JOURNAL de Théorie des Nombres de BORDEAUX

anciennement Séminaire de Théorie des Nombres de Bordeaux

Mark VAN HOEIJ et John CREMONA

Solving conics over function fields

Tome 18, n° 3 (2006), p. 595-606.

<http://jtnb.cedram.org/item?id=JTNB_2006__18_3_595_0>

Journal de Théorie des Nombres
de Bordeaux **18** (2006), 595-606

Solving conics over function fields

par Mark VAN HOEIJ et John CREMONA

Résumé. Soit F un corps de caractéristique différente de 2, et $K = F(t)$. Nous donnons un algorithme simple pour trouver, étant donné $a, b, c \in K^*$, une solution non-triviale dans K (si elle existe) à l'équation $aX^2 + bY^2 + cZ^2 = 0$. Dans certains cas, l'algorithme a besoin d'une solution d'une équation similaire à coefficients dans F ; nous obtenons alors un algorithme récursif pour résoudre les coniques diagonales sur $\mathbb{Q}(t_1, \ldots, t_n)$ (en utilisant les algorithmes existants pour telles équations sur $\mathbb{Q}$) et sur $\mathbb{F}_q(t_1, \ldots, t_n)$.

Abstract. Let F be a field whose characteristic is not 2 and $K = F(t)$. We give a simple algorithm to find, given $a, b, c \in K^*$, a nontrivial solution in K (if it exists) to the equation $aX^2 + bY^2 + cZ^2 = 0$. The algorithm requires, in certain cases, the solution of a similar equation with coefficients in F; hence we obtain a recursive algorithm for solving diagonal conics over $\mathbb{Q}(t_1, \ldots, t_n)$ (using existing algorithms for such equations over $\mathbb{Q}$) and over $\mathbb{F}_q(t_1, \ldots, t_n)$.

1. Introduction

Let $K = F(t)$, where F is a field of characteristic other than 2, and $a, b, c \in K^*$. Our goal is to find, if it exists, a solution $X, Y, Z \in K$, not all zero, for which

$$aX^2 + bY^2 + cZ^2 = 0. \tag{1}$$

We will use projective coordinates $(X : Y : Z) \in \mathbb{P}^2(K)$ to indicate that at least one of $X, Y, Z \in K$ must be non-zero, and that $(sX : sY : sZ)$ will be identified with $(X : Y : Z)$ for any nonzero $s \in K$.

The case $K = \mathbb{Q}$ was solved by Legendre, who gave a criterion (the local-global principle for conics) to decide if a solution $(X, Y, Z) \in \mathbb{P}^2(\mathbb{Q})$ exists. He also gave an algorithm to find a solution. See Algorithm I in [CR03], where a number of other algorithms are given for this case.

Manuscrit reçu le 5 janvier 2006.
MvH acknowledges support from the A. von Humboldt Foundation and NSF grant 0511544.

Legendre showed that a solution $(X : Y : Z) \in \mathbb{P}^2(\mathbb{Q})$ exists if and only if a so-called *solubility certificate* exists. The non-trivial part is to show that existence of this certificate implies the existence of a solution $(X : Y : Z)$. We sketch a textbook proof, with an improvement given in [CM98], because our algorithm will follow the same approach. Use the solubility certificate to construct a certain lattice $L \leq \mathbb{Z}^3$ which has the property that $aX^2 + bY^2 + cZ^2 \equiv 0 \pmod{abc}$ for every $(X, Y, Z) \in L$, and use Minkowski's theorem to show that L contains a non-zero element (X, Y, Z) for which $|a|X^2 + |b|Y^2 + |c|Z^2 < 2abc$. If $aX^2 + bY^2 + cZ^2$ is not already 0, it must be $\pm abc$, and a simple transformation yields new X, Y, Z for which $aX^2 + bY^2 + cZ^2 = 0$. Alternatively, in [CM98] it is shown that one can construct a lattice $L' \subset L$ such that $aX^2 + bY^2 + cZ^2 \equiv 0 \pmod{2abc}$ for every $(X, Y, Z) \in L'$ and moreover that this L' still contains a non-zero element (X, Y, Z) for which $|a|X^2 + |b|Y^2 + |c|Z^2 < 2abc$ and hence $aX^2 + bY^2 + cZ^2 = 0$. Given a solubility certificate, one can quickly find such (X, Y, Z) with the LLL lattice reduction algorithm (see [LLL82]) and Lemma 2.7 in [CR03] (where an even faster algorithm was given as well).

In the proof of Theorem 1 below we will follow the same ideas for $K = F(t)$. Although this involves additional notation, some details will be easier since the lattice arguments for $F = \mathbb{Q}$ simplify to linear algebra in the function field case. Moreover, the solution $(X : Y : Z)$ produced by our algorithm satisfies the degree bounds $\deg(X) \leq \deg(bc)/2$, $\deg(Y) \leq \deg(ac)/2$, $\deg(Z) \leq \deg(ab)/2$.

In the algorithm for conics over $K = F(t)$, it will sometimes be necessary to solve a conic over the base field F: namely, when all irreducible factors of abc have even degree. Hence our algorithm should be regarded as a recursive one, which may be used to solve conics over $F(t_1, \ldots, t_n)$ whenever we are able to solve them over F. As "base cases" we have $K = \mathbb{Q}$, mentioned above, and $F = \mathbb{F}_q$ where q is an odd prime power; in the latter case, solution is trivial since a random line intersects the conic in two points which with probability $\frac{1}{2}$ are defined over $\mathbb{F}_q$.

It has been pointed out by Miles Reid that the method used is essentially the same as for proving Tsen's Theorem: see [R97, p. 59]; in that context the field F is algebraically closed.

When $F = \mathbb{R}$ (the field of real numbers) and $K = \mathbb{R}(t)$, the algorithm presented in this paper was already given by Schicho (see Lemma 5 in [Sch98], and also [Sch00]). So the algorithm presented here is a modest generalization of Schicho's algorithm, applying over more general ground fields, and with certain improvements: notably that recursion is only necessary when all irreducible factors of abc have even degree.

In the following sections we give the definition of a solubility certificate, and then present the algorithm. We end with some examples.

The algorithm has been implemented by the authors in both Maple (see [Maple]) and MAGMA(see [Magma]). The programs may be obtained from `http://www.math.fsu.edu/~hoeij/files/ConicProgram` and `http://www.maths.nott.ac.uk/personal/jec/ftp/progs/magma/` .

2. The solubility certificate

After multiplying by denominators we may assume that

$$a, b, c \in F[t]. \tag{2}$$

Let $g = \gcd(a, b)$. If $\deg(g) > 0$, then a, b, c will be replaced by $a/g, b/g, cg$. The new conic is equivalent to the old conic under $(X : Y : Z) \mapsto (X : Y : Z/g)$. The same is done for $\gcd(b, c)$ and $\gcd(c, a)$. After a finite number of such steps (finite since the degree of abc is reduced by $\deg(g)$ at each step), we have

$$\gcd(a, b) = \gcd(b, c) = \gcd(c, a) = 1. \tag{3}$$

By a "non-trivial square" we mean the square of a non-zero non-unit $d \in F[t]$ (so $\deg(d) > 0$). If a, b or c is divisible by a non-trivial square then we can divide and obtain an equivalent conic. Thus, we may assume that a, b, c are square-free (i.e., not divisible by non-trivial squares); because of (3), this is equivalent to

$$abc \quad \text{is square-free.} \tag{4}$$

Given equation (1), the above describes Legendre's procedure to compute a *reduced form* of this equation, which means:

Definition 1. *An equation* $aX^2+bY^2+cZ^2 = 0$ *is in* reduced form *if* a, b, c *satisfy assumptions (2), (3) and (4):* $a, b, c \in F[t]$ *with* abc *square-free.*

In Section 4 below, we give an algorithm (**Algorithm ReduceConic**) to reduce a given equation.

Denote by $d_a, d_b, d_c \in \mathbb{N}$ the degrees of a, b, c, and by $l_a, l_b, l_c \in F$ the leading coefficients of a, b, c. Set

$$\text{case} := \begin{cases} 0 & \text{if } d_a \equiv d_b \equiv d_c \pmod 2 \text{ and } abc \text{ has no root in } F; \\ 1 & \text{otherwise.} \end{cases}$$

Later we will see that the harder case ("case = 0") can be restricted further to only include equations where all irreducible factors of abc have even degree; but we will first give the simpler form of the algorithm with cases defined as above.

Lemma 1. *If a solution $(X : Y : Z) \in \mathbb{P}^2(K)$ of equation (1) exists, then*

$$l_a x^2 + l_b y^2 + l_c z^2 = 0 \tag{5}$$

has a solution in $\mathbb{P}^2(F)$.

Proof. After scaling we may assume that $X, Y, Z \in F[t]$. Let d be the maximum of the degrees of aX^2, bY^2, and cZ^2: either all three of these polynomials have degree d, or two have degree d and one has strictly smaller degree. In the first case (then d_a, d_b, d_c have the same parity), taking leading coefficients of the solution gives a solution $(x : y : z)$ to (5). In the second case, say with the degree of aX^2 strictly less than d, we again find a solution to (5) of the form $(0 : y : z)$ where y, z are the leading coefficients of Y, Z respectively. □

Let $\mathrm{supp}(a)$ denote the set of all monic irreducible $p \in F[t]$ that divide a. Denote

$$L_p := F[t]/(p),$$

which is a finite extension of F of degree $\deg(p)$. Introduce a new variable u, and let $f_a, f_b, f_c \in F[t][u]$ be the following polynomials

$$f_a := bu^2 + c, \qquad f_b := cu^2 + a, \qquad f_c := au^2 + b.$$

For $f \in F[t][u]$, the image in $L_p[u]$ is denoted as $f \bmod p$.

Definition 2. *Let $aX^2 + bY^2 + cZ^2 = 0$ be in reduced form (see Definition 1). Then a* solubility certificate *is a list containing the following:*

- *For every $p \in \mathrm{supp}(a)$, a root of $f_a \bmod p$ in L_p;*
- *For every $p \in \mathrm{supp}(b)$, a root of $f_b \bmod p$ in L_p;*
- *For every $p \in \mathrm{supp}(c)$, a root of $f_c \bmod p$ in L_p;*
- *If case $= 0$, either a solution or a solubility certificate for equation (5).*

Note that over $K = \mathbb{Q}$, the standard definition of a solubility certificate is formulated slightly differently, namely the roots are combined by the Chinese Remainder Theorem. So the usual definition of a solubility certificate over $\mathbb{Q}$ is a list of three integers k_a, k_b, k_c that are roots of $f_a \bmod a$, $f_b \bmod b$, and $f_c \bmod c$ respectively.

Lemma 2. *Assume equation (1) is in reduced form and has a solution in $\mathbb{P}^2(K)$. Then a solubility certificate exists.*

Proof. Let $(X : Y : Z)$ be a solution. After scaling we may assume that $X, Y, Z \in F[t]$ with $\gcd(X, Y, Z) = 1$. For case $= 0$, we have already shown that equation (5) has a solution by considering the leading coefficients of X, Y, Z. The remainder of the proof is the same as for the case $K = \mathbb{Q}$, but we include it for completeness since our notations are slightly different.

Let $p \in \mathrm{supp}(a)$. Now $0 = aX^2 + bY^2 + cZ^2$ reduces to $bY^2 + cZ^2 \bmod p$. Since b, c do not vanish mod p, it follows that $\overline{Y}, \overline{Z}$ are either both zero or both not zero, where $\overline{Y}, \overline{Z}$ denote the images of Y, Z in L_p. The first case is ruled out since it would imply that $bY^2 + cZ^2$, and hence aX^2, would be divisible by p^2, which implies that X is divisible by p, contradicting $\gcd(X, Y, Z) = 1$. Thus, $\overline{Y}, \overline{Z}$ are not zero; and $\overline{Y}/\overline{Z} \in L_p$ is a root of $f_a \bmod p$. Repeating this for f_b (respectively f_c) for the factors of b (respectively c), it follows that a solubility certificate exists. □

2.1. Algorithmic remarks. To compute (if it exists) a solubility certificate we use the following tools:

(1) A factorization algorithm for $F[t]$;
(2) An algorithm for computing square roots in L, where L is a finite extension of F;
(3) If case = 0, we need to recursively find a solubility certificate (or a solution) for a conic over F. Hence, if $K = \mathbb{Q}(t_1, \ldots, t_n)$ and if at each stage of recursion we are in case 0, we eventually need a solubility certificate (or a solution) over $\mathbb{Q}$, which will require a factorization algorithm for $\mathbb{Z}$.

The factorization algorithms required are available in standard computer algebra systems such as Maple and Magma. Integer factorization can take much time (no polynomial time algorithm is known), while the time for factorization of polynomials over number fields is small in comparison: for fixed n, the time spent on items 1 and 2 is polynomially bounded in terms of the degrees and logarithmic heights.

Thus for $K = \mathbb{Q}(t_1, \ldots, t_n)$ with fixed n one can compute a solubility certificate in polynomial time except if case = 0 at each of the recursive stages. Having to factor integers can sometimes, but not always, be avoided by interchanging variables.

2.2. A solubility certificate corresponds to a subset of the conic. If equation (1) has a solution in $\mathbb{P}^2(K)$ then the number of solubility certificates equals 2^n for some integer n, because for each p, one has to choose one of the two roots (one root if $K = \mathbb{Q}$ and $p = 2$). Every certificate corresponds to a non-empty (by Theorem 1 below) subset of the set of K-rational points on the conic in $\mathbb{P}^2$ defined by equation (1). Grouping together subsets/certificates identified under $(X : Y : Z) \mapsto (\pm X : \pm Y : \pm Z)$, one may write the conic as a disjoint union of non-empty subsets $S_1, S_2, \ldots, S_{2^m}$ for some integer $m \geq 0$. Given one point on the conic, one can write down a *parametrization*, a birational map Ψ from $\mathbb{P}^1(K)$ to the conic. Then every point on the conic can be written as $\Psi(u : v)$ for some $(u : v) \in \mathbb{P}^1(K)$. Nevertheless, one can construct examples with $K = \mathbb{Q}$, $m > 0$, with a point

in one S_i given, where it would require a breakthrough in integer factorization to find a point in any S_j with $j \neq i$. This is because going from one S_i to another is equivalent to finding a non-trivial factor of a, b or c.

Although the above remark is not needed for our algorithm, it is useful to know that a solubility certificate corresponds to a subset of the conic, and that the algorithm in the next section finds an element of that subset. So any certificate will lead to a point on the conic; however, a much nicer point (with smaller degree or height of X, Y, Z) might be found by trying one of the other certificates.

3. Finding a point on the conic

In this section we present the algorithm for finding a solution to (1), given a solubility certificate. As pointed out in the introduction, this algorithm is a modest generalization of Schicho's algorithm.

Algorithm FindPoint

Input: $a, b, c \in K^*$ satisfying assumptions (2), (3), (4), and a solubility certificate.

Output: A solution $(X : Y : Z) \in \mathbb{P}^2(K)$ of $aX^2 + bY^2 + cZ^2 = 0$.

(1) Let d_a, d_b, d_c be the degrees of a, b, c.

(2) Let $\mathcal{S}_a$, $\mathcal{S}_b$, $\mathcal{S}_c$ be the supports of a, b, c.

(3) If $d_a \equiv d_b \equiv d_c \pmod 2$ then: if $\mathcal{S}_a \cup \mathcal{S}_b \cup \mathcal{S}_c$ contains no element of degree 1 then set case := 0, else set case := 1 and remove a single degree 1 irreducible from one of $\mathcal{S}_a$, $\mathcal{S}_b$, $\mathcal{S}_c$.
Otherwise (when the degrees have different parities): set case := 1.

(4) Set

$$A := \lceil \frac{d_b + d_c}{2} \rceil - \text{case}, \quad B := \lceil \frac{d_c + d_a}{2} \rceil - \text{case}, \quad C := \lceil \frac{d_a + d_b}{2} \rceil - \text{case}$$

and

$$X := \sum_{i=0}^{A} X_i t^i, \qquad Y := \sum_{i=0}^{B} Y_i t^i, \qquad Z := \sum_{i=0}^{C} Z_i t^i$$

where the X_i, Y_i, Z_i are new variables.

(5) If case = 0, let $l_a, l_b, l_c \in F$ be the leading coefficients of a, b, c. Let $(x : y : z)$ be a solution of equation (5), see the Remark below. Introduce a new variable W and set $E := \{X_A - xW, Y_B - yW, Z_C - zW\}$. Otherwise (case = 1) let $E := \emptyset$.

(6) For every $p \in \mathcal{S}_a$, let $\alpha \in L_p$ be a root of $f_a \bmod p$. Lift α to $F[t]$, divide $Y - \alpha Z$ by p and let r be the remainder. Write $r = \sum_{i<d} r_i t^i$ where $d = \deg(p)$. Then r_i is an F-linear combination of the Y_j and Z_j. Now set $E := E \cup \{r_0, \ldots, r_{d-1}\}$.
Repeat this for f_b (respectively f_c) for all p in $\mathcal{S}_b$ (respectively $\mathcal{S}_c$).

(7) Equating all elements of E to 0 gives a system of homogeneous linear equations for the variables X_i, Y_i, Z_i (and W if case $= 0$). Solve this system and substitute a non-trivial solution into X, Y, Z. Divide by $\gcd(X, Y, Z)$. Return $(X : Y : Z)$ and stop.

Remark on step 5: The solubility certificate in the input comes from Algorithm Conic in section 4, and contains a solution of equation (5).

Theorem 1. *Assume that equation (1) is in reduced form. Then it has a solution in* $\mathbb{P}^2(K)$ *if and only if a solubility certificate exists; in that case, a solution may be found by* **Algorithm FindPoint**.

Proof. Assume we have a solubility certificate. It suffices to prove that Algorithm FindPoint works. For step 5 see the Remark above. We will show that in each case, the system E in step 7 has a non-zero solution because the number of variables is greater than the number of equations; therefore, the notation $(X : Y : Z) \in \mathbb{P}^2(K)$ in step 7 is correct. Secondly we will argue that $aX^2 + bY^2 + cZ^2$ must be zero by simultaneously bounding its degree and showing that it is divisible by a polynomial of larger degree.

Let $D = d_a + d_b + d_c = \deg(abc)$. If case $= 0$ and we have removed a degree 1 irreducible factor p from the support set, let $D' = \deg(abc/p) = D - 1$, otherwise set $D' = D$. By construction, $r = aX^2 + bY^2 + cZ^2$ has degree $\leq \max\{d_a + 2A, d_b + 2B, d_c + 2C\} = D' -$ case. Furthermore, r is divisible by every irreducible element of $\mathcal{S}_a \cup \mathcal{S}_b \cup \mathcal{S}_c$ (from step 3), because equations E are satisfied, so (since abc is square-free) r is divisible by a polynomial of degree D'. If case $= 1$ this already gives $r = 0$. If case $= 0$, we see that r can be written as $s \cdot abc$ for some constant $s \in F$; but the t^D coefficient in $aX^2 + bY^2 + cZ^2$ is (see step 5) given by the left-hand side of equation (5), which equals 0. It follows that $r = aX^2 + bY^2 + cZ^2 = 0$ in all cases.

It remains to show that the linear system has more unknowns than equations. When case $= 0$, we have $A + B + C = D$, the number of unknowns is $(1 + A) + (1 + B) + (1 + C) + 1 = D + 4$ while the number of equations is $D+3$, since each irreducible p of degree d gives d equations and we have 3 extra equations from (5). When case $= 1$ and the degree parities are equal and we deleted a degree 1 irreducible p, we have $A + B + C = D - 3$, the number of unknowns is $(1 + A) + (1 + B) + (1 + C) = D$, and the number of equations is $D - 1$. Finally, when case $= 1$ and the degree parities are not equal, we have $A + B + C = D - 2$, the number of unknowns is $(1 + A) + (1 + B) + (1 + C) = D + 1$ and the number of equations is D.

Hence the algorithm will always return a (nontrivial) solution of equation (1). □

4. Algorithmic details

We now give an explicit algorithm to reduce an equation of the form (1).

Algorithm ReduceConic
Input: $a, b, c \in F(t)^*$.
Output: $a_1, b_1, c_1 \in F[t]$ for which $a_1X^2 + b_1Y^2 + c_1Z^2 = 0$ is the reduced form of (1), together with $\lambda, \mu, \nu \in F(t)^*$ such that (X, Y, Z) satisfies $a_1X^2 + b_1Y^2 + c_1Z^2 = 0$ if and only if $(\lambda X, \mu Y, \nu Z)$ satisfies (1).

(1) Multiply (a, b, c) by the least common multiple of the denominators, and then divide (a, b, c) by $\gcd(a, b, c)$. Set $\lambda = \mu = \nu = 1$.
(2) Let $g := \gcd(a, b)$. Set $a := a/g$, $b := b/g$, $c := cg$, and $\nu := g\nu$.
(3) Let $g := \gcd(a, c)$. Set $a := a/g$, $c := c/g$, $b := bg$, and $\mu := g\mu$.
(4) Let $g := \gcd(b, c)$. Set $b := b/g$, $c := c/g$, $a := ag$, and $\lambda := g\lambda$.
(5) Repeat the last 3 steps until $g = 1$ in all cases.
(6) Apply square-free factorization to write a as $a_1a_2^2$ for some $a_1, a_2 \in F[t]$ with a_1 square-free. Set $\mu := a_2\mu$, $\nu := a_2\nu$. Apply a similar process to b and c.
(7) Return (a_1, b_1, c_1), (λ, μ, ν) and stop.

Finally we put everything together to give the complete algorithm. We include here the singular cases.

Algorithm Conic
Input: $a, b, c \in K = F(t)$.
Output: A solution $(X : Y : Z) \in \mathbb{P}^2(K)$ of equation (1) if such solution exists.

(1) If $a = 0$ then return $(1 : 0 : 0)$ and stop.
If $b = 0$ then return $(0 : 1 : 0)$ and stop.
If $c = 0$ then return $(0 : 0 : 1)$ and stop.
(2) Replace (a, b, c) by the first output (a_1, b_1, c_1) of Algorithm ReduceConic, and let (λ, μ, ν) be the second output.
(3) Compute a solubility certificate:
  (a) For each p in $\mathrm{supp}(a)$ (respectively $\mathrm{supp}(b)$, $\mathrm{supp}(c)$), factor the image of f_a (respectively f_b, f_c) in $L_p[u]$. If irreducible, then return "no solution exists" and stop. Otherwise, store one of the roots (this root will be used in step 6 of Algorithm FindPoint).
  (b) If case $= 0$ then find a solution $(x : y : z)$ of (5). If this recursive call returned "no solution exists" then return the same message and stop.
(4) Let $(X : Y : Z)$ be the output of Algorithm FindPoint.
(5) Return $(\lambda X : \mu Y : \nu Z)$ and stop.

In the recursive step 3b we assume that it is possible to solve conics over the base field F. By recursion, if F is a field for which the necessary ingredients are available (an implementation to solve a conic over F, and the factorization algorithms mentioned in section 2.1) then $F(t_1, \ldots, t_k)$ can be handled in this way. If conic solving over F is not available, Algorithm Conic as described here could still be useful, since it will still work when case $= 1$.

5. Further remarks:

1. We may interpret our division into two cases as follows. We may avoid recursion (i.e. case $= 1$) when there is a degree 1 place of the field $F(t)$ where the conic has bad reduction. For the "finite" places of $F(t)$ this occurs precisely when (for a reduced equation) there is a degree 1 irreducible factor of abc. For the infinite place of $F(t)$, with uniformizer $1/t$, this occurs when the degrees of a, b and c have different parities. So our division into cases is invariant under F-automorphisms of $F(t)$.

2. If there are no degree 1 places of bad reduction but there is a place of bad reduction of odd degree $m = 2d - 1$, one can proceed as follows, to avoid the need to solve the leading coefficient equation (5).

Let p be an irreducible factor of abc of odd degree. Extend the base field to $F' = F(\alpha)$ of odd degree m, where α is a root of p. Over $F'(t)$, we now have a degree 1 factor $t - \alpha$ of abc, so our existing algorithm gives (via case 1, with no recursion) a solution P defined over $F'(t)$. Write $m = 2d-1$, and consider functions on the conic of the form

$$f = \sum_{i=0}^{d} a_i X^i + Y \sum_{j=0}^{d-1} b_j X^j,$$

with coefficients $a_i, b_j \in F(t)$; these form a vector space of dimension $2d+1$ over $F(t)$. We may impose linear conditions on the coefficients by requiring that $f(P) = 0$: expanding in powers of α this gives a system of $2d - 1$ equations, which has a solution space of dimension exactly 2. (This follows from the Riemann-Roch Theorem: we are computing the space $L(D)$ where D is the divisor $D = dD_\infty - \sum_\sigma (P^\sigma)$, where D_∞ is the divisor of poles of X and P^σ runs over the F'/F-conjugates of P.) Taking f_1, f_2 to be given by independent solutions to these equations, the quotient f_1/f_2 will have a unique zero and a unique pole, both at $F(t)$-rational points, which may then be found easily.

If the solution we find this way does not satisfy the degree bounds guaranteed by our main algorithm, this can be arranged as follows: we may use the leading coefficients of any solution to obtain a solution to (5), and then apply the original algorithm over $F(t)$ itself.

Thus the recursive step is only necessary when all places of bad reduction have even degree. However, in practice, it may only be worth using this trick when $F = \mathbb{Q}$; otherwise, solving (5) directly may be simpler than working over a field extension.

3. In general one would like to solve conics given by more general homogeneous quadratics $Q(X, Y, Z) = 0$, where Q has coefficients in $F(t)$. In alternative notation, a conic is defined by a 3×3 symmetric matrix A over $F(t)$ which is not necessarily diagonal. Since we have excluded the case of characteristic 2, one may readily convert this to diagonal form by completing the square. There is a cost associated with this operation: we need to factorize the diagonal entries, which might involve many more "bad places" than in the original model, where the (finite) bad places are given by irreducible factors of $\det(A)$. Over $\mathbb{Q}$ this is a major problem, since integer factorization is hard, as was pointed out by D. Simon after the appearance of [CR03]; Simon produced an alternative algorithm for conics over $\mathbb{Q}$ which avoids this: see [S05].

In the function field case, since polynomial factorization is much cheaper than integer factorization, this should not be a problem. Nevertheless, one of us (Cremona) has implemented in Magma an alternative algorithm for non-diagonal conics, which proceeds as follows (following a strategy used by Gauss over $\mathbb{Z}$: see [Gauss, Art. 272]). First, we reduce to the case of a 3×3 symmetric matrix A over $F[t]$ which is unimodular (i.e., $\det(A) \in F^*$); this only requires factorization of the original determinant, computing square roots modulo its irreducible factors, and linear algebra over finite extensions of F. This step corresponds roughly to the reduction and certificate computation steps in the diagonal algorithm. Secondly, we use unimodular transformations to reduce the degrees of the entries of A; one can show that we can either reach the situation where A is both diagonal and constant (entries in F^*), or find a solution along the way. A similar method was considered by Schicho in [Sch00], and again by van de Woestijne in [W06].

There are practical difficulties with this algorithm, though it does work. If the base field F is a finite field there is no problem, but if $F = \mathbb{Q}$ or F is itself a function field, although we steadily decrease the degrees of the entries of the matrix, the coefficients of the entries are subject to expression swell, making the overall algorithm slow. The situation is entirely analogous to the use of the Euclidean Algorithm to compute the gcd of two polynomials in $\mathbb{Q}[t]$, which suffers from the same defect. In that case, the usual solution is to use modular methods instead. That may be possible here too, but we have not investigated this.

So at the moment, the most efficient method is as follows: first diagonalize, then use Algorithm Conic from section 4. If we started with a conic over $\mathbb{Q}(t_1, \ldots, t_n)$ then we may encounter a conic over $\mathbb{Q}$ later in the

computation (in the recursive step 3b of Algorithm Conic). It is possible that diagonalizing the $\mathbb{Q}(t_1, \ldots, t_n)$ conic increased the discriminant of the $\mathbb{Q}$ conic encountered later. To prevent unnecessarily large integer factorization, one must make a change at this point in our algorithm: Once we reach a diagonal $\mathbb{Q}$ conic, we must reverse the diagonalization and obtain a non-diagonal conic over $\mathbb{Q}$. This non-diagonal $\mathbb{Q}$ conic can also be obtained by specializing $t_1, \ldots, t_n$ in the original non-diagonal $\mathbb{Q}(t_1, \ldots, t_n)$ conic. Then compute a point on this non-diagonal conic with [S05]. Apply to that point the same transformation (with $t_1, \ldots, t_n$ specialized) as was used to diagonalize the original $\mathbb{Q}(t_1, \ldots, t_n)$ conic. This way one finds a point on the diagonal $\mathbb{Q}$ conic without having to factor its discriminant (which could be much larger than the discriminant of the non-diagonal $\mathbb{Q}$-conic, see [S05] for an example).

6. Examples

Example 1. Here $K = \mathbb{Q}(t_1, t_2)$. Let $a := 1$, $b := 2t_2^2 + 2t_1t_2 - t_1^2$, and $c := -3t_2^4 - 4t_1t_2^3 + 8t_1^2t_2^2 + 16t_1^3t_2 - 48t_1^4$.

The implementation does not always find the same solution, for an explanation see section 2.2. As output, one encounters for example

$$(t_2^3 + t_1t_2^2 + 4t_1^2t_2 - 4t_1^3 \ : \ t_2^2 - 4t_1^2 \ : \ t_2)$$

but also

$$(t_2^3 + 2t_1^2t_2 + 8t_1^3 \ : \ t_2^2 + 2t_1t_2 - 4t_1^2 \ : \ t_2 + t_1).$$

Example 2. $a := t_1^2 + 1$, $b := -(t_1^2 + 1)t_2^2 + (4t_1^2 + 4)t_2 - 1$, $c := (2t_1^3 - 10t_1^2 + 2t_1 - 9)t_2^4$
$-(6t_1^3 + 6t_1 - 2)t_2^3 + (t_1^4 - 8t_1^3 - 2t_1^2 - 6t_1 - 2)t_2^2 - (4t_1^4 + 2t_1^2 - 2t_1 - 2)t_2 - 1$.

The implementation may find

$$(t_2^3 - t_2^2 + t_2 - 1 \ : \ t_2^2 + t_2 + t_1 \ : \ 1)$$

but may also find a more complicated solution, depending on which solubility certificate it chose.

References

[CM98] T. Cochrane, P. Mitchell, *Small solutions of the Legendre equation*. J. Number Theory **70** (1998), no. 1, pp. 62–66.

[CR03] J. Cremona, D. Rusin, *Efficient solution of rational conics*. Math. Comp. **72** (2003), no. 243, pp. 1417–1441.

[Gauss] C. F. Gauss, *Disquisitiones Arithmeticae*. Springer-Verlag, 1986.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász, *Factoring polynomials with rational coefficients*. Math. Ann. **261** (1982), no. 4, pp. 515–534.

[Magma] W. Bosma, J. Cannon, C. Playoust, *The Magma algebra system. I. The user language*. J. Symbolic Comput., **24** (1997), 235–265. Computational algebra and number theory (London, 1993). See also `http://magma.maths.usyd.edu.au/magma/`.

[Maple] M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, *Maple 6 Programming Guide*. Waterloo Maple Inc. (Waterloo, Canada, 2000).

[R97] M. Reid, *Chapters on Algebraic Surfaces, Chapter C: Guide to the classification of surfaces*. In J. Kollár (Ed.), IAS/Park City lecture notes series **3** (1993), AMS, Providence R.I., 1997, 1–154. See also `www.maths.warwick.ac.uk/~miles/surf/ParkC/chC.ps`.

[Sch98] J. Schicho, *Rational parametrization of real algebraic surfaces*. Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (Rostock), ACM, New York, 1998, 302–308.

[Sch00] J. Schicho, *Proper parametrization of surfaces with a rational pencil*. Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (St. Andrews), ACM, New York, 2000, 292–300.

[S05] D. Simon, *Solving quadratic equations using reduced unimodular quadratic forms*. Math. Comp. **74** (2005), no. 251, pp. 1531–1543.

[W06] C. van de Woestijne, *Surface Parametrisation without Diagonalisation*. Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation (Genoa), ACM, New York, 2006, 340–344.

Mark van Hoeij
Department of Mathematics
Florida State University
Tallahassee, FL 32306-3027, USA
E-mail: `hoeij@math.fsu.edu`
URL: `http://www.math.fsu.edu/~hoeij/`

John Cremona
School of Mathematical Sciences
University of Nottingham
University Park, Nottingham, NG7 2RD, UK
E-mail: `john.cremona@nottingham.ac.uk`
URL: `http://www.maths.nott.ac.uk/personal/jec/`