The Totally Complex Sextic S_5 Extension of Minimum Discriminant

Nasser Haddad

A Thesis

in

The Department

of

Mathematics and Statistics

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Science at
Concordia University
Montreal, Quebec, Canada

July, 1996

ISBN 0-612-18396-3

ABSTRACT

The Totally Complex Sextic S_5 Extension

of Minimum Discriminant

Nasser Haddad

We demonstrate that the totally complex algebraic number field of degree six of minimum discriminant having Galois group $\mathrm{PGL}_2(\mathbb{F}_5) \simeq S_5$ is generated over the rationals by a root of the polynomial

$$f(t) = t^6 + 2t^5 + 3t^4 + 21.$$

This field has discriminant $-1778112 = -2^6 \cdot 3^4 \cdot 7^3$ and is unique up to isomorphy.

Acknowledgements

I would like to acknowledge the following persons for their kind assistance in the completion of my masters degree. Professor David Ford whose acceptance, relentless help, keen commentary and patience enabled me to discover new horizons and observe matters in a new perspective. Professor Chris Cummins who has always been ready to offer assistance and who has given me time and advice whenever needed. I would like to thank my wife Eliana and my three children who have been an example of patience and encouragement and who supported me through the hard times of my studies and completion of my degree. My brother Elia Haddad who has lent me his full support and stood by me whenever I became discouraged and disillusioned. I would like also to thank all my friends, especially Dr. John Asfour for his timely advice and for his scholarly and moral support.

Finally a kind word should be directed towards professor David Ford and the "Computational Technique in Combinatorics, Algebra and Number Theory" research team (FCAR) in the Concordia University Department of Computer Science for the financial support they provided me in the last year of my studies.

Table of Contents

List of Figures

Introduction

The computation of primitive algebraic number fields having degree 6 and minimum (absolute) discriminant has been dealt with except for the case of Galois group $\mathrm{PGL}_2(\mathbb{F}_5) \simeq S_5$. Extensive results (covering most of the Galois groups) are due to Martinet and others [**Berge et. al.**], [**Olivier 89, 90a, 90b, 91**]. But in [**Martinet**], the author says " ... for A_5, A_6, S_5, S_6 extensions, the minimum discriminants are known only in the S_6 case. However A_5 and A_6 extensions are probably out of our computational possibilities".

[**Ford & Pohst 92, 93**], gave details of computations to determine the totally real algebraic number fields of degree 6 and having Galois groups A_5 and A_6 and minimum discriminant. These computations completed the table given by Olivier excepting only the case of Galois group S_5.

This thesis describes a computation to determine the (unique) totally complex algebraic number field of degree 6 having Galois group S_5 and minimum discriminant. Chapter 1 gives definitions and results from algebraic number theory, to be used in what follows. Chapter 2 discusses Galois groups in general, and gives some specific properties of the group $\mathrm{PGL}_2(\mathbb{F}_5)$ that we exploit in our computations. Chapter 3 presents in great detail techniques for limiting the coefficients of the sextic polynomial $f(t) = t^6 + a_1t^5 + a_2a^4 + a_3t^3 + a_4t^2 + a_5t + a_6$ in order that the number of examples to be considered be minimized. Chapter 4 gives the results of our computations. Four appendices include listings and documentation of the programs that were used, and their (partial or complete) output.

Chapter One

Mathematical Preliminaries

If E is a field containing the field F, then E is said to be an extension field of F, denoted by E/F.

Let a number $\alpha \in E$ be a root of a polynomial p over F, that is, it satisfies a non-zero polynomial equation with coefficients in F. Among all polynomials over F, there exists a unique monic polynomial p of minimal degree subject to $p(\alpha) = 0$, and p is called the minimal polynomial of α over F. (A monic polynomial is one with leading coefficient 1.) The minimal polynomial of α is monic and irreducible over F. (These facts are to be found in [**Stewart 73**]). In other words, there exist elements $a_1, \ldots, a_n \in F$ such that

$$\alpha^n + a_1\alpha^{n-1} + \cdots + a_n = 0.$$

The degree n of this polynomial is called the degree of α with respect to F. The n distinct roots of this polynomial $\alpha_1, \ldots, \alpha_n$ are called the conjugates of α with respect to F. A root of a polynomial with coefficients in F is called an algebraic number over F. In particular, a root of a polynomial with rational coefficients is simply called an algebraic number. By [**Stewart 79**, Theorem 2.1] the set A of algebraic numbers is a subfield of the complex field $\mathbb{C}$. We define an algebraic number field to be a subfield K of $\mathbb{C}$ such that $[K : \mathbb{Q}]$ is finite. By [**Stewart 79**, Theorem 1.8] this implies that every element of K is algebraic, and hence $K \subseteq A$.

DEFINITION 1.1. *A complex number θ is called an algebraic integer when it is a root of a monic polynomial p integer coefficients.*

THEOREM 1.2 [**Stewart 79**, Theorem 2.2]. *If K is a number field then $K = \mathbb{Q}(\theta)$ for some algebraic number θ.*

COROLLARY 1.3 [**Stewart 79**, Corollary 2.11]. *If K is a number field then $K = \mathbb{Q}(\theta)$ for some algebraic integer θ.*

DEFINITION 1.4. *Let K be a finite extension of the rational numbers and let O_K denote the ring of all algebraic integers in K. The elements $u_1, \ldots, u_n$ of O_K form an integral basis for K if every element of O_K can be uniquely written as*

$$a_1 u_1 + \cdots + a_n u_n$$

with $a_1, \ldots, a_n \in \mathbb{Z}$.

THEOREM 1.5 [**Narkiewicz**, Theorem 9.7]. *Every field K of degree n over $\mathbb{Q}$ has an integral basis consisting of n elements.*

By choosing a basis $v_1, \ldots, v_n \in K$ as a linear space over $\mathbb{Q}$ we may assume that these elements lie in O_K since the following lemma holds.

LEMMA 1.6 [**Narkiewicz**, Lemma 9.5]. *If v is an algebraic number, then there exists a natural number N such that Nv is an algebraic integer.*

Assuming $[K : \mathbb{Q}] = n$, where both K and $\mathbb{Q}$ are subfields of $\mathbb{C}$ with $\mathbb{Q} \subset K = \mathbb{Q}(\theta)$, there exist n embeddings of K in $\mathbb{C}$. The number θ can be sent to any of its n conjugates over $\mathbb{Q}$. ([**Marcus**, p 19])

Let $\sigma_1, \ldots, \sigma_n$ be the n embeddings of K in $\mathbb{C}$. The discriminant of an integral basis is independent of which integral basis we choose. This value is called the discriminant of the algebraic number field (or of O_K). So, if $u_1, \ldots, u_n$ is an integral basis of the algebraic number field K then

$$d(K) = \mathrm{disc}(u_1, \ldots, u_n) = |\sigma_i(u_j)|^2$$

i.e., the square of the determinant of the matrix having $\sigma_i(u_j)$ in the i^{th} row, j^{th} column. For two integral bases $\{u_1, \ldots, u_n\}$, $\{v_1, \ldots, v_n\}$ of an algebraic number field K, we have

$$\mathrm{disc}(u_1, \ldots, u_n) = (\pm 1)^2 \, \mathrm{disc}(v_1, \ldots, v_n) = \mathrm{disc}(v_1, \ldots, v_n),$$

because the matrix corresponding to the change of basis is unimodular ([**Stewart 79**, p 53])

EXAMPLE: The ring of algebraic integers of $\mathbb{Q}[\sqrt{5}]$ is $\mathbb{Z}\left[\frac{1}{2}(1+\sqrt{5})\right]$. An integral basis is the set $\left\{ 1, \frac{1}{2}(1+\sqrt{5}) \right\}$ and the field discriminant is $d = 5$. Details can be found in [**Stewart 79**, p 60].

DEFINITION 1.9. *The signature of a number field is the pair (r_1, r_2) where r_1 is the number of embeddings of K whose images lie in $\mathbb{R}$, and $2r_2$ is the number of non-real complex embeddings, so that $r_1 + 2r_2 = n$.*

If f is an irreducible polynomial defining the number field K by one of its roots, the signature of K will also be called the signature of f. So, when $r_1 = 0$, we say that K and f are totally complex.

Chapter Two

Galois Groups

Here we describe briefly the idea of the Galois group of an algebraic number field, and develop the specific properties of the group $\mathrm{PGL}_2(\mathbb{F}_5)$ that are used in our computations.

DEFINITION 2.1. *A bijective mapping of a given set into itself is called a permutation.*

DEFINITION 2.2. *Let A be any nonempty set and let S_A be the set of all permutations of A. The set S_A is a group under composition. This group is called the symmetric group on the set A. In case $A = \{1, 2, \ldots, n\}$, the symmetric group on A is denoted by S_n.*

DEFINITION 2.3. *Let K be a number field of degree n. K is called Galois over $\mathbb{Q}$, if K is invariant for the n embeddings of K in $\mathbb{C}$.*

Every such embedding sends K into itself since it sends each element to one of its conjugates. All such embeddings form a group, called the *Galois group* of K, and denoted by $\mathrm{Gal}(K/\mathbb{Q})$. In other words, for $K = \mathbb{Q}[\theta]$, there exist exactly n embeddings of K in $\mathbb{C}$, given by $\theta \mapsto \theta_i$, where the θ_i are the roots in $\mathbb{C}$ of the minimal polynomial of θ. These embeddings are $\mathbb{Q}$-linear, their images K_i in $\mathbb{C}$ are called the conjugate fields of K, and the K_i are isomorphic to K. By invariant we mean that for all the embeddings σ_i of K in $\mathbb{C}$ we have $\sigma_i(K) = K$. So, the Galois group may be considered as a permutation group acting on the roots of a generating polynomial [**Cohen**, p 153].

Any automorphism $\sigma \in \mathrm{Gal}(K/\mathbb{Q})$ maps a root of an irreducible factor of the polynomial f over $\mathbb{Q}$ to another root of the irreducible factor and σ is uniquely determined by its action on these roots (since they generate K over $\mathbb{Q}$). Fixing a labelling of the roots $\alpha_1, \ldots, \alpha_n$ of f and observing that any $\sigma \in \mathrm{Gal}(K/\mathbb{Q})$ defines a unique permutation of $\alpha_1, \ldots, \alpha_n$, and also defines a unique permutation of the subscripts $\{1, 2, \ldots, n\}$. This leads to an injection

$$\mathrm{Gal}(K/\mathbb{Q}) \hookrightarrow S_n$$

of the Galois group into the symmetric group on n letters which is a homomorphism (both group operations are composition). So, we may consider Galois groups as subgroups of symmetric groups.

Definition 2.4. *Let $f \in \mathbb{Q}[x]$ with $n = \deg(f) > 1$, and let the zeros of f be $\alpha_1, \ldots, \alpha_n$. Then the discriminant of f is:*

$$\mathrm{disc}(f) = \prod_{i<j} (\alpha_i - \alpha_j)^2.$$

Let f be a polynomial with rational coefficients. We may assume that f is separable (*i.e.*, its zeros in K are distinct) and has integer coefficients. Then the discriminant D of f is an integer and is nonzero. For any prime p, consider the reduction in $\mathbb{F}_p$ of $\bar{f} \equiv f \bmod p$. If p divides D then the reduced polynomial $\bar{f}$ has discriminant $\overline{D} = 0 \in \mathbb{F}_p$, so is not separable. If p does not divide D, then $\bar{f}$ is a separable polynomial over $\mathbb{F}_p$ and we can factor $\bar{f}$ into distinct irreducibles

$$\bar{f} = \bar{f}_1 \bar{f}_2 \cdots \bar{f}_k$$

in $\mathbb{F}_p[x]$, $\deg(\bar{f}_i) = n_i$, $i = 1, 2, \ldots, k$.

THEOREM 2.5 ([**Dummit & Foot**, p 553]). *For any prime p not dividing the discriminant D of $f \in \mathbb{Z}[x]$, the Galois group over $\mathbb{F}_p$ of the reduction $\bar{f} = f \bmod p$ is permutation group isomorphic to a subgroup of the Galois group over $\mathbb{Q}$ of f*

It follows that not only is the Galois group of the reduction $\bar{f} \bmod p$ of f isomorphic to a subgroup of the Galois group of f but there is an ordering of the roots of $\bar{f}$ and of f (depending on p) so that under this isomorphism the action of the corresponding automorphisms as permutations of these roots is the same. So, there are automorphisms in the Galois group of f with the same cycle types as the automorphism of $\bar{f}$. The roots of $\bar{f}_1$ are permuted amongst themselves by the Galois group and given any two of these roots there is a Galois automorphism taking the first root to the second (since the group is transitive; i.e., given any two elements $a, b \in K$ there is some $\tau \in G$ such that $a = \tau b$). Similarly, the Galois group permutes the roots of each of the factors $\bar{f}_i, i = 1, \ldots, k$, transitively. Since these factors are relatively prime we also see that no root of one factor is mapped to a root of any other factor by any element of the Galois group.

DEFINITION 2.6. *We define the cycle type of a permutation T of degree n to be the partition of n induced by the lengths of the disjoint cycles of T. The factorization of a polynomial f modulo any prime p also induces a partition, namely the partition of $deg(f)$ formed by the degrees of the factors.*

LEMMA 2.7 ([**Lagarias & Odlyzko**], [**van der Waerden**, section 8.10]). *For any good prime p relative to a polynomial f, (i.e., p not dividing the discriminant of f) the degree partition of the factorization of $f \bmod p$ is the cycle type of some permutation in $\mathrm{Gal}(f/\mathbb{Q})$.*

LEMMA 2.8 ([**Lagarias & Odlyzko**]). *Let T be a partition of n. Then as $s \longrightarrow \infty$, the proportion of occurrences of T as the factor type of f mod p_i, $i = 1, \ldots, s$, ($p_1, \ldots, p_s$ distinct primes) tends to the proportion of permutations in* $\mathrm{Gal}(f/\mathbb{Q})$ *having the cycle type T.*

DEFINITION 2.9. *The general linear group over the field F, $\mathrm{GL}_n(F)$, is the set of nonsingular $n \times n$ matrices with entries from F. The subgroup of such matrices of determinant 1 is the special linear group $\mathrm{SL}_n(F)$.*

It is easily seen that $|\mathrm{GL}_2(\mathbb{F}_5)| = 480$. Defining

$$\left\langle \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \right\rangle$$

to be the normal subgroup T of $\mathrm{GL}_2(\mathbb{F}_5)$, we have

$$\mathrm{PGL}_2(\mathbb{F}_5) = \mathrm{GL}_2(\mathbb{F}_5)/T,$$

and therefore $|\mathrm{PGL}_2(\mathbb{F}_5)| = 120$.

The group $\mathrm{PGL}_2(\mathbb{F}_5)$ is isomorphic to the subgroup of S_6 generated by $(1\ 2\ 3\ 4\ 5)$ and $(1\ 6)(2\ 3)(4\ 5)$; the cycle types $1{\cdot}1{\cdot}1{\cdot}1{\cdot}2$, $1{\cdot}1{\cdot}1{\cdot}3$, $1{\cdot}2{\cdot}3$ and $2{\cdot}4$ do not occur in this group. So if f is a polynomial with Galois group $\mathrm{PGL}_2(\mathbb{F}_5)$ and p is a prime not dividing the discriminant of f then the degree sequence of the mod p factors of f cannot be among these four types.

According to [**Cohen**, p 325], sixth degree permutation groups are of four types. A sextic field has a quadratic subfield if and only if its Galois group is isomorphic to a subgroup of

$$G_{72} = \langle (123), (14), (25), (36), (1524)(36) \rangle.$$

Similarly, it has a cubic subfield if and only if its Galois group is isomorphic to a subgroup of $S_4 \times C_2$. Hence, it has both a quadratic and a cubic subfield if and only if its Galois group is isomorphic to C_6, S_3, or D_6 (these being the groups that arise in both cases). If the field is primitive, *i.e.*, does not have quadratic or cubic subfields, this implies that its Galois group can only be $\mathrm{PSL}_2(\mathbb{F}_5) \simeq A_5$, $\mathrm{PGL}_2(\mathbb{F}_5) \simeq S_5$, A_6 or S_6.

Chapter Three

Computing Bounds on Coefficients

For $\rho_1, \ldots, \rho_n$ the algebraic conjugates of an algebraic number ρ of degree n, and for $m \in \mathbb{Z}$, we define

$$S_m(\rho) = \rho_1^m + \cdots + \rho_n^m,$$

$$T_m(\rho) = |\rho_1|^m + \cdots + |\rho_n|^m.$$

An important theorem, originally due to Hunter (see Theorem 3.3 below), establishes the existence of an algebraic integer ρ having $T_2(\rho)$ within a bound given in terms of the field discriminant and $\mathrm{Tr}(\rho)$, and further, that $\mathrm{Tr}(\rho)$ can be taken to lie between 0 and $n/2$. Once $T_2(\rho)$ has been bounded, a theorem of Pohst (Theorem 3.2 below) can be applied to develop bounds on $T_m(\rho)$, for $m \geq 3$. Newton's relations, involving $S_1(\rho), \ldots, S_n(\rho)$ and the coefficients of the characteristic polynomial χ of ρ, combined with the obvious fact that

$$|S_m(\rho)| \leq T_m(\rho)$$

for all $m \geq 1$, then enable us to calculate explicit bounds on the coefficients of χ.

LEMMA 3.1. *Let $w > 1$ and define the function f on the real numbers as*

$$f(x) = \begin{cases} \dfrac{1 - x^w}{1 - x}, & \text{for } x \neq 1, \\ w, & \text{for } x = 1. \end{cases}$$

Then f is strictly increasing in the interval $0 < x < \infty$.

PROOF: Let $g = (1 - x)^2 \dfrac{df}{dx}$, so that

$$g = 1 - x^{w-1} - (w - 1)x^{w-1}(1 - x),$$

and let $h = \frac{dg}{dx}$, giving

$$h = w(w-1)(x-1)x^{w-2}.$$

In the interval $0 < x < 1$, h is clearly negative; hence g is decreasing. Because $g(1) = 0$, g must be positive in this interval. Hence $\frac{df}{dx}$ is positive, so f is strictly increasing in the interval $0 < x < 1$. A similar discussion shows f is strictly increasing in the interval $1 < x < \infty$. $\square$

THEOREM 3.2. *Let R, K and m be positive constants satisfying $R \geq 3K^{1/3}$ and $m > 2$. Then*

$$F_m(x) := x_1^m + x_2^m + x_3^m$$

has a global maximum on the set

$$S := \left\{ x \in \left(\mathbb{R}^{\geq 0}\right)^3 \;\middle|\; x_1^2 + x_2^2 + x_3^2 \leq R^2;\ x_1 x_2 x_3 = K \right\}$$

at a point $y = (y_1, y_2, y_3)$ with $y_2 = y_3$ and $y_1^2 + y_2^2 + y_3^2 = R^2$.

PROOF: The condition $R \geq 3K^{1/3}$ ensures S is not empty. If $R = 3K^{1/3}$ then S consists of the single point $x_0 = (K^{1/3}, K^{1/3}, K^{1/3})$ and the theorem is obvious. We therefore assume that $R > 3K^{1/3}$, and that $F_m(x)$ has its maximum on S at a point y.

Because $y_1^2 + y_2^2 + y_3^2 \leq R^2$ we may choose y_4 so that

$$y_1^2 + y_2^2 + y_3^2 + y_4^2 = R^2$$

and apply the Lagrange multiplier method to maximize

$$F_m(y) = y_1^m + y_2^m + y_3^m$$

subject to the constraints

$$y_1^2 + y_2^2 + y_3^2 + y_4^2 = R^2 \quad \text{and} \quad y_1 y_2 y_3 = K.$$

Defining

$$L(y_1, y_2, y_3, y_4, \lambda_1, \lambda_2) = y_1^m + y_2^m + y_3^m + \lambda_1(y_1^2 + y_2^2 + y_3^2 + y_4^2) + \lambda_2 y_1 y_2 y_3,$$

we have

$$\begin{aligned} 0 &= \frac{\partial L}{\partial y_1} = m y_1^{m-1} + 2\lambda_1 y_1 + \lambda_2 y_2 y_3 \\ 0 &= \frac{\partial L}{\partial y_2} = m y_2^{m-1} + 2\lambda_1 y_2 + \lambda_2 y_1 y_3 \\ 0 &= \frac{\partial L}{\partial y_3} = m y_3^{m-1} + 2\lambda_1 y_3 + \lambda_2 y_1 y_2 \\ 0 &= \frac{\partial L}{\partial y_4} = 2\lambda_1 y_4. \end{aligned}$$

If $y_4 \neq 0$ then $\lambda_1 = 0$. We may form the sum

$$0 = y_1 \frac{\partial L}{\partial y_1} + y_2 \frac{\partial L}{\partial y_2} + y_3 \frac{\partial L}{\partial y_3}$$

and solve for λ_2, which gives

$$\lambda_2 = -\frac{m(y_1^m + y_2^m + y_3^m)}{3 y_1 y_2 y_3}.$$

Substituting back and simplifying yields

$$\begin{aligned} 0 &= 2y_1^m - y_2^m - y_3^m \\ 0 &= -y_1^m + 2y_2^m - y_3^m \\ 0 &= -y_1^m - y_2^m + 2y_3^m \end{aligned}$$

which implies $y = x_0$, which is easily seen to be an absolute minimum for $F_m(x)$ on S. Thus $y_4 = 0$ if $F_m(x)$ has its maximum at y.

Because $R > 3K^{1/3}$ the coordinates of y can not all be equal; without loss of generality we assume $y_1 \neq y_2$ and $y_1 \neq y_3$. Because

$$0 = y_j \frac{\partial L}{\partial y_j} = m y_j^m + 2\lambda_1 y_j^2 + \lambda_2 y_1 y_2 y_3$$

for $j = 1, 2, 3$, we have

$$\frac{y_1^m - y_2^m}{y_1^2 - y_2^2} = -\frac{2\lambda_1}{m} = \frac{y_1^m - y_3^m}{y_1^2 - y_3^2}$$

so that

$$\frac{1 - \left(\frac{y_2^2}{y_1^2}\right)^{m/2}}{1 - \frac{y_2^2}{y_1^2}} = \frac{1 - \left(\frac{y_3^2}{y_1^2}\right)^{m/2}}{1 - \frac{y_3^2}{y_1^2}}.$$

By Lemma 3.1 above we have $\frac{y_2^2}{y_1^2} = \frac{y_3^2}{y_1^2}$, hence $y_2 = y_3$. □

REMARK: A general version of this result appears as Theorem 4 in [**Pohst**].

For totally complex roots, $f(t)$ can be written as

$$\begin{aligned} f(t) &= (t - \alpha_1)(t - \overline{\alpha}_1)(t - \alpha_2)(t - \overline{\alpha}_2)(t - \alpha_3)(t - \overline{\alpha}_3) \\ &= (t^2 - \beta_1 t + \gamma_1)(t^2 - \beta_2 t + \gamma_2)(t^2 - \beta_3 t + \gamma_3) \\ &= t^6 + a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t + a_6 \end{aligned}$$

with

$$
\begin{aligned}
\beta_j &= \alpha_j + \overline{\alpha}_j = 2\Re(\alpha_j) = 2\Re(\overline{\alpha}_j) \in \mathbb{R}, \quad & j = 1,2,3 \\
\gamma_j &= \alpha_j \overline{\alpha}_j = |\alpha_j|^2 = |\overline{\alpha}_j|^2 > 0, & j = 1,2,3 \\
-a_1 &= \beta_1 + \beta_2 + \beta_3 \\
a_2 &= \beta_1\beta_2 + \beta_1\beta_3 + \beta_2\beta_3 + \gamma_1 + \gamma_2 + \gamma_3 \\
-a_3 &= \beta_1\beta_2\beta_3 + (\beta_1 + \beta_2)\gamma_3 + (\beta_1 + \beta_3)\gamma_2 + (\beta_2 + \beta_3)\gamma_1 \\
a_4 &= \beta_1\beta_2\gamma_3 + \beta_1\beta_3\gamma_2 + \beta_2\beta_3\gamma_1 + \gamma_1\gamma_2 + \gamma_1\gamma_3 + \gamma_2\gamma_3 \\
-a_5 &= \beta_1\gamma_2\gamma_3 + \beta_2\gamma_1\gamma_3 + \beta_3\gamma_1\gamma_2 \\
a_6 &= \gamma_1\gamma_2\gamma_3 > 0.
\end{aligned}
$$

For the root α of f we have

$$
\begin{aligned}
S_m = S_m(\alpha) &= \alpha_1^m + \overline{\alpha}_1^m + \alpha_2^m + \overline{\alpha}_2^m + \alpha_3^m + \overline{\alpha}_3^m, \\
T_m = T_m(\alpha) &= |\alpha_1|^m + |\overline{\alpha}_1|^m + |\alpha_2|^m + |\overline{\alpha}_2|^m + |\alpha_3|^m + |\overline{\alpha}_3|^m \\
&= 2\left(\gamma_1^{m/2} + \gamma_2^{m/2} + \gamma_3^{m/2}\right)
\end{aligned}
$$

for $m \geq 1$. Then $S_m \in \mathbb{Z}$, and it follows by the triangle inequality that

$$
|S_m| \leq T_m. \tag{5}
$$

THEOREM 3.3. *If K is a totally complex algebraic number field of degree six with discriminant $d(K)$ then K has an integral element ρ, with $\rho \notin \mathbb{Z}$, such that*

$$
0 \leq \mathrm{Tr}(\rho) \leq 3 \quad \textit{and} \quad T_2(\rho) \leq \frac{\mathrm{Tr}(\rho)^2}{6} + \left(\frac{4}{3}|d(K)|\right)^{\frac{1}{5}}.
$$

PROOF: This is a special case of Theorem 6.4.2 of [**Cohen**], with $n = 6$. The theorem originates in [**Hunter**]. □

THEOREM 3.4. *The coefficients a_1, a_2, a_6 may be chosen to satisfy the following.*

$$a_1 = -\operatorname{Tr}(\rho) \in \{0,1,2,3\} \tag{1}$$

$$1 \le a_6 \le \left(\frac{T_2(\rho)}{6}\right)^3 \tag{2}$$

$$a_2 \le \frac{1}{2}T_2(\rho) + \frac{1}{3}a_1^2 \tag{3}$$

PROOF: Equation (1) follows by replacing ρ with $-\rho$. For (2) we apply the inequality between arithmetic and geometric means to write

$$a_6^{1/3} = (\gamma_1\gamma_2\gamma_3)^{1/3} \le \frac{1}{3}(\gamma_1 + \gamma_2 + \gamma_3) = \frac{T_2}{6}$$

and the result follows.

For (3) we apply the Cauchy-Schwartz inequality to the vectors $A = (1,1,1)$ and $B = (\beta_1, \beta_2, \beta_3)$ to yield

$$a_1^2 = (\beta_1 + \beta_2 + \beta_3)^2 = (A \cdot B)^2 \le |A|^2|B|^2 = 3(\beta_1^2 + \beta_2^2 + \beta_3^2) = 3(S_2 + T_2).$$

Substituting $S_2 = a_1^2 - 2a_2$ we have

$$a_2 \le \frac{1}{2}\left(T_2 + \frac{2}{3}a_1^2\right).$$

See also (15), (17) and (18) in [**Pohst**]. □

To determine bounds on $a_2, \ldots, a_6$ we proceed as follows.

For $m = 2$, we define $Y_2 = \lceil T_2 \rceil$ (which implies $Y_2 \ge T_2 \ge \lfloor T_2 \rfloor$).

For $m \ge 3$, values for Y_m are found by applying Theorem 3.2 with $R = \sqrt{T_2/2}$, $K = \sqrt{a_6}$. Observing that $T_m = 2F_m(\sqrt{\gamma_1}, \sqrt{\gamma_2}, \sqrt{\gamma_3})$, we conclude that

$$T_m \le \max\left\{ 2y_1^m + 4y_2^m \;\middle|\; 2y_1^2 + 4y_2^2 = T_2;\; y_1^2 y_2^4 = a_6 \right\}.$$

The conditions on y_1, y_2 immediately give

$$4u^3 - 4T_2u^2 + T_2^2u - 16a_6 = 0$$
$$4v^3 - T_2v^2 + 2a_6 = 0$$

with $u = y_1^2$ and $v = y_2^2$ (see graphs in Appendix I). Altogether these conditions permit only two solutions, (u_1, v_1), (u_2, v_2), and these solutions must satisfy

$$0 < u_1 \leq \frac{T_2}{6} \leq u_2 < \frac{T_2}{2}$$
$$0 < v_2 \leq \frac{T_2}{6} \leq v_1 < \frac{T_2}{4}.$$

The roots u_1, u_2, v_1, v_2 can be computed with sufficient accuracy so that the integer value

$$Y_m = \max\left(\left\lfloor 2u_1^{m/2} + 4v_1^{m/2}\right\rfloor, \left\lfloor 2u_2^{m/2} + 4v_2^{m/2}\right\rfloor\right) \geq \lfloor T_m \rfloor$$

can be determined.

Defining

$$X_m = \sum_{i=1}^{m-1} a_i S_{m-i}$$

for $1 \leq m \leq 6$ and applying Newton's relations [**Cohen**, Proposition 4.3.3] we have

$$S_m + X_m + ma_m = 0. \tag{7}$$

For any Y_m satisfying $Y_m \geq \lfloor T_m \rfloor$ we may apply (5) and (7) to get

$$|X_m + ma_m| = |S_m| \leq \lfloor T_m \rfloor \leq Y_m$$

so that

$$\frac{-X_m - Y_m}{m} \leq a_m \leq \frac{-X_m + Y_m}{m}$$

for $m = 2, \ldots, 6$. Also, from (2) we have

$$1 \leq a_6 \leq \left\lfloor \left(\frac{T_2}{6}\right)^3 \right\rfloor \leq \left\lfloor \frac{Y_2^3}{6^3} \right\rfloor.$$

When $a_1 \neq 0$, another bound on a_5 may be computed. Taking $m = 5$ and 6 in (7) and eliminating S_5, then solving for a_5, gives

$$a_5 = \frac{-X_6' + S_6}{6a_1}$$

where

$$X_6' = a_1 X_5 - a_2 S_4 - a_3 S_3 - a_4 S_2 - 6a_6$$

so that

$$\frac{-X_6' - Y_6}{6a_1} \leq a_5 \leq \frac{-X_6' + Y_6}{6a_1}.$$

Chapter Four

Experimental Results

For an appropriate bound B, we are to generate at least one defining polynomial

$$f(t) = t^6 + a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t + a_6$$

for each field F with $|d_F| \leq B$ and signature $= (0,3)$.

Given such a bound B, we construct a set M of monic sixth-degree polynomials such that each primitive totally complex algebraic number field F of degree 6 and discriminant $d_F \leq B$ contains a generating element $\rho \in F - \mathbb{Q}$ for which the minimal polynomial $m_\rho(t)$ belongs to M.

A preliminary computation produced $f(t) = t^6 + 2t^5 + 3t^4 + 21$ with $d_F = -1778112$, signature $= (0,3)$, $\mathrm{Gal}(f/\mathbb{Q}) = \mathrm{PGL}_2(\mathbb{F}_5)$, establishing that any totally complex sextic number field K of minimum absolute discriminant has discriminant d_K satisfying $|d_K| \leq |d_F| = 1778112$. So, by setting $B = |d_F| = 1778112$, and assuming $\mathrm{Tr}(\rho) \leq 3$, we apply Theorem 3.3 to calculate

$$T_2(\rho) \leq \frac{\mathrm{Tr}(\rho)^2}{6} + \left(\frac{4}{3}B\right)^{1/5} \leq 20.336.$$

An ALGEB program (see [**Ford**]) computed approximations to the roots u_1, u_2, v_1, v_2 (see chapter three) so that the integer values Y_3, Y_4, Y_5, Y_6 could be determined. These values, with a listing of the ALGEB program, appear in Appendix I.

Having the table above, a PASCAL program was written to read the data in this table and generate examples. This program is given in Appendix II, with detailed descriptions of its components in Appendix III.

An ALGEB program computed signatures (using Sturm sequences [**Cohen**, Algorithm 4.1.11]) and field discriminants (using the Round Two algorithm of Zassenhaus [**Cohen**, Algorithm 6.1.8]), excluding polynomials with real roots or with $|d_F| > 1778112$. Finally, MAPLE programs computed Galois groups and confirmed that all examples with $d_F = -1778112$ were isomorphic (see Appendix IV).

The polynomial generation and screening required about 82 CPU-hours on a Digital AlphaServer 2100 4/200 in the Department of Computing Services at Concordia University. The remaining computations—signatures, field discriminants, Galois groups, field isomorphisms and integral bases—took less than one CPU-hour in total, using a variety of systems at Concordia University.

THEOREM. *The discriminant of minimum absolute value for a totally complex S_5 extension of degree 6 is $d = -1778112 = -2^6 \cdot 3^4 \cdot 7^3$. There is, up to isomorphy, exactly one field F of that discriminant with Galois group S_5. It is generated by a root ρ of the polynomial*

$$f(t) = t^6 + 2t^5 + 3t^4 + 21.$$

An integral basis for F is given by

$$1, \quad \rho, \quad \rho^2, \quad \rho^3, \quad \rho^4, \quad \frac{1}{277}(29 - 106\rho + 120\rho^2 - 47\rho^3 + 135\rho^4 + \rho^5).$$

References

Berge et. al. A.-M. Berge, J. Martinet & M. Olivier, *The computation oof sextic fields with a quadratic subfield*, Math. Comp. **54** (1990 pages 869-884).

Cohen H. Cohen, "A Course in Computational Algebraic Number Theory," Springer-Verlag, Berlin, 1993.

Dummit & Foot D. Dummit & R. Foot, "Abstract Algebra," Prentice-Hall, New Jersey, 1991.

Ford D. Ford, "On the Computation of the Maximal Order in a Dedekind Domain," Ph.D. Dissertation, Ohio State University, 1978.

Ford & Pohst 92 D. Ford & M. Pohst, *The Totally Real A_5 Extension of Degree 6 with Minimum Discriminant*, Experimental Math. **1** (1992), no. 3, 231-235.

Ford & Pohst 93 D. Ford & M. Pohst, *The Totally real A_6 Extension of Degree 6 with Minimum Discriminant*, Experimental Math. **2** (1993), no. 3, 231-232.

Hunter J. Hunter, *The Minimum Discriminants of Quintic Fields*, Proc. Glasgow Math. Ass. **3** (1957), 57–67.

Lagarias & Odlyzko J. C. Lagarias & A. M. Odlyzko, *Effective Versions of the Chebotarev Density Theorem*, in "Algebraic Number Fields (L-functions and Galois Theory),," A. Frölich, ed., Academic Press, 1977, pp. 409–464.

Marcus D. Marcus, "Number Theory," Springler-Verlag, New York, 1977.

Martinet J. Martinet, *Discriminants and Permutation Groups*, in "Number Theory," Proceedings of the First Conference of the Canadian Number Theory Association, Banff, 1988 (R. A. Mollin, ed.), de Gruyter, Berlin and New York, 1990, pp. 359–385.

Narkiewicz W. Narkiewicz, "Number Theory," Polish Scientific Publishers, Warszawa, 1977.

Olivier 89 M. Olivier, *Tables de corps sextiques contenant un sous-corps quadratique (I)*, Séminaire de Théorie des Nombres de Bordeaux **1** **Sér. 2** (1989), 205-250.

Olivier 90a M. Olivier, *Corps sextiques contenant un corps quadratique (II)*, Séminaire de Théorie des Nombres de Bordeaux **2** **Sér. 2** (1990), 49–102.

Olivier 90b M. Olivier, *Corps Sextiques Primitifs*, Annales de l'Institut Fourier **40** (1990), no. 4, 757–767.

Olivier 91 M. Olivier, *Corps sextiques contenant un corps cubique (III)*, Séminaire de Théorie des Nombres de Bordeaux **3** **Sér. 2** (1991), 201–245.

Pohst M. Pohst, *On the Computation of Number Fields of Small Discriminants Including the Minimum Discriminant of Sixth Degree Fields*, J. Number Theory **14** (1982), 99–117.

Stewart 73 I. Stewart, "Galois Theory," Chapman and Hall, London, 1973.

Stewart 79 I. Stewart, "Algebraic Number Theory," Chapman and Hall, London, 1979.

van der Waerden B. L. van der Waerden, "Algebra," Ungar, New York, 1970.

APPENDIX I — BOUNDS FOR Y_3, Y_4, Y_5, Y_6

Using the results from chapter three, the ALGEB program `tmtbla.agb` computes bounds on Y_3, Y_4, Y_5 and Y_6 for each possible combination of values of $Y_2 = \lceil T_2 \rceil$ and a_6. Theorem 3.2 of chapter three implies $\gamma_2 = \gamma_3$; taking $y = \sqrt{\gamma_1}$ and $z = \sqrt{\gamma_2} = \sqrt{\gamma_3}$ we define

$$\frac{u}{\mu} = y^2 = \gamma_1, \quad \frac{v}{\mu} = z^2 = \gamma_2 = \gamma_3.$$

The denominator μ is a parameter controlling the accuracy of our approximations. We know

$$T_2 = 2y^2 + 4z^2 = 2\frac{u}{\mu} + 4\frac{v}{\mu} \implies w_1 = 2u + 4v - \mu T_2 = 0,$$

$$a_6 = \gamma_1\gamma_2\gamma_3 = y^2 z^4 = \frac{uv^2}{\mu^3} \implies w_2 = uv^2 - \mu^3 a_6 = 0$$

which together imply

$$0 = \frac{1}{4}\,\mathrm{Resultant}(w_1, w_2, u) = -\left(v^3 - \frac{\mu}{4}T_2 v^2 + \frac{\mu^3}{2}a_6\right) = -\mu^3 g_z\left(\frac{v}{\mu}\right),$$

$$0 = \frac{1}{4}\,\mathrm{Resultant}(w_1, w_2, v) = u^3 - \mu T_2 u^2 + \frac{\mu^2}{4}T_2^2 u - 4\mu^3 a_6 = \mu^3 g_y\left(\frac{u}{\mu}\right)$$

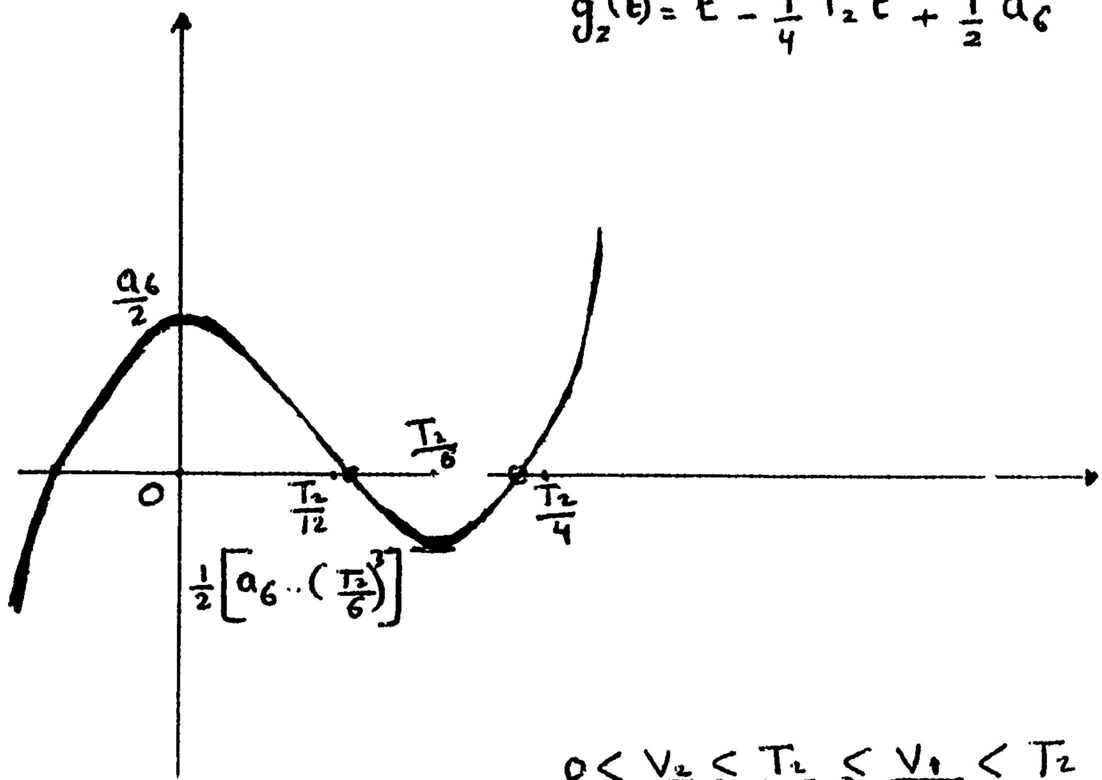
where

$$g_z(t) = t^3 - \frac{1}{4}T_2 t^2 + \frac{1}{2}a_6,$$

$$g_y(t) = t^3 - T_2 t^2 + \frac{1}{4}T_2^2 t - 4a_6.$$

The graphs of g_z and g_y are shown in figures 1 and 2 respectively. In the main loop we use successively larger values of μ, until our upper and lower approximations of $\lfloor T_3 \rfloor$, $\lfloor T_4 \rfloor$, $\lfloor T_5 \rfloor$ and $\lfloor T_6 \rfloor$ coincide.

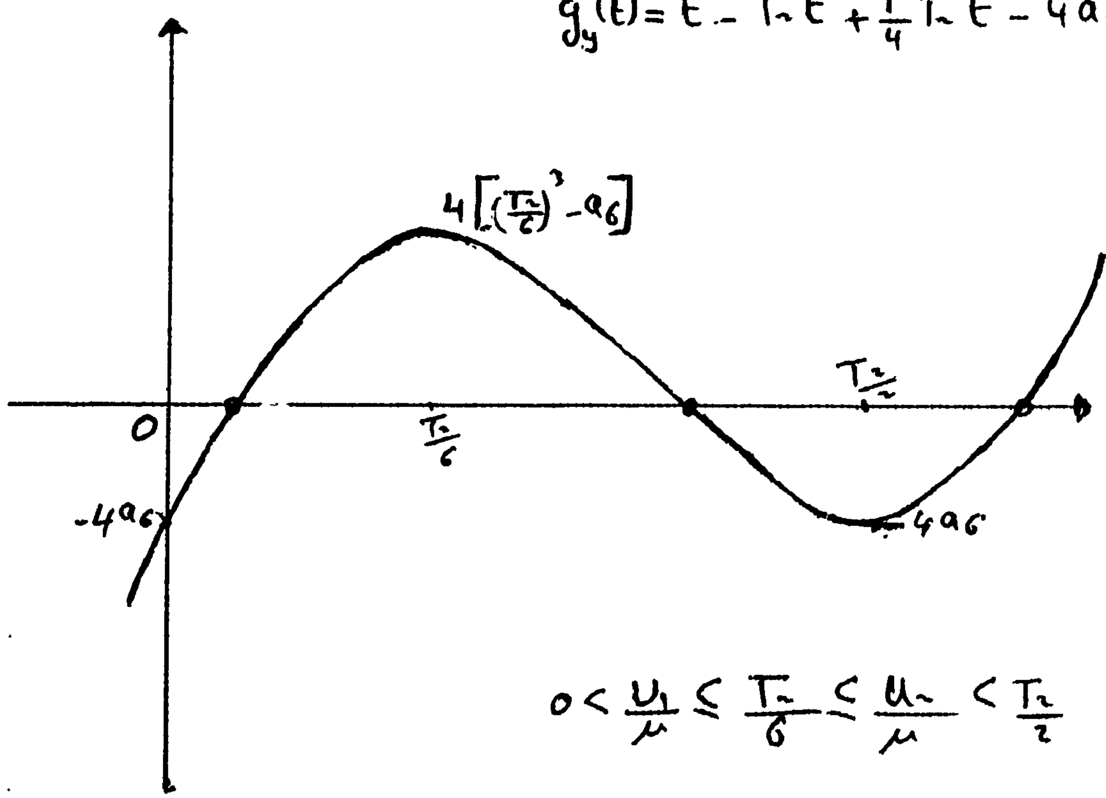
$$g_z(t) = t^3 - \frac{1}{4}T_2 t^2 + \frac{1}{2}a_6$$



$$0 < \frac{V_2}{\mu} \le \frac{T_2}{6} \le \frac{V_1}{\mu} < \frac{T_2}{4}$$

Figure 1

$$g_y(t) = t^3 - T_2 t^2 + \frac{1}{4}T_2^2 t - 4a_6$$



$$0 < \frac{U_1}{\mu} \le \frac{T_2}{6} \le \frac{U_2}{\mu} < \frac{T_2}{2}$$

Figure 2

ALGEB Program trtbla.agb

```
iochan 1;

begin
integer m, ct, ca, cb;
integer T2, a6, a6l, a6u;
integer Y2l, Y3l, Y4l, Y5l, Y6l;
integer Y2h, Y3h, Y4h, Y5h, Y6h;
integer u1l, u1h, r1l, r1h, u2l, u2h, r2l, r2h;
integer v1l, v1h, s1l, s1h, v2l, v2h, s2l, s2h;
integer T2l, T3l, T4l, T5l, T6l;
integer T2h, T3h, T4h, T5h, T6h;
boolean done;

integer procedure max (x, y);
integer x, y;
if x < y then max := y
         else max := x
end;

integer procedure ceilg (u, v);          { truncate rightwards }
value u, v;  integer u, v;
if (v < 0) then u := -u ! v := -v;
if (u < 0) then ceilg := u / v
           else ceilg := (u + v - 1) / v
end;

integer procedure floor (u, v);          { truncate leftwards }
value u, v;  integer u, v;
if (v < 0) then u := -u ! v := -v;
if (u > 0) then floor := u / v
           else floor := (u - v + 1) / v
end;

integer procedure gy (u);                { m^3 * gy(u/m) }
integer u;
gy := ((u - m*T2)*u + cb*T2*T2)*u - 4*m*m*m*a6
end;

integer procedure gz (v);                { m^3 * gz(v/m) }
integer v;
gz := (v - ct*T2)*v*v + ca*a6
end;
```

```
procedure vsolve (ul, uh, vl, vh);
integer ul, uh, vl, vh;
integer uc, vc, g, gl, gh, d;
gl := gy(ul);
gh := gy(uh);
if (gl <= 0) and (0 <= gh) then
    d := +1
else
if (gl >= 0) and (0 >= gh) then
    d := -1
else
    d := 0 ! writes(1,"You have made a y-mistake!") ! line(1,1);
if gl = 0 then uh := ul else
if gh = 0 then ul := uh else
while uh - ul > 1 do
    begin
    uc := (ul + uh)/2;
    g  := d*gy(uc);
    if g <= 0 then ul := uc;
    if g >= 0 then uh := uc
    end;
gl := gz(vl);
gh := gz(vh);
if (gl <= 0) and (0 <= gh) then
    d := +1
else
if (gl >= 0) and (0 >= gh) then
    d := -1
else
    d := 0 ! writes(1,"You have made a z-mistake!") ! line(1,1);
if gl = 0 then vh := vl else
if gh = 0 then vl := vh else
while vh - vl > 1 do
    begin
    vc := (vl + vh)/2;
    g  := d*gz(vc);
    if g <= 0 then vl := vc;
    if g >= 0 then vh := vc
    end
end;
```

```
integer procedure lsqrt(n);             { lower square root }
integer n,x,y;
x := 2;  y := x*x;
while y < n do x :=: y ! y := x*x;
while y > n do x := y + n / 2*x ! y := x*x;
lsqrt := x
end;

integer procedure usqrt (n);            { ceiling(sqrt(v)) }
integer n,x,y;
x := 2;  y := x*x;
while y < n do x :=: y ! y := x*x;
while y > n do x := y + n / 2*x ! y := x*x;
if (x*x < n) then usqrt := x + 1
             else usqrt := x
end;

output(1,"AGB$OUTPUT");                 { *** start here *** }

for T2 := 6,...,21 do
    begin
    a6l := 1;
    a6u := (T2*T2*T2)/(6*6*6);
    for a6 := a6l,...,a6u do
        begin
        m := 2^14;  done := false;
        while not done do
            begin
            m := 2*m;  ct := m/4;  ca := m*m*m/2;  cb := m*m/4;
            u1l := 0;              v1l := ceilg(m*T2,6);
            u1h := floor(m*T2,6);  v1h := floor(m*T2,4);
            vsolve(u1l,u1h,v1l,v1h);
            u2l := ceilg(m*T2,6);  v2l := 0;
            u2h := floor(m*T2,2);  v2h := floor(m*T2,6);
            vsolve(u2l,u2h,v2l,v2h);
            r1l := lsqrt(m*u1l);  s1l := lsqrt(m*v1l);
            r2l := lsqrt(m*u2l);  s2l := lsqrt(m*v2l);
            T2l := 2*max(u1l + 2*v1l,
                         u2l + 2*v2l);
            T3l := 2*max(u1l*r1l + 2*v1l*s1l,
                         u2l*r2l + 2*v2l*s2l);
            T4l := 2*max(u1l*u1l + 2*v1l*v1l,
                         u2l*u2l + 2*v2l*v2l);
            T5l := 2*max(u1l*u1l*r1l + 2*v1l*v1l*s1l,
                         u2l*u2l*r2l + 2*v2l*v2l*s2l);
```

```
            T6l := 2*max(u1l*u1l*u1l + 2*v1l*v1l*v1l,
                         u2l*u2l*u2l + 2*v2l*v2l*v2l);
            Y2l := floor(T2l,m);
            Y3l := floor(T3l,m*m);
            Y4l := floor(T4l,m*m);
            Y5l := floor(T5l,m*m*m);
            Y6l := floor(T6l,m*m*m);
            r1h := usqrt(m*u1h);  s1h := usqrt(m*v1h);
            r2h := usqrt(m*u2h);  s2h := usqrt(m*v2h);
            T2h := 2*max(u1h + 2*v1h,
                         u2h + 2*v2h);
            T3h := 2*max(u1h*r1h + 2*v1h*s1h,
                         u2h*r2h + 2*v2h*s2h);
            T4h := 2*max(u1h*u1h + 2*v1h*v1h,
                         u2h*u2h + 2*v2h*v2h);
            T5h := 2*max(u1h*u1h*r1h + 2*v1h*v1h*s1h,
                         u2h*u2h*r2h + 2*v2h*v2h*s2h);
            T6h := 2*max(u1h*u1h*u1h + 2*v1h*v1h*v1h,
                         u2h*u2h*u2h + 2*v2h*v2h*v2h);
            Y2h := floor(T2h,m);
            Y3h := floor(T3h,m*m);
            Y4h := floor(T4h,m*m);
            Y5h := floor(T5h,m*m*m);
            Y6h := floor(T6h,m*m*m);
            done := Y3l = Y3h and
                    Y4l = Y4h and
                    Y5l = Y5h and
                    Y6l = Y6h
            end;
        writen(1,a6,3);
        writen(1,T2,5);
        writen(1,Y3h,5);
        writen(1,Y4h,5);
        writen(1,Y5h,5);
        writen(1,Y6h,5);
        line(1,1)
        end
    end;

close(1)

end
```

Output from trtbla.agb (ALGEB)

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
1	6	6	6	6	6
1	7	8	10	14	20
1	8	11	17	27	45
2	8	9	12	15	20
1	9	14	24	44	80
2	9	12	18	29	46
3	9	11	14	19	25
1	10	17	33	64	128
2	10	15	26	47	86
3	10	14	22	35	58
4	10	13	18	25	37
1	11	20	42	89	190
2	11	18	35	69	139
3	11	17	30	55	104
4	11	16	26	44	77
5	11	15	23	35	55
6	11	14	20	28	39
1	12	24	53	118	268
2	12	22	45	96	208
3	12	21	40	80	165
4	12	20	36	68	132
5	12	19	32	57	104
6	12	18	29	48	82
7	12	17	26	40	63
8	12	16	24	33	48
1	13	28	64	152	362
2	13	26	57	128	293
3	13	24	51	110	243
4	13	23	46	96	203
5	13	22	42	84	170
6	13	21	39	73	142
7	13	21	35	64	117
8	13	20	33	55	96
9	13	19	30	48	78
10	13	19	28	42	63
1	14	31	77	191	476
2	14	29	69	165	396
3	14	28	63	145	339
4	14	27	58	129	292
5	14	26	54	115	254

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
6	14	25	50	103	220
7	14	24	46	93	190
8	14	23	43	83	164
9	14	23	41	75	141
10	14	22	38	67	120
11	14	22	36	60	102
12	14	21	34	53	86
1	15	35	91	235	609
2	15	33	82	206	520
3	15	32	76	185	454
4	15	31	71	167	401
5	15	30	66	152	356
6	15	29	62	139	317
7	15	28	58	127	283
8	15	27	55	116	252
9	15	27	52	107	224
10	15	26	49	98	198
11	15	25	47	89	175
12	15	25	44	82	154
13	15	24	42	74	135
14	15	24	40	68	117
15	15	23	38	62	102
1	16	40	105	284	764
2	16	38	97	253	664
3	16	36	90	230	591
4	16	35	84	211	531
5	16	34	80	194	480
6	16	33	76	180	436
7	16	32	72	167	396
8	16	31	68	155	360
9	16	31	65	144	327
10	16	30	62	134	297
11	16	29	59	125	270
12	16	29	57	116	244
13	16	28	54	108	220
14	16	28	52	100	199
15	16	27	50	93	178
16	16	27	48	86	160
17	16	26	46	80	142
18	16	26	44	74	127

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
1	17	44	121	338	943
2	17	42	112	305	832
3	17	40	105	280	750
4	17	39	99	260	683
5	17	38	94	242	626
6	17	37	90	226	576
7	17	36	86	212	531
8	17	35	82	199	490
9	17	35	79	187	453
10	17	34	76	176	418
11	17	33	73	166	386
12	17	33	70	156	356
13	17	32	67	147	329
14	17	32	65	138	303
15	17	31	62	130	278
16	17	31	60	122	255
17	17	30	58	115	234
18	17	30	56	108	214
19	17	29	54	102	195
20	17	29	52	95	177
21	17	29	50	90	161
22	17	28	49	84	146
1	18	48	138	397	1146
2	18	46	129	362	1024
3	18	45	121	336	934
4	18	43	115	314	860
5	18	42	110	295	796
6	18	41	106	278	740
7	18	41	102	263	690
8	18	40	98	249	644
9	18	39	94	236	601
10	18	38	91	224	562
11	18	38	88	212	526
12	18	37	85	202	492
13	18	37	82	192	460
14	18	36	79	182	430
15	18	35	77	173	402
16	18	35	74	165	375
17	18	34	72	156	350
18	18	34	70	149	326

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
19	18	34	68	141	303
20	18	33	66	134	282
21	18	33	64	127	261
22	18	32	62	121	242
23	18	32	60	114	223
24	18	32	58	109	206
25	18	31	57	103	190
26	18	31	55	98	175
27	18	31	54	93	162
1	19	53	156	463	1375
2	19	51	146	425	1242
3	19	49	139	397	1143
4	19	48	133	374	1061
5	19	47	127	354	991
6	19	46	122	336	929
7	19	45	118	319	873
8	19	44	114	304	822
9	19	43	110	290	775
10	19	43	107	277	732
11	19	42	104	265	691
12	19	41	100	253	653
13	19	41	97	243	617
14	19	40	95	232	583
15	19	40	92	222	550
16	19	39	89	213	520
17	19	39	87	204	491
18	19	38	85	195	463
19	19	38	82	187	437
20	19	37	80	179	411
21	19	37	78	171	387
22	19	37	76	164	364
23	19	36	74	157	342
24	19	36	72	150	321
25	19	35	70	144	301
26	19	35	69	137	282
27	19	35	67	131	264
28	19	34	65	126	246
29	19	34	64	120	230
30	19	34	62	115	214
31	19	34	61	110	200

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
1	20	57	175	534	1632
2	20	55	165	494	1487
3	20	54	157	465	1379
4	20	53	151	440	1290
5	20	51	145	418	1213
6	20	50	140	399	1145
7	20	50	136	382	1084
8	20	49	132	366	1028
9	20	48	128	351	976
10	20	47	124	337	927
11	20	47	121	324	882
12	20	46	117	311	839
13	20	45	114	299	799
14	20	45	111	288	761
15	20	44	109	278	725
16	20	44	106	267	691
17	20	43	103	257	658
18	20	43	101	248	626
19	20	42	98	239	596
20	20	42	96	230	568
21	20	41	94	222	540
22	20	41	92	214	513
23	20	41	90	206	488
24	20	40	88	198	464
25	20	40	86	191	440
26	20	39	84	184	417
27	20	39	82	177	396
28	20	39	80	171	375
29	20	38	78	164	354
30	20	38	77	158	335
31	20	38	75	152	317
32	20	37	73	147	299
33	20	37	72	141	282
34	20	37	70	136	265
35	20	37	69	131	250
36	20	36	68	126	236
37	20	36	66	121	222
1	21	62	194	611	1919
2	21	60	184	569	1762
3	21	59	176	538	1645

a_6	Y_2	Y_3	Y_4	Y_5	Y_6
4	21	57	170	512	1548
5	21	56	164	489	1464
6	21	55	159	468	1390
7	21	54	154	450	1322
8	21	53	150	433	1261
9	21	53	146	417	1204
10	21	52	142	402	1151
11	21	51	139	388	1101
12	21	51	135	375	1054
13	21	50	132	362	1009
14	21	49	129	350	967
15	21	49	126	339	927
16	21	48	123	328	889
17	21	48	121	317	852
18	21	47	118	307	817
19	21	47	116	297	784
20	21	46	113	288	751
21	21	46	111	279	720
22	21	46	108	270	690
23	21	45	106	262	661
24	21	45	104	253	634
25	21	44	102	245	607
26	21	44	100	238	581
27	21	44	98	230	556
28	21	43	96	223	531
29	21	43	94	216	508
30	21	43	92	209	485
31	21	42	91	202	463
32	21	42	89	196	442
33	21	42	87	189	422
34	21	41	86	183	402
35	21	41	84	177	383
36	21	41	82	171	364
37	21	40	81	166	347
38	21	40	79	160	330
39	21	40	78	155	313
40	21	40	77	150	297
41	21	39	75	145	282
42	21	39	74	141	268

Appendix II — Generating the Polynomials

The PASCAL program `tcdeg6` generates the polynomials that define the fields to be tested for Galois group $\mathrm{PGL}_2(\mathbb{F}_5)$. Such a polynomial is constrained to have a root θ such that coefficient $a_1 = -\operatorname{Tr}(\theta) \in \{0, 1, 2, 3\}$ and $6 \le T_2(\theta) \le B_{a_1} \le 21$; thus the controlling variable $T = \lceil T_2(\theta) \rceil$ runs from 6 to 21.

The array `pr` is initialized to contain the primes p in the range $101 \le p \le 571$.

The array `Tmb` is initialized to contain the values of Y_3, Y_4, Y_5, Y_6 for all possible pairs (a_1, Y_2), with $Y_2 = T$ by definition.

The array `B` is initialized to contain upper bounds on T for $a_1 = 0, 1, 2, 3$.

Detailed descriptions of the procedures defined in this program are to be found in Appendix III.

```
program tcdeg6 (input, output);        { totally complex, degree 6 }

const a0  =   1;
      B0  =  19;
      B1  =  20;
      B2  =  20;
      B3  =  21;
      Tmn =   6;
      Tmx =  21;
      nb  = 239;

var cpu: integer;
    T: integer;
    B: array [0..3] of integer;
    Tmb: array [1..nb,1..6] of integer;
    a1, a1l, a1u, c1l, c1u: integer;
    a2, a2l, a2u, c2l, c2u: integer;
    a3, a3l, a3u, c3l, c3u: integer;
    a4, a4l, a4u, c4l, c4u: integer;
    a5, a5l, a5u, c5l, c5u: integer;
    a6, a6l, a6u, c6l, c6u: integer;
    X1, X2, X3, X4, X5, X6: integer;
    Y1, Y2, Y3, Y4, Y5, Y6: integer;
    Z1, Z2, Z3, Z4, Z5, Z6: integer;
    S1, S2, S3, S4, S5, S6: integer;
    old1, old2, old3, old4, old5, old6: boolean;
```

```
type flag = (val, rej);

var c02: array [0..02-1, 0..02-1, 0..02-1, flag] of boolean;
    f02: boolean;
    c03: array [0..03-1, 0..03-1, 0..03-1, flag] of boolean;
    f03: boolean;
    c05: array [0..05-1, 0..05-1, 0..05-1, flag] of boolean;
    f05: boolean;
    c07: array [0..07-1, 0..07-1, 0..07-1, flag] of boolean;
    f07: boolean;
    c11: array [0..11-1, 0..11-1, 0..11-1, flag] of boolean;
    f11: boolean;
    c13: array [0..13-1, 0..13-1, 0..13-1, flag] of boolean;
    f13: boolean;
    c17: array [0..17-1, 0..17-1, 0..17-1, flag] of boolean;
    f17: boolean;
    c19: array [0..19-1, 0..19-1, 0..19-1, flag] of boolean;
    f19: boolean;
    c23: array [0..23-1, 0..23-1, 0..23-1, flag] of boolean;
    f23: boolean;
    c29: array [0..29-1, 0..29-1, 0..29-1, flag] of boolean;
    f29: boolean;
    c31: array [0..31-1, 0..31-1, 0..31-1, flag] of boolean;
    f31: boolean;
    c37: array [0..37-1, 0..37-1, 0..37-1, flag] of boolean;
    f37: boolean;
    c41: array [0..41-1, 0..41-1, 0..41-1, flag] of boolean;
    f41: boolean;
    c43: array [0..43-1, 0..43-1, 0..43-1, flag] of boolean;
    f43: boolean;
    c47: array [0..47-1, 0..47-1, 0..47-1, flag] of boolean;
    f47: boolean;
    c53: array [0..53-1, 0..53-1, 0..53-1, flag] of boolean;
    f53: boolean;
    c59: array [0..59-1, 0..59-1, 0..59-1, flag] of boolean;
    f59: boolean;
    c61: array [0..61-1, 0..61-1, 0..61-1, flag] of boolean;
    f61: boolean;
    c67: array [0..67-1, 0..67-1, 0..67-1, flag] of boolean;
    f67: boolean;
    c71: array [0..71-1, 0..71-1, 0..71-1, flag] of boolean;
    f71: boolean;
```

```
type poly = array [0..6]  of integer;


function deg (var f: poly): integer;
var k: integer;
begin
k := 6;
while (k > 0) and (f[k] = 0) do
    k := k - 1;
deg := k
end;


procedure pswap (var f, g: poly);                  { f <--> g }
var h: poly;
begin
h := f;
f := g;
g := h
end;


procedure prdif (var g, h: poly; p: integer);     { g <-- dh/dx }
const n = 6;
var j: integer;
begin
for j := 1 to n do
    g[j-1] := (j*h[j]) mod p;
g[n] := 0
end;
```

```
procedure prmdr (var f, g: poly; p: integer);
const n = 6;                                    { f <-- rem(f,g), g <> 0 }
var j, s, t, cf, cg: integer;
begin
t := deg(g);  cg := g[t];
for s := n downto t do
if f[s] <> 0 then
    begin
    cf := f[s];
    if cg <> 1 then
    for j := 0 to s do
        f[j] := (cg*f[j]) mod p;
    for j := s-t to s do
        f[j] := (f[j] - cf*g[j-s+t]) mod p
    end
end;


procedure prgcd (var f, g: poly; p: integer);   { f <-- gcd(f,g) }
begin
while deg(g) <> 0 do
    begin
    prmdr(f,g,p);
    pswap(f,g)
    end;
if g[0] <> 0 then
    pswap(f,g)
end;
```

```
procedure prprd (var w, z, f: poly; p: integer);
var j, k: integer;                              { w <-- rem (w*z, f) }
    v: poly;
begin
for j := 0 to 6 do                { v <-- 0 }
    v[j] := 0;
for k := 6 downto 0 do            { v <-- (v.x + z_k.w) mod f }
    begin
    for j := 6 downto 1 do
        v[j] := (v[j-1] + z[k]*w[j]) mod p;
    v[0] := (z[k]*w[0]) mod p;
    prmdr(v,f,p)
    end;
for j := 0 to 6 do
    w[j] := v[j]
end;


procedure prpwr (var h, f: poly; q, p: integer);
var e, r, j: integer;                      { h <-- rem (x^q - x, f) }
    y, z: poly;
begin
for j := 0 to 6 do
    begin
    y[j] := 0;
    z[j] := 0
    end;
y[0] := 1;  z[1] := 1;            { y = 1, z = x }
e := q;
while e <> 0 do
    begin
    r := e mod 2;
    e := e div 2;
    if r <> 0 then
        prprd(y,z,f,p);
    if e <> 0 then
        prprd(z,z,f,p)
    end;
y[1] := y[1] - 1;
for j := 0 to 6 do
    h[j] := y[j] mod p
end;
```

```
function s5cyc (p: integer): boolean;
var k1, k2: integer;
    f, h0, h1, h2: poly;
begin
f[6] :=  1 mod p;
f[5] := a1 mod p;
f[4] := a2 mod p;
f[3] := a3 mod p;
f[2] := a4 mod p;
f[1] := a5 mod p;
f[0] := a6 mod p;
h0 := f;
prdif(h1,f,p);
prgcd(h1,h0,p);
if deg(h1) > 0 then
    s5cyc := true                { disc(f) = 0 mod p }
else
    begin
    prpwr(h0,f,p,p);             { h0 <-- rem (x^p - x, f) }
    h1 := f;  prgcd(h1,h0,p);  k1 := deg(h1);
    if k1 = 6 then
        s5cyc := true            { 111111 }
    else
    if k1 = 4 then
        s5cyc := false           { 11112 }
    else
    if k1 = 3 then
        s5cyc := false           { 1113 }
    else
    if k1 = 2 then
        s5cyc := true            { 1122, 114 }
    else
    if k1 < 2 then
        begin
        prpwr(h0,f,p*p,p);       { h0 <-- rem (x^p^2 - x, f) }
        h2 := f;  prgcd(h2,h0,p);  k2 := deg(h2) div 2;
        if k2 = 1 then           { deg(h2) = k1 + 2 k2 }
            s5cyc := false       { 123, 24 }
        else
            s5cyc := true        { 15, 222, 33, 6 }
        end
    end
end;
```

```
procedure csinit;
begin
f02 := false;  f13 := false;  f31 := false;  f53 := false;
f03 := false;  f17 := false;  f37 := false;  f59 := false;
f05 := false;  f19 := false;  f41 := false;  f61 := false;
f07 := false;  f23 := false;  f43 := false;  f67 := false;
f11 := false;  f29 := false;  f47 := false;  f71 := false;
end;

function c02srch: boolean;
const p = 02;
var k3, k4, k5: integer;
begin
if not f02 then
    begin
    for k3 := 0 to p-1 do
    for k4 := 0 to p-1 do
    for k5 := 0 to p-1 do
        c02[k3,k4,k5,val] := false;
    f02 := true
    end;
k3 := a3 mod p;
k4 := a4 mod p;
k5 := a5 mod p;
if not c02[k3,k4,k5,val] then
    begin
    c02[k3,k4,k5,rej] := not s5cyc(p);
    c02[k3,k4,k5,val] := true
    end;
c02srch := c02[k3,k4,k5,rej]
end;

%include 'CS19PV.INC' { functions c03srch ... c71srch similarly }
```

```
function csrch: boolean;
{  csrch true   <==>  for some p, f not in S5cyc(p)  }
{               <==>  for some p, cpsrch true        }
{  csrch false  <==>  for all p, cpsrch false        }
var cs: boolean;
begin
cs := true;
if not c02srch then  if not c03srch then  if not c05srch then
if not c07srch then  if not c11srch then  if not c13srch then
if not c17srch then  if not c19srch then  if not c23srch then
if not c29srch then  if not c31srch then  if not c37srch then
if not c41srch then  if not c43srch then  if not c47srch then
if not c53srch then  if not c59srch then  if not c61srch then
if not c67srch then  if not c71srch then
    cs := false;
csrch := cs
end;

const pn = 80;

var pr: array [1..pn] of integer;

procedure prfill;
begin
pr[01] := 571;  pr[21] := 439;  pr[41] := 313;  pr[61] := 197;
pr[02] := 569;  pr[22] := 433;  pr[42] := 311;  pr[62] := 193;
pr[03] := 563;  pr[23] := 431;  pr[43] := 307;  pr[63] := 191;
pr[04] := 557;  pr[24] := 421;  pr[44] := 293;  pr[64] := 181;
pr[05] := 547;  pr[25] := 419;  pr[45] := 283;  pr[65] := 179;
pr[06] := 541;  pr[26] := 409;  pr[46] := 281;  pr[66] := 173;
pr[07] := 523;  pr[27] := 401;  pr[47] := 277;  pr[67] := 167;
pr[08] := 521;  pr[28] := 397;  pr[48] := 271;  pr[68] := 163;
pr[09] := 509;  pr[29] := 389;  pr[49] := 269;  pr[69] := 157;
pr[10] := 503;  pr[30] := 383;  pr[50] := 263;  pr[70] := 151;
pr[11] := 499;  pr[31] := 379;  pr[51] := 257;  pr[71] := 149;
pr[12] := 491;  pr[32] := 373;  pr[52] := 251;  pr[72] := 139;
pr[13] := 487;  pr[33] := 367;  pr[53] := 241;  pr[73] := 137;
pr[14] := 479;  pr[34] := 359;  pr[54] := 239;  pr[74] := 131;
pr[15] := 467;  pr[35] := 353;  pr[55] := 233;  pr[75] := 127;
pr[16] := 463;  pr[36] := 349;  pr[56] := 229;  pr[76] := 113;
pr[17] := 461;  pr[37] := 347;  pr[57] := 227;  pr[77] := 109;
pr[18] := 457;  pr[38] := 337;  pr[58] := 223;  pr[78] := 107;
pr[19] := 449;  pr[39] := 331;  pr[59] := 211;  pr[79] := 103;
pr[20] := 443;  pr[40] := 317;  pr[60] := 199;  pr[80] := 101;
end;
```

```
function s5type: boolean;
var k: integer;
    cvalid: boolean;
begin
cvalid := true;
for k := 1 to pn do
if cvalid then
    cvalid := s5cyc(pr[k]);
s5type := cvalid
end;


procedure lib$init_timer;
external;

procedure lib$stat_timer (code: integer; var value: integer);
external;


function lbound (u, v: integer): integer;
{  lower bound = u/v; truncate rightwards ("ceiling")   }
begin
if (v < 0) then
    begin  u := -u;   v := -v  end;
if (u < 0) then
    lbound := u div v
else
    lbound := (u + v - 1) div v
end;

function ubound (u, v: integer): integer;
{  upper bound = u/v; truncate leftwards ("floor")  }
begin
if (v < 0) then
    begin  u := -u;   v := -v  end;
if (u > 0) then
    ubound := u div v
else
    ubound := (u - v + 1) div v
end;
```

```
function lsqrt (v: integer): integer;    { floor(sqrt(v)) }
var r, b: integer;
begin
r := 46340;
while r*r > v do
    begin
    b := r mod 2;
    r := ((r - b) div 2) + ((b*r + v) div (2*r))
    end;
lsqrt := r
end;

function usqrt (n: integer): integer;    { ceiling(sqrt(v)) }
var u: integer;
begin
u := lsqrt(n);
if (u*u < n) then
    usqrt := u + 1
else
    usqrt := u
end;

function square (n: integer): boolean;  { is n square? }
var r: integer;
begin
if n < 0 then
    square := false
else
    begin  r := lsqrt(n);  square := (n = r*r)  end
end;

function gcd (x, y: integer): integer;
var z: integer;
begin
y := abs(y);
while y <> 0 do
    begin  z := x mod y;  x := y;  y := z  end;
gcd := abs(x)
end;
```

```
function sign (x: integer): integer;
begin
if x < 0 then sign := -1
else
if x > 0 then sign := +1
else
                sign :=  0
end;


function root (x: integer; var h: poly; n: integer): boolean;
var k, v: integer;
    r: boolean;
begin
if x = 0 then
    root := (h[0] = 0)
else
    begin
    v := 0;  k := 0;  r := true;
    repeat
        if 0 = (v - h[k]) mod abs(x) then
            v := (v - h[k]) div x
        else
            r := false;
        k := k + 1
    until (k = n) or (not r);
    if r then
        root := (h[n] = v)
    else
        root := false
    end
end;
```

```
procedure a2bd (V2: integer; var b2l, b2u: integer);
begin
b2l := lbound(-X2-V2,2);
b2u := ubound(-2*X2+3*V2,6)
end;

procedure a6bd (V2: integer; var b6l, b6u: integer);
begin
b6l := 1;
b6u := ubound(V2*V2*V2,216)
end;

procedure a3bd (V3: integer; var b3l, b3u: integer);
begin
b3l := lbound(-X3-V3,3);
b3u := ubound(-X3+V3,3)
end;

procedure a4bd (V4: integer; var b4l, b4u: integer);
begin
b4l := lbound(-X4-Y4,4);
b4u := ubound(-X4+Y4,4)
end;

procedure a5bd (V5, V6: integer; var b5l, b5u: integer);
var U6: integer;
begin
b5l := lbound(-X5-V5,5);
b5u := ubound(-X5+V5,5);
if a1 <> 0 then
    begin
    U6  := -a1*X5 + a2*S4 + a3*S3 + a4*S2 + 6*a6;
    b5l := max(b5l,lbound(U6-V6,6*a1));
    b5u := min(b5u,ubound(U6+V6,6*a1))
    end
end;
```

```
procedure tmfill;
var j, k: integer;
begin
for j := 1 to nb do
for k := 1 to 6 do
    read(Tmb[j,k])
end;

procedure inittm (var W3, W4, W5, W6: integer; V, a: integer);
var j, k: integer;
begin
for j := 1 to nb do
if (abs(a) = Tmb[j,1]) and (V = Tmb[j,2]) then
    begin
    W3 := Tmb[j,3];
    W4 := Tmb[j,4];
    W5 := Tmb[j,5];
    W6 := Tmb[j,6]
    end
end;


function cubesf (var f: poly): boolean; { f = g(h), deg(g) = 3 }
begin
cubesf :=
    (27*f[3] = f[5]*(18*f[4] - 5*f[5]**2))
    and
    (81*f[1] = f[5]*(27*f[2] - 3*f[5]**2*f[4] + f[5]**4))
end;

function quadsf (var f: poly): boolean; { f = g(h), deg(g) = 2 }
begin
quadsf :=
    (64*f[2]
        = 32*f[5]*f[3] - 24*f[5]**2*f[4] + 5*f[5]**4 + 16*f[4]**2)
    and
    (64*f[1]
        = (8*f[3] - 4*f[5]*f[4] + f[5]**3)*(4*f[4] - f[5]**2))
end;
```

```
procedure normal (var h: poly; n: integer);
var j, k: integer;             { remove factors of x from h }
begin
for k := n downto 1 do
if h[0] = 0 then
    begin
    for j := 0 to k-1 do h[j] := h[j+1];
    h[k] := 0
    end
end;


function linfac (var f: poly): boolean;
var lfac: boolean;             { does f have a linear factor? }

        procedure rtestl (v: integer);
        begin
        if root(v,f,6) then
            lfac := true
        end;

        procedure dtestl;
        var m, h, x, y: integer;
        begin
        h := abs(f[0]);
        m := lsqrt(h);
        for x := 1 to m do
        if not lfac then
        if 0 = h mod x then
            begin
            rtestl(x);  rtestl(-x);  y := h div x;
            if y <> x then
                begin  rtestl(y);  rtestl(-y)  end
            end
        end;

begin
lfac := (f[0] = 0);
if not lfac then
    dtestl;
linfac := lfac
end;
```

```
function quafac (var f: poly): boolean;
var h0: poly;   { deg 5 }         { does f have a quadratic factor? }
    h1: poly;   { deg 4 }
    qfac: boolean;

        procedure rtestq (v: integer);
        begin
        if root(v,h1,4) then
        if root(v,h0,5) then
            qfac := true
        end;

        procedure qtest (r: integer);
        var d, j, m, g, x, y: integer;
        { If g(x) = x^2 + q*x + r and f[0] = a6 = r*d }
        { then h0(q)*x + h1(q)*r = rem(f,g,x)          }
        begin
        d := f[0] div r;
        h1[6] :=                          0;      { q^6 }
        h1[5] :=                          0;      { q^5 }
        h1[4] :=                         -1;      { q^4 }
        h1[3] :=                       f[5];      { q^3 }
        h1[2] :=                -f[4] + 3*r;      { q^2 }
        h1[1] :=           -2*r*f[5] + f[3];      { q^1 }
        h1[0] := -r*r - f[2] + d + r*f[4];        { q^0 }
        h0[6] :=                          0;      { q^6 }
        h0[5] :=                         -1;      { q^5 }
        h0[4] :=                       f[5];      { q^4 }
        h0[3] :=                -f[4] + 4*r;      { q^3 }
        h0[2] :=            f[3] - 3*r*f[5];      { q^2 }
        h0[1] := -3*r*r - f[2] + 2*r*f[4];        { q^1 }
        h0[0] := f[1] + r*r*f[5] - r*f[3];        { q^0 }
```

```
        if (h0[0] = 0) and (h1[0] = 0) then
            qfac := true                    { q = 0 is a solution }
        else
            begin
            normal(h0,5);
            normal(h1,4);
            g := gcd(h0[0],h1[0]);
            m := lsqrt(g);
            for x := 1 to m do
            if not qfac then
            if 0 = g mod x then
                begin
                rtestq(x);  rtestq(-x);  y := g div x;
                if y <> x then
                    begin  rtestq(y);  rtestq(-y)   end
                end
            end
        end;

        procedure dtestq;
        var m, h, x, y: integer;
        begin
        h := abs(f[0]);
        m := lsqrt(h);
        for x := 1 to m do
        if not qfac then
        if 0 = h mod x then
            begin
            qtest(x);   qtest(-x);  y := h div x;
            if y <> x then
                begin  qtest(y);  qtest(-y)   end
            end
        end;

begin
qfac := false;
dtestq;
quafac := qfac
end;
```

```
function cubfac (var f: poly): boolean;
var p, q, r, d: integer;        { does f have a cubic factor? }
    h1, h3: poly;
    cfac: boolean;

        procedure rtestc (v: integer);
        var b, c, u, w, x1, y1, x3, y3: integer;
        begin
        if not cfac then
        if root(v,h1,3) then
        if root(v,h3,4) then
            begin
            p  := v;
            w  := (p - f[5])*p + f[4];
            x1 := w*r - f[1];
            y1 := r - d;
            if y1 <> 0 then               { solve E1: q*y1 = x1 }
               begin
               q := x1 div y1;
               if 0 = x1 mod abs(y1) then           { y1 | x1 }
               if f[2] = (d - r)*p + (w - q)*q + r*f[5] { E2 }
               then
               if f[3] = (w - 2*q)*p + q*f[5] + r + d   { E3 }
               then
                   cfac := true
               end
            else                          { r = d }
            if x1 = 0 then                { rhs(E1) = 0 }
               begin
               x3 := w*p - f[3] + r + d;
               y3 := 2*p - f[5];
               if y3 <> 0 then            { solve E3: q*y3 = x3 }
                   begin
                   q := x3 div y3;
                   if 0 = x3 mod abs(y3) then       { y3 | x3 }
                   if f[2] = (w - q)*q + r*f[5] then    { E2 }
                       cfac := true
                   end
               else                       { f[5] = 2*p }
               if x3 = 0 then             { rhs(E3) = 0 }
               if square (w*w - 4*(f[2] - r*f[5])) then
                   cfac := true
               end
            end
        end;
```

```
procedure ctest (v: integer);
var j, k, m, g, x, y: integer;
begin
r := v;
d := f[0] div r;
h1[6]  := 0;
h1[5]  := 0;
h1[4]  := 0;
h1[3]  := d + r;
h1[2]  := -(d + 2*r)*f[5];
h1[1]  := (d + r)*f[4] + r*f[5]*f[5] - 2*f[1];
h1[0]  := (f[1] - r*f[4])*f[5] + (d - r)*(d + r - f[3]);
h3[6]  := 0;
h3[5]  := 0;
h3[4]  := d*r;
h3[3]  := -2*d*r*f[5];
h3[2]  := d*r*(f[5]*f[5] + 2*f[4]) - (r + d)*f[1];
h3[1]  := ((r + d)*f[1] - 2*d*r*f[4])*f[5] + (r - d)**3;
h3[0]  := (f[2] - r*f[5])*(r - d)**2
                         + (f[1] - r*f[4])*(f[1] - d*f[4]);
if (h1[0] = 0) and (h3[0] = 0) then
    rtestc(0);
if not cfac then
    begin
    normal(h1,3);  normal(h3,4);
    g := gcd(h1[0],h3[0]);  m := lsqrt(g);
    for x := 1 to m do
    if not cfac then
    if 0 = g mod x then
        begin
        rtestc(x);  rtestc(-x);  y := g div x;
        if y <> x then
            begin  rtestc(y);  rtestc(-y)  end
        end
    end
end;
```

```
        procedure dtestc;
        var m, h, x: integer;
        begin
        h := abs(f[0]);
        m := lsqrt(h);
        for x := 1 to m do
        if not cfac then
        if 0 = h mod x then
           begin ctest(x);  ctest(-x)  end
        end;

begin
cfac := false;
dtestc;
cubfac := cfac
end;


procedure irtest;
var f: poly;
begin
f[0] := a6;
f[1] := a5;
f[2] := a4;
f[3] := a3;
f[4] := a2;
f[5] := a1;
f[6] := a0;
if not quadsf(f) then
if not cubesf(f) then
if not linfac(f) then
if not quafac(f) then
if not cubfac(f) then
if s5type then
    begin
    write(a1:4);  write(' ');
    write(a2:4);  write(' ');
    write(a3:4);  write(' ');
    write(a4:4);  write(' ');
    write(a5:4);  write(' ');
    write(a6:4);  writeln
    end
end;
```

```
procedure a5test;
begin
S5 := -X5 - 5*a5;
X6 := a1*S5 + a2*S4 + a3*S3 + a4*S2 + a5*S1;
S6 := -X6 - 6*a6;
if old4 then
    old5 := ((c5l <= a5) and (a5 <= c5u)
                              and (abs(S6) <= Z6))
else
    old5 := false;
if not old5 then
if abs(S6) <= Y6 then
if not csrch then
    irtest
end;

procedure a4test;
begin
if old3 then
    old4 := ((c4l <= a4) and (a4 <= c4u))
else
    old4 := false;
S4 := -X4 - 4*a4;
X5 := a1*S4 + a2*S3 + a3*S2 + a4*S1;
a5bd(Y5,Y6,a5l,a5u);
if old4 then
    a5bd(Z5,Z6,c5l,c5u);
for a5 := a5l to a5u do
    a5test
end;

procedure a3test;
begin
if old6 then
    old3 := ((c3l <= a3) and (a3 <= c3u))
else
    old3 := false;
S3 := -X3 - 3*a3;
X4 := a1*S3 + a2*S2 + a3*S1;
a4bd(Y4,a4l,a4u);
if old3 then
    a4bd(Z4,c4l,c4u);
for a4 := a4l to a4u do
    a4test
end;
```

```
procedure a6test;
begin
inittm(Z3,Z4,Z5,Z6,T-1,a6);
inittm(Y3,Y4,Y5,Y6,T,a6);
if old2 then
    old6 := ((c6l <= a6) and (a6 <= c6u))
else
    old6 := false;
csinit;         { requires a1, a2, a6 }
S2 := -X2 - 2*a2;
X3 := a1*S2 + a2*S1;
a3bd(Y3,a3l,a3u);
if old6 then
    a3bd(Z3,c3l,c3u);
for a3 := a3l to a3u do
    a3test
end;

procedure a2test;
begin
if old1 then
    old2 := ((c2l <= a2) and (a2 <= c2u))
else
    old2 := false;
a6bd(Y2,a6l,a6u);
if old2 then
    a6bd(Z2,c6l,c6u);
for a6 := a6l to a6u do
if a6 <> 0 then
    a6test
end;
```

```
procedure a1test;
begin
lib$init_timer;
old1 := (T > 6);
S1   := -X1 - 1*a1;
X2   := a1*S1;
a2bd(Y2,a2l,a2u);
if old1 then
    a2bd(Z2,c2l,c2u);
for a2 := a2l to a2u do
    a2test;
lib$stat_timer(2,cpu);
writeln(T:3,'::',a1:1,'::',cpu:15)
end;

procedure Tvtest;
begin
X1  :=      0;
Y2  :=      T;
Z2  := T - 1;
a1l :=      0;
a1u :=      3;
for a1 := a1l to a1u do
if T <= B[a1] then
    a1test
end;


begin                                { *** start here *** }

prfill;

tmfill;

B[0] := B0;  B[1] := B1;  B[2] := B2;  B[3] := B3;

for T := Tmn to Tmx do               { T = ceiling(T2) }
    Tvtest

end.
```

APPENDIX III — ROUTINES DEFINED IN `tcdeg6.pas`

```
function deg (var f: poly): integer;
```

Returns the degree of the polynomial f.

```
procedure prmdr (var f, g: poly; p: integer);
```

Replaces the polynomial f by its remainder upon division by the polynomial g, modulo the prime number p (possibly multiplied by a unit modulo p).

```
procedure pswap (var f, g: poly);
```

Exchanges the polynomials f and g.

```
procedure prdif (var g, h: poly; p: integer);
```

Assigns to h the derivative of the (univariate) polynomial g, reduced modulo the prime number p.

```
procedure prgcd (var f, g: poly; p: integer);
```

Given polynomials f and g and a prime number p, replaces the polynomial f by $\gcd(f, g) \bmod p$, possibly multiplied by a unit modulo p.

```
procedure prprd (var w, z, f: poly; p: integer);
```

Given polynomials w, z and f and a prime number p, replaces the polynomial w by the remainder of $w \cdot z$ upon division by the polynomial f, reduced modulo the prime number p.

`procedure prpwr (var h, f: poly; q, p: integer);`

Given the (monic) polynomial f, the positive integer q and the prime number p, assigns to h the remainder of $x^q - x$ upon division by $f(x)$, reduced modulo p.

`function s5cyc (p: integer): boolean;`

Given the prime p and defining $f(t) = t^6 + a_1t^5 + a_2t^4 + a_3t^3 + a_4t^2 + a_5t + a_6$ from the global variables $a_1, \ldots, a_6$, s5cyc returns the value TRUE if p divides the discriminant of f or if the degree sequence of the mod p factors of f corresponds to one of the cycle types that occur in $\mathrm{PGL}_2(\mathbb{F}_5)$, namely, 1·1·1·1·1·1, 1·1·2·2, 1·1·4, 1·5, 2·2·2, 3·3, 6; otherwise the degree sequence of the mod p factors of f is one of 1·1·1·1·2, 1·1·1·3, 1·2·3, 2·4, which do not occur in $\mathrm{PGL}_2(\mathbb{F}_5)$, and s5cyc returns the value FALSE. (If the Galois group of f is $\mathrm{PGL}_2(\mathbb{F}_5)$ then s5cyc returns the value TRUE, but not conversely.)

`procedure csinit;`

For each of the twenty primes $p = 2, \ldots, 71$, the flag I_p (denoted $\mathtt{f}_p$) is initialized to FALSE.

`function cpsrch: boolean;`

The number p is a prime between 2 and 71; the arrays V_p and W_p are boolean arrays of size $p \times p \times p$; the subscripts r_3, r_4, r_5 are the residues of a_3, a_4, a_5 mod p. If the flag I_p is FALSE then the array V_p has not been initialized; the entries $V_p[k_3, k_4, k_5]$ (denoted $\mathtt{c}_p\mathtt{[}k_3\mathtt{,}k_4\mathtt{,}k_5\mathtt{,val]}$) are initialized to FALSE and I_p becomes TRUE. If $V_p[r_3, r_4, r_5]$ is FALSE then $W_p[r_3, r_4, r_5]$ (denoted $\mathtt{c}_p\mathtt{[}r_3\mathtt{,}r_4\mathtt{,}r_5\mathtt{,rej]}$)

has not been computed; its value is determined to be $\neg$ `s5cyc(`p`)` and $V_p[r_3, r_4, r_5]$ becomes TRUE. The value returned is $W_p[r_3, r_4, r_5]$.

`function csrch: boolean;`

Returns the value TRUE if `s5cyc(`p`)` is FALSE for at least one of the primes $p = 2, \ldots, 71$; returns the value FALSE if `s5cyc(`p`)` is TRUE for all these primes. (If the Galois group of f is $\mathrm{PGL}_2(\mathbb{F}_5)$ then `csrch` returns the value FALSE, but not conversely.)

`procedure prfill;`

The integer array `pr[1..80]` is initialized to contain the eighty primes p in the range $101 \leq p \leq 571$.

`function s5type: boolean;`

Tests `s5cyc(`p`)` for the primes p in the range $101 \leq p \leq 571$. If the Galois group of $f(t) = t^6 + a_1t^5 + a_2t^4 + a_3t^3 + a_4t^2 + a_5t + a_6$ is $\mathrm{PGL}_2(\mathbb{F}_5)$ then `s5type` returns the value TRUE (but not conversely).

`procedure lib$init_timer;`

Initializes the system clock.

`procedure lib$stat_timer (code: integer; var value: integer);`

When code = 2, returns in cpu the elapsed CPU time for the process, expressed in hundredths of a second.

```
function lbound (u, v: integer): integer;
```

Assuming $\frac{u}{v}$ is a lower bound for a set of integers, returns $\left\lceil \frac{u}{v} \right\rceil$ (which must be a lower bound for the same set of integers).

```
function ubound (u, v: integer): integer;
```

Assuming $\frac{u}{v}$ is an upper bound for a set of integers, returns $\left\lfloor \frac{u}{v} \right\rfloor$ (which must be an upper bound for the same set of integers).

```
function lsqrt (v: integer): integer;
```

Given the non-negative integer v, returns $\lfloor \sqrt{v} \rfloor$.

```
function usqrt (w: integer): integer;
```

Given the non-negative integer w, returns $\lceil \sqrt{w} \rceil$.

```
function square (n: integer): boolean;
```

Returns TRUE if the integer n is a perfect square, otherwise returns FALSE.

```
function gcd (x, y: integer): integer;
```

Returns the greatest common divisor of the integers x and y.

```
function sign (x: integer): integer;
```

Returns the sign (-1 if $x < 0$, 0 if $x = 0$, $+1$ if $x > 0$) of the integer x.

```
function root (x: integer; var h: poly; n: integer): boolean;
```

Given the polynomial h of degree n and the integer value x, returns TRUE if x is a root of h, and returns FALSE otherwise.

```
procedure a2bd (V2: integer; var b2l, b2u: integer);
```

Given the value V_2 for $\lceil T_2 \rceil$, returns lower and upper bounds b_{2l} and b_{2u} for a_2, as explained in Chapter Three.

```
procedure a6bd (V2: integer; var b6l, b6u: integer);
```

Given the value V_2 for $\lceil T_2 \rceil$, returns lower and upper bounds b_{6l} and b_{6u} for a_6, as explained in Chapter Three.

```
procedure a3bd (V3: integer; var b3l, b3u: integer);
```

Given the value V_3 for Y_3, returns lower and upper bounds b_{3l} and b_{3u} for a_3, as explained in Chapter Three.

```
procedure a4bd (V4: integer; var b4l, b4u: integer);
```

Given the value V_4 for Y_4, returns lower and upper bounds b_{4l} and b_{4u} for a_4, as explained in Chapter Three.

```
procedure a5bd (V5, V6: integer; var b5l, b5u: integer);
```

Given the values V_5 and V_6 for Y_5 and Y_6, returns lower and upper bounds b_{5l} and b_{5u} for a_5, as explained in Chapter Three.

```
procedure tmfill;
```

For the 239 possible combinations of a_6 and Y_2, reads values for a_6, Y_2, Y_3, Y_4, Y_5, Y_6 from the standard input and stores them in the array T_{mb} (see Chapter Three).

```
procedure inittm (var W3, W4, W5, W6: integer; V, a: integer);
```

Given $Y_2 = V$ and $a_6 = a$, assigns to W_3, W_4, W_5, W_6 the values for Y_3, Y_4, Y_5, Y_6 (see Chapter Three).

`function cubesf (var f: poly): boolean;`

Returns TRUE if the polynomial $f(t)$ can be written as $g(h(t))$, with $\deg g = 3$ and $\deg h = 2$; otherwise returns FALSE. (When this condition is true, h defines a field element having cubic minimal polynomial over $\mathbb{Q}$.)

`function quadsf (var f: poly): boolean;`

Returns TRUE if the polynomial $f(t)$ can be written as $g(h(t))$, with $\deg g = 2$ and $\deg h = 3$; otherwise returns FALSE. (When this condition is true, h defines a field element having quadratic minimal polynomial over $\mathbb{Q}$.)

`procedure normal (var h: poly; n: integer);`

Removes factors of x from $h(x)$, so that $h(0) \neq 0$.

`function linfac (var f: poly): boolean;`

Returns TRUE if $f(x) = x^6 + a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$ has a linear factor, FALSE otherwise.

> `procedure rtestl (v: integer);`
>
> Gives the value TRUE to `lfac` if the integer v is a root of the polynomial f; leaves `lfac` unchanged otherwise.
>
> `procedure dtestl;`
>
> Assuming $a_6 \neq 0$, gives the value TRUE to `lfac` if there is an integer x (necessarily dividing a_6) such that $f(x) = 0$.

`function quafac (var f: poly): boolean;`

Assumes $f(x) = x^6 + a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$, with $a_6 \neq 0$. Returns TRUE if f has a quadratic factor, FALSE otherwise.

`procedure rtestq (v: integer);`

Gives `qfac` the value TRUE if $h_0(v) = h_1(v) = 0$; h_0 and h_1 are defined below.

`procedure qtest (r: integer);`

Assumes $r \mid a_6 \neq 0$. If $f(x)$ has a quadratic factor $g(x) = x^2 + qx + r$ for some integer q then `qfac` is given the value TRUE, as follows. The remainder of $f(x)$ upon division by $g(x)$ is $h_0(q)x + h_1(q)r$, where

$$\begin{aligned} h_0(q) = -q^5 + a_1q^4 - (a_2 - 4r)q^3 + (a_3 - 3ra_1)q^2 \\ - (3r^2 + a_4 - 2ra_2)q + a_5 + r^2a_1 - ra_3, \\ h_1(q) = -q^4 + a_1q^3 - (a_2 - 3r)q^2 - (2ra_1 - a_3)q - r^2 - a_4 + d + ra_2. \end{aligned}$$

If $h_0(0) = h_1(0) = 0$ then $x^2 + r$ is a factor of $f(x)$, and `qfac` becomes TRUE. Otherwise $h_0(x)$ and $h_1(x)$ are normalized so that $h_0(0) \neq 0 \neq h_1(0)$. If $h_0(v) = h_1(v) = 0$ for some $v \mid \gcd(h_0(0), h_1(0))$ then $x^2 + vx + r$ is a factor of $f(x)$, and `qfac` becomes TRUE.

`procedure dtestq;`

For each integer $r \mid a_6 \neq 0$, gives the value TRUE to `qfac` if there exists an integer q such that $x^2 + qx + r$ is a factor of $f(x)$.

```
function cubfac (var f: poly): boolean;
```

Assumes $f(x) = x^6 + a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$, with $a_6 \neq 0$. Returns TRUE if f has a cubic factor, FALSE otherwise.

```
procedure rtestc (v: integer);
```

Assumes r is a fixed integer value, with $r \mid a_6 \neq 0$. If $f(x)$ has a cubic factor $g(x) = x^3 + px^2 + qx + r$, with $p = v$, for some integer q, then `cfac` is given the value TRUE, as follows.

Taking $k_0x^2 + k_1x + k_2$ to be the remainder of $f(x)$ upon division by $g(x)$, define

$$E_1 = \frac{q}{r}k_2 - k_1, \quad E_2 = \frac{p}{r}k_2 - k_0, \quad E_3 = \frac{1}{r}k_2.$$

Note that $k_0 = k_1 = k_2 = 0 \iff E_1 = E_2 = E_3 = 0$. Defining

$$\begin{aligned} x_1 &= wr - a_5, & y_1 &= r - d, \\ x_2 &= wq - a_4 - (p - a_1)r + dp, & y_2 &= q, \\ x_3 &= wp - a_3 + r + d, & y_3 &= 2p - a_1, \end{aligned}$$

with $w = (p - a_1)p + a_2$ and $d = \dfrac{a_6}{r}$, gives

$$\begin{aligned} E_1 &= x_1 - qy_1 = wr - a_5 - (r - d)q, \\ E_2 &= x_2 - qy_2 = wq - a_4 - (p - a_1)r + pd - q^2, \\ E_3 &= x_3 - qy_3 = wp - a_3 + r + d - (2p - a_1)q. \end{aligned}$$

If $y_1 \neq 0$ take $q = \dfrac{x_1}{y_1}$. If $q \in \mathbb{Z}$ and $E_2 = E_3 = 0$ then cfac becomes TRUE.

If $y_3 \neq 0$ take $q = \dfrac{x_3}{y_3}$. If $q \in \mathbb{Z}$ and $E_1 = E_2 = 0$ then cfac becomes TRUE.

If $y_1 = y_3 = 0$ then

$$E_1 = wr - a_5, \quad E_2 = wq - a_4 + 2pr - q^2, \quad E_3 = wp - a_3 + 2r.$$

If $E_1 = E_3 = 0$, and if the discriminant of E_2 (as a quadratic polynomial in q) is a perfect square, then cfac becomes TRUE.

`procedure ctest (v: integer);`

Assumes $v \mid a_6 \neq 0$. If $f(x)$ has a cubic factor $g(x) = x^3 + px^2 + qx + r$, with $r = v$, for some integers p and q, then cfac is given the value TRUE, as follows. Taking E_1, E_2, E_3 as above, define

$$h_1(p) = \text{Resultant}(E_3, E_1, q), \quad h_3(p) = \text{Resultant}(E_1, E_2, q).$$

As polynomials in p, the degrees of h_1 and h_3 are 3 and 4 respectively. If $x^3 + px^2 + qx + r$ is a factor of $f(x)$ then $h_1(p) = h_3(p) = 0$. The value $p = 0$ is tested (by `rtestc(0)`) if $h_1(0) = h_3(0) = 0$. Then h_1 and h_3 are normalized by removing all factors of p, and all divisors v of $\gcd(h_1(0), h_3(0))$ are tested (by `rtestc(`v`)`) as possible values for p.

`procedure dtestc;`

For each integer $v \mid a_6 \neq 0$, gives the value TRUE to cfac if there exist integers p and q such that $x^3 + px^2 + qx + v$ is a factor of $f(x)$.

`procedure irtest;`

If the polynomial $f(x) = x^6 + a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$ has no linear, quadratic or cubic factors (*i.e.*, if f is irreducible) and if $f(x)$ cannot be written as $g(h(x))$, with $\deg(h) = 2$ or 3 (in which case the field generated over $\mathbb{Q}$ by a root of f would have a proper subfield), and if the degrees of the mod p factors of f are consistent with $\mathrm{PGL}_2(\mathbb{F}_5)$ cycle types for primes p in the range $101 \leq p \leq 571$, then the coefficients $a_1, \ldots, a_6$ are written to the standard output.

`procedure a5test;`

Given a_1, a_2, a_6, a_3, a_4, a_5, S_1, S_2, S_3, S_4, S_5 and X_5, assigns

$$S_5 \leftarrow -X_5 - 5a_5, \quad X_6 \leftarrow a_1S_5 + a_2S_4 + a_3S_3 + a_4S_2 + a_5S_1, \quad S_6 \leftarrow -X_6 - 6a_6.$$

Provided the current combination of values for a_1, a_2, a_6, a_3, a_4 and a_5 satisfies $T - 1 < T_2 \leq T$, and provided the degrees of the mod p factors of the polynomial $f(x) = x^6 + a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$ are consistent with $\mathrm{PGL}_2(\mathbb{F}_5)$ cycle types for primes p in the range $2 \leq p \leq 71$, and provided f satisfies the tests in `irrtest` above, the coefficients $a_1, \ldots, a_6$ are written to the standard output.

`procedure a4test;`

Having a_1, a_2, a_3, a_4, S_1, S_2, S_3, X_4, Y_5 and Y_6, assigns

$$S_4 \leftarrow -X_4 - 4a_4, \quad X_5 \leftarrow a_1S_4 + a_2S_3 + a_3S_2 + a_4S_1,$$

and determines bounds a_{5l}, a_{5u} for a_5 (as explained in chapter three). If the current values of a_1, a_2, a_6, a_3 and a_4 appear in any cases with smaller T-value, then additional bounds c_{5l}, c_{5u} for a_5 are computed (to be used in excluding cases for which $T_2 \leq T - 1$). Calls `a5test` for $a_{5l} \leq a_5 \leq a_{5u}$.

`procedure a3test;`

Having a_1, a_2, a_3, S_1, S_2, X_3 and Y_4, assigns

$$S_3 \leftarrow -X_3 - 3a_3, \quad X_4 \leftarrow a_1 S_3 + a_2 S_2 + a_3 S_1,$$

and determines bounds a_{4l}, a_{4u} for a_4 from Y_4 and X_4 (as explained in chapter three). If the current values of a_1, a_2, a_6 and a_3 appear in any cases with smaller T-value, then additional bounds c_{4l}, c_{4u} for a_4 are computed (to be used in excluding cases for which $T_2 \leq T - 1$). Calls `a4test` for $a_{4l} \leq a_4 \leq a_{4u}$.

`procedure a6test;`

Given a_6, assigns to Z_3, Z_4, Z_5, Z_6 the values of Y_3, Y_4, Y_5, Y_6 corresponding to $\lceil T_2 \rceil = T - 1$ and retrieves the values of Y_3, Y_4, Y_5, Y_6 corresponding to $\lceil T_2 \rceil = T$. Initializes the flags I_p for $2 \leq p \leq 71$. With a_1, a_2, X_2 and S_1 given, assigns

$$S_2 \leftarrow -X_2 - 2a_2 S_1, \quad X_3 \leftarrow a_1 S_2 + a_2 S_1,$$

and determines bounds a_{3l}, a_{3u} for a_3 (as explained in chapter three). If the current values of a_1, a_2 and a_6 appear in any cases with smaller T-value, then additional bounds c_{3l}, c_{3u} for a_3 are computed (to be used in excluding cases for which $T_2 \leq T - 1$). Calls `a3test` for $a_{3l} \leq a_3 \leq a_{3u}$.

`procedure a2test;`

Given $Y_2 = T = \lceil T_2 \rceil$, determines lower and upper bounds $a_{6l} = 1$, $a_{6u} = \left\lfloor \frac{Y_2^3}{216} \right\rfloor$ for a_6 (as explained in chapter three). If the current values of a_1 and a_2 appears in any cases with smaller T-value, then additional bounds c_{6l}, c_{6u} for a_6 are computed (to be used in excluding cases for which $T_2 \leq T - 1$). Calls `a6test` for $a_{6l} \leq a_6 \leq a_{6u}$.

`procedure a1test;`

Initializes the system CPU timer. Given a_1, X_1 and $Y_2 = T = \lceil T_2 \rceil$, assigns

$$S_1 \leftarrow -X_1 - a_1, \quad X_2 \leftarrow a_1 S_1$$

and determines bounds a_{2l}, a_{2u} for a_2 (as explained in chapter three). If $T > 6$, additional bounds c_{2l}, c_{2u} for a_2 are computed (to be used in excluding cases for which $T_2 \leq T - 1$). Calls `a2test` for $a_{2l} \leq a_2 \leq a_{2u}$. When `a2test` has completed, reads the system CPU clock and reports the elapsed time on the standard output.

`procedure Tvtest;`

Initializes

$$X_1 \leftarrow 0, \quad Y_2 \leftarrow T = \lceil T_2 \rceil, \quad Z_2 \leftarrow T - 1;$$

calls `a1test` for $0 \leq a_1 \leq 3$, provided $T \leq B_{a_1}$.

Appendix IV — Screening the Generated Polynomials

The ALGEB program s0s5b.agb reads the sequence of polynomials generated as described above. For each polynomial it computes the signature by means of Sturm sequences [**Cohen**, Algorithm 4.1.11]; polynomials not having signature (0,3) are discarded. For those that remain the field discriminant is computed. This is done using the Zassenhaus "Round Two" algorithm [**Cohen**, Algorithm 6.1.8]. Polynomials for which the field discriminant is too large are discarded. The polynomials which survive these tests are written in normalized form, which is to say that the leading term of $f(t)$ of odd degree has a positive coefficient, and $g(t) = t^6 f(1/t)$ replaces $f(t)$ if $|f(0)| = 1$ and g precedes f lexicographically.

ALGEB Program s0s5b.agb

```
iochan 1, 2;  integer fsig, fdmax;      { signature (0,3) }

fsig := 0;  fdmax := abs(-1778112);

begin

integer j, fd, m, ct;  array f[0:6];

external integer procedure fdisc;
external integer procedure sturm;
```

```
procedure normal;          { make leading odd term positive }
integer j, k;  array g[0:6];
if abs(f[0]) = 1 then
    begin
    k := 0;
    for j := 0,...,6 do
        begin
        g[j] := f[0]*f[6-j];
        if abs(g[j]) # abs(f[j]) then k := j
        end;
    if abs(g[k]) < abs(f[k]) then f := g
    end;
k := 1;
if f[3] # 0 then k := 3;
if f[5] # 0 then k := 5;
if f[k] < 0 then
    f[1] := -f[1] ! f[3] := -f[3] ! f[5] := -f[5]
end;

input(1,"AGB$CONTROL");          { *** start here *** }
readn(1,m);
close(1);

input(1,"AGB$INPUT");  output(2,"AGB$OUTPUT");

for ct := 1,...,m do
    begin
    f[6] := 1;  for j := 5,4,...,0 do readn(1,f[j]);  normal;
    if sturm(f) = fsig then
        begin
        fd := fdisc(f);
        if abs(fd) <= fdmax then
            begin
            writen(2,fd,12);
            for j := 5,4,...,0 do space(2,1) ! writen(2,f[j],4);
            line(2,1)
            end
        end
    end;

close(1);  close(2)

end
```

The output from the ALGEB program s0s5b.agb consists of 1190 distinct polynomials of signature (0,3), each of which is given together with the discriminant of the field generated by one of its roots.

Output from s0s5b.agb (ALGEB)

-12167	0	5	3	12	-4	8
.						
.						
.						
-1728243	3	1	-6	-5	18	27
-1778112	0	3	2	6	0	1
-1778112	0	3	4	0	6	13
-1778112	0	6	10	21	30	13
-1778112	2	1	4	14	16	9
-1778112	2	3	0	0	0	21
-1778112	2	5	0	7	-2	11
-1778112	2	6	6	9	-6	3
-1778112	2	8	18	25	10	17
-1778112	2	-1	-12	-2	10	11
-1778112	2	10	4	11	-8	3

A simple PASCAL program reads the output from s0s5b.agb and creates the MAPLE program mins0s5.mpl. This program computes the Galois group of the splitting field for each of the polynomials. (The substitution of $x-514$ for x avoids an infinite loop in the MAPLE galois routine.)

MAPLE Program mins0s5.mpl

```
rpt := proc (fx, discF)
local st, gg, et, gx:
lprint('f'=fx):
st := time():  gg := galois(subs(x=x-541,fx)):  et := time() - st:
lprint('dF'=discF,gg[1],'xT'=et):
end:
rpt(x^6+5*x^4+3*x^3+12*x^2-4*x+8,-12167):
  .
  .
  .
rpt(x^6+3*x^5+x^4-6*x^3-5*x^2+18*x+27,-1728243):
rpt(x^6+3*x^4+2*x^3+6*x^2+1,-1778112):
rpt(x^6+3*x^4+4*x^3+6*x+13,-1778112):
rpt(x^6+6*x^4+10*x^3+21*x^2+30*x+13,-1778112):
rpt(x^6+2*x^5+x^4+4*x^3+14*x^2+16*x+9,-1778112):
rpt(x^6+2*x^5+3*x^4+21,-1778112):
rpt(x^6+2*x^5+5*x^4+7*x^2-2*x+11,-1778112):
rpt(x^6+2*x^5+6*x^4+6*x^3+9*x^2-6*x+3,-1778112):
rpt(x^6+2*x^5+8*x^4+18*x^3+25*x^2+10*x+17,-1778112):
rpt(x^6+2*x^5-x^4-12*x^3-2*x^2+10*x+11,-1778112):
rpt(x^6+2*x^5+10*x^4+4*x^3+11*x^2-8*x+3,-1778112):
```

The output from this MAPLE program includes the polynomial, the discriminant of the field generated by adjoining one of its roots to $\mathbb{Q}$, the Galois group of the splitting field of the polynomial, and the time (in seconds) required to compute the Galois group.

The Galois group $PGL_2(\mathbb{F}_5) \simeq S_5$ occurs only for the ten fields with discriminant -1778112. It is shown below that these fields are all isomorphic.

Output from minsOs5.mpl (MAPLE)

```
 f = x^6 + 5*x^4 + 3*x^3 + 12*x^2 - 4*x + 8
dF = -12167    S3    xT = .600
    .
    .
    .
 f = x^6 + 3*x^5 + x^4 - 6*x^3 - 5*x^2 + 18*x + 27
dF = -1728243    D6    xT = .400
 f = x^6 + 3*x^4 + 2*x^3 + 6*x^2 + 1
dF = -1778112    PGL2(5)    xT = 4.817
 f = x^6 + 3*x^4 + 4*x^3 + 6*x + 13
dF = -1778112    PGL2(5)    xT = 4.700
 f = x^6 + 6*x^4 + 10*x^3 + 21*x^2 + 30*x + 13
dF = -1778112    PGL2(5)    xT = 5.000
 f = x^6 + 2*x^5 + x^4 + 4*x^3 + 14*x^2 + 16*x + 9
dF = -1778112    PGL2(5)    xT = 5.600
 f = x^6 + 2*x^5 + 3*x^4 + 21
dF = -1778112    PGL2(5)    xT = 5.266
 f = x^6 + 2*x^5 + 5*x^4 + 7*x^2 - 2*x + 11
dF = -1778112    PGL2(5)    xT = 9.400
 f = x^6 + 2*x^5 + 6*x^4 + 6*x^3 + 9*x^2 - 6*x + 3
dF = -1778112    PGL2(5)    xT = 5.200
 f = x^6 + 2*x^5 + 8*x^4 + 18*x^3 + 25*x^2 + 10*x + 17
dF = -1778112    PGL2(5)    xT = 4.700
 f = x^6 + 2*x^5 - x^4 - 12*x^3 - 2*x^2 + 10*x + 11
dF = -1778112    PGL2(5)    xT = 4.816
 f = x^6 + 2*x^5 + 10*x^4 + 4*x^3 + 11*x^2 - 8*x + 3
dF = -1778112    PGL2(5)    xT = 5.350

Total CPU Time:    1063.516 sec
```

For the ten fields with discriminant -1778112, defined over $\mathbb{Q}$ by the polynomials $p_1(t), \ldots, p_{10}(t)$, the MAPLE program `isofld.mpl` confirms that the field defined by $p_k(t)$ is isomorphic to the field defined by $p_5(t)$, for $1 \le k \le 10$. The MAPLE routine `lattice` is used to discover a relation between a root y of $p_k(t)$ and a root x of $p_5(t)$; the correctness of the relation is confirmed by a `resultant` computation.

MAPLE Program isofld.mpl

```
readlib(lattice):

Digits := 20:  q := 10^Digits:

relvec := proc (f, r, g, s)
local h, j, k, u, v, w, b, c:
v := array(0..6):
for k from 0 to 6 do
    v[k] := array(1..9):
    if k = 6 then w := s else w := expand(r^k) fi:
    for j from 1 to 7 do if j = k+1 then v[k][j] := 1
                                    else v[k][j] := 0 fi od:
    v[k][8] := round(q*coeff(w,I,0)):
    v[k][9] := round(q*coeff(w,I,1))
od:
u := convert(v,list):  b := lattice(u,integer):  c := b[1];
h := c[1] + c[2]*x + c[3]*x^2 + c[4]*x^3
                                 + c[5]*x^4 + c[6]*x^5 + c[7]*y:
if rem(resultant(h,g,y),f,x) = 0 then print('0'=h) fi
end:

p[01] := t^6           +  3*t^4 +  2*t^3 +  6*t^2        +  1:
p[02] := t^6           +  3*t^4 +  4*t^3          +  6*t + 13:
p[03] := t^6           +  6*t^4 + 10*t^3 + 21*t^2 + 30*t + 13:
p[04] := t^6 + 2*t^5 +    t^4 +  4*t^3 + 14*t^2 + 16*t +  9:
p[05] := t^6 + 2*t^5 +  3*t^4                          + 21:
p[06] := t^6 + 2*t^5 +  5*t^4          +  7*t^2 -  2*t + 11:
p[07] := t^6 + 2*t^5 +  6*t^4 +  6*t^3 +  9*t^2 -  6*t +  3:
p[08] := t^6 + 2*t^5 +  8*t^4 + 18*t^3 + 25*t^2 + 10*t + 17:
p[09] := t^6 + 2*t^5 + 10*t^4 +  4*t^3 + 11*t^2 -  8*t +  3:
p[10] := t^6 + 2*t^5 -    t^4 - 12*t^3 -  2*t^2 + 10*t + 11:

print('x is a root of p[5] and y is a root of p[k].');
nf := 5:
for ng from 1 to 10 do
    lprint('k'=ng):
    fx := subs(t=x,p[nf]):  rf := fsolve(fx,x,complex):
    gy := subs(t=y,p[ng]):  rg := fsolve(gy,y,complex):
    kf := 1:
    for kg from 1 to 6 do relvec(fx,rf[kf],gy,rg[kg]) od
od:
```

Output from isofld.mpl (MAPLE)

x is a root of p[5] and y is a root of p[k].

k = 1

$$0 = 103 + 82 x + 111 x^2 + 5 x^3 + 21 x^4 - 6 x^5 + 277 y$$

k = 2

$$0 = 10 - 27 x - 121 x^2 + 22 x^3 + 37 x^4 + 29 x^5 - 277 y$$

k = 3

$$0 = 58 - 212 x - 37 x^2 - 94 x^3 - 7 x^4 + 2 x^5 - 277 y$$

k = 4

$$0 = 55 - 10 x + 27 x^2 + 121 x^3 + 65 x^4 + 21 x^5 + 277 y$$

k = 5

$$0 = - x + y$$

k = 6

$$0 = 58 + 65 x - 37 x^2 - 94 x^3 - 7 x^4 + 2 x^5 - 277 y$$

k = 7

$$0 = 45 + 17 x + 148 x^2 + 99 x^3 + 28 x^4 - 8 x^5 + 277 y$$

k = 8

$$0 = 158 + 72 x + 138 x^2 + 126 x^3 + 86 x^4 + 15 x^5 + 277 y$$

k = 9

$$0 = 103 - 195 x + 111 x^2 + 5 x^3 + 21 x^4 - 6 x^5 + 277 y$$

k = 10

$$0 = - 113 + 222 x + 10 x^2 - 27 x^3 - 58 x^4 - 23 x^5 - 277 y$$

Total CPU Time: 142.783