Beiträge zur Algebra und Geometrie
Contributions to Algebra and Geometry
Volume 37 (1996), No. 2, 309-328.

On the Complexity of Parametrizing Curves

Michal Mňuk,[1] J. Rafael Sendra,[2] and Franz Winkler[1]

[1] *Research Institute for Symbolic Computation, Johannes-Kepler University, A-4040 Linz, Austria*

[2] *Departamento de Matemáticas, Universidad de Alcalá, Spain*

Abstract. Given a rational algebraic plane curve $\mathcal{C}$ in implicit representation we analyze the bit complexity of an algorithm to describe $\mathcal{C}$ by parametric equations. Estimates for subalgorithms for computing the standard decomposition of singularities and the genus are also obtained.

1. Introduction

Rational parametrization of algebraic curves is one of the basic computational problems in constructive algebraic geometry. A completely symbolic approach to this problem has been published in [11] and it has been implemented in the program system CASA ([4, 9]). The main goal of this paper is to give a thorough theoretical worst case complexity analysis of this parametrization algorithm.

In the first part we limit the complexity analysis to the case in which the curve has rational coefficients and only ordinary singularities, i.e. no blowing up is necessary to resolve them. In the Section 4 we discuss questions related to the presence of neighboring points.

Since the main steps of the parametrization algorithm consist of solving systems of algebraic equations in two variables, solving linear systems, and computing gcd's and resultants of polynomials of two or three variables, it is easy to see that the complexity is polynomial in the size of the input. In this paper we give a precise analysis in terms of bit complexity of the parametrization problem and some of its important subproblems.

[1]Supported by the "Fonds zur Förderung der wissenschaftlichen Forschung" under project number P8573-PHY and the "EC Project POSSO" No. 6846.

[2]Partially supported by DGICYT A.I. Spain–Austria HU-007 and ÖAD Int. Akt. 16.

Section 2 contains a summary of complexity bounds for basic integer and polynomial arithmetic algorithms (gcd, polynomial remainder sequences, etc.). Section 3 is dedicated to the computing time analysis of the parametrization problem. In Subsection 3.1, description and analysis of an algorithm for the decomposition of the set of singularities is given. Subsection 3.2 is devoted to the main algorithm for parametrization. A brief description of and a detailed analysis in terms of bit complexity are provided. In order to obtain tighter bounds for special cases we distinguish whether coordinates of all singularities are rational or not. In Subsections 3.2.1 and 3.2.2 we give the analysis for both cases. Section 4 discusses the complexity of algorithms designed so far in the case when the input curve possesses neighboring singularities.

2. Complexity of integer and polynomial arithmetic

In this section we briefly recall complexity bounds for basic algorithms for integers and polynomials. For some of them we give also short proofs to make the methods used here more transparent. Throughout this paper we use classical algorithms for both integers and polynomials. Even though this will not yield the best possible estimate of the complexity the main goal of this paper is to reveal computationally expensive parts of the parametrization algorithm. For detailed description of algorithms in this section we refer to [6, 7], etc. For elementary introduction the reader may refer to [2].

2.1. Integer arithmetic

The computational complexity of operations over integers is measured in terms of length of input, i.e. the number of bits needed to store the input into the memory. By length(a), $a \in \mathbb{Z}$ we denote $\lfloor \log_2(|a|) \rfloor + 2$ (we take the most significant bit as sign bit).

Remark 2.1. Throughout this paper all logarithms are taken to the base 2.

The following proposition summarizes classical results on integer arithmetic.

Proposition 2.2. *Let $a, b \in \mathbb{Z}$ and l_a, l_b be the length of a and b, respectively. Then it holds:*

(i) The worst case complexity for adding a and b is bounded by $\mathcal{O}(\max\{l_a, l_b\})$ and

$$\mathrm{length}(a+b) = \mathcal{O}(\max\{l_a, l_b\})$$

(ii) The worst case complexity for multiplying a and b is bounded by $\mathcal{O}(l_a l_b)$ and

$$\mathrm{length}(ab) = \mathcal{O}(l_a + l_b)$$

(iii) The worst case complexity for computing a^n is bounded by $\mathcal{O}((n\, l_a)^2)$.

(iv) The worst case complexity for computing $a \mod b$ ($|a| > |b|$) is bounded by $\mathcal{O}(l_b(l_a - l_b + 1))$.

(v) The worst case complexity for computing the greatest common divisor of a and b is bounded by $\mathcal{O}(l_a(l_b - l_c + 1))$, where l_c is the length of the gcd.

2.2. Polynomial arithmetic

Before we start to give complexity analysis of polynomial arithmetic we need to define some measures for the size of polynomials. Let $a = a_n x_r^n + \cdots + a_0$ be a multivariate polynomial in $\mathbb{Z}[x_1, \ldots, x_{r-1}][x_r]$ with $a_i \in \mathbb{Z}[x_1, \ldots, x_{r-1}]$. The complexity of various operations with a is determined by the degree of a (regarded as a univariate polynomial in x_r) and by the size of its coefficients. In the sequel, we recall three well-known notions of a norm. In order to obtain an estimate of the size of coefficients we define the *max-norm* of a by

$$\|a\|_{\max} := \max_{i=0,\ldots,n} \{\|a_i\|_{\max}\}.$$

However this norm does not comprise any information on the degree of the polynomial. Hence numerous complexity estimates are given in terms of the *1-norm* defined by

$$\|a\|_1 := \sum_{i=0}^{n} \|a_i\|_1$$

or the *2-norm* defined by

$$\|a\|_2 := \Big(\sum_{i=0}^{n} \|a_i\|_2^2\Big)^{1/2}.$$

For $a \in \mathbb{Z}$ we define

$$\|a\|_{\max} = \|a\|_1 = \|a\|_2 = |a|.$$

We have obvious relations among these norms. For $a \in \mathbb{Z}[x_1, \ldots, x_r]$ it holds

$$\begin{aligned} \|a\|_{\max} &\leq \|a\|_1 && (1)\\ \|a\|_{\max} &\leq \|a\|_2 && (2)\\ \|a\|_1 &\leq n^r\,\|a\|_{\max} && (3)\\ \|a\|_2 &\leq n^{r/2}\,\|a\|_{\max} && (4) \end{aligned}$$

Throughout this paper, for $a \in \mathbb{Z}[x_1, \ldots, x_r]$, we will use $L_{\max}(a)$ and $L_1(a)$ to abbreviate $\mathrm{length}(\|a\|_{\max})$ and $\mathrm{length}(\|a\|_1)$, respectively.

Proposition 2.3. *Let* $A, B \in \mathbb{Z}[x_1, \ldots, x_r]$, $n = \max\{\deg_{x_i}(A) \mid i = 1, \ldots, r\}$, $m = \max\{deg_{x_i}(B) \mid i = 1, \ldots, r\}$, *and* $L_A = L_{max}(A)$, $L_B = L_{max}(B)$. *Then we have:*

(i) The worst case complexity for adding A *and* B *is* $\mathcal{O}(k^r L)$ *where* $k = \max\{m, n\}$, $L = \max\{L_A, L_B\}$.

(ii) The worst case complexity for multiplying A *and* B *is bounded by* $\mathcal{O}(L_A L_B n^r m^r)$.

(iii) The worst case complexity for computing the pseudoquotient and pseudoremainder of A *and* B *w.r.t.* x_r ($\deg_{x_r}(A) \geq \deg_{x_r}(B)$) *is*

$$\mathcal{O}(n^r\, m^{r-1}(n - m + 1)((n - m + 2)L_B + L_A)L_B).$$

Furthermore, for univariate polynomials the length of the max-norm of quotient and remainder is dominated by $\mathcal{O}((n - m + 1)L_B + L_A)$.

Proof. First two assertions are obvious. Let $n_r := \deg_{x_r}(A)$, $m_r := \deg_{x_r}(B)$. Computation of pseudo-quotient and pseudo-remainder can be described by the following recurrence relation:

$$\begin{aligned} A^{(0)} &\leftarrow A \\ A^{(k)} &\leftarrow \mathrm{lc}(B)A^{(k-1)} - x_r^{\deg_{x_r}(A^{(k-1)})-m_r+1}\,\mathrm{lc}(A^{(k-1)})B \end{aligned}$$

where $k = 1, \ldots, \deg_{x_r}(A) - m_r + 1$. We see that

$$\|A^{(i)}\|_{\max} = \mathcal{O}(2^i \|A\|_{\max} \|B\|^i_{\max}).$$

Each loop of the above recurrence requires $\mathcal{O}(n)$ multiplications of $(r-1)$-variate polynomials with lengths of max-norms $\mathcal{O}((n-m+1)L_B + L_A)$ and L_B. Combining these results together we get a bound as stated in the theorem. □

Corollary 2.4. *Let A, B be univariate polynomials over a field, $n \geq \max\{\deg(A), \deg(B)\}$, $e \in \mathbb{N}$, $L_A = L_{max}(A)$, $L_B = L_{max}(B)$. The worst case complexity for computing $C \equiv A^e \mod B$ is bounded by*

$$\mathcal{O}(n^2 \log e(eL_A + n \log eL_B)^2)$$

Moreover, $L_{max}(C) = \mathcal{O}(eL_A + n \log eL_B)$.

Proof. Without loss of generality we may assume $e = 2^k$. We use the well-known algorithm for exponentiation based on repeated squaring. See Algorithm 1.

EXPPOLY(A, B, e)
Input: univariate polynomials A, B, and $e \in \mathbb{N}$, $e = \sum_{i=0}^{k} e_i 2^i$
Output: $A^e \mod B$

1. $S_0 \leftarrow 1$; $T_0 \leftarrow A$;
   for $i = 0$ **to** k **do**
2.   **if** $e_i = 1$ **then** $S_i \leftarrow S_{i-1}T_{i-1} \mod B$
3. $T_i \leftarrow T_{i-1}^2 \mod B$
   end

Algorithm 1:

Let us estimate the length of T_i in step 3. We easily obtain $L_{\max}(T_{i-1}^2) = \mathcal{O}(\log n + 2L_{\max}(T_{i-1}))$. Using results of Proposition 2.3 we conclude $L_{\max}(T_i) = \mathcal{O}(2L_{\max}(T_{i-1}) + nL_B)$. By induction $L_{\max}(T_i) = \mathcal{O}(2^i L_A + niL_B)$. Hence upon termination

$$L_{\max}(A^e \mod B) = \mathcal{O}(eL_A + n \log eL_B).$$

When counting bit operations we may restrict the analysis to step 3. In the i-th loop, the multiplication costs $\mathcal{O}(n^2(2^i L_A + niL_B)^2)$ and computing residuum $\mathcal{O}(n^2(nL_B + 2^i L_A + niL_B)L_B)$. The overall complexity is then

$$\mathcal{O}(n^2 \log e(eL_A + n \log eL_B)^2).$$

□

We will often need estimates of norms of polynomial divisors. The following lemma can be found in [8].

Lemma 2.5 (Landau-Mignotte-bound). *Let A and C be univariate polynomials with integral coefficients. Moreover let $C = AB$ for some $B \in \mathbb{Z}[x]$. Then*

(i) $\|B\|_{max} \leq \mathcal{O}(2^n \|C\|_2) = \mathcal{O}(2^n n^{1/2} \|C\|_{max}) = \mathcal{O}(2^n \|C\|_{max})$

(ii) $L_{max}(B) = \mathcal{O}(n + L_{max}(C))$

where $n = \deg(C)$.

In the next proposition, we summarize some basic results on polynomial remainder sequences, resultants, polynomial gcd's and on solving linear systems (cf. [1] and others).

Proposition 2.6. *Let $A, B \in \mathbb{Z}[x_1, \ldots, x_r], n$ the maximum degree in any variable of A and B, and $L_{max} := \max\{L_{max}(A), L_{max}(B)\}$. Then the following holds:*

(i) The worst case complexity of the subresultant chain algorithm over $\mathbb{Z}[x_1, \ldots, x_{r-1}][x_r]$ is bounded by $\mathcal{O}(n^{2r+2}(r \log n + L_{max})^2)$. Furthermore, the subresultant coefficients are bounded in length by $\mathcal{O}(n(\log n + L_{max}))$, and the degree of the k-th subresultant is at most k.

(ii) The worst case complexity of Brown's modular gcd algorithm is

$$\mathcal{O}(n^{2r+1}(r \log n + L_{max})^2).$$

(iii) The worst case complexity of Collins' modular resultant algorithm is

$$\mathcal{O}(n^{2r+1}(r \log n + L_{max}) + n^{2r}(r \log n + L_{max})^2) =$$
$$\mathcal{O}(n^{2r+1}(\log n + L_{max}) + n^{2r}(\log n + L_{max})^2)$$

(iv) The worst case complexity of Bareiss' algorithm for a linear system over $\mathbb{Z}[x_1, \ldots, x_r]$ is bounded by $\mathcal{O}(k^{2r+5} n^{2r} L^2)$, where k is the order of the matrix of the system, L is the maximum length of 1-norms of its entries, and n bounds the degrees.

In the course of parametrization we will have to compute primitive part of polynomials in $\mathbb{Z}[x_1][x_2]$. Thus, we will need bounds for computing the gcd of several polynomials in $\mathbb{Z}[x_1]$.

Corollary 2.7. *Let $G = \{g_1, \ldots, g_s\}$ be a set of univariate polynomials over $\mathbb{Z}$, $\Gamma := \max\{\|g_i\|_{max} | g_i \in G\}$, $L := \text{length}(\Gamma)$, and $n := \max\{\deg(g_i) | g_i \in G\}$. The worst case complexity for computing $\gcd(g_1, \ldots, g_s)$ is*

$$\mathcal{O}(sn^3(n^2 + nL + L^2)).$$

Moreover, the max-norm of $\gcd(g_1, \ldots, g_s)$ is bounded by

$$\|\gcd(g_1, \ldots, g_s)\|_{max} = \mathcal{O}(2^n \Gamma)$$
$$L_{max}(\gcd(g_1, \ldots, g_s)) = \mathcal{O}(n + L)).$$

Proof. Without loss of generality we may assume $s = 2^k$ for some k. We will proceed by divide-and-conquer. Using the following recurrence

$$S_j^{(0)} \leftarrow g_j,\ j = 1, \ldots, 2^k$$
$$S_j^{(i)} \leftarrow \gcd(S_{2j-1}^{(i-1)}, S_{2j}^{(i-1)}),\ j = 1, \ldots, 2^{k-i},\ i = 1, \ldots, k.$$

Upon termination the element $S_1^{(s)}$ will contain the result. In the i-th step 2^{k-i} gcd's of pairs of polynomials are computed. From Lemma 2.5 we see that the max-norm of polynomials in the i-th step is bounded by $\mathcal{O}(2^n\Gamma)$ (as they all divide some of g_i's). Using Proposition 2.6 we may bound the number of bit-operations by

$$O\Big(\sum_{i=1}^{k} \frac{s}{2^i} n^3 (n + L)^2)\Big).$$

Evaluating the closed form of this sum at $k = \log s$ we get the assertion of the corollary. □

3. Parametrization algorithm

In this section we briefly describe an algorithm for parametrizing plane curves, that we will analyze in next subsections. For further details we refer to [11].

Basically, the parametrization process consists of two parts. First, the standard decomposition of singularities (including the neighboring ones, in general) is determined. In this step, the set of all singularities is decomposed according to their multiplicities. In the course of this process the genus is determined yielding a decision of rationality of the curve. Second, using these informations a rational parametrization is computed.

In the sequel, algorithms corresponding to the two steps from above are outlined. For this purpose, we will assume that the given curve is rational and therefore the algorithm will not compute the genus. The curve is supposed to be in regular position (i.e. no line parallel to axes cut the curve in more than one singularity). This can always be achieved by a suitable change of coordinates (cf. [10]).

3.1. Standard decomposition of singularities

Let K be a field of characteristic zero in which all the field operations can be carried out effectively. For two univariate polynomials A, B over a field, let $\mathrm{res}(A, B)$ denote the resultant of A and B, $\mathrm{subres}(A, B, x)$ the subresultant with respect to x, and A' the derivative of A. The Algorithm 2 on the next page yields the standard decomposition of the set of singularities.

Remark 3.1. Some steps in the above algorithm have to be clarified theoretically and computationally. In step 5.3. the polynomials $\bar{A}_i$ defining the families of singularities are obtained. The x_1 coordinates of all the corresponding $(i+1)$-fold affine points of $\mathcal{C}$ are exactly the roots of $\bar{A}_i$. Then, in step 5.5. x_2-coordinates of singularities are computed. Let p be an irreducible factor of $\bar{A}_i$. As the curve is supposed to be in regular position, for each α with $p(\alpha) = 0$ there is exactly one β such that $(\alpha : \beta : 1)$ is a singular point of F. Hence in each

SINGULARITY(F)
Input: $F \in K[x_1, x_2, x_3]$ - defining homogeneous polynomial of an irreducible plane curve $\mathcal{C}$ of degree n in regular position
Output: standard decomposition of singularities

1. $\mathcal{F} \leftarrow \emptyset$;
2. $f \leftarrow F(x_1, x_2, 1)$; $B_1 \leftarrow f$;
3. $\bar{B}_1 \leftarrow \gcd(\mathrm{res}_{x_2}(f, \frac{\partial f}{\partial x_1}), \mathrm{res}_{x_2}(f, \frac{\partial f}{\partial x_2}))$;
4. $B_1 \leftarrow \frac{\bar{B}_1}{\gcd(\bar{B}_1, \bar{B}_1')}$;
5. **for** $i \geq 2$ **while** $\deg(B_{i-1}) > 0$ **do**
5.1. $\bar{B}_i \leftarrow \gcd(\mathrm{res}_{x_2}(f, \frac{\partial^i f}{\partial x_1^i}), \mathrm{res}_{x_2}(f, \frac{\partial^i f}{\partial x_1^{i-1}\partial x_2}), \ldots, \mathrm{res}_{x_2}(f, \frac{\partial^i f}{\partial x_1 \partial x_2^{i-1}}), \mathrm{res}_{x_2}(f, \frac{\partial^i f}{\partial x_2^i}))$;
5.2. $B_i \leftarrow \frac{\bar{B}_i}{\gcd(\bar{B}_i, \bar{B}_i')}$;
5.3. $\bar{A}_{i-1} \leftarrow \frac{B_{i-1}}{B_i}$;
5.4. **if** $\deg(\bar{A}_{i-1}) > 0$ **then**
5.5. $q_{i-1}(x_2) \leftarrow$
$\tilde{p}_{i-1}(x_1)x_2 - p_{i-1}(x_1) = \mathrm{subres}_1(f(x_1, x_2), \frac{\partial f}{\partial x_1}(x_1, x_2), x_2)$
$\bmod \bar{A}_{i-1}(x_1)$;
5.6. $\mathcal{F}^*_{i-1} \leftarrow \{(\alpha : \frac{p_{i-1}(\alpha)}{\tilde{p}_{i-1}(\alpha)} : 1)\}_{\bar{A}_{i-1}(\alpha)=0}$;
5.7. determine the expanded neighboring graph $\mathcal{F}_{i-1}$ of $\mathcal{F}^*_{i-1}$;
5.8. $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}_{i-1}$;
end
5.9. $i \leftarrow i + 1$;
end
6. **repeat** the process, dehomogenizing w.r.t. x_2 and then w.r.t. x_1;
7. **return** $\mathcal{F}$

Algorithm 2:

$\mathbb{Q}(\alpha)[x_2]$ ($p(\alpha) = 0$, p being a factor of $\bar{A}_i$) we have $\deg_{x_2} \gcd(f(\alpha, x_2), \frac{\partial f}{\partial x_1}(\alpha, x_2)) = 1$. This verifies the correctness of step 5.5. Finally, the set of $i+1$-fold singularities is described by

$$\{(\alpha\tilde{p}_i(\alpha) : p_i(\alpha) : \tilde{p}_i(\alpha))\}_{\bar{A}_i(\alpha)=0}.$$

In case $\bar{A}_i$ is reducible, more families of singular points may be put together.

Remark 3.2. The output of algorithm SINGULARITY is a family of singular points separated according their multiplicities. Note that $\{(m(\alpha) : n(\alpha) : p(\alpha))\}_{\bar{A}_i(\alpha)=0}$ stands for a set of $\deg(\bar{A}_i)$ of $i+1$-fold points.

Now we are going to analyze the computing time of the algorithm SINGULARITY. Let $F(x_1, x_2, x_3)$ be a homogeneous polynomial defining an algebraic rational plane curve $\mathcal{C}$ of degree n, in regular position, and such that all the singularities of $\mathcal{C}$ are ordinary. Note, that the algorithm SINGULARITY works for algebraic curves of any genus. It computes the neighborhood graph of $\mathcal{C}$ from which it is then trivial to read off the genus. Thus, at the same time, we obtain a decision procedure for determining the rationality of $\mathcal{C}$. The analysis applies to curves of arbitrary genus.

Theorem 3.3. *Let $L_F := \text{length}(\|F\|_{max})$. The worst case complexity of the algorithm SINGULARITY is $\mathcal{O}(n^{12}(n\log n + L_F)^2)$.*

Proof. One has basically to analyze the loop in step 5. The loop in step 5 will be executed at most $\mathcal{O}(n)$ times. Let $T_{\text{SING}}(i,j)$ be the time of execution of step 5.j in the i-th loop. Then if I denotes the set of all the different multiplicities of points of $\mathcal{C}$ it holds

$$T_{\text{SING}} = \sum_{i\in I}\sum_{j=1}^{11} T_{\text{SING}}(i,j).$$

Let us consider now the worst case complexity of the execution of i-th loop in step 5. We may assume to have all derivatives of F precomputed and stored in a table. It can be easily verified that this step does not affect the overall complexity.

Step 5.1. One has to calculate $i+1$ derivatives of the form $G_k = \frac{\partial^i f}{\partial x_1^k \partial x_2^{i-k}}$, $i+1$ resultants of the type $R_k = \text{res}_{x_2}(f, G_k)$, and $\gcd(R_0,\dots,R_i)$. Using Proposition 2.6 we have for $j=0,\dots,i$

$$L_{\max}(\frac{\partial^j f}{\partial x_k^j}) = \mathcal{O}(i\log n + L_F)$$
$$L_{\max}(\text{res}_{x_k}(f, \frac{\partial^j f}{\partial x_k^j})) = \mathcal{O}(n(i\log n + L_F)).$$

Then it follows that the time for computing the $i+1$ resultants is dominated by $i(n^5(i\log n + L_F) + n^4(i\log n + L_F)^2)$ which is $\mathcal{O}(in^5(i\log n + L_F)^2)$. Finally, the time to compute the greatest common divisor is $\mathcal{O}(n^6 i(n^4 + n^4(i\log n + L_F) + (n(i\log n + L_F))^2)) = \mathcal{O}(in^{10}(i\log n + L_F)^2)$. Hence

$$T_{\text{SING}}(i,1) = \mathcal{O}(in^{10}(i\log n + L_F)^2).$$

Moreover, $L_{\max}(\bar{B}_i) = \mathcal{O}(n^2 + n(i\log n + L_F))$ and $\deg(\bar{B}_i) \le n^2$.

Step 5.2. Since the modular gcd algorithm of Brown computes the gcd and the cofactors, it holds that the step 5.2. is bounded in time by the time of computing the $\gcd(\bar{B}_i, \bar{B}_i')$. Now, using Proposition 2.6, and the fact that $L_{\max}(\bar{B}_i) = \mathcal{O}(n(i\log n + L_F))$, it follows that

$$T_{\text{SING}}(i,2) = \mathcal{O}(n^8(i\log n + L_F)^2).$$

We also observe that, since B_i is a divisor of $\text{res}_{x_2}(f, \frac{\partial^i f}{\partial x_1^i})$, it holds that $\text{length}(B_i) = \mathcal{O}(n(i\log n + L_F))$.

Step 5.3. Applying Proposition 2.3, one has that

$$T_{\text{SING}}(i,3) = \mathcal{O}(n^6(i\log n + L_F)^2).$$

Step 5.5. Using Proposition 2.6 and that $L_{\max}(\frac{\partial f}{\partial x_1}) = \mathcal{O}(\log n + L_F)$, one has that the subresultant computation requires time $\mathcal{O}(n^6(\log n + L_F)^2)$. Now, we have to bound the time for computing $\tilde{p}(x_1) \mod \bar{A}_i$ and $p(x_1) \mod \bar{A}_i$. We observe that $\max\{L_{\max}(p_i), L_{\max}(\tilde{p}_i)\} = \mathcal{O}(n\log n + nL_F)$ and $\max\{\deg(p_i), \deg(\tilde{p}_i)\} = \mathcal{O}(n^2)$. Furthermore for $\bar{A}_i$, we use the fact that $\bar{A}_i$ divides $\operatorname{res}_{x_2}(f, \frac{\partial f}{\partial x_1})$ (all roots of $\bar{A}_i$ are x_1-coordinates of some singularities), and therefore $\deg(\bar{A}_i) \leq n^2$, $L_{\max}(\bar{A}_i) = \mathcal{O}(n(n + L_F))$ (Lemma 2.5). Thus applying Proposition 2.3 one concludes that the time for the two divisions is bounded by $\mathcal{O}(n^8(\log n + L_F)^2)$. Hence

$$T_{\text{SING}}(i, 5) = \mathcal{O}(n^8(\log n + L_F)^2)).$$

By the hypothesis all singularities are ordinary hence no expansion of the neighboring tree is needed and step 5.7. does not contribute to the complexity. Then the worst case complexity for algorithm SINGULARITY can be bounded by

$$T_{\text{SING}} = \mathcal{O}(\sum_{i\in I} in^{10}(i\log n + L_F)^2).$$

Thus, since $\sum_{i\in I} i^2 = \mathcal{O}(n^2)$,

$$T_{\text{SING}} = \mathcal{O}(n^{12}(n\log n + L_F)^2).$$ □

Corollary 3.4. *Let $\mathcal{C}$ be a irreducible plane algebraic curve with no neighboring singularities given by a homogeneous polynomial F of degree n. Then the genus of $\mathcal{C}$ can be computed in $\mathcal{O}(n^{12}(n\log n + L_F)^2)$ bit operations.*

Proof. Using all informations on singularities of $\mathcal{C}$, we have an algorithm for computing the genus. See Algorithm 3.

Genus($\mathcal{F}$)

1. $\mathcal{F} \leftarrow$Singularity($\mathcal{C}$); $g \leftarrow 0$;
   for all $S \in \mathcal{F}$ **do**
2. $\quad m \leftarrow$ multiplicity(S); $g \leftarrow g + m(m-1)/2$;
   end
   return $(n-1)(n-2)/2 - g$;

Algorithm 3:

We have $\text{Card}(\mathcal{F}) = \mathcal{O}(n^2)$. Hence the running time is dominated by the computation in step 1. □

In the following theorem we summarize some results on singularities which have partially been proved in course of complexity analysis of the algorithm SINGULARITY.

Theorem 3.5. *Let $F \in K[x_1, x_2, x_3]$ be a homogeneous polynomial of degree n defining an algebraic rational plane curve $\mathcal{C}$ with only ordinary singularities and let $L_F := L_{max}(F)$. Then, if*

$$\bigcup_{i\in I}\{(m_{i,1}(\alpha) : m_{i,2}(\alpha) : m_{i,3}(\alpha))\}_{A_i(\alpha)=0}$$

is the standard decomposition of the set of singularities $\mathcal{S}(\mathcal{C})$ of $\mathcal{C}$ it holds that:

(i) $\deg(A_i) \le n^2,\ \deg(m_{i,j}) \le n^2$ *for* $i \in I,\ j = 1,2,3$;

(ii) $L_{max}(A_i) = \mathcal{O}(n(n + L_F))$ *for* $i \in I$;

(iii) $L_{max}(m_{i,j}) = \mathcal{O}(n^3(n + L_F))$ *for* $i \in I,\ j = 1,2,3$.

Proof. The first two statements have already been proved in Theorem 3.3. We also saw there, that any polynomial component $m_{i,j}(t)$ in the standard decomposition is congruent modulo $\bar{A}_i(t)$ to a polynomial dominated in length by $\mathcal{O}(n(\log n + L_F))$ and in degree by $\mathcal{O}(n^2)$. Therefore, using Proposition 2.3, one has that $L_{\max}(m_{i,j}) = \mathcal{O}(n^3(n + L_F))$. □

For rational singularities a better upper bound of the length may be given.

Proposition 3.6. *Let* $F \in K[x_1, x_2, x_3]$ *be a homogeneous polynomial of degree* n *defining an algebraic plane curve* $\mathcal{C}$. *Then, the length of any coordinate of any rational singularity of* $\mathcal{C}$ *is bounded by* $\mathcal{O}(n(\log n + L_{max}(F)))$.

Proof. Let $f(x_1, x_2) = F(x_1, x_2, 1),\ R_1(x_1) = \mathrm{res}_{x_2}(f, \frac{\partial f}{\partial x_1})$, and $R_2(x_1) = \mathrm{res}_{x_2}(f, \frac{\partial f}{\partial x_2})$. Then the x_1-coordinate of any rational affine singularity is a common rational root of R_1 and R_2. Therefore, the length of any x_1-coordinate of any rational singularity is bounded by the maximum of the lengths of R_1 and R_2 (cf. [8]),that is $\mathcal{O}(n(\log n + L_{\max}(F))$. Similarly, considering resultants w.r.t. x_1 one obtains the same bound for the x_2-coordinates. □

3.2. Parametrization

In this section we design an algorithm for parametrization of plane rational curves with only ordinary singularities. See Algorithm 4 on the next page. Note, that according to Corollary 3.4 we may apply the subsequent algorithm to any irreducible plane curve regardless whether it is rational or not. In a preprocessing step we may determine the genus and refuse any non-rational curve causing no asymptotic increase of computing time. We assume that POINTS(F,k) is an auxiliary algorithm that computes k simple points on the curve given by F over the smallest algebraic extension of K. Here, we do not want to go into details of construction of such algorithm. For a description we refer to [5, 12].

Remark 3.7. We briefly comment some steps of the algorithm. In step 6, H_a is forced to pass through the standard decomposition of the set of singularities with a specified multiplicity. To achieve it, we refer to [11].

The elimination of factors corresponding to fixed common points of the curve and H_a can be done by computing the primitive part of S_1 and S_2 w.r.t. x_2 and x_1, respectively, or explicitly by dividing by these factors, which are known in advance [11].

In the sequel, the complexity of the algorithm PARAMETRIZE is analyzed. For deriving complexity bounds, two different cases are distinguished. First, we consider only curves whose singularities are ordinary and rational. Second, we deal with the most general case that we consider: curves only with ordinary singularities; probably lying on some algebraic extension of the ground field, Furthermore, as we mentioned in the introduction, we do not take care of simple points on curves. We assume that as many simple points as the algorithm PARAMETRIZE needs are given. For actually finding them, we refer to [5], [12].

PARAMETRIZE($\mathcal{C}$)
Input: $F \in K[x_1, x_2, x_3]$ - homogeneous polynomial defining an irreducible rational curve $\mathcal{C}$ of degree n
Output: rational parametrization of $\mathcal{C}$

1. $\mathcal{F} \leftarrow \text{SINGULARITY}(F)$;
2. choose $a \in \{n-2, n-1, n\}$;
3. $k \leftarrow (a-n+2)n + (n-3)$;
4. $\{R_1, \ldots, R_k\} \leftarrow \text{POINTS}(F, k)$;
5. $H_a(x_1, x_2, x_3)$ is a linear system of curves of degree a with undetermined coefficients;
6. pass $H_a(x_1, x_2, x_3)$ through $\mathcal{F}$, such that, if P is an r-fold point in $\mathcal{F}$ (including the neighboring one), then P has multiplicity $r-1$ on H_a;
7. pass $H_a(x_1, x_2, x_3)$ through regular points $\{R_1, \ldots, R_k\}$;
8. **if** $a = n$ **then** pass H_a through an additional point not on $\mathcal{C}$;
9. solve the linear system of equations in the undetermined coefficients of H_a created in steps 6, 7 and 8, and substitute the solution in H_a;
10. $S_1(x_2) \leftarrow \text{res}_{x_1}(F(x_1, x_2, 1), H_a(x_1, x_2, 1))$;
11. $S_2(x_1) \leftarrow \text{res}_{x_2}(F(x_1, x_2, 1), H_a(x_1, x_2, 1))$;
12. remove the factors in S_1 and S_2 coming from fixed intersections of F and H_a;
13. solve the linear system of equations $\{S_1(x_2) = 0, S_2(x_1) = 0\}$, where x_1, x_2 are the unknowns. Say that $(R_1(t), R_2(t))$ is the solution;
14. **return** $(R_1(t), R_2(t))$

Algorithm 4:

3.2.1. Rational ordinary singularities

In this subsection we deal with curves having only rational ordinary singularities.

Let $F(x_1, x_2, x_3) \in K[x_1, x_2, x_3]$ be a homogeneous polynomial defining an algebraic rational curve $\mathcal{C}$ of degree n, in regular position, and such that all the singularities are ordinary and rational. We consider given simple points $\{R_1, \ldots, R_k\}$ on $\mathcal{C}$ with rational coordinates, where $k = (a-n+2)\,n + (n-3)$ for some $a \in \{n-2, n-1, n\}$.

Lemma 3.8. *The worst case complexity for the algorithm PARAMETRIZE, working on rational ordinary singularities, is*

$$\mathcal{O}(n^{12}(n \log n + L_F)^2 + n^{12} \max\{L_R, \log n + L_S\}^2)$$

where n is the degree of the curve, $L_F = L_{max}(F)$, L_R is the maximum length of the given rational simple points, and L_S is the maximum length of singularities.

Proof. Let $T_{\text{PARAM}}(i)$ be the time of execution of step i of the algorithm PARAMETRIZE. Then it holds that

$$T_{\text{PARAM}} = \sum_{i=1}^{14} T_{\text{PARAM}}(i).$$

From the Theorem 3.3 it follows that $T_{PARAM}(1) = \mathcal{O}(n^{12}(n \log n + L_F)^2)$. Steps 2 and 3 are irrelevant in terms of complexity. Also since $a \leq n$ one has that

$$T_{\mathrm{PARAM}}(5) = \mathcal{O}(n^2).$$

In the step 6, let $P_1, \ldots, P_m$ be the singularities of $\mathcal{C}$ and $r_1, \ldots, r_m$ the corresponding multiplicities. Let H_a be a generic tri-variate homogeneous polynomial with undetermined coefficients, i.e. $H_a = \sum_{i+j+k=a} c_{ijk} X^i Y^j Z^k$. Then, passing H_a through $\mathcal{F}$, with the specified multiplicities, implies to compute

$$\left\{ \left\{ \frac{\partial^{r_j - 2} H_a}{\partial x_1^{i_1} \partial x_2^{i_2} \partial x_3^{i_3}} (P_j) \right\}_{i_1+i_2+i_3=r_j-2} \right\}_{j=1,\ldots,m}.$$

This follows from the fact that i-th derivatives of H_a may be written as linear combinations of $i+1$-st derivatives (this follows from Euler's Theorem, cf. [3], p. 6).

Since the number of terms of each of the $r_j(r_j-1)/2$ polynomials to be evaluated at P_j is $(a-r_j+3)(a-r_j+4)/2$, then the time for the execution is

$$T_{\mathrm{PARAM}}(6) = \mathcal{O}(\sum_{j=1}^{m} \frac{r_j(r_j-1)}{2} \frac{(a-r_j+3)(a-r_j+4)}{2} t_j),$$

where t_j is the time for evaluating any monomial $i_1(i_1-1)\cdot\cdots\cdot(i_1-\alpha_1+1)\, i_2(i_2-1)\cdot\cdots\cdot(i_2-\alpha_2+1)\, i_3(i_3-1)\cdot\cdots\cdot(i_3-\alpha_3+1)\, X^{i_1-\alpha_1} Y^{i_2-\alpha_2} Z^{i_3-\alpha_3}$, $i_1+i_2+i_3=a$, $\alpha_1+\alpha_2+\alpha_3=r_j-2$ at P_j. Then t_j is dominated by the time for evaluating an expression of the form AB^a, where A represents the coefficient of the monomial, $\mathrm{length}(A) = \mathcal{O}(r_j \log a)$, and B is the coordinate of P_j of maximal length. The time to compute B^a is dominated by $(a\, \mathrm{length}(B))^2$ hence

$$t_j = \mathcal{O}(a^2\, \mathrm{length}(B)^2 + a\, \mathrm{length}(B) r_j \log a) = \mathcal{O}(a^2\, \mathrm{length}(B)\, (\mathrm{length}(B) + r_j))$$

Hence using $a \leq n$ and $\sum_{i=1}^{m} r_j(r_j-1)/2 = (a-1)(a-2)/2$, we obtain

$$T_{\mathrm{PARAM}}(6) = \mathcal{O}(n^6 L_S(L_S+n)) = \mathcal{O}(n^6 L_S^2 + n^7 L_S).$$

In the step 7, let t be the time for passing H_a through one simple point $R \in \{R_1, \ldots, R_k\}$. Then $t \leq \frac{(n+1)(n+2)}{2} t^*$, where t^* is now the time for evaluating any monomial of H_a at R. Thus, t^* is dominated by the time for computing B^a, where B is the coordinate of R_i of maximum length. Consequently,

$$T_{\mathrm{PARAM}}(7) = \mathcal{O}(kn^2a^2L_R^2) = \mathcal{O}(n^6 L_R^2).$$

In the step 8, if $a = n$ then coordinates of an additional point not on $\mathcal{C}$ can always be taken bounded by n (not all points $(k:1:0)$, $k = 0, \ldots, n$, may lie on $\mathcal{C}$). Forcing H_a to pass through $(k:1:0)$ will result in the linear equation $c_{010} + c_{110}k + \cdots + c_{a10}k^a = 0$. The time for computing its coefficients is

$$T_{\mathrm{PARAM}}(8) = \mathcal{O}(a^2 \log^2 a) = \mathcal{O}(n^2 \log^2 n).$$

In the step 9, a linear system with $\mathcal{O}(n^2)$ equations has to be solved. On the other hand the length of the entries in the system is dominated by $\mathcal{O}(n \log n + nL_S)$, for rows coming from step 6, and by $\mathcal{O}(nL_R)$, if the equation comes from steps 7 or 8. Thus, applying the Proposition 2.6, one has that the time for solving the system is dominated by

$$n^{10} \max\{n \log n + dL_S, nL_R\}^2 = \mathcal{O}(n^{12} \max\{L_R, \log n + L_S\}^2)$$

And since the substitution in H_a of the solution takes $\mathcal{O}(n^2)$ it follows that

$$T_{\mathrm{PARAM}}(9) = \mathcal{O}(n^{12} \max\{L_R, \log n + L_S\}^2).$$

Now, we are interested in the length of the resulting pencil H_a. Then, taking into account that every coordinate of the solutions of a linear system can be expressed as a fraction of determinants, it follows that

$$L_{\max}(H_a) = \mathcal{O}(n^3 \max\{L_R, \log n + L_S\})$$

In steps 10 and 11, we first observe that the maximal length of max-norms of polynomials in $\mathbb{Z}[x_1, x_2, t]$ (t being the remaining undetermined coefficient in the solution of the linear system in step 9) involved in the resultants is bounded by $\mathcal{O}(\max\{L_F, n^3 D\})$, where $D = \max\{L_R, \log n + L_S\}$. Therefore, applying the Proposition 2.6 one has that both steps are dominated in time by $n^7(\log n + \max\{L_F, n^3 D\}) + n^6 \max\{L_F, n^3 D\}^2$. But, since $n^3 \leq \max\{L_F, n^3 D\}$ it follows that

$$\begin{aligned} T_{\mathrm{PARAM}}(10) \approx T_{\mathrm{PARAM}}(11) &= \mathcal{O}(n^7(\log n + \max\{L_F, n^3 D\}) \\ &\quad + n^6(\log n + \max\{L_F, n^3 D\}^2)) \\ &= \mathcal{O}(n^6 \max\{L_F, n^3 D\}^2). \end{aligned}$$

On the other hand, from Proposition 2.6 we have that $L_{\max}(S_i) = \mathcal{O}(n^2 \max\{L_F, n^3 D\})$.

In the step 12, the quotient of S_i and a polynomial p_i, obtained as the product of all the factors produced by singular and regular points, has to be calculated. The computation of p_i takes $\mathcal{O}(n^4 D^2)$, $\deg(p_i) = \mathcal{O}(n^2)$ and $\mathrm{length}(p_i) = \mathcal{O}(n^2 D)$. Thus

$$T_{\mathrm{PARAM}}(12) = \mathcal{O}(n^6 \max\{L_F, n^3 D\}^2).$$

Finally, since the steps 13 and 14 are irrelevant for the complexity analysis, taking in account the result of Theorem 3.3 one concludes that

$$T_{\mathrm{PARAM}} = \mathcal{O}(n^{12}(n \log n + L_F)^2 + n^{12} \max\{L_R, \log n + L_S\}^2). \qquad \square$$

Remark 3.9.

1. In deriving complexity bounds for the algorithm PARAMETRIZE, in the rational ordinary singularity case, we have not utilized any special simplified version for the algorithm SINGULARITY. We have just applied the Theorem 3.3. A suitable version for rational singularities may reduce the upper bound.

2. In the proof of the previous lemma, we have implicitly assumed that the coordinates of the rational singularities are known, although algorithm SINGULARITIES returns families of points. This is not any loss of generality, since an optional step for detecting them can be inserted in the outline of SINGULARITY, without modifying the complexity bound in the Theorem 3.3

In the last part of this subsection we apply the Proposition 3.6 and the previous lemma to derive an upper bound depending only on the rational simple points.

Theorem 3.10. *The worst case complexity for algorithm PARAMETRIZE, working on rational ordinary singularities, is*

$$\mathcal{O}(n^{12}\max\{L_R, n(\log n + L_F)\}^2)$$

where n is the degree of the curve, $L_F = L_{max}(F)$, and L_R bounds the length of the given rational simple points.

Proof. By Lemma 3.8, we know that $\mathcal{O}(n^{12}(n\log n+L_F)^2+n^{12}\max\{L_R, \log n+L_S\}^2)$ bounds the worst case complexity of the algorithm. By Proposition 3.6 $L_S = \mathcal{O}(n(\log n + L_F))$. Combining these bounds we immediately obtain the assertion of the theorem. □

Corollary 3.11. *The worst case complexity of algorithm PARAMETRIZE, working on rational ordinary singularities, is bounded by $\mathcal{O}(n^{14}\log^2 n \max\{L_R, L_F\}^2)$.*

3.2.2. Standard families of ordinary singularities

This subsection is dedicated to analysis of the complexity in general. We consider curves having only ordinary singularities (i.e. without neighboring ones), possibly lying in some algebraic extensions of the ground field.

Let $F \in K[x_1, x_2, x_3]$ be a homogeneous polynomial defining an algebraic rational curve $\mathcal{C}$ in regular position of degree n with only ordinary singularities. We consider $\{R_1, \ldots, R_k\}$ to be given rational simple points on the curve, where $k = (a - n + 2)n + (n - 3)$ and $a \in \{n-2, n-1, n\}$. Then, the following lemma gives an upper bound for the worst case complexity in terms of the degree and the length of the curve, of the rational simple points and of the standard decomposition.

Theorem 3.12. *The worst case complexity for the algorithm PARAMETRIZE is*

$$\mathcal{O}(n^{26}(n + L_F)^2 + n^6 L_R^2)$$

where n is the degree of the curve, $L_F = \text{length}(\|F\|_{max})$, L_R is the maximum length of coordinates of simple points returned from the procedure POINTS.

Proof. We will analyze the algorithm PARAMETRIZE allowing the coordinates of singularities to lie in finite extensions of the ground field. Let T_{PARAM} have the same meaning as in the proof of Lemma 3.8. The complexity of first five steps is independent of the type of singularities and has already been determined.

Let $\mathcal{F}_S$ be the standard singularity decomposition computed in step 1. Then $\mathcal{F}_S$ may be written as

$$\mathcal{F}_S = \bigcup_{i \in I} \mathcal{F}_i,$$

where $\mathcal{F}_i = \{(m_i(\alpha), n_i(\alpha), p_i(\alpha))\}_{\bar{A}_i(\alpha)=0}$. In step 6, since $\mathcal{F}_i$ contains $(i+1)$-fold points, passing H_a through $\mathcal{F}_i$ implies to force the polynomials

$$\left\{ \frac{\partial^{i-1} H_a}{\partial x_1^{i_1} \partial x_2^{i_2} \partial x_3^{i_3}} \right\}_{i_1+i_2+i_3=i-1}$$

to vanish on $\mathcal{F}_i$. Therefore, if T_i is the time for forcing one of these $i(i+1)/2$ polynomials to vanish on $\mathcal{F}_i$, it holds that $T_{\mathrm{PARAM}}(6) = \mathcal{O}(\sum_{i \in I} i^2 T_i)$. In deriving bounds for T_i, we observe that computing time for obtaining

$$G(t) = \frac{\partial^{i-1} H_a}{\partial x_1^{i_1} \partial x_2^{i_2} \partial x_3^{i_3}} (m_i(t), n_i(t), p_i(t))$$

and $g(t) \equiv G(t) \mod \bar{A}_i(t)$ have to be analyzed. Since $\frac{\partial^{i-1} H_a}{\partial x_1^{i_1} \partial x_2^{i_2} \partial x_3^{i_3}}$ is a polynomial of degree $a-i+1$ with $\mathcal{O}(a-i+2)(a-i+3)/2)$ terms, it follows that, as $a \leq n$, $T_i = \mathcal{O}(n^2 t_i)$, where t_i is the time for computing $cB(t)^{3n} \mod \bar{A}_i$, with c being an integer of length $\mathcal{O}(i \log n)$, $B \in \{m_i, n_i, p_i\}$. Recall (by Theorem 3.5), $L_{\max}(B) = \mathcal{O}(n^3(n+L_F))$, $\deg(B) = \mathcal{O}(n^2)$, $L_{\max}(\bar{A}_i) = \mathcal{O}(n(n+L_F))$, $\deg(\bar{A}_i) = \mathcal{O}(n^2)$. Using Corollary 2.4 we obtain $L_{\max}(B(t)^{3n} \mod \bar{A}_i = \mathcal{O}(n^4(n+L_F))$ and $t_i = \mathcal{O}(n^{11}(n+L_F)^2)$. Hence

$$T_{\mathrm{PARAM}}(6) = \mathcal{O}((n^{15}(n+L_F)^2).$$

For steps 7 and 8, reasoning as in the proof of Lemma 3.8, it holds that

$$\begin{aligned} T_{\mathrm{PARAM}}(7) &= \mathcal{O}(n^6 L_R^2) \\ T_{\mathrm{PARAM}}(8) &= \mathcal{O}(n^3 \log n). \end{aligned}$$

In step 9, linear system of $\mathcal{O}(n^2)$ equations has to be solved. Its entries are univariate polynomials $p(t)$ with $L_{\max}(p) = \mathcal{O}(n^4(n+L_F))$ (thus also $L_1(p) = \mathcal{O}(n^4(n+L_F))$) and $\deg(p) = \mathcal{O}(n^2)$. Using Proposition 2.6 we obtain

$$T_{\mathrm{PARAM}}(9) = \mathcal{O}(n^{26}(n+L_F)^2).$$

In steps 10 and 11, length of max-norms of polynomials in $\mathbb{Z}[x_1, x_2, t]$ involved in the resultant are bounded by $\mathcal{O}(n^4(n+L_F))$, where applying Proposition 2.6 one has

$$T_{\mathrm{PARAM}}(10) \approx T_{\mathrm{PARAM}}(11) = \mathcal{O}(n^{14}(n+L_F)^2)).$$

In step 12 we first compute $\prod(x_1 - \omega_{i,1}(t))$ and $\prod(x_2 - \omega_{i,2}(t))$ where $\omega_{i,1}$ and $\omega_{i,2}$ are first and second coordinates of a singularity of F, respectively. Then an exact division of S_i by these polynomials is performed. Since the degree in x_i is $\mathcal{O}(n)$ and the length of max-norm is bounded by $\mathcal{O}(n^3(n+L_F))$ (3.5) using Proposition 2.3 we obtain

$$T_{\mathrm{PARAM}}(12) = \mathcal{O}(n^{11}(n+L_F)^2).$$

Finally, since steps 13 and 14 are irrelevant in terms of complexity, we conclude

$$T_{\mathrm{PARAM}} = \mathcal{O}(n^{26}(n+L_F)^2 + n^5 \log n L_R^2). \qquad \square$$

4. Curves with neighboring points

Up to now we considered and analyzed algorithms for curves devoid of complicated singularities, i.e. of those having no neighborhood points. Though curves with complex singularities are encountered very often. In this section we study the classical algorithm for computing the neighborhood tree, give basic analysis of its behavior in terms of complexity, and outline the impact on the algorithm SINGULARITY and PARAMETRIZE.

4.1. Computing neighboring singularities

Now we give a sketch of the algorithm for computing the neighborhood graph for a given absolutely irreducible plane curve. See Algorithm 5. As a neighboring singularity is just a usual singularity of a transformed curve we will use the representation as in Section 3.1. Though due to the rather complicated nature of neighboring singularities we will be forced to work in multiple extensions of the underlying field.

NEIGHBORHOOD TREE(P)
Input: A singular point P of a curve
Output: Neighborhood tree rooted in P

1. determine a change of coordinates T_1 moving the projective point $(0:0:1)$ to P;
2. determine a change of coordinates T_2 such that $F' = F \circ T_1 \circ T_2$ contains $(0:0:1)$ but none of $(1:0:0)$, $(0:1:0)$ and such that no axis is tangent at $(0:0:1)$ to F';
3. perform the quadratic transformation on F' centered at $(0:0:1)$ yielding a new curve F'';
4. compute the set $\mathcal{S}$ of singularities on $F'' \cap V(Z)$;
5. **for all** $R \in \mathcal{S}$ **do**
6. call NEIGHBORHOOD TREE recursively on R;

   end

Algorithm 5:

Remark 4.1. In the general version of the parametrization algorithm the above procedure will be called in step 5.7. of the algorithm SINGULARITY in Subsection 3.1.

Let us have a closer look at this algorithm. In each recursive call we are computing singular points hence possibly introducing new algebraic extensions. We will examine the two first recursive calls to NEIGHBORHOOD TREE.

(a) Let K be the coefficient domain of the original polynomial F defining the curve. Let P be a singularity of F computed by the algorithm SINGULARITY. Hence

$$P = (m(\alpha) : n(\alpha) : p(\alpha)),\ A(\alpha) = 0$$

where $A(U)$ is a squarefree univariate polynomial (not necessarily irreducible) with a root α. We will mimic the computing in $K[\alpha]$ by performing all computations in $K[U]/(A(U))$. However, care is needed when doing divisions.

(b) In the step 1. of the previous algorithm we set up a transformation T_1 which moves the point $(0:0:1)$ to one singularity. Then T_1 may contain algebraic numbers as coefficients. Following the transformed polynomial F'' has coefficients in $K[\alpha]$. So we have to perform all computations modulo $A(U)$.

(c) Obviously, each recursive call may introduce a new extension of the previously obtained field so that finally we will be computing in $K[\alpha_1, \ldots, \alpha_l]$. All computations in this field will be mimicked in the ring $K[U_1, \ldots, U_l]$ modulo the triangular system of polynomials

$$\begin{array}{r} A_1(U_1) \\ A_2(U_1, U_2) \\ \vdots \\ A_l(U_1, \ldots, U_l). \end{array}$$

For the sake of complexity consideration it is important to estimate the depth of the neighborhood graph. As we saw in the Section 3 the number of new variables has an immense impact on the theoretical complexity of the parametrization algorithm. In the sequel we will show that this number may be of the same order as the degree of the curve.

We prove some facts about the depth of the neighborhood graph now.

Theorem 4.2. *There is an infinite family of rational plane curves $\{\mathcal{C}_k\}$ where the depth of the neighborhood graph is $\Theta(\deg(\mathcal{C}_k))$ (i.e. of the same order as $\deg(\mathcal{C}_k)$).*

We will prove some auxiliary lemmas from which the theorem will immediately follow.

Lemma 4.3. *Let $F_k(X,Y,Z) = X^{2k+1} - Y^2 Z^{2k-1}$ and*

$$\Phi_i^{(k)} = Z^{2k-2i-1}(X+Y)^{2k-2i+1} A_i^{(k)}(X,Y,Z) - (XY)^{2k-1}(X-Y)^2 B_i^{(k)}(X,Y,Z) \tag{5}$$

where

$$A_1^{(k)} = 1 \tag{6}$$

$$A_i^{(k)} = A_{i-1}^{(k)}(XY, XY + XZ - YZ, (X+Y)Z)(2XY + XZ - YZ)^{2k-2i+3} \tag{7}$$

$$B_1^{(k)} = 1 \tag{8}$$

$$B_i^{(k)} = B_{i-1}^{(k)}(XY, XY + XZ - YZ, (X+Y)Z)(XY + XZ - YZ)^{2k-1}. \tag{9}$$

Then it holds

(i) For all $k \geq 1$, $\mathcal{C}_k = V(F_k)$ is a parametrizable plane curve.

(ii) $\Phi_i^{(k)}$ determines points in the i-th neighborhood of the point $(0:0:1)$ on $\mathcal{C}_k$.

(iii) For all $i \leq k-1$, $\Phi_i^{(k)}$ has a double non-ordinary point $(1:1:0)$.

Proof. It is easy to show the rationality of $\mathcal{C}_k$. The transformation $X = T^2$, $Y = T^{2k+1}$, $Z = 1$ yields the desired birational mapping.

In order to verify the second assertion we perform the following changes of coordinates in $\mathbb{P}^2$:

$$\begin{aligned} T_1 : X &\leftarrow Z \\ Y &\leftarrow Y+Z \\ Z &\leftarrow X \end{aligned} \qquad \begin{aligned} T_2 : X &\leftarrow X+Y \\ Y &\leftarrow -X+Y \\ Z &\leftarrow Z \end{aligned} \qquad \begin{aligned} T_3 : X &\leftarrow YZ \\ Y &\leftarrow XZ \\ Z &\leftarrow XY. \end{aligned}$$

The transformation T_1 moves the point $(0:0:1)$ to $(1:1:0)$, T_2 assures that $\Phi_i^{(k)}$ does not pass through $(1:0:0)$ and $(0:1:0)$, and no axis is a tangent to $\Phi_i^{(k)}$ at $(0:0:1)$. The mapping T_3 is the well known quadratic transformation. Putting them together we obtain

$$\begin{aligned} T : X &\leftarrow XY \\ Y &\leftarrow XY + XZ - YZ \\ Z &\leftarrow (X+Y)Z. \end{aligned}$$

First we apply T_1 and then T_3 to $\mathcal{C}_k$ getting $\Phi_1^{(k)}$. By induction on i it is not hard to verify Formulae 5 and 6.

For the last assertion of the lemma it is sufficient to consider

$$\frac{\partial \Phi_i^{(k)}(X,1,Z)}{\partial X} \text{ and } \frac{\partial \Phi_i^{(k)}(X,1,Z)}{\partial Z}$$

at $X = 1,\ Z = 0$. It is easy to see that $(1:1:0)$ is exactly a double point on $\Phi_i^{(k)}$, for $1 \leq i \leq k-1$. □

As we agreed on using multiple extensions whenever an enlargement of the ground field is required every such new extension is represented by a polynomial in one additional variable. From the Theorem 4.2 we see that the computation of the neighborhood graph may extend the ground field several times hence introducing a number of variables asymptotically equivalent to the degree of the curve. The complexity estimates from the Section 2 imply an exponential time growth for the algorithm SINGULARITY and hence for the parametrization when run on curves with neighboring points. This is due to the non-constant number of new variables introduced in the process of expanding the neighborhood graph.

Even though we have an explicit bound for the total number of points in the neighborhood graph of the curve $\mathcal{O}(n^2)$, n being the degree of the curve, in the course of the computation we perform arithmetic operations over polynomials with a growing number of variables. In order to stay below a polynomial time bound we would have to require a constant depth of the neighborhood graph of curves under consideration. The Theorem 4.2 shows that this is impossible.

Resuming, we have shown that there is an infinite family of rational curves on which our algorithm takes exponential time in the degree of the curve. Thus no polynomial time bound is possible using algorithms described above to parametrize curves with neighboring points. Only in a special case where the depth of the neighborhood graph is bounded by a constant we may achieve a polynomial running time.

5. Conclusions

Even though the parametrization algorithm is in use for a long time only little was known about its complexity. Mainly due to the prohibitive running time it is not feasible to determine its average behavior by testing it on a large set of data. In this paper, the worst case complexity for the special case of curves lacking neighboring singularities has been proved. Practical examples reveal that the worst case complexity determined in this paper is too pessimistic for most cases (curves of degree 10 have been parametrized in a reasonable amount of time). This is partially due to the high estimate for the length of coordinates of singularities. If this parameter was taken as an input measure, finer theoretical complexity could be obtained and also some bounds observed in practical computations might be matched closely.

In the course of the analysis of parametrization algorithm we obtained an estimate for the time needed to determine the genus of an irreducible curve and an algorithm to decompose the set of singularities.

Moreover, we have shown that allowing neighboring points may considerably increase the amount of time needed to parametrize curves. It turns out that using strategies described in this paper the time to parametrize a curve may be exponential in the degree of the input curve.

All algorithms described in this paper were implemented in the Maple package CASA ([9]) developed at the Research Institute for Symbolic Computation, Linz, Austria.

References

[1] Buchberger, B.; Collins, G. E.; Loos, R.: *Computer Algebra; Symbolic and Algebraic Computation.* Springer Verlag Wien–New York, second edition, 1982.

[2] Collins, G. E.; Mignotte, M.; Winkler, F.: *Arithmetic in basic algebraic domains.* In: Computer Algebra; Symbolic and Algebraic Computation (vgl. [1]), 184–220.

[3] Fulton, W.: *Algebraic Curves.* Addison–Wesley, 1989.

[4] Gebauer, R.; Kalkbrener, M.; Wall, B.; Winkler, F.: *CASA: A Computer Algebra Package for Constructive Algebraic Geometry.* In: S. M. Watt, editor, *ISSAC 91*, 403–410, Bonn, Germany, July 1991.

[5] Hilbert, D.; Hurwitz, A.: *Über die Diophantischen Gleichungen vom Geschlecht Null.* Acta Math. 1890, 217–224.

[6] Knuth, D. E.: *The Art of Computer Programming, vol. 2: Seminumerical Algorithms.* Addison–Wesley, second edition, 1981.

[7] Loos, R.: *Generalized polynomial remainder sequences.* In: Computer Algebra; Symbolic and Algebraic Computation (vgl. [1]), 115–137.

[8] Mignotte, M.: *Some useful bounds.* In: Computer Algebra; Symbolic and Algebraic Computation (vgl. [1]), 259–263.

[9] Mňuk, M.; Wall, B.; Winkler, F.: *CASA reference manual (version 2.2).* Technical report, Research Institute for Symbolic Computation, Linz, Austria, 1994. See also: http://www.risc.uni-linz.ac.at:/labs-info/compal/casa/casa.html.

[10] Sakkalis, T.; Farouki, R.: *Singular points of algebraic curves.* J. Symb. Comput. **9** (1990), 405–421.

[11] Sendra, J. R.; Winkler, F.: *Symbolic parametrization of curves.* J. Symb. Comput. **12**(6) (1991), 607–631.

[12] Sendra, J. R.; Winkler, F.: *Determining simple points on rational algebraic curves.* Technical Report RISC-93-23, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, 1993.

Received July 14, 1995