Resultants, Resolvents and the Computation of Galois Groups

Alexander D. Healy
ahealy@fas.harvard.edu

Abstract

We will develop the machinery of resultants and resolvent polynomials with the ultimate goal of understanding the so-called "resolvent method" for computing Galois groups over $\mathbb{Q}$. Along the way, we will encounter tangential applications of these tools, such as the determination of the minimal polynomial of, say, the sum of two algebraic numbers using resultants. In this way, this paper can serve at once as an exposition on computing Galois groups of rational polynomials and as an introduction to some techniques of computation in number fields.

1 Preliminaries

The following notational conventions will be used throughout the remaining sections. Given a polynomial $A(X) = a_n x^n + \cdots + a_1 x + a_0$, $\ell(A)$ denotes the leading coefficient, a_n, of A, and $\mathrm{cont}(A)$, the *content* of A, is equal to $\gcd(a_n, a_{n-1}, \ldots, a_0)$. We will assume a familiarity with basic mathematical/numerical algorithms such as the Euclidean algorithm (for computing the gcd of two elements of a Euclidean domain) and Newton's method (for approximating the real and complex roots of polynomials over $\mathbb{R}$). Such topics, as well as the majority of the material in the following sections, are covered in either [Coh93], [Knu73] or [Knu81].

2 Resultants

Definition 2.1. *Let $\mathcal{R}$ be an integral domain. Given two polynomials $A(X)$, $B(X) \in \mathcal{R}[X]$ with roots $\alpha_1, \ldots, \alpha_m$ and $\beta_1, \ldots, \beta_n$ respectively (in some extension of the field of fractions of $\mathcal{R}$), the resultant $R(A,B)$ of A and B is defined to be*

$$R(A,B) = \ell(A)^n \ell(B)^m \prod_{i,j} (\alpha_i - \beta_j)$$

which is equivalent to both

$$R(A,B) = \ell(A)^n B(\alpha_1) \cdots B(\alpha_m)$$

and

$$R(A,B) = (-1)^{nm} \ell(B)^m A(\beta_1) \cdots A(\beta_n)$$

The resultant has several properties which make it useful and easy to compute. For instance, $R(A,B) \in \mathcal{R}$, which is a corollary to the following proposition:

Proposition 2.2. *Let* $A(X) = a_m X^m + \cdots + a_0$ *and* $B(X) = b_n X^n + \cdots + b_0$, *and let*

$$S = \begin{bmatrix} a_m & \cdots & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_m & \cdots & \cdots & a_0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_m & \cdots & \cdots & a_0 \\ b_n & \cdots & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_n & \cdots & \cdots & b_0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & b_n & \cdots & \cdots & b_0 \end{bmatrix}$$

be Sylvester's matrix. Then $R(A, B) = \det(S)$.

Proof. We follow [Coh93] and use Vandermonde determinants to show this proposition. Recall the following Vandermonde determinant identity:

$$\begin{vmatrix} x_1^{k-1} & \cdots & x_k^{k-1} \\ \vdots & \vdots & \vdots \\ x_1^0 & \cdots & x_k^0 \end{vmatrix} = \prod_{i<j} (x_i - x_j)$$

Let

$$V = \begin{bmatrix} \beta_1^{m+n-1} & \cdots & \beta_n^{m+n-1} & \alpha_1^{m+n-1} & \cdots & \alpha_m^{m+n-1} \\ \beta_1^{m+n-2} & \cdots & \beta_n^{m+n-2} & \alpha_1^{m+n-2} & \cdots & \alpha_m^{m+n-2} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \beta_1^0 & \cdots & \beta_n^0 & \alpha_1^0 & \cdots & \alpha_m^0 \end{bmatrix}$$

where we take the α_i, β_i to be (distinct) formal variables defined by $A(\alpha_i) = 0, B(\beta_i) = 0$, and note that

$$\det(V) = \prod_{1 \le i < j \le n} (\beta_i - \beta_j) \prod_{1 \le i < j \le m} (\alpha_i - \alpha_j) \prod_{\substack{1 \le i \le n \\ 1 \le j \le m}} (\beta_i - \alpha_j)$$

by the Vandermonde identity. Now consider

$$SV = \begin{bmatrix} \beta_1^{n-1} A(\beta_1) & \cdots & \beta_n^{n-1} A(\beta_n) & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \beta_1^0 A(\beta_1) & \cdots & \beta_n^0 A(\beta_n) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \alpha_1^{m-1} B(\alpha_1) & \cdots & \alpha_m^{m-1} B(\alpha_m) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \alpha_1^0 B(\alpha_1) & \cdots & \alpha_m^0 B(\alpha_m) \end{bmatrix}$$

Naturally, $\det(SV)$ is equal to the product of the determinants of the upper-left block and the lower-right block. Each of these blocks is a Vandermonde matrix where the columns have been scaled by $A(\beta_i)$ or $B(\alpha_i)$, so by applying the Vandermonde identity to each block, we have

$$\begin{aligned} \det(SV) &= A(\beta_1) \cdots A(\beta_n) B(\alpha_1) \cdots B(\alpha_m) \prod_{1 \le i < j \le n} (\beta_i - \beta_j) \prod_{1 \le i < j \le m} (\alpha_i - \alpha_j) \\ &= a_m^n B(\alpha_1) \cdots B(\alpha_m) \prod_{1 \le i < j \le n} (\beta_i - \beta_j) \prod_{1 \le i < j \le m} (\alpha_i - \alpha_j) \prod_{\substack{1 \le i \le n \\ 1 \le j \le m}} (\beta_i - \alpha_j) \\ &= a_m^n B(\alpha_1) \cdots B(\alpha_m) \det(V) \end{aligned}$$

Recall that α_i and β_i are distinct formal variables so $\det(V) \neq 0$. Hence, $\det(S) = a_m^n B(\alpha_1) \cdots B(\alpha_m)$, and when we replace the formal variables α_i and β_i, with the actual roots of A and B, we get $\det(S) = R(A, B)$. □

This gives us a method for computing resultants explicitly as the determinant of Sylvester's matrix. When $\mathcal{R} = \mathbb{Z}$, another approach to computing the resultant of $A(X)$ and $B(X)$ would be to find the roots of both polynomials in $\mathbb{C}$ to high precision, and then explicitly calculate $\ell(A)^n \ell(B)^m \prod_{i,j} (\alpha_i - \beta_j)$, rounding the result to the nearest integer. This is quite costly however, and is hopeless when $\mathcal{R}$ is a more complicated ring, for example when $\mathcal{R} = \mathbb{Q}[Y]$.

The following algorithm (from [Coh93]) allows us to compute the resultant $R(A, B)$ very efficiently, while only performing computations in $\mathcal{R}[X]$:

Algorithm 2.3 (Sub-Resultant).
SUB-RESULTANT(A, B)
    IF $A = 0$ *or* $B = 0$ THEN RETURN 0;
    IF $\deg(A) < \deg(B)$ THEN
        $\text{swap}(A, B)$;
        $s = (-1)^{\deg(A) \cdot \deg(B)}$;
    END IF
    $a = \text{cont}(A)$;
    $b = \text{cont}(B)$;
    $A = A/a$;
    $B = B/b$;
    $t = a^{\deg(B)} \cdot b^{\deg(A)}$;
    $g = 1$;
    $h = 1$;
    $\delta = \deg(A) - \deg(B)$;
    WHILE $\deg(B) > 0$ DO
        $R = \ell(B)^{\delta+1} A \pmod{B}$;   \\ *i.e. R is such that* $\ell(B)^{\delta+1} A = BQ + R$
        $A = B$;
        $B = R/(g \cdot h^{\delta})$;
        $g = \ell(A)$;
        $h = h^{1-\delta} \cdot g^{\delta}$;
        $\delta = \deg(A) - \deg(B)$;
        $s = (-1)^{\deg(A) \cdot \deg(B)} \cdot s$;
    END WHILE
    RETURN$(s \cdot t \cdot h^{1-\deg(A)} \cdot B^{\deg(A)})$;
END *SUB-RESULTANT*

Most of the steps of this algorithm are very straightforward computationally, except perhaps the line that reads "$R = \ell(B)^{\delta+1} A \pmod{B}$;". The assignment in this line is accomplished by using a variant of Euclidean division of polynomials known as "pseudo-division", which is very fast in part because all the intermediate computations remain in $\mathcal{R}[X]$. The details of this calculation are discussed in [Knu81].

Next we will prepare to prove the correctness of the sub-resultant algorithm. Let A_i be the value of the variable A at the beginning of the i-th iteration of the WHILE loop ($i = 0, 1, 2, \ldots$), and let g_i, h_i, δ_i, s_i be the values of g, h, δ, s respectively at the beginning of the i-th iteration of the WHILE loop. For notational convenience, set $\ell_i = \ell(A_i)$ and $d_i = \deg(A_i)$. Following [Coh93], we will prove the validity of Algorithm 2.3 by proving an inductive relationship between $R(A_k, A_{k+1})$ and $R(A_{k+1}, A_{k+2})$. Applying this relationship $r+1$ times, where r is the index of the final iteration, we will see that the quantity that is returned, namely $s_{r+1} \cdot t \cdot h_{r+1}^{1-d_{r+1}} \cdot A_{r+2}^{d_{r+1}}$, is exactly equal to $R(A, B)$.

Proof of correctness of Algorithm 2.3. Clearly, if either $A_0 = 0$ or $A_1 = 0$, then $R(A_0, A_1) = 0$. All the other lines in the algorithm that precede the WHILE loop serve to ensure that $\deg(A_0) \geq \deg(A_1)$, to guarantee that A_0 and A_1 are primitive, i.e. their content is a unit, and to initialize $g_0 = h_0 = 1, \delta_0 = d_0 - d_1$. From the definition of the resultant, it is clear that $R(aA_0, bA_1) = a^{d_1} b^{d_0} R(A_0, A_1) = tR(A_0, A_1)$, so we may assume that A_0 and A_1 are primitive and simply show that $R(A_0, A_1) = s_{r+1} \cdot h_{r+1}^{1-d_{r+1}} \cdot A_{r+2}^{d_{r+1}}$.

By the definition of the resultant,

$$R(A_k, A_{k+1}) = (-1)^{d_k d_{k+1}} \cdot \ell_{k+1}^{d_k} \cdot \prod_{i=0}^{d_{k+1}} A_k(\beta_i)$$

where β_i are the roots of A_{k+1}. Since $\ell_{k+1}^{\delta_k+1} A_k = Q_{k+1} \cdot A_{k+1} + R_{k+1}$, $A_k(\beta_i) = \frac{R_{k+1}(\beta_i)}{\ell_{k+1}^{\delta_k+1}}$, so this is equal to

$$\begin{aligned}(-1)^{d_k d_{k+1}} \cdot \ell_{k+1}^{d_k} \cdot \prod_{i=0}^{d_{k+1}} \frac{R_{k+1}(\beta_i)}{\ell_{k+1}^{\delta_k+1}} &= (-1)^{d_k d_{k+1}} \cdot \ell_{k+1}^{d_k - d_{k+1}(\delta_k+1)} \cdot \prod_{i=0}^{d_{k+1}} R_{k+1}(\beta_i) \\ &= (-1)^{d_k d_{k+1}} \cdot \ell_{k+1}^{d_k - d_{k+1}(\delta_k+1)} \cdot \frac{1}{\ell_{k+1}^{d_{k+2}}} R(A_{k+1}, g_k h_k^{\delta_k} A_{k+2})\end{aligned}$$

since $\frac{R_{k+1}}{g_k h_k^{\delta_k}} = A_{k+2}$. Again, from the definition of the resultant, it is easy to see that $R(A, cB) = c^{\deg(A)} R(A, B)$, so the above is equal to

$$(-1)^{d_k d_{k+1}} \cdot \ell_{k+1}^{d_k - d_{k+1}(\delta_k+1) - d_{k+2}} \cdot \left(g_k h_k^{\delta_k}\right)^{d_{k+1}} R(A_{k+1}, A_{k+2})$$

For $i > 0$ we have $\ell_i = g_i$ and $h_i = h_{i-1}^{1-\delta_{i-1}} \cdot g_i^{\delta_{i-1}}$, allowing us to write this as

$$\begin{aligned} R(A_k, A_{k+1}) &= (-1)^{d_k d_{k+1}} \cdot g_{k+1}^{d_k - d_{k+1}(\delta_k+1) - d_{k+2}} \cdot \left(g_k h_k^{\delta_k}\right)^{d_{k+1}} \cdot R(A_{k+1}, A_{k+2}) \\ &= (-1)^{d_k d_{k+1}} \cdot \frac{g_{k+1}^{d_k} g_k^{d_{k+1}} h_k^{\delta_k d_{k+1}}}{g_{k+1}^{\delta_k d_{k+1}} g_{k+1}^{d_{k+1}} g_{k+1}^{d_{k+2}}} R(A_{k+1}, A_{k+2}) \\ &= (-1)^{d_k d_{k+1}} \frac{g_k^{d_{k+1}} h_k^{\delta_k d_{k+1}}}{g_{k+1}^{\delta_k(d_{k+1}-1)} g_{k+1}^{d_{k+2}}} R(A_{k+1}, A_{k+2}) \\ &= (-1)^{d_k d_{k+1}} \frac{g_k^{d_{k+1}} h_k^{\delta_k d_{k+1}}}{\left(\frac{h_{k+1}}{h_k^{1-\delta_k}}\right)^{d_{k+1}-1} g_{k+1}^{d_{k+2}}} R(A_{k+1}, A_{k+2}) \\ &= (-1)^{d_k d_{k+1}} \frac{g_k^{d_{k+1}} h_k^{d_k - 1}}{g_{k+1}^{d_{k+2}} h_{k+1}^{d_{k+1}-1}} R(A_{k+1}, A_{k+2}) \end{aligned}$$

Recall that r is the index of the final iteration of the algorithm, so by applying the above relationship between $R(A_k, A_{k+1})$ and $R(A_{k+1}, A_{k+2})$ a total of $r+1$ times, we have

$$R(A_0, A_1) = (-1)^{\sum_{i=0}^{r+1} d_i d_{i+1}} \frac{g_0^{d_1} h_0^{d_0 - 1}}{g_{r+1}^{d_{r+2}} h_{r+1}^{d_{r+1}-1}} R(A_{r+1}, A_{r+2}) = s_{r+1} h_{r+1}^{1-d_{r+1}} A_{r+2}^{d_{r+1}}$$

since $g_0 = h_0 = 1$ and A_{r+2} is a constant (so $d_{r+2} = 0$ and $R(A_{r+1}, A_{r+2}) = A_{r+2}^{d_{r+1}}$). Hence, Algorithm 2.3 returns the correct value of $R(A_0, A_1)$ when A_0 and A_1 are primitive. We noted earlier that this is sufficient to prove the correctness of the algorithm when A_0 and A_1 are not necessarily primitive, and so the proof is complete. $\square$

Another remarkable fact about this algorithm, is that despite the numerous divisions in the line that reads "$B = R/(g \cdot h^\delta)$;", all constants remain in the ring $\mathcal{R}$ and all polynomials remain in $\mathcal{R}[X]$, (see [Knu81]). Finally, it is interesting to note that by simply changing the line of Algorithm 2.3 that reads

$$\text{RETURN}(s \cdot t \cdot h^{1-\deg(A)} \cdot B^{\deg(A)});$$

to

$$\text{RETURN}(\gcd(a,b)A/\operatorname{cont}(A));$$

we obtain an efficient algorithm for computing the greatest common divisor of two polynomials in $\mathcal{R}[X]$. In fact, this is the algorithm which is studied in [Knu81], and the algorithm which is recommended for computing the greatest common divisor of polynomials in [Coh93].

Applications of Resultants

Algorithm 2.3 is particularly powerful because it gives us an efficient way to implicitly manipulate the roots of polynomials, while only performing computations in the ring $\mathcal{R}[X]$. For instance, if we let α, β be two algebraic numbers with minimal polynomials $f(X), g(X)$ respectively, then the quantity $R_Y(f(X-Y), g(Y))$, (i.e. the resultant with respect to the variable Y, taking $\mathcal{R} = \mathbb{Q}[X]$), is equal to:

$$\ell(g)^{\deg(f)} f(X-\beta_1) f(X-\beta_2) \cdots f(X-\beta_n)$$

which has roots $\alpha_i + \beta_j$ where α_i are the roots of f and β_j are the roots of g. In particular, $\alpha + \beta$ is a root of $h(X) = R_Y(f(X-Y), g(Y))$, so if $h(X)$ is irreducible, it is the minimal polynomial of $\alpha+\beta$. On the other hand, if $h(X)$ is not irreducible, then one of its irreducible factors must be the minimal polynomial of $\alpha+\beta$, so we can factor $h(X)$ as $h_1(X)\cdots h_k(X)$ and evaluate each of $h_i(X)$ at a high-precision numerical approximation of $\alpha+\beta$ to find the factor that has $\alpha+\beta$ as a root. (For this example, we will assume that we have a procedure for factoring polynomials over $\mathbb{Z}[X]$, as this topic is far beyond the scope of this paper. The subject of modern polynomial factoring methods is treated in [Coh93].)

For example, $\alpha = \sqrt{2} + \sqrt{3}$ and $\beta = \sqrt{5} + \sqrt{6}$ are roots of $f(X) = X^4 - 10X^2 + 1$ and $g(X) = X^4 - 22X^2 + 1$, respectively. *A priori*, however, it is not apparent what the minimal polynomial of $\alpha + \beta = \sqrt{2} + \sqrt{3} + \sqrt{5} + \sqrt{6}$ is. Nonetheless, we can compute the resultant $h(X) = R_Y(f(X-Y), g(Y))$ using, for instance, the GP/PARI command `polresultant` (see [BBB+00]):

```
?polresultant((X - Y)^4 - 10*(X - Y)^2 + 1, Y^4 - 22*Y^2 + 1, Y)
```

which returns

$$\begin{aligned} h(X) &= X^{16} - 128X^{14} + 5712X^{12} - 117248X^{10} + 1169248X^8 \\ &\qquad -5289984X^6 + 8195328X^4 - 1990656X^2 + 20736 \\ &= \left(X^8 - 64X^6 + 96X^5 + 808X^4 - 1152X^3 - 2304X^2 + 1152X + 144\right) \\ &\qquad \left(X^8 - 64X^6 - 96X^5 + 808X^4 + 1152X^3 - 2304X^2 - 1152X + 144\right) \\ &= h_1(X) \cdot h_2(X) \end{aligned}$$

Now we need to determine which of $h_1(X)$ and $h_2(X)$ actually has $\alpha + \beta$ as a root. $\alpha + \beta \approx 7.83182209$, and $h_1(7.83182209) = 4,568,619.29\ldots$, whereas $h_2(7.83182209) = -0.000689531\ldots$, so the minimal polynomial of $\alpha + \beta$ is

$$h_2(X) = X^8 - 64X^6 - 96X^5 + 808X^4 + 1152X^3 - 2304X^2 - 1152X + 144$$

Similarly, $R_Y(f(X+Y), g(Y)), R_Y(f(X/Y), g(Y)), R_Y(f(XY), g(Y))$ will give polynomials with roots $\alpha + \beta, \alpha\beta$ and α/β, respectively.

Another application of the sub-resultant algorithm is the *Tschirnhausen transformation*, which will play an important role in section 4 when we examine the computation of Galois groups. The purpose of this transformation is to take a monic irreducible polynomial $P(X) \in \mathbb{Z}[X]$ of degree n and return another monic irreducible polynomial $Q(X)$ of degree n with the same splitting field, and hence the same Galois group. The algorithm is as follows:

Algorithm 2.4 (Random Tschirnhausen Transformation).
TSCHIRNHAUSEN$(P(X))$
    $Q = X^2$;
    WHILE $gcd(Q, Q') \notin \mathbb{Z}$ DO
        Choose $A \in \mathbb{Z}[X]$ *randomly, with* $\deg(A) = n - 1$
        (e.g. choose n *coefficients at random from* $\{-100, -99, \ldots, 99, 100\}$)
        $Q = R_Y(P(X), X - A(Y))$;
    END WHILE
    RETURN Q
END *TSCHIRNHAUSEN*

In the following sections we will employ this transformation when the "resolvent" we are considering has integer roots all of which have multiplicity greater than one. In this case, we hope that by applying a Tschirnhausen transformation to the input polynomial we will obtain a new resolvent with either a simple integer root or no integer roots at all. While we do not prove that the transformation will achieve this in a reasonable amount of time (or that the transformation itself is an efficient computation), it is very effective and runs very rapidly in practice. We will, nonetheless, prove the correctness of the algorithm.

Proof of correctness. Clearly, when the algorithm terminates, it returns a square-free polynomial $Q(X) = R_Y(P(X), X - A(Y))$ for some $A(X) \in \mathbb{Z}[X]$. By the definition of the resultant, $Q(X) = (X - A(\alpha_1)) \cdots (X - A(\alpha_n))$, so the splitting field of $Q(X)$, $\mathrm{Spl}(Q) = \mathbb{Q}(A(\alpha_1), \ldots, A(\alpha_n))$, is contained in the splitting field of $P(X)$, $\mathrm{Spl}(P) = \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$. Every $\sigma \in \mathrm{Gal}(P)$ acts by permuting the α_i, and hence by permuting the $A(\alpha_i)$. Now we consider the action of $\mathrm{Gal}(P)$ on the roots of $Q(X)$. In particular, every automorphism of $\mathrm{Spl}(P)$ restricts to an automorphism of the splitting field of $\mathrm{Spl}(Q)$.

Since the $A(\alpha_i)$ are distinct by assumption, the restriction of each $\sigma \in \mathrm{Gal}(P)$ to $\mathrm{Spl}(Q)$, is in fact a distinct automorphism. In particular, $|\mathrm{Aut}(\mathrm{Spl}(Q))| = [\mathrm{Spl}(P) : \mathbb{Q}] \geq [\mathrm{Spl}(Q) : \mathbb{Q}]$. Since it is true in general that $|\mathrm{Aut}(\mathrm{Spl}(Q))| \leq [\mathrm{Spl}(Q) : \mathbb{Q}]$, we have that $|\mathrm{Aut}(\mathrm{Spl}(Q))| = [\mathrm{Spl}(Q) : \mathbb{Q}]$. Because $|\mathrm{Aut}(\mathrm{Spl}(Q))| = [\mathrm{Spl}(P) : \mathbb{Q}] = [\mathrm{Spl}(Q) : \mathbb{Q}]$ and $\mathrm{Spl}(Q) \subseteq \mathrm{Spl}(P)$, we can conclude that $\mathrm{Spl}(Q) = \mathrm{Spl}(P)$. □

Finally, we will show how the sub-resultant algorithm can be used to efficiently compute the discriminant of a polynomial $A(X) \in \mathcal{R}[X]$. Recall that the discriminant of a polynomial $A(X)$ of degree n with roots $\alpha_1, \ldots, \alpha_n$, is defined as (assuming $\mathrm{char}\mathcal{R} = 0$)

$$\begin{aligned}
\mathrm{disc}(P) &= \ell(A)^{2n-2} \prod_{1 \leq i < j \leq n} (\alpha_i - \alpha_j)^2 \\
&= (-1)^{\binom{n}{2}} \ell(A)^{n-2} \left[\ell(A) \prod_{i \neq 1} (\alpha_1 - \alpha_i)\right] \cdots \left[\ell(A) \prod_{i \neq n} (\alpha_n - \alpha_i)\right] \\
&= (-1)^{\binom{n}{2}} \ell(A)^{n-2} A'(\alpha_1) \cdots A'(\alpha_n) \\
&= (-1)^{\binom{n}{2}} R(A, A')/\ell(A)
\end{aligned}$$

giving us an efficient way to compute the discriminant using the sub-resultant algorithm. Note that every entry in the first column of Sylvester's matrix (where $B = A'$) is divisible by $\ell(A)$ (since the only entries in the first column are zero and $\ell(A) = \ell(A')$), so $R(A, A') = \det(S)$ is in fact divisible by $\ell(A)$, which proves that $\mathrm{disc}(A) \in \mathcal{R}$.

3 Resolvents

Definition 3.1. *Given a polynomial $F(X_1, \ldots, X_n) \in \mathbb{Z}[X_1, \ldots, X_n]$, a subgroup $G \subseteq S_n$, and a polynomial $P(X) \in \mathbb{Z}[X]$ with roots $\alpha_1, \ldots, \alpha_n$, let*

$$H = \mathrm{Stab}_G(F) = \{\sigma \in G | F(X_{\sigma(1)}, \ldots, X_{\sigma(n)}) = F(X_1, \ldots, X_n)\}$$

be the stabilizer of F in G. Then the resolvent $\mathrm{Res}_G(F, P)$ *is defined by*

$$\mathrm{Res}_G(F, P)(X) = \prod_{\sigma \in G/H} \left(X - F(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})\right)$$

where σ ranges over a set of $|G/H|$coset representatives of G/H.

The following result will play an important role in the determination of Galois groups.

Proposition 3.2. *If $\mathrm{Gal}(P)$ is conjugate (in G) to a subgroup of $H = \mathrm{Stab}_G(F)$, then $\mathrm{Res}_G(F, P)$ has a root in $\mathbb{Z}$. Furthermore, if $\mathrm{Res}_G(F, P)$ has a* simple *root in $\mathbb{Z}$ then $\mathrm{Gal}(P)$ is conjugate to a subgroup of $H = \mathrm{Stab}_G(F)$.*

Proof. First, suppose $\mathrm{Gal}(P)$ is conjugate to subgroup $N \leq H$, i.e. $\mathrm{Gal}(P) = \sigma^{-1} N \sigma$ for a fixed $\sigma \in G$. Pick an arbitrary $\tau \in \mathrm{Gal}(P)$. $\tau = \sigma^{-1} \nu \sigma$ for some $\nu \in N$. Let $\eta \in H$ be such that $\sigma^{-1}\eta \in G/H$ is the representative of the coset $\sigma^{-1}H$. Then one of the roots of $\mathrm{Res}_G(F,P)$ is $F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})$, by definition. If we apply τ to this root, we have

$$\begin{aligned}\tau F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)}) &= \sigma^{-1}\nu\sigma F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})\\ &= \sigma^{-1}\nu F(\alpha_{\eta(1)},\ldots,\alpha_{\eta(n)})\\ &= \sigma^{-1}F(\alpha_{\nu\eta(1)},\ldots,\alpha_{\nu\eta(n)})\\ &= \sigma^{-1}F(\alpha_1,\ldots,\alpha_n)\end{aligned}$$

since both ν and η are in $H=\mathrm{Stab}_G(F)$. Hence,

$$\tau F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)}) = F(\alpha_{\sigma^{-1}(1)},\ldots,\alpha_{\sigma^{-1}(n)})$$

which is exactly $F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})$, since they only differ by $\eta\in\mathrm{Stab}(F)$. In particular, the root $F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})$, is fixed by $\mathrm{Gal}(P)$, so it must be in $\mathbb{Q}$, and hence in $\mathbb{Z}$ since the α_i are integral, and thus so is $F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})$.

On the other hand, if $F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})\in\mathbb{Z}$ is a simple root, it must be fixed by $\mathrm{Gal}(P)$. Therefore, we have that for any $\tau\in\mathrm{Gal}(P)$,

$$\tau F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)}) = F(\alpha_{\sigma^{-1}\eta(1)},\ldots,\alpha_{\sigma^{-1}\eta(n)})$$

and since $\eta\in H=\mathrm{Stab}_G(F)$, this is equivalent to

$$\tau F(\alpha_{\sigma^{-1}(1)},\ldots,\alpha_{\sigma^{-1}(n)}) = F(\alpha_{\sigma^{-1}(1)},\ldots,\alpha_{\sigma^{-1}(n)})$$

It is an easy exercise to verify that $\mathrm{Stab}_G(\sigma^{-1}F)=\sigma^{-1}\,\mathrm{Stab}(F)\sigma$ for any $\sigma^{-1}\in G$, and consequently that $\mathrm{Res}_G(\sigma^{-1}F,P)=\mathrm{Res}_G(F,P)$. Therefore, we can renumber the roots of $P(X)$, by setting $\alpha'_i=\alpha_{\sigma^{-1}(i)}$, to get

$$\tau F(\alpha'_1,\ldots,\alpha'_n) = F(\alpha'_1,\ldots,\alpha'_n)$$

where $F(\alpha'_1,\ldots,\alpha'_n)\in\mathbb{Z}$ is a simple root of $\mathrm{Res}_G(\sigma^{-1}F,P)=\mathrm{Res}_G(F,P)$. Therefore, $\tau\in\mathrm{Stab}_G(\sigma^{-1}F)$, for if it were not then τ would belong to another coset of $\mathrm{Stab}(\sigma^{-1}F)$, and hence $\tau F(\alpha'_1,\ldots,\alpha'_n)$ would be a different root (of $\mathrm{Res}_G(\sigma^{-1}F,P)=\mathrm{Res}_G(F,P)$) from $F(\alpha'_1,\ldots,\alpha'_n)$. This cannot be the case since we assumed that $F(\alpha'_1,\ldots,\alpha'_n)$ was a simple root. So $\tau\in\mathrm{Stab}_G(\sigma^{-1}F)=\sigma^{-1}\,\mathrm{Stab}_G(F)\sigma=\sigma^{-1}H\sigma$. Since the choice of τ was arbitrary, this suffices to show that $\mathrm{Gal}(P)\subseteq\sigma^{-1}H\sigma$, i.e. $\mathrm{Gal}(P)$ is a subgroup of a conjugate of H. □

To apply this result in practice, we compute numerical approximations to the roots of $P(X)$ and then use these to determine $F(\alpha_{\sigma(1)},\ldots,\alpha_{\sigma(n)})$ numerically. Naturally, these approximations need to be accurate enough to guarantee that we can correctly recognize when $\mathrm{Res}_G(F,P)$ has an integer root and recognize that this root is in fact simple. In practice this is not usually very problematic (at least for small degree polynomials with small coefficients). So from this point onward we will suppress the details of the numerical approximations and assume that all numerical computations are carried out to sufficient accuracy to ensure that no roundoff-related difficulties occur.

The following result is not difficult to prove directly, but in preparation for the next section, and to demonstrate the applicability of Proposition 3.2, we will use resolvent polynomials.

Proposition 3.3. *Given an irreducible polynomial $P(X)\in\mathbb{Z}[X]$ of degree n, $\mathrm{Gal}(P)$ is a subgroup of $A_n\subseteq S_n$ if and only if $\mathrm{disc}(P)$ is a square, i.e. $\sqrt{\mathrm{disc}(P)}\in\mathbb{Z}$.*

Proof. Let $F(X_1, \dots, X_n) = \ell(P)^{n-1} \prod_{i \neq j} (X_i - X_j)$ and let $G = S_n$. It is not hard to see $H = \mathrm{Stab}_G(F) = A_n$, and so

$$\mathrm{Res}_G(F,P)(X) = \left(X - \ell(P)^{n-1} \prod_{i \neq j} (\alpha_i - \alpha_j) \right) \left(X + \ell(P)^{n-1} \prod_{i \neq j} (\alpha_i - \alpha_j) \right) = X^2 - \mathrm{disc}(P)$$

By assumption, $P(X)$ is irreducible, and hence separable, so $\mathrm{disc}(P) \neq 0$. Therefore $X^2 - \mathrm{disc}(P)$ has a simple root in $\mathbb{Z}$ if and only if $\sqrt{\mathrm{disc}(P)} \in \mathbb{Z}$, and in particular, by Proposition 3.2, $\mathrm{Gal}(P)$ is a subgroup of a conjugate of $A_n \subseteq S_n$ if and only if $\sqrt{\mathrm{disc}(P)} \in \mathbb{Z}$. Since A_n is the only subgroup of index 2 in S_n, this proves the proposition. □

4 Determining Gal(P)

In the previous section we had our first glance (Proposition 3.2) at how judiciously chosen resolvent polynomials can be used to limit the possibilities for $\mathrm{Gal}(P)$. In this section, we will consider some more general algorithms that will determine $\mathrm{Gal}(P)$ for irreducible monic polynomials $P(X) \in \mathbb{Z}[X]$ with $\deg(P) \leq 5$. Note that any irreducible polynomial in $\mathbb{Z}[X]$ can be transformed into a monic irreducible polynomial with the same splitting field by computing

$$\ell(P)^{\deg(P)-1} P(X/\ell(P)) = R_Y(P(Y), X - \ell(P))/\ell(P)$$

which is simply a Tschirnhausen transformation (so it has the same splitting field as $P(X)$) and it is easy to check that its coefficients are in $\mathbb{Z}$. Since there are only a finite number of possible Galois groups $\mathrm{Gal}(P)$, namely the transitive subgroups of $S_{\deg(P)}$, we will actually give a separate algorithm for each degree, which is tailored to distinguish between the possible Galois groups in each case. The subsequent algorithms are adapted from those suggested in [Coh93].

For $\deg(P) = 1$ and $\deg(P) = 2$, there is only one transitive subgroup of S_n, namely $S_1 = C_1$ and $S_2 = C_2$ respectively. The first non-trivial case is when $\deg(P) = 3$.

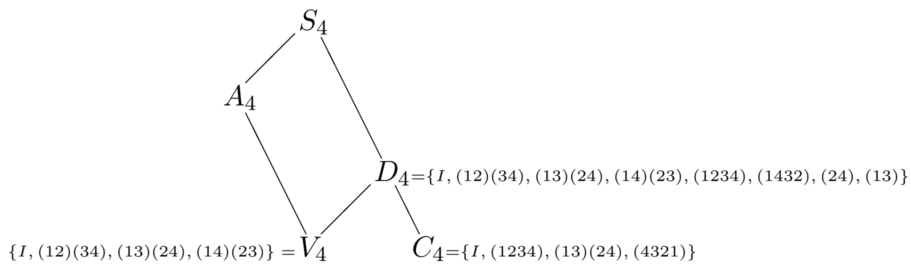
deg(P) = 3

Since the only transitive subgroups of S_3 are S_3 and A_3, we only need Proposition 3.3 to determine $\mathrm{Gal}(P)$, i.e.

Algorithm 4.1 (Degree 3 Galois group).
3*GALOIS(P)*
 IF $\sqrt{\mathrm{disc}(P)} \in \mathbb{Z}$ THEN
  RETURN A_3
 ELSE
  RETURN S_3
END 3*GALOIS*

deg(P) = 4

It is an exercise in group theory to show that the transitive subgroups of S_4 are all isomorphic to either C_4, V_4, D_4, A_4 or S_4, and that any two isomorphic transitive subgroups are conjugate. Given this, we can use the following inclusion lattice and Proposition 3.2 (together with Proposition 3.3) to determine $\mathrm{Gal}(P)$:



The following algorithm determines the Galois group of a monic irreducible quartic polynomial $P \in \mathbb{Z}[X]$:

Algorithm 4.2 (Degree 4 Galois group).
4*GALOIS(P)*
S4:

$F_1(X_1, X_2, X_3, X_4) = X_1X_3 + X_2X_4$;
\\\\ *Is* $\mathrm{Gal}(P)$ *contained in a conjugate of* D_4*? (Using* $S_4/\operatorname{Stab}_{S_4}(F_1) = \{I, (12), (14)\}$*)*
IF $\operatorname{Res}_{S_4}(F_1, P)$ *has no roots in* $\mathbb{Z}$ THEN
IF $\sqrt{\operatorname{disc}(P)} \in \mathbb{Z}$ THEN
RETURN A_4;
ELSE
RETURN S_4;
END IF
ELSE IF $\operatorname{Res}_{S_4}(F_1, P)$ *has a simple root in* $\mathbb{Z}$ THEN
GOTO *D4*;
ELSE
P = TSCHIRNHAUSEN(P);
GOTO *S4*;
END IF

D4:

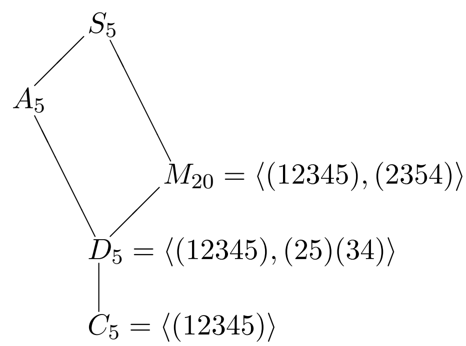
\\\\ *Now* $\operatorname{Res}_{S_4}(F_1, P)$ *has a simple root in* $\mathbb{Z}$*, so* $\mathrm{Gal}(P)$ *is contained in a conjugate of* D_4.
IF $\sqrt{\operatorname{disc}(P)} \in \mathbb{Z}$ THEN RETURN V_4;
σ = *the element of* $\{I, (12), (14)\}$ *corresponding to the simple root of* $\operatorname{Res}_{S_n}(F_1, P)$;
\\\\ *Renumber the roots of* P *so that* $\alpha_1\alpha_3 + \alpha_2\alpha_4 \in \mathbb{Z}$
Set $\alpha_i = \alpha_{\sigma(i)}$;
$F_2(X_1, X_2, X_3, X_4) = X_1X_2^2 + X_2X_3^2 + X_3X_4^2 + X_4X_1^2$;
\\\\ *Is* $\mathrm{Gal}(P)$ *contained in a conjugate of* C_4*? (Using* $D_4/\operatorname{Stab}_{D_4}(F_2) = \{I, (13)\}$*)*
IF $\operatorname{Res}_{D_4}(F_2, P)$ *has no root in* $\mathbb{Z}$ THEN
RETURN D_4;
ELSE IF $\operatorname{Res}_{D_4}(F_2, P)$ *has a simple root in* $\mathbb{Z}$ THEN
RETURN C_4;
ELSE
P = TSCHIRNHAUSEN(P);
GOTO *D4*;
END IF

END 4*GALOIS*

At the point in the algorithm where we encounter the line that reads "Set $\alpha_i = \alpha_{\sigma(i)}$;", we know that $\mathrm{Gal}(P)$ is a subgroup of a conjugate of D_4. However, we need to ensure that we are working inside the appropriate conjugate of D_4. That is, in order to apply Proposition 3.2 with $G = D_4$, we need the explicit action of $\mathrm{Gal}(P)$ on the roots of the resolvent. By reordering the roots, we ensure that we have the correct conjugate of D_4 for the choice of resolvent F_2 and corresponding coset representatives. Bearing this in mind, and noting that $\mathrm{Stab}(F_1) = D_4$ and $\mathrm{Stab}(F_2) = C_4$, the correctness of this algorithm follows from Proposition 3.2.

deg(P) = 5

Again, it is an exercise in group theory to show that the transitive subgroups of S_5 are isomorphic to $S_5, A_5, M_{20}, D_5, C_5$ (where M_{20} is the "meta-cyclic" group $\langle(12345),(2354)\rangle$, and that isomorphic transitive groups are conjugate.



The following algorithm determines the Galois group of a monic irreducible quintic polynomial $P \in \mathbb{Z}[X]$:

Algorithm 4.3 (Degree 5 Galois group).
5*GALOIS(P)*
S5:

$F_1(X_1,X_2,X_3,X_4,X_5) = X_1^2(X_2X_5+X_3X_4)+X_2^2(X_1X_3+X_4X_5)+X_3^2(X_1X_5+X_2X_4)+X_4^2(X_1X_2+X_3X_5)+X_5^2(X_1X_4+X_2X_3)$;
\\ *Is* $\mathrm{Gal}(P)$ *contained in a conjugate of* M_{20}*? (Using* $S_5/\mathrm{Stab}_{S_5}(F_1) = \{I,(12),(13),(14),(15),(25)\}$*)*
IF $\mathrm{Res}_{S_5}(F_1,P)$ *has no roots in* $\mathbb{Z}$ THEN
    IF $\sqrt{\mathrm{disc}(P)} \in \mathbb{Z}$ THEN
        RETURN A_5;
    ELSE
        RETURN S_5;
    END IF
ELSE IF $\mathrm{Res}_{S_5}(F_1,P)$ *has a simple root in* $\mathbb{Z}$ THEN
    GOTO *M20*;
ELSE
    $P =$ TSCHIRNHAUSEN(P);
    GOTO *S5*;
END IF

M20:

\\ *Now* $\mathrm{Res}_{S_5}(F_1,P)$ *has a simple root in* $\mathbb{Z}$*, so* $\mathrm{Gal}(P)$ *is contained in a conjugate of* M_{20}*.*
IF $\sqrt{\mathrm{disc}(P)} \notin \mathbb{Z}$ THEN RETURN M_{20};

$\sigma =$ *the element of* $\{I, (12), (13), (14), (15), (25)\}$ *corresponding to the simple root of* $\mathrm{Res}_{S_5}(F_1, P)$;
\\ *Renumber the roots of* P *so that* $F_1(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) \in \mathbb{Z}$
Set $\alpha_i = \alpha_{\sigma(i)}$;
$F_2(X_1, X_2, X_3, X_4, X_5) = X_1 X_2^2 + X_2 X_3^2 + X_3 X_4^2 + X_4 X_5^2 + X_5 X_1^2$;
\\ *Is* $\mathrm{Gal}(P)$ *contained in a conjugate of* C_5*? (Using* $D_5 / \mathrm{Stab}_{D_5}(F_2) = \{I, (12)(35)\}$*)*
IF $\mathrm{Res}_{D_5}(F_2, P)$ *has no root in* $\mathbb{Z}$ THEN
    RETURN D_5;
ELSE IF $\mathrm{Res}_{D_5}(F_2, P)$ *has a simple root in* $\mathbb{Z}$ THEN
    RETURN C_5;
ELSE
    $P =$ TSCHIRNHAUSEN(P);
    GOTO *M20*;
END IF
END 5*GALOIS*

Just as with the algorithm for polynomials of degree 4, the correctness of this algorithm follows from Proposition 3.2.

These algorithms demonstrate how we can use resolvent polynomials to navigate through the lattice of possible Galois groups and ultimately determine $\mathrm{Gal}(P)$. Unfortunately, the complexity of such an algorithm grows with the complexity of the inclusion lattice of possible Galois groups. Algorithms for degree 6 and degree 7 polynomials are given in [Coh93], and publicly available packages such a GP/PARI ([BBB$^+$00]) are capable of determining Galois groups of irreducible polynomials up to degree 11.

In [SM85], the authors discuss a similar technique for determining Galois groups over $\mathbb{Q}$ using linear resolvent polynomials. However, the "resolvent method" is by no means the only way to compute the Galois groups of rational polynomials. Various techniques for computing Galois groups, including the method presented here, are surveyed in [Hul].

5 Conclusion

We have seen how resolvent polynomials and resultant polynomials are not only theoretically useful tools, but also how they lend themselves to efficient computation. This is fortunate since we can use resultants to perform exact computations on (irrational) algebraic numbers, or to find discriminants. Such computations proved essential in section 4 where we considered the problem of determining $\mathrm{Gal}(P)$ algorithmically, and are basic to the study of computational algebraic number theory. As far as further reading is concerned, [Coh93] builds upon the machinery developed here, and Knuth's *The Art of Computer Programming*(particularly volumes 1 and 2, [Knu73], [Knu81]) is an excellent reference for many of the underlying algorithms which were not covered in detail here.

References

[BBB+00] C. Batut, K. Belabas, D. Benardi, H. Cohen, and M. Olivier. *User's Guide to PARI-GP*. by anonymous ftp from `ftp://megrez.math.u-bordeaux.fr/pub/pari`, 2000. see `http://pari.home.ml.org`.

[Coh93] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[Hul] Alexander Hulpke. Techniques for the computation of galois groups. Available for download at `http://citeseer.nj.nec.com/465712.html`.

[Knu73] D. E. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Second Edition*. Addison-Wesley, 1973.

[Knu81] D. E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, Second Edition*. Addison-Wesley, 1981.

[SM85] L. Soicher and J. McKay. Computing galois groups over the rationals. *J. Number Theory*, 20:273–281, 1985.