL **then** Go back to step 5. **end if** `// a zero divisor was encountered in URES`
11: $R_p = \phi_\gamma^{-1}(R_p)$ `// Convert` R_p `over` $\bar{L}_p$ `to its corresponding polynomial over` L_p
12: **if** $M = 1$ **then**
13: $R, M := R_p, p$; `// First iteration`
14: **else**
15: Using the CRT, compute $R' \equiv R \mod M$ and $R' \equiv R_p \mod p$
16: Set $R = R'$ and $M = M \cdot p$
17: **end if**
18: $H :=$ Rational Number Reconstruction of R mod M
19: **if** $H \neq$ FAIL **then if** $H = presult$ **return**(H) **else** $presult = H$ **end if**
20: **end while**

As mentioned before, not all the primes result in a successful reconstruction of the resultant. Definition 6 distinguishes four types of primes.

Definition 6. *Let $f_1, f_2 \in L[x_1, \ldots, x_k][y]$ and p be a prime.*

- ***Lc-bad Primes:*** *If p divides either $\mathrm{lc}(\check{f}_1)$, $\mathrm{lc}(\check{f}_2)$, or any $\mathrm{lc}(\check{M}_i(z_i))$ for $1 \leq i \leq n$, we call p an lc-bad prime.*
- ***Det-bad Primes:*** *Let A be the matrix obtained from Algorithm 1 over $F = \mathbb{Z}_p$. If $\det(A) = 0$, then p is called a* det-bad *prime.*
- ***Zero-Divisor Primes:*** *If p is neither an lc-bad nor a det-bad prime and there exists r_i among the m.p.r.s., Definition 4, such that $\mathrm{lc}(r_i)$ is not invertible over $\bar{L}_p$, then p is called a zero-divisor prime. In other words, p is a zero-divisor prime if Algorithm 3 fails for p.*
- ***Good Primes:*** *If p is neither lc-bad, det-bad, nor a zero-divisor prime, we define it as a good prime.*

Example 8. Let $f_1 = 23z_2x + z_1y$ and $f_2 = (z_2 + 5)x + z_1y$ be two polynomials listed in the lexicographic order with $x > y$ over $L[x, y]$ where $L = \mathbb{Q}[z_1, z_2]/\langle z_1^2 - 2, z_2^2 - 3\rangle$. Then, $p = 23$ is an lc-bad prime since $\mathrm{lc}(f_1) = 0 \mod p$. Moreover, $p = 11$ is a zero-divisor prime because $\mathrm{lc}(f_2) = z_2 + 5$ is not invertible over $\mathbb{Z}_{11}[z_1, z_2]/\langle z_1^2 - 2, z_2^2 - 3\rangle$ as $z_2^2 - 3 \mod 11 = (z_2 + 5)(z_2 + 6)$.

4 Implementation and Benchmarks

We have implemented algorithm MRES and its subalgorithms in Maple [9]. We use the recursive dense data structure from [11] to represent elements of $L = \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ and polynomials in $L[x_1, \ldots, x_k]$. For the set of primes $\mathbb{P}$ we use 31 bit primes.

We present two timing benchmarks. All timings were obtained on Intel Core i7-6700. In both Table 2 and Table 3, column N denotes the number of primes needed by MRES, and column MRES 1 is the time for our algorithm, MRES, using ϕ_γ and computing over $\bar{L}_p$. Column MRES 2 is the time for MRES if we do not use ϕ_γ and compute over L_p. Column LAMP is the time spent in Algorithm 1. For both algorithms, column PRES is the time spent in Algorithm 4. The speedup achieved by employing ϕ_γ can be observed by comparing columns PRES for MRES 1 and MRES 2.

The first benchmark, Table 2, presents timings of the resultant computations in $L[x, y]$ where the number field $L = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7})$ has degree 16. In Table 2, the input polynomials f_1 and f_2 have degree m in x and y and $\mathrm{res}(f_1, f_2, x)$ has degree r_y in y. The coefficients of the input polynomials, f_1 and f_2, are polynomials in z_1, z_2, z_3, and z_4 with coefficients chosen randomly from $[1, 9)$.

Table 2. Timings in CPU seconds for computing $\mathrm{res}(f_1, f_2, x)$, the resultant of f_1 and f_2 of degree m in $L[x, y]$.

m	r_y	N	MRES 1			MRES 2	
			time	LAMP	PRES	time	PRES
2	4	4	0.313	0.126	0.140	0.828	0.828
4	16	4	0.828	0.187	0.501	8.609	8.563
6	36	7	3.938	0.189	3.218	59.938	59.610
8	64	11	14.171	0.218	11.891	291.281	289.875
10	100	16	47.500	0.468	48.842	967.437	962.609
12	144	22	119.766	0.596	103.016	> 1000	> 1000
14	196	29	282.844	0.798	244.189	> 1000	> 1000

The second benchmark, Table 3, shows timings for computing the resultant of two polynomials f_1 and f_2 in $L[x, y]$, where $L = \mathbb{Q}(\alpha_1, \alpha_2, \alpha_3)$. Let $M_1 = z_1^2 - 2$, $M_2 = z_2^2 - 3$, and $M_3 = \sum_{j=0}^{d_3} z_3^j + z_1z_2$ be the minimal polynomials of α_1, α_2, and α_3, respectively. Thus, L is an algebraic number field of degree $d = 2 \times 2 \times d_3$. To consider various degrees for L, we change d_3. In Table 3, the input polynomials f_1 and f_2 have degree 16 in x and y and L has degree d. The Maple codes and benchmarks are available at `http://www.cecm.sfu.ca/~mmonagan/code/MRES`.

Table 3. Timings in CPU seconds for computing $\mathrm{res}(f_1, f_2, x)$ over an algebraic number field $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \alpha_3)$ of degree d.

d	N	MRES 1			MRES 2	
		time	LAMP	PRES	time	PRES
16	5	43.688	0.095	42.562	288.265	287.811
24	5	55.203	0.156	53.688	379.735	379.077
32	5	57.234	0.249	55.517	513.797	513.078
40	5	67.719	0.375	65.641	628.547	627.361
48	5	80.687	0.624	78.094	745.578	744.703
56	5	100.953	0.922	97.031	894.734	893.921
64	5	114.062	1.375	110.171	> 1000	> 1000

5 Complexity

Let $f_1, f_2 \in \bar{L}_p[x_1, x_2, ..., x_k, y]$ and $r = \mathrm{res}(f_1, f_2, y) \in L_p[x_1, x_2, ..., x_k]$. Let d be the degree of the number field L. Let $\#f$ denote the number of terms of f in the variables $x_1, x_2, ..., x_k, y$. Let $T_f = \#f_1 + \#f_2$ and $T_r = \#r$. So T_f is the number of terms in the input and T_r is the number of terms in the output r. Let $m = \deg(f_1, y)$, $n = \deg(f_2, y)$, $dx = \max_{i,j} \deg(f_i, x_j)$ and $D = (m+n)dx$. We have $T_f \leq (m+n+2)(dx+1)^k$ and $T_r \leq (D+1)^k$. Since our implementation currently uses classical quadratic polynomial arithmetic, we assume that multiplication and inverses in $\bar{L}_p$ cost $O(d^2)$.

Definition 7. *Given $f \in L_{\mathbb{Z}}[x_1, \ldots, x_k, y]$, we can represent $f = \sum_\alpha C_\alpha X^\alpha$ as a polynomial over $\mathbb{Z}[z_1, \ldots, z_n, x_1, \ldots, x_k]$ where $X^\alpha = \prod_{i=1}^n z_i^{\alpha_i} \prod_{j=1}^k x_j^{\beta_j}$ such that $\alpha_i, \beta_j \in \mathbb{Z}$. If we use this representation, we denote the height of f by $\|f\|_\infty$ and define it as*

$$H(f) = \|f\|_\infty = \max_\alpha(| C_\alpha |).$$

Theorem 5. *Algorithm PRES does*

$$O(T_f d D^k + mnd^2 D^k + kdD^{k+1}) = O(dD^k(T_f + mnd + kD)$$

arithmetic operations in $\mathbb{Z}_p$.

Proof. The following costs count the arithmetic operations in $\mathbb{Z}_p$. The dominating steps of PRES are the evaluations at $x_k = \beta$ in Step 6, the cost of the MEA in Step 1, and the interpolation cost in Steps 13–15. To interpolate $x_1, \ldots, x_k$

in r we need to bound $\deg(r, x_i)$. From Sylvesters matrix for $f_1(y)$ and $f_2(y)$ we have

$$\deg(r, x_i) \leq m \deg(f_2, x_i) + n \deg(f_1, x_i) \leq (n+m)dx = D.$$

Thus, to interpolate $x_1, \ldots, x_k$ in r we need $(D+1)^k$ values using dense interpolation.

We have to evaluate the input polynomials f_1 and f_2 at $x_k = \beta$ for $\beta \in \mathbb{Z}_p$ in line 9 of PRES for $D+1$ choices of β. To speed this up, we precompute the powers β^i for $0 \leq i \leq dx$ which has a negligible cost. The evaluation cost is dominated by evaluating at $x_1 = \beta$ which costs $O(T_f d)$ multiplications. This is done for $D+1$ choices of β and for $(D+1)^{k-1}$ calls to PRES. The total evaluation cost is $O(T_f dD^k)$.

Algorithm PRES makes $(D+1)^k$ calls to URES in Step 1. URES calls the MEA which does $O(mn)$ arithmetic operations in $\bar{L}_p$ each of which costs $O(d^2)$ thus URES costs $O(mnd^2D^k)$ in total.

Algorithm PRES is called once to interpolate x_k from $D+1$ values of $r(z, x_1, \ldots, x_{k-1}, x_k = \beta)$. It does at most $d(D+1)^{k-1}$ univariate interpolations in x_k each of which costs $O(D^2)$ for a total cost of $O(dD^{k+1})$. In general Algorithm PRES is called $(D+1)^{k-i}$ times to interpolate x_i from $D+1$ values of $r(z, x_1, \ldots, x_{i-1}, x_i = \beta)$. It does at most $d(D+1)^{i-1}$ univariate interpolations in x_i, each of which costs $O(D^2)$, which in total costs $O(dD^{k+1})$. Thus the total interpolation cost is $O(kD^{k+1}d)$.

Adding the three costs gives the result.

Let N be the number of good primes needed to reconstruct the resultant r. Let $M = \log \max_{i=1}^{n} H(\check{m}_i)$ and $C = \log \max(H(\check{f}_1), H(\check{f}_2))$.

Theorem 6. *Algorithm MRES costs*

$$O(N(M + CT_M)d + Nd^3 + Nd^2T_f + Nd^2T_r + NdD^k(T_f + mnd + kD) + N^2dT_r) =$$
$$O(Nd(M + CT_M + d^2 + d(T_f + T_r) + D^k(T_f + mnd + kD) + NT_r))$$

arithmetic operations.

Proof. Algorithm MRES reduces the minimal polynomials $\check{M}_1, \ldots, \check{M}_n$ and the input polynomials $\check{f}_1$ and $\check{f}_2$ mod N primes which costs $O(N(M + CT_M)d)$.

The time complexity of building the matrix A in Algorithm 1 for N primes is $O(Nd^3)$. The running time complexity of applying ϕ_γ to the T_f non-zero terms of f_1 and f_2 for N primes is $O(Nd^2T_f)$. Let R_p be the output of Algorithm PRES in Step 9 of MRES, then the time complexity of calling ϕ_γ^{-1} for R_p in Step 11 for N primes is $O(Nd^2T_r)$.

According to Theorem 5, calling PRES in Step 9 of MRES costs $O(dD^k(T_f + mnd + kD)$.

Finally, Algorithm MRES reconstructs $O(dT_r)$ rational coefficients in Step 15 and 18 which costs $O(N^2)$ each hence $O(N^2dT_r)$ in total. The theorem follows by adding the costs explained above.

6 Failure Probability

In this section, we compute the probability of encountering problematic primes and evaluation points. Let $\mathbb{P}_{31} = \{\text{all 31 bit primes}\}$, that is, primes in $(2^{30}, 2^{31})$, and let $N_p = |\ \mathbb{P}_{31}\ | = 50,697,537$ denote the cardinality of $\mathbb{P}_{31}$.

6.1 Lc-bad Primes and Evaluation Points

Theorem 7. *Let $f_1, f_2 \in L[x_1, \ldots, x_k, y]$. Let $H = \max(\|\mathrm{lc}(\check{f}_1)\|)_\infty, \|\mathrm{lc}(\check{f}_2)\|)_\infty) < 2^h$, and $\mathrm{lc}(\check{M}_i) < 2^m$ for $1 \le i \le n$. If p is chosen at random from $\mathbb{P}_{31}$ then* $\mathrm{Prob}[p \textit{ is an lc-bad prime}] \le \frac{2\lfloor \frac{h}{30} \rfloor + n\lfloor \frac{m}{30} \rfloor}{N_p}$.

Proof. Let A denote the event that $p \mid \mathrm{lc}(\check{f}_1)$, B denote the event that $p \mid \mathrm{lc}(\check{f}_2)$, and C denote the event that $p \mid \mathrm{lc}(\check{M}_i)$ for some $1 \le i \le n$. Then

$$\mathrm{Prob}[p \text{ is an lc-bad prime}] = \mathrm{Prob}[A \vee B \vee C] \le \mathrm{Prob}[A] + \mathrm{Prob}[B] + \mathrm{Prob}[C]$$

To compute $\mathrm{Prob}[A]$, we first notice that $\mathrm{lc}(\check{f}_1) = \sum_{i=1}^{N} a_{\alpha_i} Z^{\alpha_i} \in L_{\mathbb{Z}}$ where the sum is over a finite number of $n-$tuples $\alpha_i = (\alpha_{i_1}, \ldots, \alpha_{i_n}) \in \mathbb{Z}^n_{\ge 0}$ such that $Z^{\alpha_i} = z_1^{\alpha_{i_1}} \cdots z_n^{\alpha_{i_n}}$. Since $\mathrm{lc}(\check{f}_1) \ne 0$ there is at least one j with $a_{\alpha_j} \ne 0$. Thus,

$$\begin{aligned}
\mathrm{Prob}[A] &= \mathrm{Prob}[\mathrm{lc}(\check{f}_1) = 0 \mod p] \\
&= \mathrm{Prob}[p \mid a_{\alpha_1} \wedge p \mid a_{\alpha_2} \wedge \ldots \wedge p \mid a_{\alpha_N}] \\
&\le \mathrm{Prob}[p \mid a_{\alpha_j}]. \\
&\le \frac{\lfloor \frac{h}{30} \rfloor}{N_p}.
\end{aligned}$$

Similarly, we have $\mathrm{Prob}[B] \le \frac{\lfloor \frac{h}{30} \rfloor}{N_p}$. For C we have

$$\begin{aligned}
\mathrm{Prob}[C] &= \mathrm{Prob}[p \mid \mathrm{lc}(\check{M}_1) \vee \ldots \vee p \mid \mathrm{lc}(\check{M}_n)] \\
&\le \sum_{i=1}^{n} \mathrm{Prob}[p \mid \mathrm{lc}(\check{M}_i)] \\
&\le n \frac{\lfloor \frac{m}{30} \rfloor}{N_p}.
\end{aligned}$$

Adding the three probabilities implies the theorem.

To compute the probability of encountering an lc-bad evaluation point, we represent $\check{f}_1 \in \bar{L}_p[x_1, \ldots, x_k, y]$ as a non-zero polynomial over $\mathbb{Z}_p[z][x_1, \ldots, x_k][y]$ so $\mathrm{lc}(\check{f}_1) \in \mathbb{Z}_p[z][x_1, \ldots, x_k]$. Thus $\beta \in \mathbb{Z}^k$ is an lc-bad evaluation point if $\mathrm{lc}(\check{f}_1)(\beta)$ vanishes.

Theorem 8. *Let $\beta \in \mathbb{Z}^k$ be chosen at random, then*

$$\mathrm{Prob}[\beta \textit{ is an lc-bad evaluation point}] \le \frac{\deg(\check{f}_1)}{p}.$$

Proof. Let $\mathrm{lc}(\check{f}_1) = \sum_{i=0}^{d-1} a_i(x_1, \ldots, x_k) z^i \neq 0$. Thus, there exists $0 \leq j \leq d-1$ such that $a_j(x_1, \ldots, x_k) \neq 0$. We have,

$$\begin{aligned} \mathrm{Prob}[\mathrm{lc}(\check{f}_1)(\beta) = 0] &\leq \mathrm{Prob}[a_j(\beta) = 0] \\ &\leq \frac{\deg(a_j)}{p} \leq \frac{\deg(\check{f}_1)}{p} \end{aligned}$$

6.2 Det-bad Primes

We recall Hadamard's bound for the determinant of an integer matrix.

Theorem 9. *Let A be an $n \times n$ matrix with $A_{i,j} \in \mathbb{Z}$. Then* $\mid \det(A) \mid \leq \prod_{i=1}^{n} \sqrt{\sum_{j=1}^{n} A_{i,j}^2}$.

Let $\gamma = z_1 + C_1 z_2 + \cdots + C_{n-1} z_n$ where $0 \neq C_i \in \mathbb{Z}$ for $1 \leq i \leq n-1$. Recall that p is a det-bad prime if $\det(A) \mod p = 0$ where A is the coefficient matrix of powers of γ. We consider the case where $\check{m}_i \in \mathbb{Z}[z_1, \ldots, z_i]$ are monic for $1 \leq i \leq n$ so $A \in \mathbb{Z}^{d \times d}$. To compute the probability that p is a det-bad prime, we must first compute an upper bound for $\mid \det(A) \mid$. If we get an upper bound for the entries of A, we can use Hadamard's bound, Theorem 9, to get an upper bound for the $\mid \det(A) \mid$. To do so, we first compute γ^i for $1 \leq i \leq d-1$ over $\mathbb{F} = \mathbb{Z}$. In this case, the largest entry of matrix A will appear in its last column, $[\gamma^{d-1}]_{B_L}$. Thus we need an upper bound for the height of the remainder of γ^{d-1} divided by $\check{m}_n, \ldots, \check{m}_1$. However, before dividing by the monic minimal polynomials, we have $\|\gamma^{d-1}\|_\infty < \|\gamma^d\|_\infty$. Accordingly, if we compute a bound for the remainder of γ^d divided by $\check{m}_n, \ldots, \check{m}_1$, we can use it as an upper bound for $A_{i,j}$. Notice that $\deg(\gamma^j, z_i) = j$ for $1 \leq i \leq n$. Recall that $d_i = \deg(\check{m}_i, z_i)$, and $d = \prod_{i=1}^{n} d_i$ is the degree of our algebraic number field.

Lemma 2. *Let $f, g \in \mathbb{Z}[z_1, \ldots, z_n]$ and $\check{m}_i = z_i^{d_i} + \sum_{j=0}^{d_i-1} a_j z_i^j$ where $a_j \in \mathbb{Z}[z_1, \ldots, z_{i-1}]$. we have,*

(i) $\|fg\|_\infty \leq \|f\|_\infty \|g\|_\infty \min(T_f, T_g)$.
(ii) $\deg(a_j, z_k) \leq d_k - 1$ *for* $1 \leq k \leq i-1$ *and* $T_{a_j} \leq \prod_{k=1}^{i-1} d_k < d$.

In [3], Chen and Monagan introduced an upper bound for the remainder of division by a univariate monic polynomial. Using the same strategy, we prove Theorem 10.

Theorem 10. *Let $f \in \mathbb{Z}[z_1, \ldots, z_n]$ and $d = \deg(f, z_i) > 0$ where $d = \prod_{i=1}^{n} d_i$. Let r be the remainder of f divided by $\check{m}_n$ and $\delta = d - d_n + 1$ be the maximum number of division steps. Then,*

(i) $\deg(r, z_n) \leq d_n - 1$ *and* $\deg(r, z_i) \leq d + \delta(d_i - 1)$, *for* $1 \leq i \leq n-1$.
(ii) $\|r\|_\infty \leq \|f\|_\infty (1 + d/d_n \|\check{m}_n\|_\infty)^\delta$.

Proof. Let $f = \sum_{i=0}^{d} f_i z_n^i$ and $\check{m}_n = z_n^{d_n} + \sum_{j=0}^{d_n-1} a_j z_n^j$ such that $f_i, a_j \in \mathbb{Z}[z_1, \ldots, z_{n-1}]$ for $1 \leq i \leq d$ and $0 \leq j \leq d_n - 1$.

(i) The quotient of f divided by $\check{m}_n$ has degree $d - d_n$ so the division of f by $\check{m}_n$ has up to $\delta = d - d_n + 1$ steps. In the first step, we have $r_1 = f - f_d z_n^{d-d_n} \check{m}_n$. Thus $\deg(r_1, z_n) \leq d - 1$. Moreover, for $1 \leq i \leq n-1$, we have $\deg(f_d, z_i) \leq \deg(f, z_i) = d$ and $\deg(\check{m}_n, z_i) \leq d_i - 1$. Consequently,

$$\begin{aligned}\deg(r_1, z_i) &= \max\{\deg(f, z_i), \deg(f_d, z_i) + \deg(\check{m}_n, z_i)\} \\ &\leq \deg(f, z_i) + \deg(\check{m}_n, z_i) \\ &\leq d + d_i - 1.\end{aligned}$$

If $\deg(r_1, z_n) \geq d_n$, we continue the division. Let $b_1 = \mathrm{lc}(r_1, z_n)$ and $\deg(r_1, z_n) = d - 1$. In the second division step, we have $r_2 = r_1 - b_1 z_n^{d-d_n-1} \check{m}_n$. Hence, $\deg(r_2, z_n) \leq \deg(r_1, z_n) - 1 \leq d - 2$ and

$$\begin{aligned}\deg(r_2, z_i) &\leq \deg(r_1, z_i) + \deg(\check{m}_n, z_i) \\ &\leq d + 2(d_i - 1).\end{aligned}$$

Since the division algorithm has at most δ steps, in the last step, we have $\deg(r, z_n) \leq d - \delta = d_n - 1$ and

$$\deg(r, z_i) \leq d + \delta(d_i - 1).$$

(ii) In the first step of the division, we have $r_1 = f - f_d z^{d-d_n} \check{m}_n$. Thus $\|r_1\|_\infty \leq \|f\|_\infty + \|f_d \check{m}_n\|_\infty$. To compute a bound for $\|f_d \check{m}_n\|_\infty$, it is sufficient to get a bound for $\|f_d a_j\|_\infty$ where $a_j \in \mathbb{Z}[z_1, \ldots, z_{n-1}]$. Using Lemma 2, we have $T_{a_j} < d/d_n$ and

$$\|f_d a_j\|_\infty \leq \|f_d\|_\infty \|\check{m}_n\|_\infty \min(T_{a_j}, T_{f_d}) \leq d/d_n \|f_d\|_\infty \|\check{m}_n\|_\infty$$

for $1 \leq j \leq d_n - 1$. Thus,

$$\|r_1\|_\infty \leq \|f\|_\infty + \|f_d \check{m}_n\|_\infty \leq \|f\|_\infty + \|f\|_\infty \|\check{m}_n\|_\infty d/d_n \leq \|f\|_\infty (1 + d/d_n \|\check{m}_n\|_\infty).$$

Furthermore, $\deg(r_1, z_n) \leq d - 1$. If $\deg(r_1, z_n) \geq d_n$, in the second division step, we have $r_2 = r_1 - b_1 z_n^{d-d_n-1} \check{m}_n$ where $b_1 = \mathrm{lc}(r_1, z_n)$. Since $\|b_1\|_\infty \leq \|r_1\|_\infty$, using the same strategy as the first step, we have

$$\begin{aligned}\|r_2\|_\infty &\leq \|r_1\|_\infty + \|b_1 \check{m}_n\|_\infty \leq \|r_1\|_\infty + d/d_n \|r_1\|_\infty \|\check{m}_n\|_\infty \\ &\leq \|r_1\|_\infty (1 + d/d_n \|\check{m}_n\|_\infty) \leq \|f\|_\infty (1 + d/d_n \|\check{m}_n\|_\infty)^2.\end{aligned}$$

Continuing this argument, the result is obtained.

Theorem 11. *Let $f \in \mathbb{Z}[z_1, \ldots, z_n]$ and $\check{m}_i \in \mathbb{Z}[z_1, \ldots, z_i]$ for $1 \leq i \leq n$ be monic minimal polynomials. Suppose that $\deg(f, z_i) \leq d$ where $d = \prod_{i=1}^n d_i$. Let r be the remainder of f divided by $\check{m}_n, \ldots, \check{m}_1$. Then*

$$\|r\|_\infty \leq \|f\|_\infty \prod_{i=1}^{n} (1 + D_i \|\check{m}_{n-i+1}\|_\infty)^{\delta_i}$$

where $D_i = \frac{d}{\prod_{j=1}^i d_{n-j+1}}$, $\delta_1 = d - d_n + 1$, and $\delta_i = d - d_{n-i+1} + 1 + (d_{n-i+1} - 1) \sum_{j=1}^{i-1} \delta_j$ for $2 \leq i \leq n$.

Proof. Let r_1 be the remainder of f divided by $\check{m}_n$ w.r.t. z_n and $\delta_1 = d - d_n + 1$ be the maximum number of division steps. From Theorem 10, we have

$$\|r_1\|_\infty \leq \|f\|_\infty (1 + \frac{d}{d_n}\|\check{m}_n\|_\infty)^{\delta_1}.$$

Now, let r_2 be the remainder of r_1 divided by $\check{m}_{n-1}$ w.r.t. z_{n-1}. From part (i) of Theorem 10, we have $\deg(r_1, z_{n-1}) \leq d + \delta_1(d_{n-1} - 1)$, thus

$$\deg(r_1, z_{n-1}) - d_{n-1} + 1 \leq d + \delta_1(d_{n-1} - 1) - d_{n-1} + 1$$

and $\delta_2 = d + \delta_1(d_{n-1} - 1) - d_{n-1} + 1$ is the maximum number of division steps. Let $\check{m}_{n-1} = z_{n-1}^{d_{n-1}} + \sum_{j=0}^{d_{n-1}-1} b_j z_{n-1}^j$ such that $b_j \in \mathbb{Z}[z_1, \ldots, z_{n-2}]$ for $0 \leq j \leq d_{n-1} - 1$. Thus $T_{b_j} \leq \frac{d}{d_n d_{n-1}}$. Using the same strategy as the proof of part (ii) of Theorem 10, we have

$$\begin{aligned}\|r_2\|_\infty \leq& \|r_1\|_\infty (1 + \frac{d}{d_n d_{n-1}})\|\check{m}_{n-1}\|_\infty)^{\delta_2} \\ \leq& \|f\|_\infty (1 + \frac{d}{d_n}\|\check{m}_n\|_\infty)^{\delta_1} (1 + \frac{d}{d_n d_{n-1}}\|\check{m}_{n-1}\|_\infty)^{\delta_2}.\end{aligned}$$

The result is obtained by repeating this process for all n minimal polynomials.

Using Theorem 11, we are well-equipped to compute a bound for the entries of A i.e. $A_{i,j}$.

Corollary 1. *Let $\gamma = z_1 + C_1 z_2 + \cdots + C_{n-1} z_n$ where $0 \neq C_i \in \mathbb{Z}$ for $1 \leq i \leq n-1$ and $\|\gamma^d\|_\infty \leq 2^C$. Let r be the remainder of γ^d divided by the monic minimal polynomials $\check{m}_n, \ldots, \check{m}_1$. Let A be the coefficient matrix obtained from Algorithm 1. Let D_i and δ_i be as in Theorem 11. Then,*

$$A_{i,j} \leq \|r\|_\infty \leq 2^C \prod_{i=1}^{n} (1 + D_i \|\check{m}_{n-i+1}\|_\infty)^{\delta_i}.$$

Proof. This is a consequence of Theorem 11.

We have determined that $\delta_n \leq d^2/d_n$ by computational experiment but we can only prove this for $d_1 = d_2 = \cdots = d_n$. Thus Corollary 1 implies $\log\|r\|_\infty$ is polynomial in d, C and $\|\check{m}_i\|_\infty$.

Suppose Algorithm MRES chooses p at random from $\mathbb{P}_{31}$. Theorem 12 bounds the probability that p is a det-bad prime, that is $p| \det(A)$.

Theorem 12. *Let $\gamma = z_1 + C_1 z_2 + \cdots + C_{n-1} z_n$ where $0 \neq C_i \in \mathbb{Z}$ for $1 \leq i \leq n-1$ and $\|\gamma^d\|_\infty \leq 2^C$. Let D_i and δ_i be as in Theorem 11. Suppose $\det(A) \neq 0$. If p is chosen at random from $\mathbb{P}_{31}$ then*

$$\text{Prob}[p| \det(A)] \leq \frac{\lfloor (d/2 \log_2 d + d(C + \sum_{i=1}^{n} \delta_i \log_2(1 + D_i \|\check{m}_{n-i+1}\|_\infty))) \rfloor}{30 N_p}.$$

Proof. To compute the probability that $p|\det(A)$ we first bound $|\det(A)|$. Using Theorem 9 and Corollary 1,

$$|\det(A)| \leq \prod_{i=1}^{d} \sqrt{\sum_{j=1}^{d} A_{j,i}^2} \leq d^{d/2}(2^C \prod_{i=1}^{n}(1 + D_i \|\check{m}_{n-i+1}\|_\infty)^{\delta_i})^d.$$

Since $p \in \mathbb{P}_{31}$ implies $p > 2^{30}$,

$$\begin{aligned}\text{Prob}[p|\det(A)] &\leq \frac{\lfloor \log_2(|\det(A)|)/\log_2 2^{30} \rfloor}{N_p} \\ &\leq \frac{\lfloor (d/2 \log_2 d + dC + d\sum_{i=1}^{n} \delta_i \log_2(1 + D_i\|\check{m}_{n-i+1}\|_\infty)) \rfloor}{30N_p}.\end{aligned}$$

Now we can get a bound for $\|M(z)\|_\infty$ where $M(z)$ is the characteristic polynomial obtained from Algorithm 1.

Theorem 13. *Let $M(z)$ be the characteristic polynomial obtained from Algorithm 1. We have,*

$$\|M(z)\|_\infty \leq d^{d/2}(2^C \prod_{i=1}^{n}(1 + D_i\|\check{m}_{n-i+1}\|_\infty)^{\delta_i})^d.$$

Proof. To construct the characteristic polynomial, $M(z)$, we can solve the linear system $Aq = -[\gamma^d]_{B_L}$ for $q \in \mathbb{Q}^d$. Using the Cramer's rule, $q_k = \frac{\det(A^{(k)})}{\det(A)}$ where $A^{(k)}$ is the matrix formed by replacing the k-th column of A by $[\gamma^d]_{B_L}$ for $1 \leq k \leq d$. Thus, the largest entries of $A^{(k)}$ appear in the k-th column. Now, using Theorem 9, we have

$$|\det(A^{(k)})| \leq \prod_{i=1}^{d} \sqrt{\sum_{j=1}^{d} {A_{j,i}^{(k)}}^2} \leq d^{d/2}(2^C \prod_{i=1}^{n}(1 + D_i\|\check{m}_{n-i+1}\|_\infty)^{\delta_i})^d.$$

Since $\check{m}_i \in \mathbb{Z}[z_1, \ldots, z_i]$, we have $M(z) \in \mathbb{Z}$ which implies that $\det(A) \mid \det(A^{(k)})$. Thus, $q_k \in \mathbb{Z}$ and $q_k \leq |\det(A^{(k)})| \leq d^{d/2}(2^C \prod_{i=1}^{n}(1 + D_i\|\check{m}_{n-i+1}\|_\infty)^{\delta_i})^d$.

We still must compute the failure probability of hitting a zero-divisor prime and evaluation point.

7 Conclusion

We have contributed a new modular algorithm to compute the resultant of two polynomials in $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)[x_1, \ldots, x_k]$. Our algorithm has been implemented in Maple, and its efficacy has been demonstrated through the presentation of two benchmarks. Furthermore, we gave a complexity analysis with failure probabilities. Nevertheless, there remains the task of computing the failure probabilities associated with encountering zero-divisor primes and evaluation points.

Acknowledgement

This work was supported by Maplesoft and the National Science and Engineering Research Council (NSERC) of Canada.

References

1. Mahsa Ansari and Michael Monagan. Computing GCDs of multivariate polynomials over algebraic number fields presented with multiple extensions. In *Computer Algebra in Scientific Computing*, volume 14139 of *LNCS*, page 1–20. Springer, 2023.
2. W. S. Brown and Joseph F. Traub. On Euclid's algorithm and the theory of subresultants. *J. ACM*, 18:505–514, 1971.
3. Liang Chen and Michael Monagan. Algorithms for solving linear systems over Cyclotomic fields. *Journal of Symbolic Computation*, 45(9):902–917, 2010.
4. George E. Collins. The calculation of multivariate polynomial resultants. *J. ACM*, 18(4):515–532, oct 1971.
5. David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2007.
6. Mark Encarnacion. Computing GCDs of polynomials over algebraic number fields. *Journal of Symbolic Computation*, 20:299–313, 1995.
7. K.O. Geddes, S.R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Springer US, 1992.
8. Michael Monagan. Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. In *Proceedings of ISSAC 2004*, pages 243–249. ACM, 2004.
9. Michael Monagan, Keith Geddes, K. Heal, G. Labahn, S. Vorkoetter, J. Mccarron, and P. Demarco. Maple 8 Introductory Programming Guide. 01 2003.
10. Barry M. Trager. Algebraic factoring and rational function integration. In *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation*, SYMSAC '76, page 219–226, New York, NY, USA, 1976. Association for Computing Machinery.
11. Mark van Hoeij and Michael Monagan. A Modular GCD Algorithm over Number Fields Presented with Multiple Extensions. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, page 109–116. ACM, 2002.
12. Paul Wang, M. J. T. Guy, and J. H. Davenport. P-adic Reconstruction of Rational Numbers. *SIGSAM Bull.*, 16(2):2–3, May 1982.