Modular Algorithms for Resultants *

M.B. Monagan, S.C. Tse and A.D. Wittkopf
Department of Mathematics, Simon Fraser University,
Burnaby, British Columbia, V5A 1S6 Canada.
E-mail: {monagan,stephent,wittkopf}@cecm.sfu.ca

Keywords: Resultants, Solving Equations, Modular Algorithms, Polynomial Data Structures.
Intended Audience: Users of Maple and Designers of Modular Algorithms.

Abstract

We present a modular algorithm for efficiently computing resultants over finite fields and number fields. The algorithm is implemented in Maple using a new data structure for multivariate polynomials that supports zero or more field extensions over $\mathrm{GF}(p)$ and $\mathbb{Q}$.

1 Introduction

At Simon Fraser University, the Computational Algebra Group has developed a new data structure for Maple for representing multivariate polynomials over finite fields and number fields. The data structure is designed to facilitate the implementation of what is known as the "modular method" for computing polynomial operations such as GCDs and resultants. The main contributions we make in this paper are:

1. We extend the modular algorithm of Collins [1] for computing resultants of multivariate polynomials over $\mathbb{Z}$ to over finite fields and number fields with an arbitrary number of extensions. (Section 2 and 3).
2. The use of a probabilistic algorithm to compute sharp degree bounds on the degree of the resultant in all variables (Section 4).
3. In the number field case, to avoid the expensive cost of rational reconstruction, we pre-compute a multiple of the denominator of the resultant so that integer reconstruction can be used (Section 5).
4. We use primitive elements to convert multiple extensions into one large extension for improved efficiency (Section 6).
5. The recursive dense data structure that we use for the implementation (Section 7).

Background The theory of resultants was developed in late nineteenth century to study the roots of a system of polynomial equations. For example, suppose we want to solve the system $\{f(x,y)=0, g(x,y)=0\}$ with

$$\begin{aligned} f(x,y) &= x^2-y^2, \\ g(x,y) &= x^2-x+2y^2-y-1. \end{aligned}$$

The idea of the resultant is to eliminate one of the variables. We will eliminate x. We recall the definition of the resultant.

*This work is supported by NSERC of Canada and the MITACS NCE of Canada.

Definition 1 (Resultants). *Let R be any ring and $f(x) = a_n x^n + \ldots + a_0$, $g(x) = b_m x^m + \ldots + b_0$ be polynomials in $R[x]$ with $n, m > 0$.* Sylvester's matrix *of f and g is the $m+n$ by $m+n$ matrix*

$$M = \begin{bmatrix} a_n & a_{n-1} & \cdots & \cdots & a_0 & & & \\ & a_n & a_{n-1} & \cdots & \cdots & a_0 & & \\ & & & & \vdots & & & \\ & & & a_n & a_{n-1} & \cdots & \cdots & a_0 \\ b_m & b_{m-1} & \cdots & b_0 & & & & \\ & b_m & b_{m-1} & \cdots & b_0 & & & \\ & & & & & \vdots & & \\ & & & & & \vdots & & \\ & & & & b_m & b_{m-1} & \cdots & b_0 \end{bmatrix}$$

in which the first m rows are the coefficients of f, the next n rows are the coefficients of g, and the elements not shown are all zero. The resultant *of f, g, denoted by* $\mathrm{res}_x(f, g)$, *is the determinant of M which, note, is an element of R.*

In our example we think of f, g as elements of $\mathbb{Z}[y][x]$, that is, $R = \mathbb{Z}[y]$. We have

$$\begin{aligned} \mathrm{res}_x(f, g) &= \det \begin{pmatrix} 1 & 0 & -y^2 & 0 \\ 0 & 1 & 0 & -y^2 \\ 1 & -1 & 2y^2 - y - 1 & 0 \\ 0 & 1 & -1 & 2y^2 - y - 1 \end{pmatrix} \\ &= 9y^4 - 6y^3 - 6y^2 + 2y + 1 \\ &= (y-1)(3y+1)(3y^2 - 1) \end{aligned}$$

a polynomial in y. The following theorem tells us how we can use the resultant to solve our system.

Theorem 2 (Fundamental Theorem of Resultants). *Let F be an algebraically closed field and let $f = a_n x^n + \cdots + a_0$, $g = b_m x^m + \cdots + b_0$ be polynomials in $F[y][x]$ where $m, n > 0$. Let $r = \mathrm{res}_x(f, g) \in F[y]$. If (x^*, y^*) is a common zero of $\{f(x, y) = 0, g(x, y) = 0\}$, then $r(y^*) = 0$. Conversely, if $r(y^*) = 0$, then at least one of the following holds:*

1. $a_n(y^*) = \cdots = a_0(y^*) = 0$;
2. $b_m(y^*) = \cdots = b_0(y^*) = 0$;
3. $a_n(y^*) = b_m(y^*) = 0$; *or,*
4. $f(x^*, y^*) = g(x^*, y^*) = 0$ *for some* $x^* \in F$.

Since cases 1-3 do not apply in our example, the solutions for y are the zeroes of the resultant, namely, $\{1, \frac{-1}{3}, \pm\frac{1}{\sqrt{3}}\}$. Back-substituting, we obtain the solutions for x, from which we determine the solutions of the system are $\{(\pm 1, 1), (\pm\frac{1}{3}, \frac{-1}{3}), (\pm\frac{1}{\sqrt{3}}, \pm\frac{1}{\sqrt{3}})\}$.

But how do we compute the resultant? If we use Gaussian elimination to compute the determinant of M, the time complexity is $\mathrm{O}((m+n)^3)$ arithmetic operations in R. One can do better. Let $\mathrm{REM}_x(f, g)$ denote the remainder of f divided by g in x. Let l be the degree of $\mathrm{REM}_x(f, g)$ in x. The following identity

$$\mathrm{res}_x(g, f) = b_m^{n-l}\, \mathrm{res}_x(g, \mathrm{REM}_x(f, g))$$

tells us that we can use the Euclidean algorithm to compute resultants. The advantage is that the Euclidean algorithm is inherently quadratic not cubic in $m+n$. However, an expression swell occurs in R and it is more efficient to combine the Euclidean algorithm with a modular approach. The first person to do this for resultants of multivariate polynomials over $\mathbb{Z}$, as is the case with our example, was Collins [1]. The idea is quite simple. To compute the resultant of $f, g \in \mathbb{Z}[y][x]$, we compute the resultant modulo enough primes p_i

$$
\begin{tikzcd}
\mathbb{Q}[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{NRES}"] \arrow[d, "\phi_{p_i}"', "i=1,\ldots,N"] & \mathbb{Q}[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k] \\
\mathbb{Z}_p[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{CRES}"] \arrow[d, "|_{z_i=\sigma_i(\tilde{z})}"', "i=1,\ldots,t"] & \mathbb{Z}_p[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k] \arrow[u, "\text{Chinese Remainder}"] \\
\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{PRES}"] \arrow[d, "|_{y_i=\beta_{ij}}"', "i=1,\ldots,k;\ j=0,\ldots,\deg_{y_i}(r)"] & \mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[y_1,\ldots,y_k] \arrow[u, "|_{\tilde{z}=z_1+\cdots+z_t}"'] \\
\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[x] \arrow[r, "\text{URES}"] & \mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle \arrow[u, "\text{Interpolation}"]
\end{tikzcd}
$$

Figure 1: Homomorphism Diagram for Multivariate Resultants

so that the answer over $\mathbb{Z}$ can be obtained by applying the Chinese remainder theorem. For each prime, we evaluate the polynomials f and g at enough points $y = \beta_j$ such that the modular resultant can be obtained by polynomial interpolation. This reduces the computation to many resultants in one variable over a finite field where we can use the Euclidean algorithm with no expression swell.

But how many primes and evaluation points do we need? Coefficient bounds and degree bounds can be obtained from Sylvester's matrix. And are there any restrictions on the primes and evaluation points? Yes, one must simply avoid primes and evaluation points that make the leading coefficients $a_n(y)$ and $b_m(y)$ vanish. In the rest of the paper we will fill in the details for how to generalize this to number fields and propose optimizations.

Notation Before we proceed, let us fix some notation. Let $\deg_x(f)$ denote the degree of f in x and $\mathrm{lc}_x(f)$ denote the leading coefficient of f in x. Let $\phi_p(f) : R[x_1,\ldots,x_k] \to R_p[x_1,\ldots,x_k]$ denote the modular homomorphism for any ring R and for any p in $\mathbb{Z}$ (usually a prime). For example, let $f = 10yx^3 + 8yx + 5$ be a polynomial in $\mathbb{Z}[y][x]$. Then, $\deg_x(f) = 3$, $\mathrm{lc}_x(f) = 10y$, and $\phi_3(f) = yx^3 + 2yx + 2 \in \mathbb{Z}_3[y][x]$.

Let $\mathbb{Q}[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle$ denote a number field over $\mathbb{Q}$ with t extensions. Each z_i is an extension variable and each $m_i(z_i) \in \mathbb{Q}[z_1,\ldots,z_{i-1}]/\langle m_1,\ldots,m_{i-1}\rangle[z_i]$ is the minimal polynomial for z_i. Likewise let $\mathbb{Z}_p[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle$ denote a finite ring over $\mathbb{Z}_p$ with t extensions.

2 Over Number Fields

Figure 1 is a homomorphism diagram for computing multivariate resultants over number fields. It gives an overview of the modular algorithm. Here we fix some of the details:

NRES computes multivariate resultants over number fields. N is the number of primes whose product must be large enough to reconstruct the rational coefficients of the resultant. In the algorithm, rather than computing a bound on the size of the coefficients, we simply run until the current value for the resultant does not change for two additional primes. Its pseudo-code is shown below.

CRES computes multivariate resultants over finite rings by collapsing extensions. $\sigma_i(\tilde{z})$ is the mapping function for z_i in term of $\tilde{z}$. It is discussed in Section 6.

PRES computes multivariate resultants over finite rings with extensions. $\deg_{y_i}(r)$ is the degree bound on y_i of the resultant; β_{ij} is the j-th evaluation point of y_i. The algorithm over $\mathbb{Z}_p$ is from [1] but we extend it to support extensions over $\mathbb{Z}_p$. This requires a new data structure that supports zero or more ring extensions, which is discussed in Section 7.

URES computes univariate resultants over finite rings using the Euclidean algorithm. It is discussed in Section 3.

Remark Because CRES collapses extensions, NRES calls PRES, which calls URES, with input over $\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle$ (a finite ring with only one extension). However, users can directly call PRES and URES over $\mathbb{Z}_p[z_1, \ldots, z_t]/\langle m_1, \ldots, m_t\rangle$ (a finite ring with t extensions).

Below is the pseudo-code for NRES. This intends to show the main idea of the modular algorithm and thus we assume the input and the minimal polynomials are over $\mathbb{Z}$. We extend it to over $\mathbb{Q}$ in Section 5. Algorithm PDEG applies a probabilistic algorithm to obtain sharp degree bounds, which is discussed in Section 4.

NRES Multivariate resultants over number fields $F = \mathbb{Q}[z_1, \ldots, z_t]/\langle m_1, \ldots, m_t\rangle$.
input: $f, g \in F[Y][x]\backslash\{0\}$
output: $r = \mathrm{res}_x(f, g) \in F[Y]$

1. {initialization}
   (a) {probabilistic degree bounds}
   $\Pi_\theta \leftarrow \mathrm{PDEG}(f, g)$
   (b) {big modulus prime}
   $p \leftarrow 10^5$
   (c) {image agreement count}
   $\eta \leftarrow 0$
   (d) {resultant so far}
   $r \leftarrow$ **null**
2. {main computation}
   while $\eta < 3$ **do**
   (a) {find next good prime}
   do $p \leftarrow$ the next prime after p
   until $\phi_p(\mathrm{lc}_x(f)), \phi_p(\mathrm{lc}_x(g)) \neq 0$
   and $\phi_p(\mathrm{lc}_{z_1}(m_1)), \ldots, \phi_p(\mathrm{lc}_{z_t}(m_t)) \neq 0$
   (b) {solve image problem[1]}
   $\bar{r} \leftarrow \mathrm{CRES}(\phi_p(f), \phi_p(g), \Pi_\theta)$
   (c) {lift modulus images}
   $r_0 \leftarrow$ CHINESE-REMAINDER($[r, \bar{r}]$)
   (d) {stop if three agreements in a row}
   if $r_0 = r$ **then** $\eta \leftarrow \eta + 1$
   else $\eta \leftarrow 1$ **and** $r \leftarrow r_0$
3. **output** r

3 Euclidean Algorithm over Rings

Algorithm URES implements the Euclidean algorithm over any field F. It computes the *Euclidean polynomial remainder sequence* of $f, g \in F[x]$, or simply the *remainder sequence*, $r_0, \ldots, r_{n+1} \in F[x]$ satisfying

1. $r_0 = f$,
2. $r_1 = g$,
3. $r_{i+1} = \mathrm{REM}_x(r_{i-1}, r_i)$, and

[1] if dividing by a zero divisor occurs in URES and a failure is generated, goto Step 2a; see Section 3

4. $r_{n+1} = 0$.

Let $d_0, \dots, d_n$ be the degrees of $r_0, \dots, r_n$ in x. Then we can compute the resultant of f, g as follows [2],

$$\mathrm{res}_x(f,g) = \mathrm{lc}_x(r_n)^{d_{n-1}} \prod_{i=0}^{n-2} (-1)^{d_i d_{i+1}} \mathrm{lc}_x(r_{i+1})^{d_i - d_{i+2}}$$

To avoid the intermediate coefficient swell, it is highly desirable to apply the Euclidean algorithm over a finite field by reducing the input modulo some prime. Consider a number field over $\mathbb{Q}$. Since a minimal polynomial modulo a prime may become reducible, the modular algorithm may call URES over a finite ring instead of a finite field. That is, there can be zero divisors, which would cause URES to fail.

Let us illustrate the problem with an example. Consider a finite ring $\mathbb{Z}_5[u]/\langle m \rangle$ where $m = 3u^3 - 1 \in \mathbb{Z}_5[u]$ is the minimal polynomial[2]. Note that m can be factored as $3(u+2)(u^2+3u+4)$ and thus $\mathbb{Z}_5[u]/\langle m \rangle$ is not a field. Let $f = x^3 + u^2x^2 + 1$ and $g = x^2 + x + u + 4$ be polynomials in $\mathbb{Z}_5[u]/\langle m \rangle[x]$. We compute $\mathrm{res}_x(f,g)$ by URES as follows,

$$\begin{aligned} r_0 &= f = x^3 + u^2x^2 + 1 \\ r_1 &= g = x^2 + x + u + 4 \\ r_2 &= \mathrm{REM}_x(r_0, r_1) = 4(u+2)(u+4)x + u^2 + u + 3 \\ r_3 &= \mathrm{REM}_x(r_1, r_2) = \cdots \end{aligned}$$

It fails in computing r_3 because the leading coefficient of r_2, $4(u+2)(u+4)$, is a zero divisor of the ring.

If this occurs, a failure is generated; we simply trap the failure and try another prime. If the primes are large, the number of zero divisors will be small, hence, failures will occur infrequently. Typically, our primes are $O(10^8)$. Thus with a very low probability of dividing by a zero divisor, we simply proceed with the Euclidean algorithm over the ring.

The following theorems guarantee the correctness and the termination of the modular algorithm. They can be generalized to a number field with an arbitrary number of extensions. That is, for a resultant over any number field, the algorithm outputs the resultant modulo the prime if it terminates. Secondly, it terminates for all but finitely many primes. Then we will eventually have enough primes to reconstruct the resultant over $\mathbb{Q}$. Proofs are based on [3].

Theorem 3 (Correctness). *Let $f, g \in \mathbb{Q}[z]/\langle m \rangle[x]$ of positive degree. Let p be a prime such that*

1. *$\phi_p(m)$ exists,*
2. *$\bar{f} = \phi_p(f)$ and $\bar{g} = \phi_p(g)$ exist, and*
3. *$\phi_p(\mathrm{lc}_x(f)) \neq 0$ and $\phi_p(\mathrm{lc}_x(g)) \neq 0$, and*
4. *$\mathrm{URES}(\bar{f}, \bar{g})$ terminates.*

Let $r = \mathrm{res}_x(f,g)$, $\bar{r} = \mathrm{res}_x(\bar{f}, \bar{g})$, and $\omega = \mathrm{URES}(\bar{f}, \bar{g})$. Then $\bar{r} = \phi_p(r)$ and $\omega = \phi_p(r)$.

Theorem 4 (Termination). *The conditions in Theorem 3 hold for all but finitely many primes.*

4 Probabilistic Degree Bounds

The modular algorithm requires a degree bound on each variable of the resultant in order to interpolate that variable. Because degree bounds derived from Sylvester's matrix can be very pessimistic for multivariate resultants, we develop a probabilistic algorithm to obtain sharp degree bounds. Let us illustrate the algorithm with an example. Let $f = 2zx^4 + yz^2x + 2yz + y^4z + 2yz^4$ and $g = 2y^2x^3 + 2y^4 + 2y^5 + y^3z^2$ be polynomials

[2]By the common definition, minimal polynomials are monic and irreducible. For efficiency, instead of monic polynomials with fractional coefficients, we always work with their associates over $\mathbb{Z}$. For brevity, instead of *characteristic polynomials*, we call them *minimal polynomials* even when they are reducible, in case of finite rings.

in $\mathbb{Z}[y,z][x]$. Suppose we want to obtain a degree bound on the variable y of $\mathrm{res}_x(f,g)$. First we reduce the input field modulo a large prime. In the example, let us choose $p = 1009$. Then we choose a random element $\alpha_z = 213 \in \mathbb{Z}_p$ and evaluate both f, g at $z = \alpha_z$ to obtain $\bar{f} = 426x^4 - 36yx - 9y + 213y^4, \bar{g} = 2y^2x^3 + 2y^4 + 2y^5 - 36y^3 \in \mathbb{Z}_p[y][x]$. Next we compute $\mathrm{res}_x(\bar{f},\bar{g})$ using a degree bound based on Sylvester's matrix. Since it is a bivariate problem, the bound now should be reasonably good. We call PRES to obtain $\mathrm{res}_x(\bar{f},\bar{g}) = -346y^{20} + 3y^{19} - 163y^{18} + 422y^{17} + 2y^{16} - 169y^{15} + 392y^{14} - 119y^{13} + 485y^{12} + 444y^{11} \in \mathbb{Z}_p[y]$, and read off its degree 20. We repeat this procedure to obtain 12 for the degree bound on z. If p is large, the degree bounds $[20, 12]$ are the degrees of $\mathrm{res}_x(f,g)$ with high probability.

Let us determine the probability of obtaining the wrong degree bounds. We will do this for a single extension but the idea can be generalized to an arbitrary number of extensions. Let $R = \mathbb{Z}_p[u]/\langle m\rangle$ be a finite ring where $m \in \mathbb{Z}_p[u]$ is the minimal polynomial of degree q. Let $f, g \in R[y,z][x]$. We want to apply the probabilistic algorithm to obtain the degree bound on y of $r = \mathrm{res}_x(f,g)$. Let $\deg_z(r) = n$. Let $\bar{f}, \bar{g} \in R[z][x]$ be f, g evaluated at $y = \alpha$ for a random evaluation point $\alpha \in R$. Let $\bar{r} = \mathrm{res}_x(\bar{f},\bar{g}) \in R[z]$ with $\deg_z(\bar{r}) = \bar{n} \le n$. Let E denote the probability that the output is wrong, that is, $\bar{n} < n$. It happens when α is a root of the leading coefficients of r in y.

We can show that, for any finite ring, $E \le 1/p$. But, it can be arranged so that, for most cases, $E \le (n/p)^q$. Furthermore, if we repeat the probabilistic algorithm k times and take the maximum of the output, the error probability becomes $E^k \le (1/p)^{qk}$. Hence we can choose a large prime, choose a minimal polynomial of high degree, or repeat the process more times to make the error probability as small as we allow it [7].

Lemma 5 (Number of Roots in Finite Rings). *Let $R = \mathbb{Z}_p[u]/\langle m\rangle$ be a finite ring where $m \in \mathbb{Z}_p[u]$ is the minimal polynomial of degree q. Let $f \in R[x]$ have degree n.*

1. *If m is irreducible, then the number of roots of f in R is at most n (by Fundamental Theorem of Algebra).*
2. *If m is square-free, then the number of roots of f in R is at most n^q (by applying the Chinese remainder theorem).*
3. *If m is reducible, then the number of roots of f in R is at most p^{q-1}.*

Since we can easily[3] choose our prime p such that m is square-free modulo p, but we cannot in general choose p such that m is irreducible modulo p, we have

Theorem 6 (Error Probability in Simple Finite Rings). *Let R as in Lemma 5. If m is square-free, then $E \le (n/p)^q$, hence $E^k \le (n/p)^{qk}$.*

For example, for $n = 1000$, $p > 10^6$, $q = 4$, the probability of an error is less than $1/10^{12}$.

5 Denominator Multiples

In Section 2, NRES computes resultants when the coefficients of the input and the minimal polynomials are over $\mathbb{Z}$. In order to extend NRES to work over $\mathbb{Q}$, we first clear the fractions from the input by multiplying through by a suitable integer. However, the denominator problem with the minimal polynomials is not as easy. Consider the number field $\mathbb{Q}[w,z]/\langle m_1, m_2\rangle$ where $m_1 = 3w^3 - 1 \in \mathbb{Q}[w]$ and $m_2 = 2z^3 - 3zw^2 + w + 3 \in \mathbb{Q}[w]/\langle m_1\rangle[z]$. Let $f = x^3 + x + 2wz$ and $g = x^2 + (1+w)x + w + z$ be polynomials in $\mathbb{Q}[w,z]/\langle m_1, m_2\rangle[x]$. Note that the coefficients of f, g are over $\mathbb{Z}$ but those of $\mathrm{res}_x(f,g) = \frac{7}{6} + 2z + \left(2 - \frac{20}{3}z + \frac{7}{2}z^2\right)w + \left(-2 + 9z + 10z^2\right)w^2$ are not. The denominator 6 of the resultant is introduced from divisions by the minimal polynomials which have non-unit leading coefficients. To recover a resultant with fractions we can apply either rational reconstruction or, if we know a multiple of the denominator, integer reconstruction. The time complexity of the former is in general higher than that of the latter.

To compute a multiple of the denominator 6 in the example, we start with the minimal polynomial of the last extension, which is $m_1(w,z)$ in the example. First, we compute the degree bound on z of the

[3] by checking that $\mathrm{GCD}(m', m) \ne 1$

resultant, $d_z = 3$ using the probabilistic degree bound[4] in Section 4. Then we know there is at most $\kappa_z = \tilde{\theta}_z - \deg_z(m_1) + 1 = 3 - 3 + 1 = 1$ division by m_1. Hence the contribution of the extension $m_1(w,z)$ to the denominator multiple is $\mathrm{lc}_z(m_1)^{-\kappa_z} = 1/2$.

But, since m_1 is in $\mathbb{Q}[w]/\langle m_1\rangle[z]$, the division by m_1 also contribute to the degree in w. Let σ_w denote this accumulated degree from $m_1(w,z)$ for w. Working out the details, $\sigma_w = \deg_w(\lambda_w)\lceil\kappa_z/(\deg_z(m_1) - \deg_z(\lambda_w))\rceil = 2\lceil 2/(3-1)\rceil = 2$ where $\lambda_w = -3zw^2$ is the leading term of m_1 in w. In general, we update the accumulated degrees of all other extension variables recursively.

Now we can proceed to compute the contribution of the extension $m_2(w)$ with the accumulated degree $\sigma_w = 2$. Again, we compute the degree bound on w of the resultant, $d_w = 4$. Then, we compute the bound on the number of divisions by m_2, $\kappa_w = (\sigma_w + d_w) - \deg_w(m_2) + 1 = (2+4) - 3 + 1 = 4$. Hence the denominator multiple from all extensions is $\mathrm{lc}_z(m_1)^{-\kappa_z}\mathrm{lc}_w(m_2)^{-\kappa_w} = 1/(2^1 \times 3^4) = 1/162$.

Here are the recursive formulas for the general case. Let $f, g \in \mathbb{Q}[z_1, \dots, z_t]/\langle m_1, \dots, m_t\rangle[Y][x]$. Let θ_i denote the degree and c_i denote the leading coefficient of m_i in z_i. Let d_i denote the degree bound on z_i of $\mathrm{res}_x(f,g)$. Let σ_i denote the accumulated degree, κ_i denote the bound on the number of divisions by m_i. Let λ_{ij} denote the leading term of m_i in z_j, $\lambda_{ij,i}$ denote the degree of λ_{ij} in z_i, and $\lambda_{ij,i}$ denote the degree of λ_{ij} in z_j. Let D denote the denominator multiple of $\mathrm{res}_x(f,g)$. Then,

$$\begin{aligned}
\sigma_i &= \sum_{j=1}^{i-1} \lambda_{ij,i}\lceil\kappa_j/(\theta_j - \lambda_{ij,j})\rceil \\
\kappa_i &= \max(0, \sigma_i + d_i - \theta_i + 1) \\
D &= c_1^{-\kappa_1}\cdots c_t^{-\kappa_t}
\end{aligned}$$

6 Collapsing Extensions

We observe that, due to overhead in data structure and program control, arithmetic operations are faster in a ring with one big extension than in a ring with lots of small extensions. Therefore, we can speed up the resultant computations by collapsing extensions of the input ring to one big extension. For a typical problem this can speed up the resultant computation by a factor of 10. Although it causes severe expression swell over $\mathbb{Q}$, the conversion is rather efficient over $\mathbb{Z}_p$.

Let us illustrate the algorithm with an example. Consider a finite field $\mathbb{Z}_7[w,z]/\langle m_1, m_2\rangle$ where $m_1 = 3w^3 - 1 \in \mathbb{Z}_7[w]$ and $m_2 = 2z^3 - 3zw^2 + w + 3 \in \mathbb{Z}_7[w]/\langle m_1\rangle[z]$. Let $f = x^3 + x + 2wz$ and $g = x^2 + (1+w)x + w + z$ be polynomials in $\mathbb{Z}_7[w,z]/\langle m_1, m_2\rangle[x]$. We first compute a primitive element $u = w + z$ of the two extensions, that is, computing the minimal polynomial for u and the mapping functions of w, z in u. One way[5] to do this is to use resultants and GCDs [5]. We compute a sequence of resultants as follows,

$$\begin{aligned}
r_2 &= u - w - z \bmod \langle m_1, m_2\rangle \in \mathbb{Z}_7[u,w][z] \\
r_1 &= \mathrm{res}_z(r_2, m_2) \bmod \langle m_1\rangle \in \mathbb{Z}_7[u][w] \\
&= 4uw^2 + (6 + 6u^2)w + 5u^3 + 6 \\
r_0 &= \mathrm{res}_w(r_1, m_1) \in \mathbb{Z}_7[u] \\
&= 6u^9 + 3u^6 + 5u^4 + u^3 + 6u^2 + 3u + 1
\end{aligned}$$

The minimal polynomial for u is $m = r_0$. Let $\mathrm{GCD}_x(f,g)$ denote the monic greatest common divisor of f, g in x. We compute $\mathrm{GCD}_w(m_1, r_1) = w - (3u^8 + 5u^7 + 2u^6 + 2u^5 + 3u^4 + 3u^2 + 3u + 6) \in \mathbb{Z}_7[u]/\langle m\rangle[w]$ and solve for w in $\mathbb{Z}_7[u]$. Hence the mapping function of w in u is $3u^8 + 5u^7 + 2u^6 + 2u^5 + 3u^4 + 3u^2 + 3u + 6$ and that of z is $u - w = 4u^8 + 2u^7 + 5u^6 + 5u^5 + 4u^4 + 4u^2 + 5u + 1$ in $\mathbb{Z}_7[u]/\langle m\rangle$.

We make the substitution into f, g to obtain $\tilde{f} = x^3 + x + 2u^8 + 2u^7 + 3u^5 + 3u^4 + u^3 + 5u^2 + 6u + 3$, $\tilde{g} = x^2 + (4u^8 + 2u^7 + 5u^6 + 5u^5 + 4u^4 + 4u^2 + 5u + 2)x + u \in \mathbb{Z}_7[u]/\langle m\rangle[x]$. Then we can proceed in this ring of only one extension of degree 9 to obtain $\tilde{r} = \mathrm{res}_x(\tilde{f}, \tilde{g}) = 2u^8 + 4u^7 + 6u^4 + 3u^2 + 6u + 4 \in \mathbb{Z}_7[u]/\langle m\rangle$. At last

[4] by ignoring the minimal polynomials and treating extension variable as polynomial variables

[5] Another way is to use linear algebra [4].

we back-substitute $u = w + z$ into $\tilde{r}$ to obtain $r = \mathrm{res}_x(f, g) = 3w^2z^2 + 2wz^2 + 5wz + 5z^2 + 5w + 2z + 5 \in \mathbb{Z}_7[w, z]/\langle m_1, m_2\rangle$.

The following theorems can be generalized to a number field with an arbitrary number of extensions.

Theorem 7 (Correctness). *Let $F[w, z]/\langle m_1, m_2\rangle$ be a number field where $m_1 \in F[w]$ and $m_2 \in F[w]/\langle m_1\rangle[z]$ are the minimal polynomials. Let*

$$\begin{aligned} r_2 &= u - w - z \bmod \langle m_1, m_2\rangle \in F[u, w][z] \\ r_1 &= \mathrm{res}_z(r_2, m_2) \bmod \langle m_1\rangle \in F[u][w] \\ r_0 &= \mathrm{res}_w(r_1, m_1) \in F[u] \end{aligned}$$

Let $F[u]/\langle m\rangle$ be another number field where $m = r_0 \in F[u]$ is the minimal polynomial. Then $F[u]/\langle m\rangle \cong F[w, z]/\langle m_1, m_2\rangle$. Let $\sigma : F[w, z]/\langle m_1, m_2\rangle \to F[u]/\langle m\rangle$ be the mapping of the isomorphism. Then, in addition, $\mathrm{GCD}_w(m_1, r_1) \bmod \langle m\rangle = w - \sigma(w) \in F[u]/\langle m\rangle[w]$.

The following theorem is stated for the equal degree case, namely, $\deg_w(m_1) = \deg_z(m_2) = \sqrt{n}$ in order to obtain a meaningful measure. The degree of the extension field $[F[w, z]/\langle m_1, m_2\rangle : F]$ is $\sqrt{n} \times \sqrt{n} = n = \deg_u(m)$.

Theorem 8 (Time Complexity). *The time complexity of the conversion is $O(n^3)$ arithmetic operations over F.*

Over Finite Rings When the input are over finite rings, however, the conversion may fail because of dividing by a zero divisor. Let us illustrate the problem with an example. Consider a finite ring $\mathbb{Z}_{11}[w, z]/\langle m_1, m_2\rangle$ where $m_1 = w^3 + 2 \in \mathbb{Z}_{11}[w]$ and $m_2 = z^2 + 1 \in \mathbb{Z}_{11}[w]/\langle m_1\rangle[z]$. Note that m_1 can be factored as $(w^2 + 4\,w + 5)\,(w + 7)$ and thus $\mathbb{Z}_{11}[w, z]/\langle m_1, m_2\rangle$ is not a field. Let us try to convert it to $\mathbb{Z}_{11}[u]/\langle m\rangle$ anyway. To obtain the minimal polynomial for u, we compute the sequence of resultants as follows,

$$\begin{aligned} r_2 &= u - w - z \bmod \langle m_1, m_2\rangle \in \mathbb{Z}_{11}[u, w][z] \\ r_1 &= \mathrm{res}_z(r_2, m_2) \bmod \langle m_1\rangle \in \mathbb{Z}_{11}[u][w] \\ &= w^2 + 9\,wu + u^2 + 1 \\ r_0 &= \mathrm{res}_w(r_1, m_1) \in \mathbb{Z}_{11}[u] \\ &= u^6 + 3\,u^4 + 4\,u^3 + 3\,u^2 + 10\,u + 5 \end{aligned}$$

But $m = r_0$ can be factored as $(u^2 + 4\,u + 8)\,(u^2 + 3\,u + 6)\,(u + 2)^2$ over $\mathbb{Z}_{11}$ and thus $\mathbb{Z}_{11}[u]/\langle m\rangle$ is not a field. To obtain the mapping function of w in u, we compute $\mathrm{GCD}_w(m_1, r_1)$ by the Euclidean algorithm as follows,

$$\begin{aligned} s_0 &= m_1 \;=\; w^3 + 2 \\ s_1 &= r_1 \;=\; w^2 + 9\,wu + u^2 + 1 \\ s_2 &= \mathrm{REM}_x(s_0, s_1) \;=\; 3(u + 2)(u + 9)w + 9\,u^3 + 9\,u + 2 \\ s_3 &= \mathrm{REM}_x(s_0, s_1) \;=\; \ldots \end{aligned}$$

It fails in computing s_3 because the leading coefficient of s_2, $3(u + 2)(u + 9)$, is a zero divisor of the ring.

Similar to computing resultants over finite rings in Section 3, the conversion will trap the failure and try another prime. Alternatively, we could use other primitive elements such as $u = w + sz$ where s is an integer.

7 Recursive Dense Data Structure

Algorithms in this paper are implemented using the `recden` package in Maple [3]. `recden` is a *recursive dense* representation of polynomials that facilitates modular algorithms. Formally `recden` polynomials are encoded in Maple as follows:

```
<poly>   ::= POLYNOMIAL( R::<ring>, C::<coeffs> )
<ring>   ::= [ m::nonnegint, X::list(symbol), E::list(<coeffs>) ]
<coeffs> ::= 0 or list(<coeff>)
<coeff>  ::= rational or <coeffs>
```

Thus a `recden` polynomial has two operands: the polynomial ring R and the coefficients C. The ring R consists of the modulus m (or the characteristic), the list of variables X, and the list of ring extensions over the base ring $\mathbb{Z}_m$ or $\mathbb{Q}$. By convention, $m = 0$ means the ring is over $\mathbb{Q}$. The minimal polynomials in E are stored in reverse order, that is, E_1 is the last extension and so on. The data structure for C is recursive dense with the integer 0 indicating a zero coefficient. This implies a polynomial is a zero polynomial if and only if $C = 0$.

The other key requirement is that the bottom level of the data structure, which represents a polynomial in one variable, uses a dense array of coefficients. Thus the representation for $\mathbb{Z}_p[x]$, which is what most computations in the modular algorithms are reduced to, is an array of machine integers. This allows fast arithmetic operations in $\mathbb{Z}_p[x]$ [6].

The following gives some examples of `recden` for polynomials over $\mathbb{Z}$, a finite field, and a number field.

Example 1 **(Integer Polynomial.)** Let $f_1 = x^2 + y^2 + 2 \in \mathbb{Z}[y][x]$. Its `recden` representation is:

```
POLYNOMIAL([0,[x,y],[]], [[2,0,1],0,[1]])
```

Here $m = 0$ because f_1 is over $\mathbb{Z}$, $X =$ `[x,y]` because x, y are the polynomial variables, and $E =$ `[]` because there is no extension. Also, $C_1 =$ `[2,0,1]` is the coefficient in $\mathbb{Z}[y]$ of x^0, that is, $2y^0 + 0y^1 + 1y^2$. Similarly $C_2 = 0$ (the integer 0) is the coefficient in $\mathbb{Z}[y]$ of x^1 and $C_3 =$ `[2]` is that of x^2.

Example 2 **(Finite Field.)** Let F be the finite field of order 3^4. We can represent F by $\mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle$. Let $f_2 = x^2 + y^2 + 2 \in F[x]$. Its `recden` representation is:

```
POLYNOMIAL([3,[x,y], [[1,1,0,0,1]] ],[[2,0,1],0,[1]])
```

Here $m = 3$ because f_2 is over $\mathbb{Z}_3$ and $E_1 =$ `[1,1,0,0,1]` is the encoding of the minimal polynomial $1y^0 + 1y^1 + 0y^2 + 0y^3 + 1y^4$. Note that X and C are the same as those of f_1 but they are interpreted differently. For f_1 both x, y are polynomial variables, but for f_2 only x is a polynomial variable while y is an extension variable. This is determined by the number of extensions in E. Likewise, for f_1 we interpret $C_1 =$ `[2,0,1]` as a polynomial, but for f_2 we interpret it as an element of the finite field.

Besides the efficient representation, `recden` also provides a user-friendly interface. The input command and the user display of f_2 in Maple are:

```
> f2 := rpoly(x^2+y^2+2, [x,y], 3, y^4+y+1);
```

$$f_2 := x^2 + y^2 + 2 \bmod \langle y^4 + y + 1, 3\rangle$$

Example 3 **(Number Field.)** Let F be the number field $\mathbb{Q}(\sqrt[3]{5}, \sqrt{-2 - (\sqrt[3]{5})^2})$. We can represent F by $\mathbb{Q}[w, z]/\langle w^3 - 5, z^2 + w^2 + 2\rangle$. Let $f_3 = z + w + 1 \in F$. Its input command, its user display and its `recden` representation are:

```
> f3 := rpoly(z+w+1, [z,w], [z^2+w^2+2,w^3-5]);
```

$$f_3 := z + w + 1 \bmod \langle z^2 + w^2 + 2, w^3 - 5\rangle$$

```
> lprint(f3);
      POLYNOMIAL([0,[z,w],[ [[2,0,1],0,[1]], [-5,0,0,1] ]],[[1,1],[1]])
```

Here $E_2 =$ `[-5,0,0,1]` is the first extension with the minimal polynomial $w^3 - 5$ and $E_1 =$ `[[2,0,1],0,[1]]` is the last extension with $z^2 + w^2 + 3$. Since the same encoding is used for any coefficients, the E_1 of f_3 ($z^2 + w^2 + 2 \in \mathbb{Q}[w]/\langle w^3 - 5\rangle[z]$) is identical to the C of f_1, f_2 ($x^2 + y^2 + 2 \in \mathbb{Z}[y][x]$ and $x^2 + y^2 + 2 \in \mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle[x]$). That is, the encoding is independent of variables and base rings.

```
## f,g are of positive degree in F[x]
## 'projring' returns the coefficient field F
URES := proc(ff,gg)
local f,g,F,n,m,res,r,k;
     (f,g) := (ff,gg);
     (n,m) := degrpoly(f),degrpoly(g);
     F := projring(getring(f));
     res := rpoly(1,F);
     while m>0 do
          r := remrpoly(f,g);
          if iszerorpoly(r) then return r; fi;
          k := degrpoly(r);
          ## := res (-1)^(nm) lc(g)^(n-k)
          res := mulrpoly(res,powrpoly(lcrpoly(g),n-k));
          if irem(n*m,2)=1 then res := negrpoly(res); fi;
          (f,g) := (g,r);
          (n,m) := (m,k);
     end do;
     ## res lc(g)^n
     mulrpoly(res,powrpoly(lcrpoly(g),n));
end proc:
```

Figure 2: Maple Code for Univariate Resultants

Code Snippet To give a taste of the implementation, we give the code for Algorithm URES in Figure 2. It computes the resultant of two polynomials over any field using the Euclidean algorithm. The following example shows URES executing over the finite ring $\mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle$.

```
> f := rpoly(x^2+y^2+2, [x,y], 3, y^4+y+1);
> g := rpoly(x^3+x^2*y+x*y+2, [x,y], 3, y^4+y+1);
```

$$f := x^2 + y^2 + 2 \bmod \langle y^4 + y + 1, 3\rangle$$
$$g := x^3 + x^2 y + xy + 2 \bmod \langle y^4 + y + 1, 3\rangle$$

```
> URES(f,g);
```

$$1 + 2y + y^3 \bmod \langle y^4 + y + 1, 3\rangle$$

References

[1] George Collins. The Calculation of Multivariate Polynomial Resultants. *JACM 18(4)*, 1971.

[2] Keith Geddes, Stephen Czapor, and George Labahn. *Algorithms for Computer Algebra*, 1996.

[3] Mark van Hoeij, Michael Monagan. A Modular GCD Algorithm over Number Fields Presented with Multiple Field Extensions. *Proceedings of ISSAC '02*, 2002.

[4] Rudolf Lidl and Harald Niederreiter. Construction of Irreducible Polynomials. *Finite fields, Encyclopedia of Mathematics and its Application*, 1997.

[5] Rüdiger Loos. Computing in Algebraic Extensions. *Symbolic and Algebraic Computation Series: Computing Supplementa, Volume 4*, 1983.

[6] Michael Monagan. In-place Arithmetic for Polynomials over $\mathbb{Z}_n$. *DISCO '92*, 1992

[7] Jacob Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *JACM 27(4)*, 1980.