A Modular Resultant Algorithm for Number Fields *

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

ABSTRACT

We present a modular algorithm for efficiently computing the Sylvester resultant of two multivariate polynomials over an algebraic number field. The algorithm is implemented in Maple using a new data structure which is designed to support modular algorithms. The algorithm and the implementation support multiple field extensions.

1. INTRODUCTION

Let R be a commutative ring and let $f(x) = a_n x^n + ... + a_1 x + a_0$ and $g(x) = b_m x^m + ... + b_1 x + b_0$ be polynomials in $R[x]$. The resultant of $f(x)$ and $g(x)$, denoted $\mathrm{res}_x(f, g)$, is the determinant of Sylvester's matrix. Assuming division in R, the most direct approach to compute $\mathrm{res}_x(f, g)$ is to compute the determinant of Sylvester's matrix using Gaussian elimination. If $n \geq m$, this requires $O(mn^2)$ arithmetic operations in R. Faster, the Euclidean algorithm can be modified to compute the resultant in $O(mn)$ operations.

If R is a polynomial ring in one or more variables, as is the case with most applications of resultants in computer algebra, these algorithms lead to computations in the fraction field of R where the cost of polynomial gcds is relatively expensive. The gcds can be avoided in both cases by using fraction-free algorithms which make only exact polynomial divisions in R. The best fraction-free modification of the Euclidean algorithm for computing $\mathrm{res}_x(f, g)$ is the subresultant algorithm [5]. However, in general, in all of these algorithms, an expression swell occurs in the intermediate steps and it is more efficient to apply a modular approach. The first person to do this was Collins in [1] for $R = \mathbb{Z}[y_1, y_2, ..., y_k]$. To compute the resultant $r(y_1) = \mathrm{res}_x(f, g)$ for $R = \mathbb{Z}[y_1]$ using Collin's algorithm, or the modular resultant algorithm as it is often called, one computes the resultant modulo enough primes so that the answer over $\mathbb{Z}$ can be obtained by applying the Chinese remainder theorem. For each prime p_i, one computes the resultant in $\mathbb{Z}_{p_i}[y_1]$ at enough points $y_1 = \beta_j$ so that the resultant can be obtained by polynomial interpolation. This reduces the computation to many resultants in one variable over a finite field $\mathbb{Z}_p$ where no expression swell occurs. Collins proves that provided the primes and evaluation points are chosen so that the leading coefficients a_n and b_m do not vanish, the resultant can be recovered when sufficiently many primes and evaluation points are used. A bound on the largest integer coefficient and on the degree of any y_i appearing in the resultant can be obtained from inspecting Sylvester's matrix. We refer to [3] for a better coefficient bound.

In this paper we consider the case $R = K[y_1, y_2, ..., y_k]$ where $K = \mathbb{Q}(\alpha_1, \alpha_2, ..., \alpha_t)$ and the α_i are algebraic over $\mathbb{Q}$, that is, R is a polynomial ring over an algebraic number field. One application is Trager's algorithm [9] for factoring polynomials in $K[x]$ where resultants are used to compute norms of polynomials. Also, resultants over number fields will arise naturally when solving systems of polynomial equations where the coefficients involve algebraic numbers. The main contributions we make in the paper are:

1. We generalize the modular algorithm of Collins [1] to compute resultants of multivariate polynomials over number fields with one or more field extensions (Sections 2 and 3).

2. We improve the efficiency of Collin's algorithm in practice by using a probabilistic algorithm to compute the degree and also the low degree of the resultant in all variables. We give an error analysis for the case $R = \mathbb{Q}(\alpha_1)[y_1, y_2]$ (Section 4).

3. To avoid rational reconstruction in the recovery of the coefficients of the resultant, we pre-compute a multiple of the denominator of the resultant (Section 5).

4. We use primitive elements to convert multiple field extensions into one large extension *after* reducing modulo the prime for improved efficiency. The algorithm for the conversion uses gcds and resultants. We give a running time analysis for it (Section 6).

5. We have implemented the algorithms in a new recursive dense data structure in Maple [4] that is designed to speed up modular algorithms. We give some details of the data structure and implementation (Section 7).

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

We remark that for the case $R = \mathbb{F}_q[y_1, ..., y_k]$, where $\mathbb{F}_q$ is a finite field with q elements, the only difficulty in generalizing Collins' algorithm is when q is too small, that is, there are not enough evaluation points in $\mathbb{F}_q$ to interpolate some y_i. This can be resolved by taking evaluation points from $\mathbb{F}_{q^l}$, where q^l is chosen to be sufficiently large.

Before we look at the number field case we make some remarks about the efficiency of Collin's algorithm in comparison with the subresultant algorithm. Specifically, we want to justify our use of a probabilistic algorithm. Consider the following problem for $R = \mathbb{Z}[y]$. Suppose $\deg_x(f)$, $\deg_x(g)$, $\deg_y(f)$ and $\deg_y(g)$ are all n, and the length of the integer coefficients in f and g is $O(n)$. Thus the size of f and g is $O(n^3)$ and the size of $\text{res}_x(f, g)$ is $O(n^4)$ (neglecting logarithmic factors). One can show that the time complexity of Collin's algorithm is $O(n^6)$ and that of the subresultant algorithm is $O(n^{10})$. To give the reader a feel for the kind of improvement that one obtains on a specific example, for $f, g \in \mathbb{Z}[y][x]$ of total degree 10, with random 10 digit coefficients, the Maple implementation of Collin's algorithm (`resultant/modular2`) is 63 times faster than the Maple implementation of the subresultant algorithm (`resultant/subres`). As impressive as these results appear, we make the following critical observations.

Suppose $f = c_n x^n + c_{n-1}x^{n-1} + ... + c_1 x + c_0$ and $g = f + 1$. If $c_n = 1$ then $\text{res}_x(f, g) = 1$. If the subresultant algorithm is used it will terminate in one step since $\text{rem}_x(f, g) = 1$. If Collin's algorithm is used the bounds for the number of primes and evaluation points will depend on the size of the coefficients and n. We can see that Collin's algorithm will take arbitrarily longer than the subresultant algorithm on this class of problems. Do we care about such degenerate cases? We say yes, for it has been our experience that the degree bounds and coefficient bounds on many real problems are often very bad. This increases the cost of the modular resultant algorithm but has no effect on the subresultant algorithm. Space prevents us from properly substantiating this claim, however, we give two examples.

Example 1.1 (Cyclotomic Polynomials[1]): Let $\Phi_n(x)$ denote the n'th cyclotomic polynomial. Let $f = \Phi_n(x)$ and $g = \Phi_m(x)$. For $100 < m < n \leq 200$ we find that $\text{res}_x(f, g) = 1$! But for $m = 197$ and $n = 199$, Hadamards bound on the determinant of Sylvester's matrix is a 453 digit integer! Even for the exceptional case of $m = 100, n = 200$ where the resultant $\text{res}_x(f, g) = 2^{40}$, Hadamards bound is over three times longer than the resultant.

Example 1.2 (The Logistic Map[2]): Another way to construct resultants with degree and coefficient bounds which are poor estimates is to compose polynomials. Let $f(x) = ax(1 - x)$ for parameter a. Let $g(x) = f^{(n)}(x)$. The bifurcation points of f are the roots of the polynomial $r(a) = \text{res}_x(g(x) - x, g'(x) + 1)$. These resultants have much smaller coefficients than expected and are divisible by a large power of a.

[1]This example is given to us by Chris Smith, Department of Mathematics, University of Edinburgh.

[2]This example is given to us by Robert Corless, Department of Applied Mathematics, University of Western Ontario.

To improve the performance of the modular resultant algorithm we will require that the number of primes and the number of evaluation points used is proportional to the size of $\text{res}_x(f, g)$ and not dependent on the inputs f and g. To achieve this we will use a probabilistic approach for both the coefficients and the degree of the resultant.

We make one further observation. In our experience, if R is a polynomial ring, $\text{res}_x(f, g)$ is almost always *denser* than f and g. This contrasts with the gcd problem where the $\gcd(f, g)$ is usually *sparser* than the f and g. The reason for this is the following: when R is a polynomial ring, the resultant is an integer linear combination of products of polynomials and the product of two polynomials is usually denser than the inputs. Therefore, whereas sparse algorithms have practical advantages for the gcd problem, they are of much less value for the resultant problem. Consequently, we do not pursue a sparse approach here.

2. RESULTANTS OVER NUMBER FIELDS

Let $\deg_x(f)$ denote the degree of f in x and $\text{lc}_x(f)$ denote the leading coefficient of f in x. Let p be a prime and let $\phi_p : R[x_1, \ldots, x_k] \to R_p[x_1, \ldots, x_k]$ denote the modular homomorphism $\phi_p(f) = f \bmod p$.

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.

Figure 1 is a homomorphism diagram for computing multivariate resultants over number fields. It provides an overview of our modular algorithm. 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 five additional primes.

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

PRES computes multivariate resultants over finite rings with zero or more extensions. The algorithm over $\mathbb{Z}_p$ is from [1] but extended to support extensions over $\mathbb{Z}_p$. It uses dense evaluation and interpolation where $D_i \geq \deg_{y_i}(r)$ is a degree bound on the degree of y_i of the resultant.

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

REMARK In the diagram the inputs of PRES and URES are over $\mathbb{Z}_p[\tilde{z}]/\langle \tilde{m} \rangle$ (a finite ring with only one extension). This is because CRES collapses many extensions into one extension. However, our implementation of PRES and URES support multiple extensions, including finite fields.

$$\begin{tikzcd}
\mathbb{Q}[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{NRES}"] \arrow[d, "\phi_p"', "p\in\{p_1,\ldots,p_N\}"] & \mathbb{Q}[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k] \\
\mathbb{Z}_p[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{CRES}"] \arrow[d, "|_{z_i=\sigma_i(\tilde{z})}"', "i=1,\ldots,t"] & \mathbb{Z}_p[z_1,\ldots,z_t]/\langle m_1,\ldots,m_t\rangle[y_1,\ldots,y_k] \arrow[u, "\text{Chinese Remainder}", "\text{Rational Reconstruction}"'] \\
\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[y_1,\ldots,y_k][x] \arrow[r, "\text{PRES}"] \arrow[d, "|_{y_i=\beta_{ij}\in\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle}"', "i=1,\ldots,k;\ j=0,\ldots,D_i"] & \mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[y_1,\ldots,y_k] \arrow[u, "|_{\tilde{z}=z_1+\cdots+z_t}"] \\
\mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle[x] \arrow[r, "\text{URES}"] & \mathbb{Z}_p[\tilde{z}]/\langle\tilde{m}\rangle \arrow[u, "\text{Polynomial}", "\text{Interpolation}"']
\end{tikzcd}$$

Figure 1: Homomorphism Diagram for Multivariate Resultants

3. THE EUCLIDEAN ALGORITHM OVER A RING

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

$$\begin{aligned}
r_0 &= f, \\
r_1 &= g, \\
r_{i+1} &= \text{rem}_x(r_{i-1}, r_i), \text{ and} \\
r_{n+1} &= \text{rem}_x(r_{n-1}, r_n) = 0.
\end{aligned}$$

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

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

To avoid the coefficient swell, it is desirable to apply the Euclidean algorithm over a finite ring by first reducing the inputs f and g modulo some prime p. If F is a number field, since a minimal polynomial may become reducible modulo a prime, algorithm NRES may call URES over a finite ring with zero divisors which can cause URES to fail. For example, consider the finite ring $\mathbb{Z}_5[u]/\langle m\rangle$ where $m = 3u^3 - 1$ is the minimal polynomial[3]. Note that over $\mathbb{Z}_5$, $m = 3(u+2)(u^2+3u+4)$ 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 $\text{res}_x(f, g)$ by URES as follows,

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

[3]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.

The computation of r_3 fails because $\text{lc}_x(r_2) = 4(u+2)(u+4)$ is a zero divisor. If this occurs, a failure is generated; we simply trap the failure in NRES and try another prime. If the primes are large (typically, our primes are greater than 10^9), the proportion of zero divisors of the ring will be small, hence, failures will occur infrequently. The following theorems guarantee the correctness and termination of the modular resultant algorithm NRES for one field extension and one variable x. They can be generalized to a number field with $k > 1$ extensions. Proofs are based on [4].

THEOREM 1 (CORRECTNESS). *Let $f = a_sx^s + \cdots + a_0$, $g = b_tx^t + \cdots + b_0 \in \mathbb{Q}(\alpha)[x] = \mathbb{Q}[z]/\langle m\rangle[x]$ where $m \in \mathbb{Z}[z]$ is the minimal polynomial for α and where $s, t > 0$ and $a_s, b_t \neq 0$. Let p be a prime such that*

1. *$\phi_p(\text{lc}_z(m)) \neq 0$,*
2. *$\bar{f} = \phi_p(f)$ and $\bar{g} = \phi_p(g)$ exist,*
3. *$\phi_p(a_s) \neq 0$ and $\phi_p(b_t) \neq 0$, and*
4. *$\text{URES}(\bar{f}, \bar{g})$ terminates without failure.*

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

PROOF. Let M be the Sylvester matrix of f, g and $\bar{M}$ be the Sylvester matrix of $\bar{f}, \bar{g}$. Since $\phi_p(a_s) \neq 0$ and $\phi_p(b_t) \neq 0$, $\deg_x(\bar{f}) = \deg_x(f) = s$ and $\deg_x(\bar{g}) = \deg_x(g) = t$. Hence M and $\bar{M}$ have the same dimension of $s+t$. By definition, each entry $\bar{M}_{ij}$ of $\bar{M}$ is the image of the entry M_{ij} of M, that is, each $\bar{M}_{ij} = \phi_p(M_{ij})$. Hence $\bar{M} = \phi_p(M)$. Therefore, $\bar{r} = \det(\bar{M}) = \det(\phi_p(M)) = \phi_p(\det(M)) = \phi_p(r)$ because det is homomorphic. In addition, if $\text{URES}(\bar{f}, \bar{g})$ terminates, $\omega = \bar{r} = \phi_p(r)$ because all arithmetic operations in URES are homomorphic. $\square$

LEMMA 2 (EXISTENCE OF POLYNOMIALS). *Let $f \in \mathbb{Q}[x]$ or $f \in \mathbb{Q}[z]/\langle m\rangle[x]$ where m is the minimal polynomial. Let p be a prime. Let $\text{D} : \mathbb{Q}(\alpha)[x] \to \mathbb{Z}$ denote the integer denominator, which is the smallest integer such that $\text{D}(f) \times f \in \mathbb{Z}[\alpha][x]$. Then $\phi_p(f)$ exists if and only if $\text{D}(f) \not\equiv 0 \bmod p$.*

LEMMA 3 (NON-TRIVIAL ELEMENTS). *Let $f = a_{n-1}\alpha^{n-1} + \cdots + a_1\alpha + a_0 \in \mathbb{Q}(\alpha)$ where n is the degree of α. Let p be a prime such that $\phi_p(f)$ exists. Let each u_i denote the numerator of a_i. Let $\mathrm{C} : \mathbb{Q}(\alpha) \to \mathbb{Z}$ denote the greatest common divisor of $u_0, \ldots, u_{n-1}$. Then $\phi_p(f) \neq 0$ if and only if $\mathrm{C}(f) \not\equiv 0 \bmod p$.*

LEMMA 4 (EXISTENCE OF INVERSES). *Let $f \in \mathbb{Q}(\alpha)$ and p be a prime. Let $\mathrm{N} : \mathbb{Q}(\alpha) \to \mathbb{Q}$ denote the norm function. Then $\phi_p(f)^{-1}$ exists if and only if $\mathrm{N}(f) \not\equiv 0 \bmod p$.*

THEOREM 5 (TERMINATION). *If a prime satisfies the conditions in Theorem 1 we say it is a good prime, otherwise, we say it is a bad prime. All but finitely many primes are good.*

PROOF. Intuitively, if one would apply the Euclidean algorithm directly over $\mathbb{Q}(\alpha)$, the computation would be finite and hence the number of divisions is also finite. For each of the divisions there are only finitely many primes modulo which the inverse of an element of $\mathbb{Q}(\alpha)$ does not exist. Therefore, all but finitely many primes are good.

By Lemma 2, $\bar{f}, \bar{g}$ exist if and only if $\mathrm{D}(f), \mathrm{D}(g) \not\equiv 0 \bmod p$. By Lemma 3, $\phi_p(a_s), \phi_p(b_t) \neq 0$ if and only if $\mathrm{C}(a_s), \mathrm{C}(b_t) \not\equiv 0 \bmod p$. Since all $\mathrm{lc}_z(m)$, $\mathrm{D}(m)$, $\mathrm{D}(f)$, $\mathrm{D}(g)$, $\mathrm{C}(a_s)$ and $\mathrm{C}(b_t)$ are finite integers, there are only finitely many primes that divide any of them. Hence the first three conditions hold for all but finitely many primes.

Let $r_0, \ldots, r_{n+1} \in \mathbb{Q}(\alpha)[x]$ be the remainder sequence of f, g, as described at the beginning of this section. Let $\bar{r}_0, \ldots, \bar{r}_{k+1} \in \mathbb{Z}_p(\alpha)[x]$ be the remainder sequence of $\bar{f}, \bar{g}$ if the sequence exists. We prove by induction on k that, for all but finitely many primes, $\bar{r}_i$ exists and $\bar{r}_i = \phi_p(r_i)$ for $i = 0, \ldots, n+1$.

For $i = 0$ and $i = 1$, $\bar{r}_0 = \bar{f}_s = \phi_p(a_s) = \phi_p(r_0)$ and $\bar{r}_1 = \bar{g}_t = \phi_p(b_t) = \phi_p(r_1)$. Assume the statement is true for all $i \leq n$. Given $\bar{r}_{i-1}$ and $\bar{r}_i$ we obtain $\bar{r}_{i+1}$ by computing the remainder of $\bar{r}_{i-1}$ by $\bar{r}_i$. Let c_i be the leading coefficient of r_i in x. $\bar{r}_{i+1}$ exists and $\bar{r}_{i+1} = \phi_p(r_{i+1})$ if and only if the inverse of $\phi_p(c_i)$ exists. By Lemma 4, $\phi_p(c_i)^{-1}$ exists if and only if $\mathrm{N}(c_i) \not\equiv 0 \bmod p$. Since $\mathrm{N}(c_i)$ is a finite rational, there are only finitely many primes that divide it. Hence, for all but finitely many primes, $\bar{r}_i$ exists and $\bar{r}_i = \phi_p(r_i)$ for $i = 0, \ldots, n+1$. $\square$

Theorem 1 tells us that if algorithm URES terminates without failure for a prime p, it outputs $\phi_p(r)$, the resultant modulo p. Theorem 5 tells us that URES terminates without failure for all but finitely many primes. Therefore, algorithm NRES will eventually have the resultant modulo enough primes to reconstruct the resultant over $\mathbb{Q}$.

Below is the pseudo-code for NRES assuming that the input polynomials are over $\mathbb{Z}$, the minimal polynomials are monic over $\mathbb{Z}$ and the coefficients are multivariate polynomials in $Y = y_1, ..., y_k$. In Section 5 we extend it to $\mathbb{Q}$. Algorithm PDEG applies a probabilistic algorithm to obtain sharp degree bounds, which is discussed in Section 4.

Algorithm NRES: Number Field Resultant.
Input: $f, g \in K[Y][x]\backslash\{0\}$, $K = \mathbb{Z}[z_1, \ldots, z_t]/\langle m_1, \ldots, m_t\rangle$
Output: $r = \mathrm{res}_x(f, g) \in K[Y]$

1. {initialization}
   (a) {probabilistic degree bounds}
   $\Pi_\theta \leftarrow \mathrm{PDEG}(f, g)$
   (b) {big modulus prime}
   $p \leftarrow 10^8$
   (c) {image agreement count}
   $\eta \leftarrow 0$
   (d) {resultant so far}
   $r \leftarrow$ **null**

2. {main computation}
   while $\eta < 5$ **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}
   $\bar{r} \leftarrow \mathrm{CRES}(\phi_p(f), \phi_p(g), \Pi_\theta)$
   if $\bar{r} = \mathrm{FAIL}$ then goto step 2a.
   (c) {lift modulus images}
   $r_0 \leftarrow \mathrm{CHINESE\text{-}REMAINDER}([r, \bar{r}])$
   (d) {stop if five 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

4. DEGREE BOUNDS

Algorithm PRES 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 off, we will develop a probabilistic algorithm to obtain sharp degree bounds before we compute the resultant modulo a prime. We make a further observation. Very often in practice the resultant is divisible by large powers of some variables. Let $r = \mathrm{res}_x(f, g)$, $d = \deg_y(r)$ and $\mathrm{ldeg}_y(r)$ be the largest integer l such that $y^l | r$. If we know l in advance, it is easy to modify the interpolation process so that we need only $d - l + 1$ points to interpolate y.

Let us illustrate our algorithm with an example over $\mathbb{Z}$. Let $f(x, y, z) = y^2z^2x - y^3z^2x^2 + 2\,y^3zx^3 - y^2zx^2 - y^3x^4 - x$, $g(x, y, z) = y^2z^2 - 2\,y^3z^2x + 6\,y^3zx^2 - 2\,y^2zx - 4\,y^3x^3 + 1$ and let $r(y, z) = \mathrm{res}_x(f, g)$. We have

$$r = \left(y^2z^2 + 1\right)\left(y^2z^2 + 2\,yz - 5\right)^2\left(y^2z^2 + 4\,yz + 5\right)y^9.$$

We compute $\bar{r} = \mathrm{res}_x(\phi_p\, f(x, y, \alpha), \phi_p\, g(x, y, \alpha))$ for a suitable prime p and α chosen at random from $\mathbb{Z}_p$ using degree bounds for $\deg_y(r(y, z))$ based on Sylvester's matrix.[4] Then $\deg_y(\bar{r}) \leq \deg_y(r)$ and $\mathrm{ldeg}_y(\bar{r}) \geq \mathrm{ldeg}_y(r)$. If p is large then the probability of equality here is high. We repeat this to obtain degree estimates for $\deg_z(r)$ and $\mathrm{ldeg}_z(r)$.

[4]We could also use a probabilistic algorithm here, similar to that for NRES, such that we interpolate resultant images at different evaluation points until there are several agreements in a row.

In this example we have $\deg_y(r) = 17$, $\deg_z(r) = 8$, $\mathrm{ldeg}_y(r) = 11$, and $\mathrm{ldeg}_z(r) = 0$. Since the degree bounds we obtain from Sylvester's matrix are $\deg_y(r) \le 20$ and $\deg_z(r) \le 11$, the number of univariate resultants computed over $\mathbb{Z}_p$ is reduced from $(20+1)\times(11+1) = 252$ to $(17-11+1)\times(8-0+1) = 63$ for a savings of a factor of 4. If we do not take into account $\mathrm{ldeg}_y r$ we would need $(17+1)\times(8+1) = 162$ univariate resultants and our savings is a factor of only 1.56.

Let us determine the probability of obtaining a wrong degree bound for one field extension, i.e. for $K = \mathbb{Q}(\alpha)$. Let $S = \mathbb{Z}_p[u]/\langle m\rangle$ be a finite ring where $m \in \mathbb{Z}_p[u]$ is the minimal polynomial of degree q. Let $f, g \in S[y,z][x]$. We want to apply the probabilistic algorithm to obtain the degree bound on y of $r = \mathrm{res}_x(f,g) \in S[y,z]$. Let $\deg_y(r) = d$. We can write the resultant as $r = c_d(z)y^d + \cdots + c_1(z)y + c_0(z)$ where each $c_i(z) \in S[z]$. Let $\deg_z(c_d) = n$. Let $\bar{f}, \bar{g} \in S[y][x]$ be f, g evaluated at $z = \alpha$ for a random evaluation point $\alpha \in S$. Let $\bar{r} = \mathrm{res}_x(\bar{f}, \bar{g}) \in S[y]$ with $\deg_y(\bar{r}) = \bar{d}$. Let E denote the probability that the degree bound is wrong, that is, $E = \mathrm{Prob}(\bar{d} < d)$. This happens when α is a root of $c_d(z)$. The following lemma tells us what E is and suggests we should do.

Lemma 6 (Number of Roots in Finite Rings). *Let p be a prime, $m \in \mathbb{Z}_p[u]$ of degree $q > 0$, and $S = \mathbb{Z}_p[u]/\langle m\rangle$. Let $f \in S[z]$ have degree n. Let N be the number of roots of f in S.*

1. *If m is irreducible, then $N \le n$.*
2. *If m is square-free, then $N \le n^q$.*
3. *Otherwise $N \le np^{q-1}$.*

Remark The bound in case 3 cannot be improved significantly in general for the polynomial z^q has p^{q-1} roots in $\mathbb{Z}_p[u]/\langle u^q\rangle$.

Proof. For Case 1, S is a finite field. The statement is equivalent to the Fundamental Theorem of Algebra.

For Case 2, S is a simple finite ring. The worst case for E occurs when m has q distinct linear factors, that is, $m = (u-\beta_1)\cdots(u-\beta_q)$. The statement then follows from the Chinese remainder theorem.

For Case 3, S is an arbitrary finite ring. The worst case for E occurs when m has only one linear factor of multiplicity q, that is, $m = (u-\beta)^q$. We present a counting argument on the number of roots using p-adic lifting as follows. Let $S^i = \mathbb{Z}_p[u]/\langle (u-\beta)^i\rangle$ for $i = 1, \ldots, q$. Consider the base case when $q = 1$. Since $u - \beta$ is irreducible as in Case 1, f has at most n roots in S^1. Let $\alpha_1, \ldots, \alpha_n$ be the roots of f in S^1. Using p-adic lifting, we can lift these roots from S^1 to $S^2 = \mathbb{Z}_p[u]/\langle (u-\beta)^2\rangle$ so that f has at most np^1 roots in S^2 constructed by p-adic lifting. We can continue to lift to $S^q = \mathbb{Z}_p[u]/\langle (u-\beta)^q\rangle$ so that f has at most np^{q-1} roots in $S^q = S$ constructed by p-adic lifting. $\square$

We cannot always choose a prime p such that m is irreducible modulo p (Case 1). But we can with high probability choose p such that m is square-free modulo p (Case 2). Case 2 does not hold when $r = \mathrm{res}_u(m,' m) \equiv 0 \bmod p$. Since r is an integer of a finite length, Case 2 holds for all but finitely many primes.

Theorem 7 (Error Probability in Simple Rings). *Let S be as in Lemma 6. If m is square-free then $E \le (n/p)^q$.*

Proof. As above, let $c_d \in S[z]$ of degree n be the leading coefficient of r in y. Note that there are p^q possible evaluation points in S. Since S is a simple finite ring, by Lemma 6, c_d has at most n^q roots in S. Therefore, $E \le n^q/p^q = (n/p)^q$. $\square$

This is a good result. For example, for $n = 100$, $p > 10^8$, and $q = 4$, the probability of an error is less than $1/10^{24}$. Repeating this with a second prime $q > 10^8$ we would reduce the error to less than $1/10^{48}$.

Remark When the inputs are over $\mathbb{Q}$, it is possible that $c_d(z)$ becomes 0 modulo the prime we choose and thus the degree bound is wrong for all random points $\alpha \in S$. This happens when the prime divides all the coefficients of $c_d(z)$. Hence, instead of using the same prime but different β in algorithm PDEG, we use new primes to reduce the probability that this happens.

Remark When the inputs $f, g \in S[y_1, \ldots, y_k][x]$ where $k > 2$, Lemma 6 and Theorem 7 can be extended by replacing $S[z]$ in Lemma 6 and Theorem 7 by $S[z_1, \ldots, z_{k-1}]$ and generalizing the result of Schwartz [8] on the number of roots of $f \in \mathbb{Z}_p[z_1, \ldots, z_{k-1}]$ to $f \in S[z_1, \ldots., z_{k-1}]$.

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 make the inputs primitive over $\mathbb{Z}$, that is, we clear the fractions from the input by multiplying through by a suitable integer and divide out by any common factor of the numerators of the rational coefficients. However, the denominator problem with the minimal polynomials is not as easy to resolve. 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. Both reconstructions have a quadratic time complexity[5], but the former has

[5]Rational reconstruction has a cubic time complexity if it is applied after each image; [4] point out that, if it is applied after each F_i image (the Fibonacci numbers F_i: $1, 2, 3, 5, 8, 13, \ldots$), the reconstruction has a quadratic complexity.

a big constant. Hence, we will use integer reconstruction in the following.

To compute a multiple of the denominator 6 in the example, we start with the minimal polynomial of the last extension, which is $m_2(w,z)$ in the example. First, we compute the degree bound on z of the resultant, $d_z = 3$ using the probabilistic degree bound[6] in Section 4. Then we know there is at most $\kappa_z = d_z - \deg_z(m_2) + 1 = 3 - 3 + 1 = 1$ division by m_2. Hence the contribution of the extension $m_2(w,z)$ to the denominator multiple is $\mathrm{lc}_z(m_2)^{-\kappa_z} = 1/2$.

But, since m_2 is in $\mathbb{Q}[w]/\langle m_1\rangle[z]$, the division by m_2 also contribute to the degree in w. Let σ_w denote this accumulated degree from $m_2(w,z)$ for w. Working out the details[7], $\sigma_w = \deg_w(\lambda_w)\lceil \kappa_z/(\deg_z(m_2) - \deg_z(\lambda_w))\rceil = 2\lceil 2/(3-1)\rceil = 2$ where $\lambda_w = -3zw^2$ is the leading term of m_2 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 $m_1(w)$ 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_1, $\kappa_w = (\sigma_w + d_w) - \deg_w(m_1) + 1 = (2+4) - 3 + 1 = 4$. Hence the denominator multiple from all extensions is $\mathrm{lc}_w(m_1)^{-\kappa_w}\mathrm{lc}_z(m_2)^{-\kappa_z} = 1/(3^4 \times 2^1) = 1/162$.

Here are the recursive formulas for the general case. Let $f, g \in \mathbb{Q}[z_1, \ldots, z_t]/\langle m_1, \ldots, m_t\rangle[Y][x]$. Let θ_i denote the degree and c_i denote the leading coefficient of m_i in z_i. Let λ_j^k denote the term of degree k of m_j and λ_{ij}^k denote the degree of λ_j^k 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 and κ_i denote the bound on the number of divisions by m_i. Let η_{ij}^k denote the bound on the number of divisions by m_i due to λ_j^k. Let D denote the denominator multiple of $\mathrm{res}_x(f,g)$. Then,

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

6. COLLAPSING EXTENSIONS

Suppose the number field is of degree $n = 2^m$. The modular resultant algorithm will be considerably faster if the number field is presented by a single field extension of degree 2^m than by m quadratic field extensions. This is because of the overhead of storage management of small vectors and program control when working with extensions of low degree in comparison with the arithmetic operations in $\mathbb{Z}_p$ that are being done. This overhead can be measured. We have timed Maple's overhead (see Appendix) for polynomial multiplication and gcd operations in $\mathbb{Z}_p[z]/\langle m\rangle[x]$. We determined that the overhead for $\deg_z(m) = 4$ is a factor of 15 to 16 and the overhead for $\deg_z(m) = 8$ is a factor of 4 to 5. Therefore, we can get an non-asymptotic, but still very significant speed up for resultant computations by collapsing multiple extensions into one large extension. For a typical problem with a quadratic extension, this can speed up the entire resultant computation by a factor of 10. Although this causes severe expression swell over $\mathbb{Q}$, the conversion is rather efficient over $\mathbb{Z}_p$. We give an $O(n^3)$ conversion algorithm. Another reason to collapse the extensions into one large extension would be to take advantage of fast multiplication and division in $\mathbb{Z}_p[z]$.

Let us illustrate the conversion algorithm with an example. Consider a finite ring $\mathbb{Z}_7[w,z]/\langle m_1, m_2\rangle$ 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 representation of the primitive element $u = w + z$ of the two extensions[8], that is, we compute the minimal polynomial for u and the substitutions of w, z in u. One way to do this is to use linear algebra. Another, potentially faster way, is to use resultants and gcds. The idea, attributed to Collins by Loos in [6], is as follows. First, by computing a sequence of resultants, we eliminate the extension variables w, z from the minimal polynomials for w, z and the defining equation $u - w - z$. The last resultant in the sequence will be the minimal polynomial for u. Then we can recover the substitutions of w, z by computing in the new field $\mathbb{Z}_7[u]/\langle m\rangle$ the gcds of the minimal polynomials and the intermediate resultants.

Let us apply the algorithm to the example. First 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 $\gcd_x(f,g)$ denote the monic greatest common divisor of f, g in x. We compute $\gcd_w(g, r_1) \bmod \langle m\rangle = w - (3u^8 + 5u^7 + 2u^6 + 2u^5 + 3u^4 + 3u^2 + 3u + 6)$ and solve for w in $\mathbb{Z}_7[u]/\langle m\rangle$. Hence the substitution 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 proceed in this ring of 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 we back-substitute $u = w + z$ into $\tilde{r}$ to obtain $r = \mathrm{res}_x(f,g) = 3w^2z^2 + 2wz^2 + 5wz + 5z^2 + 5w + 2z + 5 \in \mathbb{Z}_7[w,z]/\langle m_1, m_2\rangle$.

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

[7] consider the Newton's polygon of the monomials of w, z and the monomial of the steepest slope; for example, given $m_1 = 3w^7 + 2$ and $m_2 = 5z^5 + w^4z^4 + w^6z^3 + 1$, the term w^4z^4 "dominates" the term w^6z^3 in calculating the accumulated degree for w

[8] At the end of this section we say what to do when $u = w + z$ is not a primitive element.

Theorem 8 below is taken from Loos [6]. It can be generalized to a number field with an arbitrary number of extensions by iterating the procedure, that is, eliminating the last field extension until there is only one left.

THEOREM 8 (CORRECTNESS). *Let $F[w,z]/\langle m_1, m_2\rangle$ be a number field where $m_1 \in F[w]$ of degree s and $m_2 \in F[w]/\langle m_1\rangle[z]$ of degree t 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 $m = r_0$. If $u = w+z$ is a primitive element then m is the minimal polynomial for u and $F[u]/\langle m\rangle \cong F[w,z]/\langle m_1, m_2\rangle$. Let $\sigma : F[w,z]/\langle m_1, m_2\rangle \to F[u]/\langle m\rangle$ be the mapping of the isomorphism. Then σ is given by

$$\gcd_w(m_1, r_1) \bmod \langle m\rangle = w - \sigma(w) \in F[u]/\langle m\rangle[w].$$

PROOF. Let $\alpha_1, \ldots, \alpha_s$ be the distinct roots of m_1 and $\beta_1, \ldots, \beta_t$ be the distinct roots of m_2. By an alternative definition of resultants [2],

$$\begin{aligned} r_1 &= \prod_{1\le j\le t} (u - w - \beta_j) \bmod \langle m_1\rangle \\ &= (u - w - \beta_1)\cdots(u - w - \beta_t) \in F[u][w] \\ r_0 &= \prod_{1\le i\le s}\prod_{1\le j\le t} (u - \alpha_i - \beta_j) \\ &= (u - \alpha_1 - \beta_1)\cdots(u - \alpha_s - \beta_t) \in F[u] \end{aligned}$$

Let each $\gamma_{ij} = \alpha_i + \beta_j$ be a root of $r_0 = m$. Since α_i, β_j are distinct, there are $s \times t$ distinct roots of m. Since both fields have the same dimension, $F[u]/\langle m\rangle \cong F[w,z]/\langle m_1, m_2\rangle$.

By construction, $\sigma(w) = \alpha_i = \gamma_{ij} - \beta_j$ for some i, j. Hence $w - \sigma(w) = w - \alpha_i$ is a factor of m_1 and $w - \sigma(w) = w - (\gamma_{ij} - \beta_j) = -(\gamma_{ij} - w - \beta_j)$ is also a factor of r_1. Suppose $w - \tilde{\alpha}$ is another common factor of r_1, m_1. Since any factor of r_1 is of the form $w - (\gamma_{ij} - \tilde{\beta})$ for some $\tilde{\beta}$, $\tilde{\alpha} = \gamma_{ij} - \tilde{\beta}$. This implies $\tilde{\alpha} = \alpha_i = \sigma(w)$ because α_i, β_j are distinct. Therefore, $\gcd_w(m_1, r_1) = w - \sigma(\alpha)$. ☐

The following theorem is stated for the equal degree case to obtain a meaningful measure for the time complexity of the conversion.

THEOREM 9 (TIME COMPLEXITY). *Let $\deg_w(m_1) = \sqrt{n}$ and $\deg_z(m_2) = \sqrt{n}$. Then $\deg_u(m) = [F[w,z]/\langle m_1, m_2\rangle : F] = n$ and the time complexity of the conversion is $O(n^3)$ arithmetic operations in F.*

REMARK This is the same complexity that one obtains if one uses linear algebra. However, it will be easier to obtain an asymptotic speedup using asymptotically fast polynomial operations than using asymptotically fast matrix operations. Moreover, we think an $O(n^2)$ conversion algorithm may be possible using this approach. In particular, the minimal polynomial is computed in $O(n^2)$.

PROOF. There are three main computations in the conversion. The first two are computing r_1 and $\gcd(m_1, r_1) \bmod \langle m\rangle$ in Theorem 8. The third is computing the other substitutions $\sigma(w^i z^j)$, for $i, j = 0, \ldots, \sqrt{n} - 1$, given $\sigma(w), \sigma(z)$. Note, $\sigma(z)$ is easily computed from $u = w + z$.

Let us first determine the time complexity for computing $r_0 = \mathrm{res}_w(r_1, m_1) \in F[u]$. Note that $\deg_u(r_0) = n$ and $\deg_w(r_1) = \deg_w(m_1) = \sqrt{n}$. To interpolate the monic r_0 we need n points for u, taking $O(n^2)$ operations in F. There are n image problems in $F[w]$. The Euclidean algorithm in URES takes $O(\sqrt{n} \times \sqrt{n})$ operations in F for each. Hence r_0 can be computed in $O(n^2 + n \times n) = O(n^2)$ operations in F.

To compute $\gcd_w(m_1, r_1) \bmod \langle m\rangle \in F[u]/\langle m\rangle[w]$, the Euclidean algorithm requires $O(\sqrt{n}\times\sqrt{n})$ operations in $F[u]/\langle m\rangle$ since $\deg_w(r_1) = \deg_w(m_1) = \sqrt{n}$. Using classical algorithms, arithmetic operations in $F[u]/\langle m\rangle$ are $O(n^2)$ since $\deg_u(m) = n$. Hence computing $\sigma(w)$ requires $O(n \times n^2) = O(n^3)$ operations in F.

Finally, given $\sigma(w), \sigma(z)$, we compute other substitutions $\sigma(w^i z^j) = \sigma(w)^i \sigma(z)^j \bmod \langle m\rangle \in F[u]/\langle m\rangle$. This requires $O(n)$ operations in $F[u]/\langle m\rangle$, or $O(n \times n^2) = O(n^3)$ operations in F. Hence the overall time complexity of the conversion is $O(n^3)$ operations in F. ☐

Over Finite Rings: When the inputs are over finite rings, however, the conversion may fail because of division 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 $(w^2 + 4w + 5)(w + 7)$ 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,

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

Note that $m = r_0 = (u^2 + 4u + 8)(u^2 + 3u + 6)(u + 2)^2$ over $\mathbb{Z}_{11}$ and thus $\mathbb{Z}_{11}[u]/\langle m\rangle$ is not a field. To obtain the substitution of w in u, we compute $\gcd_w(m_1, r_1)$ 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.

Primitive Elements: We have assumed that the sum of extension variables $u = w + z$ is a primitive element. This is not true in general. For example, consider the number field $\mathbb{Q}(\sqrt{2}, \sqrt{3} - \sqrt{2})$. Letting $w = \sqrt{2}$ and $z = \sqrt{3} - \sqrt{2}$, if we define $u = w + z = \sqrt{2} + (\sqrt{3} - \sqrt{2}) = \sqrt{3}$, then we cannot represent $\sqrt{2}$ in the new field $\mathbb{Q}u$. In the implementation, we check if substitutions of both extension variables exist and if not, try some other linear combination $u = w + sz$ of the extension variables and repeat the conversion modulo another prime p. Since the primes is large, choosing $s \in \mathbb{Z}_p$ randomly from $[100, \ldots, p - 100]$ will give us a primitive element with high probability. Moreover, the cost is cheap since all arithmetic is being done modulo p.

REMARK If all the minimal polynomials of the input field are square-free over $\mathbb{Q}$, we choose a prime and a linear combination of the extension variables such that the new minimal $h(u)$ polynomial is square-free modulo the prime. This ensures the existence of the substitutions and ensures a low probability of obtaining wrong degree bounds in Section 4.

7. IMPLEMENTATION

The algorithms in this paper are implemented using the `recden` package in Maple [4]. The source codes and test examples can be downloaded from [10]. `recden` is a *recursive dense* representation of polynomials. Formally `recden` polynomials are encoded in Maple as follows:

```
<poly>   ::= RPOLY( 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 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. Polynomial operations in $\mathbb{Z}_p[x]$ are carefully coded in Maple's kernel in the C language for efficiency. See Monagan [7].

In the following we give examples of `recden` for polynomials over $\mathbb{Z}$, a finite field, and a number field.

Example 7.1 (Integer Polynomial): Let $f_1 = x^2 + y^2 + 2 \in \mathbb{Z}[y][x]$. Its `recden` representation is:

```
RPOLY( [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 are no extensions. 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 $C_3 =$ `[1]` is that of x^2.

Example 7.2 (Finite Ring): Let $F = \mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle$ be a finite ring of cardinality 3^4. Let $f_2 = x^2 + y^2 + 2 \in F[x]$. Its `recden` representation is:

```
RPOLY([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 a finite ring.

Besides the efficient representation, `recden` also provides a user-friendly interface. The input command for 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 7.3 (Number Field): Let F be the number field $\mathbb{Q}(\sqrt[3]{5}, \sqrt{-2 - (\sqrt[3]{5})^2})$. Note that F has two algebraic extensions. 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], [w^3-5,z^2+w^2+2]);
```

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

```
> lprint(f3);
   RPOLY([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 $z^2 + w^2 + 2$. 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 ($x^2 + y^2 + 2 \in \mathbb{Z}[y][x]$) and of f_2 ($x^2 + y^2 + 2 \in \mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle[x]$). That is, the encoding is independent of variables and base rings.

Code Snippet: To give the reader a taste of the implementation, we show the code for Algorithm URES in Figure 2. It computes the resultant of two polynomials using the Euclidean algorithm. The following example shows URES executing over the number field $\mathbb{Q}[y]/\langle y^4 + y + 1\rangle$ then, after applying ϕ_3 to the polynomials, over the finite ring $\mathbb{Z}_3[y]/\langle y^4 + y + 1\rangle$.

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

Figure 2: Maple Code for Univariate Resultants

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

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

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

$$1 - 25y + 18y^2 - 14y^3 \bmod \langle y^4 + y + 1 \rangle$$

```
> URES(phirpoly(f,3),phirpoly(g,3));
```

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

8. 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*, Kluwer Academic, 1996.

[3] A. Goldstein, G. Graham, A Hadamard type bound on the coefficients of a determinant of polynomials. *SIAM Review*, **16**, pp. 394–395, 1974.

[4] Mark van Hoeij, Michael Monagan. A Modular GCD Algorithm over Number Fields Presented with Multiple Field Extensions. *Proceedings of ISSAC '02*, pp. 109–116, 2002.

[5] Rüdiger Loos. Generalized Polynomial Remainder Sequences. *Computer Algebra, Symbolic and Algebraic Computation*, second edition, Springer Verlag, pp. 115–138, 1983.

[6] Rüdiger Loos. Computing in Algebraic Extensions. *Computer Algebra, Symbolic and Algebraic Computation*, second edition, Springer Verlag, pp. 173–188, 1983.

[7] Michael Monagan. In-place Arithmetic for Polynomials over $\mathbb{Z}_n$. *DISCO '92*, pp. 22–34, 1992

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

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

[10] Stephen Chun-To Tse. Algorithms and Bounds for Resultants. Master Thesis, Computing Science Department, Simon Fraser University, 2002. `http://www.sfu.ca/~stephent/resultant`.

Appendix

The following timings show that, in Maple, when computing in $R[x]$ where $R = \mathbb{Z}_p[z]/\langle m(z) \rangle$, there is a loss of efficiency of a factor of 15 to 16 more than expected in theory when the degree of m is 4 (compared with 256).

```
> p := 46327:
> k1 := 4;
                              k1 := 4

> m := z^4-10*z^2+1:
> c := () -> randpoly(z,degree=k1-1,dense):
> n1 := 100:
> f := randpoly(x,dense,degree=n1,coeffs=c):
> g := randpoly(x,dense,degree=n1,coeffs=c):
> m := modp1(ConvertIn(m,z),p):
> F := evalgf1(ConvertIn(f,z,x),m,p):
> G := evalgf1(ConvertIn(g,z,x),m,p):
> st := time():
> evalgf1(Multiply(F,G),m,p):
> mt1 := time()-st: st := time():
> evalgf1(Gcd(F,G),m,p):
> gt1 := time()-st:
> k2 := 256;
                             k2 := 256

> c := () -> randpoly(z,degree=k2-1,dense):
> m := z^128+c():
> n2 := 10:
> f := randpoly(x,dense,degree=n2,coeffs=c):
> g := randpoly(x,dense,degree=n2,coeffs=c):
> p := modp1(Prime(1)):
> m := modp1(ConvertIn(m,z),p):
> F := evalgf1(ConvertIn(f,z,x),m,p):
> G := evalgf1(ConvertIn(g,z,x),m,p):
> st := time():
> evalgf1(Multiply(F,G),m,p):
> mt2 := time()-st: st := time():
> evalgf1(Gcd(F,G),m,p):
> gt2 := time()-st:
> Loss1 := mt1/n1^2/k1^2  * n2^2*k2^2/mt2;

                        Loss1 := 14.86883354

> Loss2 := gt1/n1^2/k1^2  * n2^2*k2^2/gt2;

                        Loss2 := 16.09043949
```