

RECOGNIZING AND PARAMETRIZING CURVES ISOMORPHIC TO A LINE

CHI-MING LAM, VLADIMIR SHPILRAIN, AND JIE-TAI YU

ABSTRACT. Some time ago, Shpilrain and Yu reported an algorithm for deciding whether or not a polynomial $p \in K[x, y]$ is a coordinate, or, equivalently, whether or not a plane curve $p(x, y) = 0$ is isomorphic to a line. Here K is any constructible field of characteristic 0. In this paper, we show that their algorithm requires $O(n^2)$ field operations, where n is the degree of a given polynomial. We also show how their algorithm can be used to find a polynomial parametrization of a plane curve $p(x, y) = 0$ which is isomorphic to a line. This requires $O(n^2 \log_2 n)$ field operations.

1. INTRODUCTION

Let $K[x, y]$ be the polynomial algebra in 2 variables over a field K of characteristic 0. We say that a polynomial $p \in K[x, y]$ is a coordinate if there is an automorphism of $K[x, y]$ that takes x to p . By the famous theorem of Abhyankar and Moh [1], this is equivalent to saying that the curve $p(x, y) = 0$ in the affine plane K^2 is isomorphic to a line.

By the well known result of Jung [6] and van der Kulk [7], every coordinate of $K[x, y]$ is tame, where K is any field. That means, every coordinate of $K[x, y]$ can be obtained from x by applying a sequence of affine transformations and triangular automorphisms of the following types:

- (i) $(x, y) \longrightarrow (x + h(y), y)$ for some $h(y) \in K[y]$, $\deg_y h(y) \geq 2$;
- (ii) $(x, y) \longrightarrow (x, y + h(x))$ for some $h(x) \in K[x]$, $\deg_x h(x) \geq 2$.

By now, many algorithms for recognizing coordinates of $K[x, y]$ are known; see e.g. [5], [10], [11], or monographs [3], [9] and references thereto. With so many algorithms around, it becomes natural to ask for the one which is faster and/or consumes less resources than other ones.

Recently, Gutierrez, Rubio and Schicho [5] have addressed the question of the complexity of one of the algorithms for recognizing coordinates of $K[x, y]$ as well as of the auxiliary algorithm for finding a polynomial parametrization of a plane curve $p(x, y) = 0$ which is isomorphic to a line. They showed that either algorithm requires $O(n^3 \log_2 n)$ field operations, where n is the degree of a given polynomial $p(x, y)$.

The purpose of the present paper is to estimate the complexity of the algorithm for recognizing coordinates of $K[x, y]$ given by Shpilrain and Yu [10]. This complexity

1991 *Mathematics Subject Classification*. Primary 14Q05; Secondary 13B25, 13P10, 68W30.

Key words and phrases. Polynomial algebras, automorphisms, parametrization, plane curves, complexity.

The first author was partially supported by HKU Postgraduate Studentship in 2001-03. The third author was partially supported by a Hong Kong RGC-CERG grant.

turns out to be $O(n^2)$ field operations, which makes us believe that this algorithm is actually the best possible (as far as complexity is concerned). It also requires very little computer memory since (1) the complexity of the output decreases with every recursion step of the algorithm, and (2) at every recursion step, the algorithm only utilizes the output of the immediately preceding step. A brief description of the algorithm is given in Section 2, and in Section 3, we estimate the complexity of this and the auxiliary algorithm for finding a polynomial parametrization of a plane curve $p(x, y) = 0$ which is isomorphic to a line. As pointed out in [5], parametric representations of curves are important for geometric modeling; in particular, they are industrial standard in CAD systems. We show that one of the algorithms in [10] yields a parametrization algorithm that requires $O(n^2 \log_2 n)$ field operations. The most time-consuming part of the latter algorithm is evaluating the image of a given polynomial under a triangular automorphism of $K[x, y]$. It seems that this kind of computation cannot be avoided by any known algorithm, which makes us believe that the $O(n^2 \log_2 n)$ estimate cannot be improved.

Finally, we note that the degree of a polynomial p does not seem to be the most adequate measure of its complexity as far as most real-life applications are concerned. A more adequate measure appears to be the number of monomials that occur in p with non-zero coefficients, together with the set of exponents and the set of coefficients; this more accurately reflects the amount of information one has to input in order to describe a polynomial. For example, to describe a polynomial of the form x^N , we only need $\log_2 N$ bits of information, not N . This “informational complexity” was introduced by Kolmogorov and is now called Kolmogorov complexity, see [8].

2. THE ALGORITHMS

Let $>$ be the lexicographic ordering on the set of monomials in x and y , with $x > y$. For any $p \in K[x, y]$, by $lm(p)$ we denote the leading monomial of p with respect to $>$, and by $lc(p)$ the coefficient at the leading monomial of p .

We start by reproducing the algorithm for recognizing coordinate of $K[x, y]$ given by Shpilrain and Yu in [10].

Algorithm 2.1. *Let $p := p(x, y)$ be a polynomial of $K[x, y]$.*

Step 1: Take $q_1 := p_x = \frac{\partial p}{\partial x}$ and $q_2 := p_y = \frac{\partial p}{\partial y}$.

Step 2: If $lm(q_1)$ is not divisible by $lm(q_2)$ or vice versa, then p is not a coordinate. If $lm(q_1) = h \cdot lm(q_2)$ (respectively $lm(q_2) = h \cdot lm(q_1)$) for a monomial h , then go to Step 3.

Step 3: Set $q'_1 = q_1 - \frac{lc(q_1)}{lc(q_2)} \cdot h \cdot q_2$ (respectively $q'_2 = q_2 - \frac{lc(q_2)}{lc(q_1)} \cdot h \cdot q_1$). If $lm(q'_1)$ (respectively $lm(q'_2)$) is 1, then p is a coordinate. If $q'_1 = 0$ (respectively $q'_2 = 0$), then p is a coordinate if and only if $lm(q_2) = 1$ (respectively $lm(q_1) = 1$). If $q'_1 \neq 0$ and $lm(q'_1) \neq 1$, replace (q_1, q_2) by (q'_1, q_2) . (Respectively, if $q'_2 \neq 0$ and $lm(q'_2) \neq 1$, replace (q_1, q_2) by (q_1, q'_2)). Then go to Step 2.

We note that if $p = p(x, y)$ is a coordinate polynomial, then, by [2, Theorem 6.8.5] and [10, Theorem 1.4], at Step 3 of Algorithm 2.1 one can choose a polynomial (not necessarily a monomial) h such that either $h = h(y) \in K[y]$ and $lm(q_1 - h(y) \cdot q_2) <$

$lm(q_2)$ or $h = h(x) \in K[x]$ and $lm(q_2 - h(x) \cdot q_1) < lm(q_1)$. Thus, Algorithm 2.1 is very similar to the familiar Euclidean algorithm for finding the g.c.d.

The next algorithm applies to a given coordinate polynomial $p = p(x, y)$ and outputs a sequence of elementary (i.e., triangular or affine) automorphisms that takes p to x . This algorithm can be also considered a refinement of Algorithm 2.1, i.e., it can be also used for recognizing coordinates because if the input is not a coordinate, then Step 2 will eventually fail.

Algorithm 2.2. INPUT. $p := p(x, y)$, a coordinate of $K[x, y]$.

OUTPUT. A sequence of elementary automorphisms of $K[x, y]$ that takes $p(x, y)$ to x .
 Step 1 (INITIALIZATION): Let $(q_1, q_2) := (p_x, p_y)$. Let $S = \emptyset$ be the empty sequence (this is going to be a collector of elementary automorphisms).

Step 2: From [2, Theorem 6.8.5] and [10, Theorem 1.4], we know that there is either $h = h(y) \in K[y]$ or $h = h(x) \in K[x]$ such that $lm(q_1 - h(y) \cdot q_2) < lm(q_2)$ (respectively $lm(q_2 - h(x) \cdot q_1) < lm(q_1)$). Find such h (see the observation (3) below), then go to Step 3.

Step 3: Define the automorphism $\psi := (x - \int h(y)dy, y)$ (respectively, $\psi := (x, y - \int h(x)dx)$). Set $q'_1 = q_1 - h(y) \cdot q_2$ (respectively, $q'_2 = q_2 - h(x) \cdot q_1$). If $q'_1 = 0$, then $\pi\psi(p) = \mu x + \nu$ for some $\mu \in K^*$, $\nu \in K$, where $\pi = (y, x)$, the permutation automorphism. In this case, add ψ and π into S and go to Step 4. (If $q'_2 = 0$, then $\psi(p) = \mu x + \nu$. In this case, add ψ into S and go to Step 4.) Otherwise, replace $p = p(x, y)$ by $p = p(x - \int h(y)dy, y)$ (respectively, by $p = p(x, y - \int h(x)dx)$), and add ψ into S . Then go to Step 2.

Step 4: Add the automorphism $\rho_{\mu, \nu} : x \rightarrow \frac{1}{\mu}(x - \nu)$, $y \rightarrow y$ into S , output S , and stop.

The output of Algorithm 2.2 (i.e., the sequence of elementary automorphisms that takes $p(x, y)$ to x) can be used to obtain a parametrization of the curve $p(x, y) = 0$ if we apply this sequence to the pair $(0, t)$ (which is a parametrization of the line $x = 0$) instead of the pair (x, y) in the natural way. For example, a parametrization of the curve $x + y^2 = 0$ would be $(0 - t^2, t) = (-t^2, t)$. It corresponds to the automorphism $\varphi : x \rightarrow x - y^2$, $y \rightarrow y$ that takes $x + y^2$ to x .

We now make three observations relevant to Algorithms 2.1 and 2.2.

- (1) $h(y)$ may be a constant only if $p(x, y)$ is linear. Therefore, at all recursion steps of Algorithm 2.2 except, perhaps, the last one, we have $\deg_y h(y) \geq 1$.
- (2) Suppose $h(y)$ is in $K[y]$ with $\deg_y h(y) = m$. To evaluate $\int h(y)dy$, it takes $m + 1$ divisions. More specifically, if $h(y) = a_m y^m + \dots + a_0$, we may denote by $h(y)$ the $(m + 1)$ -tuple $h = (a_m, \dots, a_0)$. Then integration of $h(y)$ is equivalent to the following transformation of tuples:

$$(a_m, \dots, a_i, \dots, a_0) \rightarrow \left(\frac{a_m}{m}, \dots, \frac{a_i}{i}, \dots, \frac{a_0}{1}, 0 \right).$$

- (3) Suppose there is $h(y) \in K[y]$ such that $lm(p_x(x, y) - h(y)p_y(x, y)) < lm(p_y(x, y))$. Such an $h(y)$ can then be found from division of $p_x(x, y)$ by $p_y(x, y)$. If $\deg(p) = n$, then both $p_x(x, y)$ and $p_y(x, y)$ obviously have $O(n^2)$ monomials, so the division requires $O(n^2)$ field operations.

3. THE COMPLEXITY OF RECOGNIZING AND PARAMETRIZING COORDINATES

In this section, we estimate the complexity of Algorithm 2.2 for finding an automorphism φ (if it exists) of $K[x, y]$ such that $\varphi(p) = x$. As we have mentioned in the previous section, finding such an automorphism immediately leads to a parametrization of the curve $p(x, y) = 0$. The estimate will be given with respect to the degree $\deg p$ of a given polynomial p . First, we will find an upper bound for the number of field operations at each recursion step of Algorithm 2.2. We need some preliminary observations. The following lemma is well known, see e.g. [3, Lemma 5.1.2].

Lemma 3.1. *Let $\varphi = \sigma^{\delta_1} \tau_1 \tau_2 \cdots \tau_s$ be an automorphism of $K[x, y]$, where $\delta_1 = 0$ or 1 , σ and τ_i are affine and triangular automorphisms respectively. Let $\tau_i = (x + f_i(y), y)$ (“type 1”) or $(x, y + f_i(x))$ (“type 2”) with $d_i = \deg f_i \geq 2$, and suppose that for every i , the automorphisms τ_i and τ_{i+1} are not of the same type. Then*

$$\deg(\varphi) = d_1 \cdot d_2 \cdot \dots \cdot d_s.$$

We are now ready to prove the following

Lemma 3.2. *Suppose p is a coordinate of $K[x, y]$ with $\deg p = n$. In Algorithm 2.2, the number of field operations required for finding an elementary automorphism $\varphi := (x - \int h(y)dy, y)$ (or $\varphi := (x, y - \int h(x)dx)$) is $O(n^2)$.*

Proof. Without loss of generality, we may assume there is $h(y)$ in $K[y]$ such that $lm(p_x(x, y) - h(y)p_y(x, y)) < lm(p_y(x, y))$. Obviously, $m = \deg h(y) \leq \deg p_x = n - 1$. By observation 3 in the end of the previous section, the number of field operations for finding $h(y)$ is $O(n^2)$.

Furthermore, to construct an elementary automorphism, integration of $h(y)$ is required. By observation 2, this takes $m + 1 \leq n$ field operations. Therefore, the total number of field operations required for finding the automorphism is $O(n^2)$. \square

Lemma 3.3. *Let the polynomial $p(x, y)$ at Step 3 of Algorithm 2.2 have degree n . Then the number of field operations required for evaluating $p(x - \int h(y)dy, y)$ at Step 3 is $O(n^2 \log_2 n)$.*

Proof. It is sufficient to show that the number of field operations required for evaluating $p(x - \mu \cdot y^k, y)$, $\mu \in K^*$, $1 \leq k \leq n$, is $O(n^2 \log_2 n)$. This follows from the fact that computing a composition of two polynomials of degree at most n requires $O(n^2 \log_2 n)$ field operations. This, in turn, follows from two known facts: (1) evaluating a polynomial $q(x)$ of degree less than m at m different points requires $O(m \log_2 m)$ field operations (see e.g. [4, p. 295]), and (2) interpolating a polynomial of degree less than m using (at most) m different points requires $O(m \log_2 m)$ field operations (see e.g. [4, p. 297]). If we now let $m = n^2$ and observe that the degree of $p(x - \int h(y)dy, y)$ is less than n^2 (since the degree of $h(y)$ is less than n), we get the result. \square

Lemma 3.4. *The number of recursion steps in Algorithm 2.2 is $O(\log_2 n)$.*

Proof. Each recursion step in Algorithm 2.2 corresponds to a triangular automorphism either of type 1 (i.e., of the form $(x + f_i(y), y)$) or of type 2 (i.e., of the form $(x, y + f_i(x))$), where $\deg f_i \geq 2$ (because $\deg h(y) \geq 1$, see observation (1) in the end of the previous

section), and no two successive recursion steps correspond to triangular automorphisms of the same type.

Thus, the output of Algorithm 2.2 is an automorphism φ of the form $\sigma^{\delta_1} \tau_1 \tau_2 \cdots \tau_s$, where $\delta_1 = 0$ or 1 , σ and τ_i are affine and triangular automorphisms respectively, and for every i , τ_i and τ_{i+1} are not of the same type. Then, by Lemma 3.1, $n = \deg(\varphi) = d_1 \cdot d_2 \cdot \dots \cdot d_s \geq 2^s$ since $d_i = \deg \tau_i \geq 2$. Therefore, $\log_2 n \geq s$, hence the number of recursion steps is $O(\log_2 n)$. \square

Theorem 3.5. (1) *The number of field operations in Algorithm 2.1 for recognizing coordinates is $O(n^2)$, where n is the degree of the given polynomial $p(x, y)$.*

(2) *The number of field operations in Algorithm 2.2 for finding a parametrization of a given curve $p(x, y) = 0$ isomorphic to a line, is $O(n^2 \log_2 n)$.*

Proof. (1) To prove the statement about Algorithm 2.1, we recall that, by observation (3) in the end of the previous section, a single step of that algorithm requires $O(m^2)$ field operations, where m is the maximum degree of the two polynomials which are processed by the algorithm at this particular step. Note that at the first step of Algorithm 2.1, $m = n - 1$, so the first step requires $O(n^2)$ field operations. At each subsequent step (except, perhaps, the last one), the maximum degree of the two polynomials (q_1, q_2) is reduced by at least the factor of 2, by Lemma 3.1. Therefore, the complexity of Algorithm 2.1 is $O(n^2 + (\frac{n}{2})^2 + (\frac{n}{4})^2 + \dots) = O(n^2)$ field operations.

(2) As we see from Lemmas 3.2 and 3.3, the most time-consuming part of each recursion step of Algorithm 2.2 is evaluating $p(x - \int h(y)dy, y)$. For a polynomial $p(x, y)$ of degree n , this takes $O(n^2 \log_2 n)$ field operations. By Lemma 3.4, the number of recursion steps in Algorithm 2.2 is $O(\log_2 n)$. This may make it seem that the complexity of the algorithm is $O(n^2 \log_2^2 n)$. However, only at the first step of the recursion is the degree of $p(x, y)$ equal to n ; at each subsequent step (except, perhaps, the last one), the degree of $p(x, y)$ is reduced by at least the factor of 2, by Lemma 3.1. Therefore, by Lemma 3.3, the complexity of the algorithm is actually bounded, for some constant C , by $C \cdot (n^2 \log_2 n + (\frac{n}{2})^2 \log_2(\frac{n}{2}) + (\frac{n}{4})^2 \log_2(\frac{n}{4}) + \dots) \leq C \cdot n^2 \log_2 n \cdot \sum_k \frac{1}{2^k} = O(n^2 \log_2 n)$.

This completes the proof. \square

Example 3.6. *In [5], two recursion steps are required for finding a parametrization of $p(x, y) = 4x^2 + 8xy^3 - 8xy - 2x + 4y^6 - 8y^4 - 2y^3 + 4y^2 + 3y - 1$. The parametrization itself is:*

$$x = t - \frac{29}{64} + \frac{59}{4}t^2 - 60t^4 + 64t^6$$

$$y = \frac{5}{4} - 4t^2$$

Since $p_x = 8x + 8y^3 - 8y - 2$ and $p_y = 24xy^2 - 8x + 24y^5 - 32y^3 - 6y^2 + 8y + 3$, we have $p_y - (3y^2 - 1)p_x = 1$. After integrating $h(y) = 3y^2 - 1$, we obtain the automorphism $\varphi := (x - y^3 + y, y)$. Then evaluate: $\varphi(p(x, y)) = -1 - 2x + 4x^2 + y$. This latter polynomial (more formally, the corresponding curve) can already be parametrized, just "by inspection", as $x = t$, $y = 1 + 2t - 4t^2$. Then we compute:

$$\begin{aligned}(t, 1 + 2t - 4t^2)\varphi &= (t - (1 + 2t - 4t^2)^3 + (1 + 2t - 4t^2), 1 + 2t - 4t^2) \\ &= (-3t - 4t^2 + 40t^3 - 96t^5 + 64t^6, 1 + 2t - 4t^2).\end{aligned}$$

Thus, our parametrization of $p(x, y) = 0$ is:

$$\begin{aligned}x &= -3t - 4t^2 + 40t^3 - 96t^5 + 64t^6 \\ y &= 1 + 2t - 4t^2\end{aligned}$$

We see therefore that our parametrization is different from that of [5]. Both parametrizations have the same degree, but our parametrization has smaller space complexity (at least, in this particular example). Indeed, the collection of coefficients that corresponds to our parametrization is $\{-3, -4, 40, -96, 64, 1, 2, -4\}$, which takes up approximately 30 bits of memory, whereas the collection corresponding to the parametrization in [5] is $\{1, -29, 64, 59, 4, -60, 64, 5, 4, -4\}$, which takes up approximately 45 bits of memory.

Acknowledgement. We are grateful to the referee for numerous helpful comments.

REFERENCES

- [1] S. S. Abhyankar, T.-T. Moh, *Embeddings of the line in the plane*, J. Reine Angew. Math. **276** (1975), 148–166.
- [2] P. M. Cohn, *Free rings and their relations*, Second edition, Academic Press, 1985.
- [3] A. van den Essen, *Polynomial automorphisms and the Jacobian Conjecture*, Progress in Math., Birkhäuser, Basel 2000.
- [4] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2nd edition, 2003.
- [5] J. Gutierrez, R. Rubio, J. Schicho, *Polynomial parametrization of curves without affine singularities*, Computer Aided Geometric Design **19** (2002), 223–234.
- [6] H. W. E. Jung, *Über ganze birationale Transformationen der Ebene*, J. Reine und Angew. Math. **184** (1942), 167–174.
- [7] W. van der Kulk, *On polynomial rings in two variables*, Nieuw Archief voor Wiskunde (3) **1** (1953), 33–41.
- [8] M. Li and P. M. B. Vitanyi, *An introduction to Kolmogorov complexity and its applications*. Graduate Texts in Computer Science, 2nd ed., Springer, 1997.
- [9] A. A. Mikhalev, V. Shpilrain, and J.-T. Yu, *Combinatorial Methods: Free Groups, Polynomials, and Free Algebras*, Springer-Verlag, New York, 2003.
- [10] V. Shpilrain and J.-T. Yu, *Polynomial automorphisms and Gröbner reductions*, J. Algebra **197** (1997), 546–558.
- [11] P. Wightwick, *Equivalence of polynomials under automorphisms of \mathbb{C}^2* , J. Pure Appl. Algebra **157** (2001), 341–367.

DEPARTMENT OF MATHEMATICS, THE UNIVERSITY OF HONG KONG, POKFULAM ROAD, HONG KONG

E-mail address: tommylam@hkusua.hku.hk

DEPARTMENT OF MATHEMATICS, THE CITY COLLEGE OF NEW YORK, NEW YORK, NY 10031

E-mail address: shpil@groups.sci.cuny.cuny.edu

DEPARTMENT OF MATHEMATICS, THE UNIVERSITY OF HONG KONG, POKFULAM ROAD, HONG KONG

E-mail address: yujt@hkusua.hku.hk