MATHEMATICS OF COMPUTATION
VOLUME 63, NUMBER 208
OCTOBER 1994, PAGES 749–757

SYMMETRIC FUNCTIONS, m-SETS, AND GALOIS GROUPS

DAVID CASPERSON AND JOHN MCKAY

ABSTRACT. Given the elementary symmetric functions in $\{r_i\}$ $(i = 1, \ldots, n)$, we describe algorithms to compute the elementary symmetric functions in the products $\{r_{i_1} r_{i_2} \cdots r_{i_m}\}$ $(1 \le i_1 < \cdots < i_m \le n)$ and in the sums $\{r_{i_1} + r_{i_2} + \cdots + r_{i_m}\}$ $(1 \le i_1 < \cdots < i_m \le n)$. The computation is performed over the coefficient ring generated by the elementary symmetric functions. We apply FFT multiplication of series to reduce the complexity of the algorithm for sums. An application to computing Galois groups is given.

1. INTRODUCTION

Let

$$\mathscr{E}_k(r_1, r_2, \ldots, r_n) = \sum_{1 \le i_1 < \cdots < i_k \le n} r_{i_1} r_{i_2} \cdots r_{i_k} \tag{1}$$

denote the elementary symmetric function of degree k in $\{r_i\}$, $i = 1, \ldots, n$.

We compute the power sum functions $\mathscr{P}_i(r_1, r_2, \ldots) = \sum_j r_j^i$ from the elementary symmetric functions, and vice versa, using Newton's method consisting of iteratively finding $\mathscr{P}_1(f), \mathscr{P}_2(f), \ldots$ from the identity

$$\mathscr{P}_k + k(-1)^k \mathscr{E}_k = -\sum_{i=1}^{k-1} (-1)^i \mathscr{E}_i \mathscr{P}_{k-i}, \tag{2}$$

and in the reverse direction $\mathscr{E}_1, \mathscr{E}_2, \ldots$ are derived from

$$\mathscr{E}_k = \frac{1}{k} \sum_{i=1}^{k} (-1)^{i+1} \mathscr{P}_i \mathscr{E}_{k-i}. \tag{3}$$

We find the elementary symmetric functions of the sums $\{r_{i_1} + r_{i_2} + \cdots + r_{i_m}\}$ $(1 \le i_1 < \cdots < i_m \le n)$ and the products $\{r_{i_1} r_{i_2} \cdots r_{i_m}\}$ $(1 \le i_1 < \cdots < i_m \le n)$, given the elementary symmetric functions of the $\{r_i\}$ $(i = 1, \ldots, n)$. Equivalently, let $f(x) \in K[x]$ be a monic polynomial of degree n with roots $r_1, \ldots, r_n \in L$, where K is a field of characteristic 0, and L an extension of K. We are given the coefficients of $f(x)$, and we wish to calculate the coefficients of $f^{+m}(x)$, the monic polynomial in $K[x]$ whose roots are $r_{i_1} + r_{i_2} + \cdots + r_{i_m}$ $(1 \le i_1 < i_2 < \cdots < i_m \le n)$, and of $f^{\times m}(x)$, whose roots are $r_{i_1} r_{i_2} \cdots r_{i_m}$ $(1 \le i_1 < i_2 < \cdots < i_m \le n)$. Furthermore, our calculations are *exact*: the

Received by the editor October 7, 1992.
1991 *Mathematics Subject Classification.* Primary 05E05; Secondary 12F10, 12Y05.
Research supported by NSERC and FCAR grants.

0025-5718/94 $1.00 + $.25 per page

arithmetic is performed entirely in K and does not involve estimating the roots of $f(x)$ (compare [20, 21]).

Newton's method may be applied to give algorithms for computing $f^{+m}(x)$ and $f^{\times m}(x)$. A method for computing $f^{+m}(x)$ has appeared elsewhere [2], but we believe our method for computing $f^{+m}(x)$ is new.

2. The algorithms

We adopt the following notation and conventions throughout: the monic polynomial $f(x) = \sum_{i=0}^{n} a_i x^{n-i}$ of degree n has roots $\{r_i\}$ $(i = 1, \ldots, n)$ denoted as a set by $\mathbf{R}(f)$. We let $\mathbf{R}_m(f)$ denote $\{A \subseteq \mathbf{R}(f)\colon \operatorname{card} A = m\}$, $\mathscr{E}_k(f) = \mathscr{E}_k(\mathbf{R}(f)) = (-1)^k a_k$, and $\mathscr{P}_k(f) = \mathscr{P}_k(\mathbf{R}(f))$. We adopt the conventions that $\sum \varnothing = 0$, $\prod \varnothing = 1$, and that $0^0 = 1$: thus for any nonconstant polynomial $g(x)$, we have $g^{+0}(x) = x$ and $g^{\times 0}(x) = x - 1$, so that $\mathscr{P}_k(g^{+0}) = \delta_{k,0}$ and $\mathscr{P}_k(g^{\times 0}) = 1$. We define $\|f(x)\|_\infty$ to be $\max_i |a_i|$.

We obtain relations between $\mathscr{P}_k(f)$ and either $\mathscr{P}_k(f^{+m})$ or $\mathscr{P}_k(f^{\times m})$. Computing $\mathscr{P}_k(f)$ using (2), and $\mathscr{E}_k(f^{+m})$ using (3) provides fast algorithms for computing $f^{+m}(x)$ or $f^{\times m}(x)$. To derive relations between $\mathscr{P}_l(f)$ and either $\mathscr{P}_m(f^{+k})$ or $\mathscr{P}_m(f^{\times k})$, we apply Newton's relations to a polynomial $F(X) = \prod_{r \in \mathbf{R}(f)}(X - q(r))$ for some suitably chosen function q. The polynomial $F(X)$ serves as a convenient notational device.

2.1. **Computing symmetric functions of sums of roots.** For $a \in K$ (a field of characteristic 0) define $\exp(as) = \sum_{h=0}^{\infty}(a^h/h!)s^h \in K[[s]]$ as a formal power series; and for any monic polynomial $g(x) \in K[x]$ define the formal power series $H_g(s) \in K[[s]]$ by $H_g(s) = \sum_{h=0}^{\infty} s^h \mathscr{P}_h(g)/h!$. Note that $H_g(s) = \sum_{r \in \mathbf{R}(g)} \exp(rs)$.

For a given monic $f(x) \in K[x]$, let $F(X) = \prod_{r \in \mathbf{R}(f)}(X - \exp(rs)) \in K[[s]](X)$. We have

$$\mathscr{P}_h(F) = \sum_{r \in \mathbf{R}(f)} (\exp(rs))^h \tag{4}$$

$$= \sum_{r \in \mathbf{R}(f)} (\exp(rhs)) = H_f(hs), \tag{5}$$

and

$$\mathscr{E}_m(F) = \sum_{R \in \mathbf{R}_m(f)} \left(\prod_{r \in R} \exp(rs) \right) \tag{6}$$

$$= \sum_{R \in \mathbf{R}_m(f)} \exp\left(s \sum_{r \in R} r \right) \tag{7}$$

$$= \sum_{\rho \in \mathbf{R}(f^{+m})} \exp(s\rho) = H_{f^{+m}}(s), \tag{8}$$

so, from (3), we have

$$H_{f^{+m}}(s) = \frac{1}{m} \sum_{h=1}^{m} (-1)^{h+1} H_f(hs) H_{f^{+(m-h)}}(s) \quad \text{for } m \geq 1. \tag{9}$$

Noting that the coefficient of s^k in $H_f(hs)$ is h^k times the corresponding coefficient of $H_f(s)$, we can match coefficients of powers of s in (9) to obtain, for $m \geq 1$, $k \geq 0$,

$$\mathscr{P}_k(f^{+m}) = \frac{1}{m}\sum_{h=1}^{m}(-1)^{h+1}\sum_{j=0}^{k}\binom{k}{j}h^j\mathscr{P}_j(f)\mathscr{P}_{k-j}(f^{+(m+h)}). \tag{10}$$

This equation can be used directly to give an efficient algorithm to compute $f^{+m}(x)$ from $f(x)$ when m and n are small, but for larger values, equation (9) gives the more efficient

Algorithm S: Computation of $f^{+m}(x)$.

Input: The coefficients of $f(x) = x^n + \sum_{i=1}^{n} a_i x^{n-i}$ and an integer m, $1 \leq m \leq n$.

Output: the coefficients of the polynomial $f^{+m}(x)$.

S–1 Let N denote $\binom{n}{m} = \deg f^{+m}$. Compute $\mathscr{P}_k(f)$ for $k = 0, \ldots, N$ using (2), giving

$$H_f(s) \pmod{s^{N+1}}.$$

S–2 For $i = 2, \ldots, m$ compute $H_{f+i}(s) \pmod{s^{N+1}}$ using (9).

Note: $H_{f+0}(s) = 1$ and $H_{f+1}(s) = H_f(s)$.

S–3 From

$$H_{f+m}(s) \equiv \sum_{k=0}^{N}\mathscr{P}_k(f^{+m})s^k/k! \pmod{s^{N+1}} \tag{11}$$

find the values of $\mathscr{P}_k(f^{+m})$ for $k = 0, \ldots, N$; and then compute $f^{+m}(x)$ using (3).

End.

2.2. **Computing symmetric functions of products of roots.** Here we take $F(X) \in K[X]$ to be the polynomial $F(X) = \prod_{r\in\mathbf{R}(f)}(X - r^k)$, for some *fixed* integer $k \geq 0$. (Recall that $0^0 = 1$.) We have

$$\mathscr{P}_h(F) = \sum_{r\in\mathbf{R}(f)}(r^k)^h = \mathscr{P}_{kh}(f) \tag{12}$$

and

$$\mathscr{E}_m(F) = \sum_{R\in\mathbf{R}_m(f)}\left(\prod_{r\in R}r^k\right) = \sum_{R\in\mathbf{R}_m(f)}\left(\prod_{r\in R}r\right)^k \tag{13}$$

$$= \sum_{\rho\in\mathbf{R}(f^{\times m})}\rho^k = \mathscr{P}_k(f^{\times m}), \tag{14}$$

so, using (3), we get

$$\mathscr{P}_k(f^{\times m}) = \frac{1}{m}\sum_{h=1}^{m}(-1)^{h+1}\mathscr{P}_{kh}(f)\mathscr{P}_k(f^{\times(m-h)}) \quad \text{for } m \geq 1,\ k \geq 0. \tag{15}$$

This gives the following algorithm for computing $f^{\times m}(x)$ from $f(x)$.

Algorithm P: Computation of $f^{\times m}(x)$.

Input: The coefficients of $f(x) = x^n + \sum_{i=1}^{n} a_i x^{n-i}$ and an integer m, $1 \le m \le n$.

Output: the coefficients of the polynomial $f^{\times m}(x)$.

P–1 Compute $\mathscr{P}_k(f)$ for $k = 0, \ldots, m\binom{n}{m}$ using (2).

P–2 For $i = 2, \ldots, m$ compute

$$\{\mathscr{P}_k(f^{\times i}) | k = 0, \ldots, \binom{n}{m}\}$$

using (15).

Note: We have $f^{\times 0}(x) = x - 1$ (i.e., $\mathscr{P}_k(f^{\times 0}) = 1$ for all k), and $f^{\times 1}(x) = f(x)$.

P–3 Compute the coefficients of $f^{\times m}(x)$ from $\{\mathscr{P}_k(f^{\times m})\}$ $(k = 0, \ldots, \binom{n}{m})$ using (3)

End.

3. Implementation of algorithms

We comment on the implementation of Algorithms **S** and **P**.

Exact results require an unlimited-precision integer arithmetic package.[1] For our applications we found that care was required in implementing the arithmetic for two reasons: first, low-level implementation issues such as the allocation and reclamation of storage were delicate because intermediate quantities in these calculations were much larger than indicated by the max norm of the polynomial computed; second, the coefficients in our applications were several thousand digits long, making fast arithmetic important.

3.1. **Implementation of Algorithm S.** Step 2 of Algorithm **S** requires $\binom{m}{2}$ series multiplications, each series having $\binom{n}{m}$ terms. The use of conventional technique for multiplying series would require $O(\binom{m}{2}\binom{n}{m}^2)$ rational arithmetic operations. When $\binom{n}{m}$ is large, it pays to use Fast Fourier Transforms (FFT) to perform the series multiplications to reduce this to $O(\binom{m}{2}\binom{n}{m}\log\binom{n}{m})$ arithmetic operations.

The FFT algorithm (see [12]) requires at least $\binom{n}{m}$ roots of unity in the coefficient ring. To meet this requirement, yet still have exact arithmetic, we compute the canonical homomorphic image of $f^{+m}(x)$ in the finite field $\mathbb{F}_p$, where p is a prime chosen so that the FFT algorithm works. After computing the image of $f^{+m}(x)$ in $\mathbb{F}_p[x]$ for sufficiently many primes p, we recover $f^{+m}(x)$ using the Chinese Remainder Algorithm.

We choose the primes p_i as follows: let n_0 be an integer such that $2^{n_0} > \binom{n}{m}$. (Note that we may take $n_0 \le n$, as $\binom{n}{m} \le \binom{n}{\lfloor n/2 \rfloor} \le 2^n$.) Let M be an integer provably greater than $\|f^{+m}(x)\|_\infty$. Then pick a sequence of primes $p_1, p_2, \ldots, p_k$ such that

[1] Such packages are provided by PARI [1], ALGEB [9], as well as many other commercial symbolic computation packages.

$$\prod_{i=1}^{k} p_i \geq 2M, \tag{16}$$

$$p_i > \binom{n}{m} \quad \text{for } i = 1, \ldots, k, \text{ and} \tag{17}$$

$$2^{n_0} \equiv 1 \pmod{p_i} \quad \text{for } i = 1, \ldots, k. \tag{18}$$

Condition (16) allows us to use the Chinese Remainder Algorithm to recover $f^{+m}(x)$ from its modular images (see [6]). We require condition (17) so that the homomorphic image of $H_{f^{+m}}(s)$ is well defined modulo s^{N+1}, and so that the $\mathscr{E}_k(f^{+m})$ is recoverable from $\mathscr{P}_k(f^{+m})$ using (3). Condition (18) allows us to use FFT techniques to perform the series multiplications.

Combining this, gives Algorithm **SI**.

Algorithm SI: Implementation of Algorithm **S**.

Input: The coefficients of $f(x) = x^n + \sum_{i=1}^{n} a_i x^{n-i} \in \mathbb{Z}[x]$ and an integer m, $1 \leq m \leq n$.

Output: the coefficients of the polynomial $f^{+m}(x) \in \mathbb{Z}[x]$.

SI–1 Let N denote $\binom{n}{m} = \deg f^{+m}$. Choose n_0 to be an integer such that $2^{n_0} \geq N$. Find an M provably greater than $\|f^{+m}\|_\infty$.

SI–2 Compute $\mathscr{P}_k(f)$ for $k = 0, \ldots, N$ using (2), giving

$$H_f(s) \pmod{s^{N+1}}.$$

SI–3 Find a sequence of word-size primes $p_1, p_2, \ldots, p_k$ such that

1. $\prod_{i=1}^{k} p_i \geq 2M$,
2. $p_i > \binom{n}{m}$ for $i = 1, \ldots, k$, and
3. $2^{n_0} \equiv 1 \pmod{p_i}$ for $i = 1, \ldots, k$.

Let $^- : \mathbb{Z}[x] \to \mathbb{F}_{p_i}[x]$ denote the canonical homomorphism. For $i = 1, \ldots, k$ perform the following two steps in $\mathbb{F}_{p_i}$:

SI–3(a) Compute $\overline{H_{f^{+j}}(s)} \pmod{s^{N+1}}$ using (9) for $j = 2, \ldots, m$. Use standard FFT techniques to perform the series multiplications in $O(N \log N)$ arithmetic operations over $\mathbb{F}_{p_i}$.

SI–3(b) From

$$\overline{H_{f^{+m}}(s)} = \sum_{l=0}^{N} \overline{\mathscr{P}_l(f^{+m})/l!} \cdot s^l \pmod{s^{N+1}} \tag{19}$$

read off the values of $\overline{\mathscr{P}_l(f^{+m})}$ for $l = 0, \ldots, N$, and then compute $\overline{f^{+m}(x)}$ using (3).

SI–4 Compute $f^{+m}(x)$ from its images modulo p_i, $i = 1, \ldots, k$, using the Chinese Remainder Algorithm.

End.

In addition to increasing the speed of series multiplication, doing the computation in $\mathbb{F}_p$ has another advantage: the intermediate quantities in computations using (9) are bounded by p, which in practice we choose to be less than the computer wordsize. A disadvantage is the need to find a provable upper bound $M \geq \|f^{+m}\|_\infty$ before computing f^{+m}.

Note that the coefficients of $H_f(s)$ are rational, even when $f(x) \in \mathbb{Z}[x]$, so a direct implementation of Algorithm **S** for monic integer polynomials may be speeded up by storing the coefficients of $\widehat{H}_{f+i}(s) = \sum_{l\geq 0} \mathscr{P}_l(f^{+i})s^l$ rather than those of $H_{f+i}(s)$, and making modifications so that the algorithm uses only integer (rather than rational) arithmetic.

3.2. **Implementation of Algorithm P.** The equation (15) is a relation between integers, so Algorithm **P** can be implemented directly using a multi-precision integer package. Our greatest difficulty in using this algorithm was the very large size of the coefficients generated. When computing a degree-924 polynomial with coefficients of about 5000 digits (see §5), we found it simplest to create m temporary files, with the ith file containing the numbers $\mathscr{P}_k(f^{\times i})$ $(k = 0, \ldots, \binom{n}{m})$. This gave us tight control over the internal storage required in step 2 of Algorithm **P** at the expense of a few sequential file operations.

4. Applications to computational Galois theory

Let $f(x) = \mathbb{Z}[x]$ be monic and irreducible. A theoretical method for computing $G = \mathrm{Gal}_{\mathbb{Q}}(f)$, the Galois group of $f(x)$ over the rationals, has long been known [23], but, as it involves factoring a polynomial of total degree $n!$ in $n+1$ variables, it is impracticable in cases of interest.

Practical techniques are much more recent. See [20, 21, 13, 19, 18]. A package for computing $\mathrm{Gal}_{\mathbb{Q}}(f)$ for $\deg f \leq 7$ is available in Maple [5],[2] and the transitive permutation groups of degree ≤ 15 have been catalogued [3, 4, 16, 17], but much remains to be done in developing feasible computational techniques for finding the Galois groups of polynomials of such a degree.

One invariant for distinguishing among potential candidates for the Galois group, G, of a given polynomial is the orbit structure of m-sets of roots under the Galois action. The lengths of the orbits of $\mathbf{R}_m(f)$ under the action of G form a partition of $\binom{n}{m}$, which is a G-invariant and may be used to distinguish between potential Galois groups.

Let $F(X)$ be $f^{+m}(X)$ (or $f^{\times m}(X)$). We have $F(X) \in \mathbb{Z}[X]$, and if $F(X)$ is squarefree, we can write $F = F_1F_2\cdots F_k$, where the F_i are distinct monic irreducible polynomials. Furthermore, these F_i correspond to orbits of $\mathbf{R}_m(f)$ under G. That is, we can partition $\mathbf{R}_m(f)$ as $\bigcup_{i=1}^{k} \mathscr{R}_i$ so that $F_i = \prod_{R\in\mathscr{R}_i}(x - \sum R)$ (or $\prod_{R\in\mathscr{R}_i}(x - \prod R)$). Assuming that $F(X)$ is squarefree is not a major restriction; for, if needed, we can apply a Tschirnhaus transformation to f in order to make this so. Thus, the degrees of the irreducible factors of $f^{+m}(x)$ (or $f^{\times m}(x)$) are Galois invariants.

In §5 we see that it is generically faster to compute $f^{+m}(x)$ than $f^{\times m}(x)$. The following shows that computing $f^{\times m}(x)$ may nevertheless uncover interesting information (see [8, 11, 14]).

Let $f(z, t, u) \in \mathbb{Q}(t, u)[z]$ be

$$f(z, t, u) = (z-1)g_3(z)h_3(z) - tz^3(z+1), \tag{20}$$

where

$$g_3(z) = z^3 - (2u+2)z^2 - (4u+2)z - 2u, \tag{21}$$

[2]Now extended to $\mathrm{Gal}_K(f)$, $K = \mathbb{Q}(t_1, t_2, \ldots, t_k)$.

with roots $\{z_g\}$, and

$$h_3(z) = z^3 - (u-1)z^2 - uz - u^2/2, \tag{22}$$

with roots $\{z_h\}$. We find that

$$u^2 z^3 g_3(1/z) + 2h_3(u(z+1)) = 0, \tag{23}$$

so that $1/z_g = z_h/u - 1$. Because $\mathrm{Gal}_{\mathbb{Q}(t,u)} f = PGL(3,2)$, we know that $f^{\times 3}(z, t, u)$ has a unique irreducible degree-seven factor: $u^7 f(z/u-1, -t, u)$. For the same reason, $f^{+3}(z, t, u)$ also has an irreducible factor of degree seven, but in this case it is

$$\begin{aligned} &zt^2 - (2z + uz - u^2)(3z^2 - 6z + 1 - 6uz - u)t \\ &\quad - (z - 2u - 2)(2z^3 - 8uz^2 + (10u^2 - 4u - 6)z - 4u^3 + 3u^2) \\ &\quad (2z^3 - (6u+8)z^2 + (2 - 8u)z - 3u^2 + 2u)/4, \end{aligned} \tag{24}$$

which appears to have no such nice interpretation. Note that the computations with (20) are more efficient after putting $u = -2a$, so that the coefficients are integers.

5. Performance

The examples in this section arose from computer verification that the Mathieu groups M_{11} and M_{12} are Galois groups over $\mathbb{Q}$. (Darmon and Ford's verification of Matzat's result may be found in [7]. They used p-adic approximation to prove directly that a polynomial invariant (derived from the Steiner system and evaluated on the roots) was integer-valued, rather than use symmetric function methods to obtain their result. In this instance it appears that this p-adic technique is faster.)

The algorithms discussed in §2 and §3 were implemented by one of the authors (Casperson) in Pari. These algorithms were tested on the polynomials shown in Tables 1 and 2. Similar polynomials, first given in [15] and also used

Table 1. Coefficients of $f_{11}(x)$
The polynomial $f_{11}(x) = x^{11} + \sum_{i=1}^{11} a_i x^{11-i}$

i	a_i	i	a_i	i	a_i
		4	+85520	8	+105904640
1	+2	5	+15392	9	+252830720
2	−484	6	−6191296	10	+27555840
3	−520	7	+3032192	11	+1753436160

Table 2. Coefficients of $f_{12}(x)$
The polynomial $f_{12}(x) = x^{12} + \sum_{i=1}^{12} a_i x^{12-i}$

i	a_i	i	a_i	i	a_i
1	+4	5	−20856	9	−5349260
2	−526	6	−9429444	10	−1475917191
3	−940	7	+14732616	11	−44569205004
4	+106095	8	+282523695	12	+137613183361

TABLE 3. Degrees, norms, and computation time (CPU-seconds on a MIPS M-120/5) for the polynomials calculated

Polynomial F	$\deg F$	$\|F\|_\infty$	time
$f_{11}(x)^{+5}$	462	1.98×10^{487}	3885 s
$f_{11}(x)^{\times 5}$	462	1.65×10^{1941}	5855 s
$f_{12}(x)^{+6}$	924	7.83×10^{997}	31185 s
$f_{12}(x)^{\times 6}$	924	1.15×10^{5146}	80753 s

in [7], exhibit the Mathieu groups M_{11} and M_{12} as Galois groups over $\mathbb{Q}$. Table 3 summarizes the results of these calculations. Note the ratio of the norms of f_{11}^{+5} and $f_{11}^{\times 5}$, and of f_{12}^{+6} and $f_{12}^{\times 6}$. In terms of the polynomials given here, those given in [15, 7] are $f_{11}(x+9)$ and $f_{12}(x+8)$. We have shifted the roots to reduce the norms.

5.1. **Choosing between Algorithm S and Algorithm P.** In applications where $f^{+m}(x)$ and $f^{\times m}(x)$ provide essentially the same information, such as in §4, one is free to choose between Algorithm **S** and Algorithm **P**. The latter has the practical advantage of taking less effort to implement and seems preferable when the coefficient size is small throughout the computation. On the other hand, in our experience Algorithm **S** is faster than Algorithm **P** when one has $\|f^{+m}\|_\infty \ll \|f^{\times m}\|_\infty$. We do not have a rigorous argument to prove that this is so, but we note that the roots of f^{+m} are homogeneous of weight 1 in the roots of $f(x)$, whereas the roots of $f^{\times m}$ are homogeneous of weight m. Finally, when $\|f^{+m}\|_\infty \ll \|f^{\times m}\|_\infty$, the cost of extracting further information from either $f^{+m}(x)$ or $f^{\times m}(x)$ may lead one to favor Algorithm **S**. The ratio of times for computing $f^{+3}(z, t, u)$: $f^{\times 3}(z, t, u)$ in (20) is about 10:3.

5.2. **Comparison with other techniques.** Unlike Stauduhar's method [20, 21], the symmetric function methods are largely independent of the coefficient ring, and so apply directly to, say, coefficients in $\mathbb{Q}(t_1, t_2, \ldots, t_k)$.

Algorithm **SI** is the fastest such algorithm for computing $f^{+m}(x)$ known to us, and it seems to be significantly faster than the symmetric function algorithm of [10] and [22].

For f in $\mathbb{Q}[x]$, it is possible to use approximations over $\mathbb{C}$ or $\mathbb{Q}_p$ to the roots of f to build $f^{+m}(x)$ or $f^{\times m}(x)$, but a proof is needed that the coefficients of these are the rounded values of the coefficients of the approximate computed polynomials. Rigorous error bounds are, as usual, easier to find using a non-Archimedean valuation. Over $\mathbb{Q}_p$, the integer p should be a splitting prime, but these have a density of only $1/|\mathrm{Gal}_{\mathbb{Q}}(f)|$; however, implementation of arithmetic with nonsplitting primes is much less efficient. The target polynomial is built recursively by multiplying pairs of approximately equal-degree polynomials. Over $\mathbb{C}$ we start with quadratic and linear factors over $\mathbb{R}$.

ACKNOWLEDGMENTS

We thank Ian G. Macdonald for his perceptive comments, which have improved this paper, and Gene Smith and Thomas Mattman for sharing their computational experience with us.

BIBLIOGRAPHY

1. C. Batut, D. Bernardi, H. Cohen, and M. Olivier, *User's guide to PARI-GP*, April 1990.
2. David Boyd, *Reciprocal algebraic integers whose Mahler measures are non-reciprocal*, Canad. Math. Bull. **30** (1987), 3–8.
3. G. Butler and J. McKay, *The transitive groups of degree up to* 11, Comm. Algebra **11** (1983), 863–911.
4. ———, *The transitive groups of degree up to* 15, Comm. Algebra, 1994 (to appear).
5. B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, *Maple library reference manual*, Springer-Verlag, New York, 1991.
6. Lindsay Childs, *A concrete introduction to higher algebra*, Graduate Texts in Math., Springer-Verlag, New York, 1979.
7. Henri Darmon and David Ford, *Computational verification of* M_{11} *and of* M_{12} *as Galois groups over* $\mathbb{Q}$, Comm. Algebra **17** (1989), 2941–2943.
8. D. W. Erbach, J. Fischer, and J. McKay, *Polynomials with* $PSL(2, 7)$ *as Galois group*, J. Number Theory **11** (1979), 69–75.
9. David Ford, *On the computation of the maximal order in a Dedekind domain*, Ph.D. thesis, Ohio State University, 1978.
10. M. Guisti, D. Lazard, and A. Valibouze, *Algebraic transformations of polynomial equations, symmetric polynomials and elimination*, Symbolic and Algebraic Computation: International Symposium ISSAC '88 (P. Gianni, ed.), Lecture Notes in Comput. Sci., vol. 358, Springer-Verlag, 1988, pp. 309–314.
11. S. LaMacchia, *Polynomials with Galois group* $PSL(2, 7)$, Comm. Algebra **8** (1980), 983–992.
12. John Lipson, *Elements of algebra and algebraic computing*, Addison-Wesley, Reading, MA, 1981.
13. John McKay, *Advances in computational Galois theory*, Computers in Algebra (Martin C. Tangora, ed.), Marcel Dekker, New York, 1988, pp. 99–101.
14. G. Malle and B. H. Matzat, *Realisierung von* $PSL_2(F_p)$ *als Galoisgruppen über* $\mathbb{Q}$, Math. Ann. **272** (1985), 549–565.
15. B. H. Matzat and A. Zeh-Marschke, *Realisierung der Mathieugruppen* M_{11} *und* M_{12} *als Galoisgruppen über* $\mathbb{Q}$, J. Number Theory **23** (1986), 195–202.
16. G. F. Royle, *The transitive groups of degree* 12, J. Symbolic Comput. **4** (1987), 255–268.
17. C. C. Sims, *Computational methods in the study of permutation groups*, Computational Problems in Abstract Algebra (John Leech, ed.), Pergamon Press, Oxford, 1970, pp. 169–183.
18. L. H. Soicher and J. McKay, *Computing Galois groups over the rationals*, J. Number Theory **20** (1985), 273–281.
19. Leonard Soicher, *The computation of Galois groups*, Master's thesis, Concordia University, Montréal, Québec, Canada, April 1981.
20. R. P. Stauduhar, *The automatic determination of Galois groups*, Ph.D. thesis, University of California, Berkeley, 1969.
21. ———, *The determination of Galois groups*, Math. Comp. **27** (1973), 981–996.
22. A. Valibouze, *Fonctions symétriques et changements de bases*, Proc. European Conference on Computer Algebra, EUROCAL 1987 (James H. Davenport, ed.), Springer-Verlag, New York, 1989, pp. 323–332.
23. B. L. van der Waerden, *Algebra*, Vol. 1, Ungar, New York, 1970.

DEPARTMENT OF COMPUTER SCIENCES, CONCORDIA UNIVERSITY, MONTRÉAL, QUÉBEC, CANADA H3G 1M8

E-mail address: mckay@vax2.concordia.ca