

ON THE PROBLEM OF DETECTING WHEN TWO IMPLICIT PLANE ALGEBRAIC CURVES ARE SIMILAR

Juan Gerardo Alcázar¹, Gema M. Diaz–Toca², and Carlos Hermoso³

¹Universidad de Alcalá (Spain)

²Universidad de Murcia (Spain)

³Universidad de Alcalá (Spain)

Abstract

We make use of the complex implicit representation in order to provide a deterministic algorithm for checking whether or not two implicit algebraic curves are related by a similarity. The algorithm has been implemented in the computer algebra system Maple 2016. The implementation can be freely downloaded from the webpage of one of the authors. Examples and evidence of the good practical performance of the algorithm are given.

Keywords: Similarities; implicit algebraic curves; curves recognition.
Mathematics Subject Classification 2010: 14C05, 68W30.

1 Introduction

We say that two algebraic curves are *similar* when there is a *similarity*, i.e. the composition of an isometry and a scaling, transforming one into the other. In other words, two curves are similar when they have the same shape, excepts perhaps for the position and the size. In this paper, we address the problem of checking whether or not two given algebraic curves are similar, under the assumption that the coefficients of the curves are rational numbers of elements of an algebraic extension (and therefore given in *exact* arithmetic). For this case, we provide a *symbolic*, therefore *deterministic*, algorithm to solve the problem. In the affirmative case, our algorithm can also find the similarities transforming one curve into the other.

The motivation for addressing this problem comes from computer algebra systems. Assume that a database of classical curves is stored in your favourite computer algebra system. Using the algorithms in this paper, the system can recognise a certain curve introduced by the user as one of the curves in the database. For instance, a user of a Dynamic Geometry package can be interested in checking whether a curve resulting from a certain construction is, say, a cardioid, a lemniscata, an epitrochoid, a deltoid, etc.

In order to solve the problem, we use the *complex representations* of the curves to be compared. Complex representations have already been used in [23, 25], where the *pose-estimation* problem is addressed, and in [13], where the computation of the symmetries of an algebraic planar curve is studied. In [23] the complex representation is exploited and combined with numerical strategies in order to give a solution to the problem. In [25] the transformation to be sought is decomposed into a rotation, which is approached in a deterministic way, and a translation, which is approximated. The approach in [13] is deterministic, but they treat a different problem. In our case, we use the complex representation of the curves to translate the problem into a bivariate polynomial system of equations with complex coefficients. Furthermore, except in certain singular cases we can also add a univariate equation to the system (corresponding to the rotation angle), which allows to solve the system in a fast and efficient way.

Our problem is related to some applied fields, and in fact some of the above references appear in these fields. For instance, a central problem in Pattern Recognition is to classify a given object. If the object to be studied is represented by a curve, the problem reduces to comparing this curve with other curves, previously classified and stored in a database. But this comparison must be carried out up to a similarity. In Computer Vision we find a close problem, called *pose estimation*, which has to do with identifying a certain object in a scene. Geometrically, the problem is equivalent to recognising a same object in two different positions. In turn, this amounts to checking if two objects are the same up to the composition of a translation and a rotation, so that the scaling factor does not change. The problem also appears in Computer Aided Geometric Design, since recognising similarities or even partial similarities in an image allows to reduce the amount of space needed to store it.

In the above references the input is typically a noisy image, sometimes with occluded parts, and quite often a point cloud; a common strategy is to adjust first an algebraic curve to the input, so that the comparison is carried out between algebraic curves that approximate two objects to be compared. The strategies used so far to solve the problem for algebraic curves are very

diverse: B-splines [12], Fourier descriptors [20], complex representations [23], Möebius transformations in the parameter space [1], statistics [8, 14, 16] (also in the 3D case), moments [21, 24], geometric invariants [26, 28, 29], Newton-Puiseux parametrizations [19], or differential invariants [4, 5, 27]. The interested reader may consult the bibliographies in these papers to find other references on the matter; the list is really very long.

Compared to these strategies, besides complex representations, we also employ usual tools in the field of computer algebra, like resultants, gcds and polynomial system solving. Our method has been implemented in the computer algebra system Maple 2016; the code can be freely downloaded from [9]. Our experiments confirm that the algorithm is efficient and fast.

It is worth, however, to compare our results with those of two recent papers, namely [11] and [3]. In the case of [11], the authors also consider a deterministic point of view to solve the more general problem of checking whether two rational curves, in arbitrary dimension, are related by an affine or projective transformation. However, the implicit case is not addressed in [11]. In [3], the problem of deterministically finding the symmetries of a given implicit algebraic curve or the similarities relating two given algebraic curves is investigated using certain properties and harmonic functions, in special the fact that the Laplacian operator commutes with orthogonal transformations. We can emphasize three main differences between our paper and [3]: (1) our algorithm is fully implemented in the computer algebra system Maple, and the implementation can be downloaded and freely used from [9]; even though the ideas in [3] are sound and fast, a complete implementation is not available. (2) The algorithm in [3] is not easy to adapt for curves with approximate coefficients, i.e. with floating point coefficients; even though a complete analysis of the case of curves with approximate coefficients is out of the scope of this paper, we believe that our algorithm is better suited for that. (3) The algorithm in [3] cannot be adapted to the more general problem of checking whether or not two given implicit algebraic curves are related by an affine, non-necessarily orthogonal, transformation, since the Laplacian operator does not commute with non-orthogonal transformations. Our algorithm, however, is again better suited for that, even though, again, a complete analysis would require a separate paper.

The structure of the paper is the following. Preliminaries are introduced in Section 2. In Section 3 we give an overview of the method, and show how to perform the first steps. In Section 4 we provide the algorithm itself. Details of the experimentation carried out in Maple, as well as examples and timings are provided in Section 5. Our conclusions and some future lines of work are provided in Section 6. Additionally, the paper has one appendix,

where we provide the proofs of several technical results in the paper.

2 Preliminaries.

We consider two plane algebraic curves \mathcal{C}_1 and \mathcal{C}_2 implicitly defined by two real, square-free polynomials (i.e. without irreducible factors of multiplicity higher than one) $f(x, y)$ and $g(x, y)$ of the same degree n , with coefficients in a field \mathbb{K} , where \mathbb{K} is either \mathbb{Q} or an algebraic extension of \mathbb{Q} . Additionally, we will assume that all the irreducible factors of f, g define *real* curves, i.e. curves with infinitely many real points. Thus, a curve like $f(x, y) = (x^2 + y^2)(x^2 - y^2 - 1)$, which has one factor, namely $x^2 + y^2$, not defining a real curve, would be excluded from our study. Finally, we assume that $\mathcal{C}_1, \mathcal{C}_2$ are not either lines or circles. We will justify these assumptions further in this section. Moreover, we will write

$$f(x, y) = f_n(x, y) + f_{n-1}(x, y) + \cdots + f_0(x, y),$$

where for $p = 0, 1, \dots, n$, $f_p(x, y)$ represents the homogeneous form of degree p of f ,

$$f_p(x, y) = \sum_{j=0}^p a_{p-j,j} x^{p-j} y^j.$$

Analogously,

$$g(x, y) = g_n(x, y) + g_{n-1}(x, y) + \cdots + g_0(x, y), \quad g_p(x, y) = \sum_{j=0}^p b_{p-j,j} x^{p-j} y^j.$$

In this paper we will use the *complexification* of the curves $\mathcal{C}_1, \mathcal{C}_2$ (see [13]). In more detail, given the mapping $(x, y) \rightarrow z = x + iy$ between the Euclidean plane and the complex line, the inverse of this mapping is

$$x = \frac{z + \bar{z}}{2}, \quad y = \frac{z - \bar{z}}{2i}, \tag{1}$$

where \bar{z} denotes the conjugate of the complex number z . Using this inverse map, the polynomial $f(x, y)$ is transformed into $F(z, \bar{z})$,

$$\begin{aligned} F(z, \bar{z}) &= f\left(\frac{z + \bar{z}}{2}, \frac{z - \bar{z}}{2i}\right) = \\ &= F_n(z, \bar{z}) + F_{n-1}(z, \bar{z}) + \cdots + F_0(z, \bar{z}), \end{aligned}$$

where $F_p(z, \bar{z})$, for $p \geq 1$, is a homogeneous polynomial of degree p in the variables z, \bar{z} , i.e.

$$F_p(z, \bar{z}) = \sum_{j=0}^p \alpha_{p-j,j} z^{p-j} \bar{z}^j. \quad (2)$$

We will refer to $F(z, \bar{z})$ as the *complex implicit representation* of \mathcal{C}_1 . One can check that for $p = 0, 1, \dots, n$, $f_p(x, y)$ gives rise to $F_p(z, \bar{z})$ and conversely. We will say that a term in $F(z, \bar{z})$ has *bidegree* $(n-k, k)$ if it can be written as $\xi \cdot z^{n-k} \bar{z}^k$, with $\xi \in \mathbb{C}$.

Analogously, applying (1) to $g(x, y)$ we reach $G(z, \bar{z})$,

$$G(z, \bar{z}) = G_n(z, \bar{z}) + G_{n-1}(z, \bar{z}) + \dots + G_0(z, \bar{z}), \quad G_p(z, \bar{z}) = \sum_{j=0}^p \beta_{p-j,j} z^{p-j} \bar{z}^j. \quad (3)$$

Definition 1 An affine map $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a similarity if it is bijective and there exists a real number $R > 0$ such that for each pair of points \mathbf{x}, \mathbf{y} ,

$$\|h(\mathbf{x}) - h(\mathbf{y})\|_2 = R \cdot \|\mathbf{x} - \mathbf{y}\|_2$$

where $\|\cdot\|_2$ denotes the Euclidean norm. We refer to R as the ratio of the similarity.

Notice that if $R = 1$ then h is an (*affine*) *isometry*, i.e. it preserves distances. Planar isometries are completely classified [7]. Under the composition operation, similarities form a group, and isometries form a subgroup of this group. Furthermore, any similarity can be decomposed into an isometry and a uniform scaling, where R is the similarity ratio.

Definition 2 Two curves $\mathcal{C}_1, \mathcal{C}_2$ are similar if and only if there exists a similarity h such that $h(\mathcal{C}_1) = \mathcal{C}_2$.

Notice that Definition 2 implicitly considers $\mathcal{C}_1, \mathcal{C}_2$ as embedded in \mathbb{C}^2 : indeed, if $h(\mathcal{C}_1) = \mathcal{C}_2$, every real or complex point $(x, y) \in \mathcal{C}_1$ satisfies that $h(x, y) \in \mathcal{C}_2$, and conversely. Thus, for instance the curves defined by $f(x, y) = (x^2 + y^2 + 1)(x^2 - y^2 - 1)$ and $g(x, y) = (x^2 + y^2 + 3)(y^2 - x^2 - 1)$ are *not* similar according to Definition 2: their real parts are certainly similar (two mutually orthogonal hyperbolas), but the complex parts do not match. This is the reason to require that $\mathcal{C}_1, \mathcal{C}_2$ do not have factors defining non-real curves. Under this assumption, every similarity is, in fact, a similarity of the real parts of $\mathcal{C}_1, \mathcal{C}_2$.

Let us justify now why we require that $\mathcal{C}_1, \mathcal{C}_2$ do not have multiple components. Under this assumption and since $\mathcal{C}_1, \mathcal{C}_2$ are algebraic, using Study's Lemma (see Section 1.3 of [10]) we can translate the *geometric* property of being similar into the following *algebraic* property:

$$g(h(x, y)) = \lambda f(x, y), \quad \lambda \in \mathbb{R}. \quad (4)$$

However, this translation does not necessarily work if $\mathcal{C}_1, \mathcal{C}_2$ have multiple factors. Consider for instance the curves defined by $f(x, y) = x^2(x + 1)y$, and $g(x, y) = x(x + 1)^2y$. Even though the varieties defined by both curves are the same, the identity map $h(x, y) = (x, y)$, which is, in particular, a (trivial) similarity does not satisfy Eq. (4). So we need the assumption on $\mathcal{C}_1, \mathcal{C}_2$ being square-free to benefit from Eq. (4). Notice that we can always check whether or not a bivariate polynomial $p \in \mathbb{K}[x, y]$ is square-free by checking whether $\gcd(p, p_x, p_y) = 1$, where p_x, p_y represent the partial derivatives of p with respect to x, y . For the computation of this gcd when \mathbb{K} is an algebraic extension, the interested reader can check [17].

In particular, since h is affine, Eq. (4) guarantees that f, g have the same degree, which is another of our assumptions. Finally, we require that $\mathcal{C}_1, \mathcal{C}_2$ are not lines or circles, because in that case the number of similarities between them is finite; in particular, if $\mathcal{C}_1, \mathcal{C}_2$ are not symmetric then there exists at most one similarity mapping one onto the other [1].

A similarity can either preserve or reverse the orientation. In the first case we say that it is *orientation-preserving*, and in the second case we say that it is *orientation-reversing*. By identifying \mathbb{R}^2 and \mathbb{C} , any orientation-preserving similarity can be written as $h(z) = \mathbf{a}z + \mathbf{b}$, where $\mathbf{a}, \mathbf{b} \in \mathbb{C}$. In the same way, any orientation-reversing similarity can be written as $h(z) = \mathbf{a}\bar{z} + \mathbf{b}$. In each case, the ratio R of the similarity is equal to $|\mathbf{a}|$.

Under our assumptions and taking advantage of Eq. (4), the curves $\mathcal{C}_1, \mathcal{C}_2$ are related by an orientation-preserving similarity $h(z) = \mathbf{a}z + \mathbf{b}$, if and only if there exists $\lambda \in \mathbb{R}, \lambda \neq 0$, such that

$$G(\mathbf{a}z + \mathbf{b}, \bar{\mathbf{a}}\bar{z} + \bar{\mathbf{b}}) = \lambda \cdot F(z, \bar{z}) \quad (5)$$

Also, $\mathcal{C}_1, \mathcal{C}_2$ are related by an orientation-reversing similarity $h(z) = \mathbf{a}\bar{z} + \mathbf{b}$, if and only if there exists $\lambda \neq 0$ such that

$$G(\mathbf{a}\bar{z} + \mathbf{b}, \bar{\mathbf{a}}z + \bar{\mathbf{b}}) = \lambda \cdot F(z, \bar{z}) \quad (6)$$

Notice that in the orientation-preserving case, $h(z) = \mathbf{a}z + \mathbf{b}$ is the composition of: (1) a rotation around the origin, of angle equal to θ , where

$\mathbf{a} = |\mathbf{a}| \cdot e^{i\theta}$; (2) a scaling, where the scaling factor is $|\mathbf{a}|$ and the center is the origin; (3) a translation of vector \mathbf{b} . In the orientation-reversing case, $h(z) = \mathbf{a}\bar{z} + \mathbf{b}$ is the composition of: (1) a symmetry with respect to the x -axis; (2) a rotation around the origin, of angle equal to θ , where $\mathbf{a} = |\mathbf{a}| \cdot e^{i\theta}$; (3) a scaling, where the scaling factor is $|\mathbf{a}|$ and the center is the origin; (4) a translation of vector \mathbf{b} .

In the following sections, we will focus on orientation-preserving similarities, since this case already captures the main aspects of our approach. The interested reader can check [2] for further details on the orientation-reversing case.

3 Overview of the method and first steps.

Equations (5) and (6), with $\mathbf{a}, \lambda \neq 0$, provide necessary and sufficient conditions for $\mathcal{C}_1, \mathcal{C}_2$ to be similar. In the orientation-preserving case, comparing the terms in $z^{m-j}\bar{z}^j$, $0 \leq j \leq m$, $0 \leq m \leq n$ at both sides of (5), we obtain a polynomial system \mathcal{S} , of at most $N = \binom{n+2}{2}$ equations with complex coefficients in the variables $\mathbf{a}, \bar{\mathbf{a}}, \mathbf{b}, \bar{\mathbf{b}}, \lambda$. In turn, this yields a real polynomial system in five variables, λ and the real and complex parts of \mathbf{a} and \mathbf{b} , of degree $n + 1$, where n is the degree of $\mathcal{C}_1, \mathcal{C}_2$. Then $\mathcal{C}_1, \mathcal{C}_2$ are related by an orientation-preserving similarity iff this polynomial system is consistent and has solutions with $\mathbf{a}, \lambda \neq 0$. This provides a naive way to check if $\mathcal{C}_1, \mathcal{C}_2$ are related by an orientation-preserving similarity.

In order to improve this approach, we will proceed in the following way:

- (i) *Compute the rotation:* if $\mathcal{C}_1, \mathcal{C}_2$ are related by an orientation-preserving similarity $h(z) = \mathbf{a}z + \mathbf{b}$, the curves $\mathcal{C}_1^{(n)}, \mathcal{C}_2^{(n)}$, defined by the forms of highest degree of $\mathcal{C}_1, \mathcal{C}_2$, i.e. the polynomials $f_n(x, y), g_n(x, y)$, are related by the (also orientation-preserving) similarity $\tilde{h}(z) = \mathbf{a}z$. Comparing $\mathcal{C}_1^{(n)}, \mathcal{C}_2^{(n)}$ we can capture the rotation angle θ (see Subsection 3.1). More precisely, calling $\omega = \text{tg}(\theta)$, we can write $\mathbf{a} = r \cdot (1 + i\omega)$. Furthermore, we can compute a polynomial P such that $P(\omega) = 0$, i.e. ω is a real root of P . Similarities with $\theta = \frac{\pi}{2}, -\frac{\pi}{2}$ must be studied separately.
- (ii) *Rewrite the constant λ :* by comparing appropriate terms at the left hand-side and the right hand-side of Eq. (5), we can write λ in terms of $\mathbf{a}, \bar{\mathbf{a}}$, i.e. in terms of r, ω .
- (iii) *Rewrite the variable \mathbf{b} :* again by comparing appropriate terms at the

left hand-side and the right hand-side of Eq. (5), we can write \mathbf{b} in terms of $\mathbf{a}, \bar{\mathbf{a}}$, i.e. in terms of r, ω .

- (iv) *Reduction to an advantageous bivariate system:* Taking (i), (ii), (iii) into account, Eq. (5) gives rise to a polynomial system in just two unknowns, r, ω , where ω is a solution of a univariate equation computed in (i). This system can be efficiently solved, thanks to the condition $P(\omega) = 0$, so the orientation-preserving similarities can be efficiently computed.

The scheme (i)-(iv) shows the generic situation. Some special situations must be treated differently. We want to point out that some of the manipulations carried out in (ii) and (iii) are similar, although not identical, to the ones used in [23, 25].

3.1 Computing the angle.

Let $\mathcal{C}_1^{(n)}, \mathcal{C}_2^{(n)}$ be the curves defined by the forms $f_n(x, y), g_n(x, y)$ of highest degree of the polynomials defining $\mathcal{C}_1, \mathcal{C}_2$. From Eq. (5), we have that $\mathcal{C}_1^{(n)}$ is transformed into $\mathcal{C}_2^{(n)}$ by the similarity $\tilde{h}(z) = \mathbf{a}z$. Furthermore, since $f_n(x, y)$ and $g_n(x, y)$ are homogeneous polynomials of degree n , both $\mathcal{C}_1^{(n)}, \mathcal{C}_2^{(n)}$ are the union of n possibly complex lines, counted with multiplicity, intersecting at the origin. Let $\mathcal{L}_1, \dots, \mathcal{L}_n$ be the lines corresponding to the curve $f_n(x, y) = 0$, and let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be the lines corresponding to the curve $g_n(x, y) = 0$. If \mathcal{C}_1 and \mathcal{C}_2 are similar, the \mathcal{M}_i 's are the result of rotating the \mathcal{L}_i 's about the origin by an angle equal to θ , where θ is the angle of the rotation involved in the similarity.

Fig. 1 illustrates this idea: in this case, both $\mathcal{C}_1^{(n)}, \mathcal{C}_2^{(n)}$ are the union of four real lines, shown in different colors. Lines of the same color at the left and the right of Fig. 1 are related by $\tilde{h}(z) = \mathbf{a}z$; the angle θ is also shown.

By comparing the forms $f_n(x, y), g_n(x, y)$, in general we will find finitely many values for θ , that need to be tested afterwards. In order to show how to find these values, let $\mathbf{a} = |\mathbf{a}| \cdot e^{i\theta} = |\mathbf{a}| \cdot (\cos \theta + i \sin \theta)$, $\theta \in (-\pi, \pi]$. Whenever $\cos \theta \neq 0$, we can write $\mathbf{a} = r \cdot (1 + i\omega)$, where $r = |\mathbf{a}| \cdot \cos \theta$ and $\omega = \text{tg}(\theta)$. The values where $\cos \theta = 0$, i.e. $\theta = -\frac{\pi}{2}, \frac{\pi}{2}$ need to be examined separately.

When $\cos \theta \neq 0$ and except in one, exceptional, situation, we can compute a polynomial P satisfying $P(\omega) = 0$, i.e. such that ω is a real root of P . In order to compute P , let us assume that x is not a factor of either $f_n(x, y)$ or $g_n(x, y)$; if x is such a factor, the analysis is easier. Under this assumption,

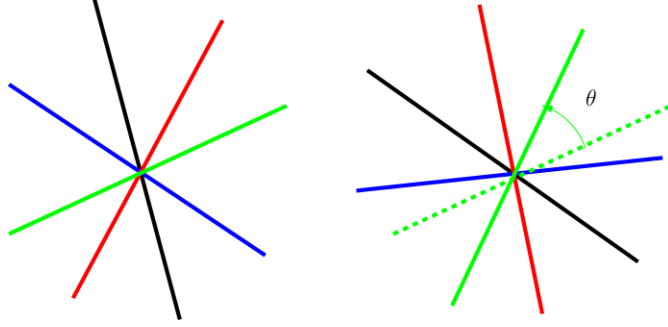


Figure 1: Capturing the rotation.

both f_n and g_n have factors corresponding to lines $y = mx$. In the case of $f_n(x, y)$, the m 's are the (possibly complex) roots of $p(y) = f_n(1, y)$; in the case of $g_n(x, y)$, the m 's are the (possibly complex) roots of $q(y) = g_n(1, y)$. Now $\theta = \gamma_2 - \gamma_1$, where γ_1 is the angle formed by one of the \mathcal{L}_i 's and the positive x -axis, and γ_2 is the angle for the corresponding \mathcal{M}_i . Writing $\gamma_2 = \gamma_1 + \theta$ and denoting $m = \text{tg}(\gamma_1)$, $\tilde{m} = \text{tg}(\gamma_2)$, from the tangent angle addition formula we get

$$\tilde{m} = \frac{m + \omega}{1 - m\omega}.$$

Notice that $m\omega \neq 1$, because we can assume that we are relating two linear factors of $f_n(x, y)$ and $g_n(x, y)$ different from x . Since \tilde{m} is a root of $q(y)$ and m is a root of $p(y)$, writing

$$Q(\omega, y) := q\left(\frac{y + \omega}{1 - y\omega}\right) \quad (7)$$

and taking the resultant of $Q(\omega, y)$ and $p(y)$ with respect to y , we get

$$P(\omega) = \text{Res}_y(p(y), \text{num}(Q(\omega, y))) = 0, \quad (8)$$

where “num” denotes the numerator of the expression in brackets; observe that by well-known properties of resultants, the degree of $P(\omega)$ is n^2 . Thus, whenever $\cos(\theta) \neq 0$, the real roots of $P(\omega)$ are the tentative values for $\omega = \text{tg}(\theta)$.

One can wonder if the polynomial $P(\omega)$ in (8) can be identically zero, in which case the advantage of having a univariate equation is lost. The answer is affirmative, as shown by the following proposition, proven in Appendix I.

Proposition 3.1 *The polynomial $P(\omega)$ in (8) is identically zero iff $p(y)$ and $q(y)$ are both divisible by $y^2 + 1$.*

The polynomials $p(y)$ and $q(y)$ are both divisible by $y^2 + 1$ when $f_n(x, y)$ and $g_n(x, y)$ both contain the factor $x^2 + y^2$. In fact, the idea behind Proposition 3.1 is that the form $x^2 + y^2$ is invariant under *any* rotation about the origin.

Remark 1 *If $f_n(x, y), g_n(x, y)$ are not multiples of $x^2 + y^2$, we can remove the factor $x^2 + y^2$ from both f_n, g_n , and proceed as before with the resulting polynomials. It is only when $f_n(x, y), g_n(x, y)$ are powers of $x^2 + y^2$ that we cannot compute ω this way.*

3.2 Rewriting λ .

The goal here is to write λ in terms of $\mathbf{a}, \bar{\mathbf{a}}$. Since

$$G(\mathbf{a}z + \mathbf{b}, \bar{\mathbf{a}}\bar{z} + \bar{\mathbf{b}}) = \cdots + \beta_{n-j,j}(\mathbf{a}z + \mathbf{b})^{n-j}(\bar{\mathbf{a}}\bar{z} + \bar{\mathbf{b}})^j + \cdots,$$

comparing the coefficients of $z^{n-j}\bar{z}^j$ at both sides of Eq. (5), we get

$$\lambda = \frac{\beta_{n-j,j}\mathbf{a}^{n-j}\bar{\mathbf{a}}^j}{\alpha_{n-j,j}} \quad (9)$$

for any $\alpha_{n-j,j} \neq 0$. Observe that if we have $\alpha_{n-j,j} \neq 0$ but $\beta_{n-j,j} = 0$ for some $j \in \{0, 1, \dots, n\}$, the equality (5) cannot be satisfied, and therefore $\mathcal{C}_1, \mathcal{C}_2$ are not related by an orientation-preserving similarity. Thus, in the rest of the paper, whenever $\alpha_{n-j,j} \neq 0$ we will assume $\beta_{n-j,j} \neq 0$. Furthermore we have the following result, which is also used in [13] and [23].

Lemma 3.1 *Let $\alpha_{n-j,j}, \beta_{n-j,j}$ be the coefficients of the terms of $F(z, \bar{z})$ and $G(z, \bar{z})$ (see Eqs. (2) and (3)). For $j \in \{0, 1, \dots, n\}$ we have $\alpha_{n-j,j} = \bar{\alpha}_{j,n-j}, \beta_{n-j,j} = \bar{\beta}_{j,n-j}$.*

A direct consequence of this lemma is that we can always find $j \in \{0, 1, \dots, n-1\}$ such that $\alpha_{n-j,j} \neq 0$.

3.3 Rewriting \mathbf{b} .

The goal now is to write \mathbf{b} (and therefore $\bar{\mathbf{b}}$ too) in terms of $\mathbf{a}, \bar{\mathbf{a}}$. In order to do this, we compare the coefficients of $z^{n-j-1}\bar{z}^j$ for $j \in \{0, \dots, n-1\}$ at both sides of (5). The terms of $G(z, \bar{z})$ contributing to the term $z^{n-j-1}\bar{z}^j$ in $G(\mathbf{a}z + \mathbf{b}, \bar{\mathbf{a}}\bar{z} + \bar{\mathbf{b}})$ are the terms with bidegrees $(n-j, j), (n-j-1, j+1)$ and $(n-j-1, j)$. After substituting $z \rightarrow \mathbf{a}z + \mathbf{b}$, the coefficient of $z^{n-j-1}\bar{z}^j$ at the left hand-side of (5) is

$$(n-j)\beta_{n-j,j}\mathbf{a}^{n-j-1}\bar{\mathbf{a}}^j\mathbf{b} + (j+1)\beta_{n-j-1,j+1}\cdot\mathbf{a}^{n-j-1}\bar{\mathbf{a}}^j\bar{\mathbf{b}} + \beta_{n-j-1,j}\cdot\mathbf{a}^{n-j-1}\bar{\mathbf{a}}^j.$$

This expression must be equal to the coefficient of $z^{n-j-1}\bar{z}^j$ in $\lambda F(z, \bar{z})$, namely $\lambda \cdot \alpha_{n-j-1,j}$. Taking (9) into account and dividing by $\mathbf{a}^{n-j-1}\bar{\mathbf{a}}^j$, we get a linear equation in $\mathbf{b}, \bar{\mathbf{b}}, \mathbf{a}$, and conjugating this equation we reach another linear equation in $\mathbf{b}, \bar{\mathbf{b}}, \bar{\mathbf{a}}$. Together, these two linear expressions provide the following system:

$$\begin{pmatrix} (n-j)\beta_{n-j,j} & (j+1)\beta_{n-j-1,j+1} \\ (j+1)\bar{\beta}_{n-j-1,j+1} & (n-j)\bar{\beta}_{n-j,j} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \bar{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \frac{\beta_{n-j,j} \cdot \alpha_{n-j-1,j}}{\alpha_{n-j,j}} \cdot \mathbf{a} - \beta_{n-j-1,j} \\ \frac{\bar{\beta}_{n-j,j} \cdot \bar{\alpha}_{n-j-1,j}}{\bar{\alpha}_{n-j,j}} \cdot \bar{\mathbf{a}} - \bar{\beta}_{n-j-1,j} \end{pmatrix} \quad (10)$$

Let $\Delta_j = (n-j)^2 \cdot |\beta_{n-j,j}|^2 - (j+1)^2 \cdot |\beta_{n-j-1,j+1}|^2$. If $\Delta_j \neq 0$, then we can write \mathbf{b} in terms of \mathbf{a} , i.e. $\mathbf{b} = \eta(\mathbf{a}, \bar{\mathbf{a}})$. In a generic situation, we will be able to do so. However, the special case when for all $j \in \{0, \dots, n-1\}$ with $\alpha_{n-j,j} \neq 0$, we have $\Delta_j = 0$, can occur; we will refer to this case as the Δ -special case. This special case is characterized in the following result. Here, let \mathbf{j} be the minimum of the $j \in \{0, \dots, n-1\}$ such that $\alpha_{n-j,j}, \beta_{n-j,j} \neq 0$ (observe that $\mathbf{j} \leq \lceil \frac{n}{2} \rceil$).

Lemma 3.2 *Assume that $\alpha_{n-j,j} \neq 0$ iff $\beta_{n-j,j} \neq 0$. The Δ -special case occurs if and only if $|\beta_{n-j,j}| = \binom{n}{j} \cdot |\beta_{n-j,j}|$ for $j = 0, \dots, n-1$. In particular, if the Δ -special case occurs, then $\mathbf{j} = 0$.*

In fact, in the Δ -special case, whenever $\mathcal{C}_1, \mathcal{C}_2$ are similar the coefficients of \mathcal{C}_1 also satisfy the relationships in Lemma 3.2; the proof is given in Appendix I.

Corollary 3.1 *If $\mathcal{C}_1, \mathcal{C}_2$ are similar, $|\beta_{n-j,j}| = \binom{n}{j} \cdot |\beta_{n-j,j}|$ for $j = 0, \dots, n-1$ if and only if $|\alpha_{n-j,j}| = \binom{n}{j} \cdot |\alpha_{n-j,j}|$ for $j = 0, \dots, n-1$.*

We want to mention that we were not able to find a geometric characterization of the Δ -special case. It seems to be an algebraic phenomenon, with an unclear interpretation in geometry terms.

3.4 Self-similarities (i.e. symmetries).

If $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$, we compute the self-similarities, i.e. the symmetries, of the curve \mathcal{C} . However, in this case we can do better. Here $\mathbf{a} = e^{i\theta}$, i.e. $|\mathbf{a}| = 1$, so in the generic case we do not need to make use of the ideas in Subsection 3.1. Indeed, in the generic case we write $\lambda, \mathbf{b}, \bar{\mathbf{b}}$ in terms of $\mathbf{a}, \bar{\mathbf{a}}$ as shown in

the previous subsections, and then we use that $\bar{\mathbf{a}} = 1/\mathbf{a}$. Thus, we have a collection of univariate polynomials in \mathbf{a} , with complex coefficients, and we just need to compute the gcd of these polynomials. In the Δ -*special* case, however, this strategy fails and we must proceed as shown in the following section.

Additionally, notice that whenever we find more than one similarity between two different curves $\mathcal{C}_1, \mathcal{C}_2$, then we can deduce that $\mathcal{C}_1, \mathcal{C}_2$ have non-trivial symmetries. Indeed, if f_1, f_2 are similarities mapping \mathcal{C}_1 onto \mathcal{C}_2 , then $f_1 \circ f_2^{-1}, f_2 \circ f_1^{-1}$ are self-similarities, i.e. symmetries, of \mathcal{C}_1 ; similarly for \mathcal{C}_2 .

4 Algorithms for computing the orientation-preserving similarities.

In this section, we will describe the algorithms for solving the problem. In this sense, we first address the generic situation, and then we consider some special situations, already mentioned in Section 3.

4.1 The generic case.

In the generic case, the polynomial $P(\omega)$ in Eq. (8) is not identically zero (maybe, according to Remark 1, after removing the factor $x^2 + y^2$ from both f_n, g_n). Furthermore, also in the generic case, there exists $j \in \{0, \dots, n-1\}$ with $\alpha_{n-j,j} \neq 0$, where $\Delta_j \neq 0$, i.e. we are not in the Δ -*special* case.

Under these conditions, we separately look for similarities where $\cos(\theta) \neq 0$, and similarities where $\cos(\theta) = 0$. For the first ones, from Subsection 3.1 we can write $\mathbf{a} = r \cdot (1 + i\omega)$. Additionally, we compute the polynomial $P(\omega)$. Since from Subsection 3.2 we can write $\lambda = \lambda(\mathbf{a}, \bar{\mathbf{a}})$, in turn we have $\lambda = \lambda(r, \omega)$. Additionally, from Subsection 3.3 we can also write $\mathbf{b} = \eta(\mathbf{a}, \bar{\mathbf{a}})$, so again we have $\mathbf{b} = \mathbf{b}(r, \omega)$. Replacing $\lambda, \mathbf{a}, \mathbf{b}$ in the system \mathcal{S} derived from Eq. (5) by its expressions in terms of λ, ω , we finally reach a bivariate system in the variables r, ω , which includes the univariate equation $P(\omega) = 0$.

Concerning similarities where $\cos(\theta) = 0$, in this case we can write $\mathbf{a} = i\mu$, where $\mu = |\mathbf{a}| \cdot \sin \theta$, and therefore $\bar{\mathbf{a}} = -i\mu$. Therefore, applying the ideas in Subsection 3.2 and Subsection 3.3 we can write $\lambda = \lambda(\mu), \mathbf{b} = \mathbf{b}(\mu)$. Therefore, the system \mathcal{S} derived from Eq. (5) turns out to be a system of univariate equations in μ , with complex coefficients, which can be very efficiently solved.

We can now state the Algorithm `Impl-Sim-General` to check similarity in the general case.

Algorithm Impl-Sim-General

Require: Two square-free, implicit algebraic curves $f(x, y) = 0$, $g(x, y) = 0$, neither lines nor circles, not in the Δ -*special* case.

Ensure: Whether or not they are related by an orientation-preserving similarity.

- 1: Compute the system \mathcal{S} from Eq. (5).
 - 2: Find j such that $\alpha_{n-j,j} \neq 0$ and $\Delta_j \neq 0$.
 - 3: Find $\lambda(\mathbf{a}, \bar{\mathbf{a}})$ from (9).
 - 4: Find $\mathbf{b} = \eta(\mathbf{a}, \bar{\mathbf{a}})$ and $\bar{\mathbf{b}} = \bar{\eta}(\mathbf{a}, \bar{\mathbf{a}})$ solving Eq. (10).
 - 5: Substitute $\lambda := \lambda(\mathbf{a}, \bar{\mathbf{a}})$, $\mathbf{b} := \eta(\mathbf{a}, \bar{\mathbf{a}})$, $\bar{\mathbf{b}} = \bar{\eta}(\mathbf{a}, \bar{\mathbf{a}})$ in the system \mathcal{S} .
[Similarities with $\cos \theta \neq 0$:]
 - 6: Compute the polynomial $P(\omega)$.
 - 7: Substitute $\mathbf{a} = r \cdot (1 + i\omega)$, $\bar{\mathbf{a}} = r \cdot (1 - i\omega)$, $\mathbf{b} = \eta(\mathbf{a}, \bar{\mathbf{a}})$, $\bar{\mathbf{b}} = \bar{\eta}(\mathbf{a}, \bar{\mathbf{a}})$ in \mathcal{S} . Add the equation $P(\omega)$ to the system.
 - 8: Check if the system has any solution with $r \in \mathbb{R}$, $r \neq 0$ and $\omega \in \mathbb{R}$.
 - 9: In the affirmative case, return “The curves are related by an orientation-preserving similarity”.
[Similarities with $\cos \theta = 0$.]
 - 10: Substitute $\mathbf{a} = i\mu$, $\bar{\mathbf{a}} = -i\mu$, $\mathbf{b} = \eta(\mathbf{a}, \bar{\mathbf{a}})$, $\bar{\mathbf{b}} = \bar{\eta}(\mathbf{a}, \bar{\mathbf{a}})$ in \mathcal{S} .
 - 11: Check if the system has real solutions.
 - 12: In the affirmative case, return “The curves are related by an orientation-preserving similarity”.
 - 13: If no similarities have been found, return “The curves are not related by an orientation-preserving similarity”.
-

4.2 Special cases.

Here we treat two special situations. The first one corresponds to the case when $f_n(x, y), g_n(x, y)$ are both powers of $x^2 + y^2$. In this case (see Proposition 3.1 and Remark 1), the polynomial $P(\omega)$ is identically zero. However, in this situation we can still apply Algorithm **Impl-Sim-General**, omitting the addition of $P(\omega) = 0$ to the system (see Step 7 of Algorithm **Impl-Sim-General**). In this case we just proceed to solving the bivariate system in r, ω , although we lose the advantage of having a univariate equation in the system.

The second special situation corresponds to the Δ -special case. From Lemma 3.2 in Subsection 3.3, in this case we have $\mathbf{j} = 0$. Furthermore, $\alpha_{n,0} \neq 0$ and $\beta_{n,0} \neq 0$. Now if $\alpha_{n-1,0} \neq 0$, then from Eq. (10), with $j = 0$, we can write \mathbf{a} as a linear function of $\mathbf{b}, \bar{\mathbf{b}}$, i.e. $\mathbf{a} = \rho(\mathbf{b}, \bar{\mathbf{b}})$. Let us see that this requirement can always be fulfilled by means of an appropriate translation. The proof of the following lemma is provided in Appendix I.

Lemma 4.1 *Let $T(z) = z + \kappa$ represent a translation by $\kappa \in \mathbb{C}$, and let $\tilde{\mathcal{C}}_1 = T(\mathcal{C}_1)$. Also, let $j \in \{0, \dots, n-1\}$. The coefficient of $\tilde{\mathcal{C}}_1$ of bidegree $(n-j-1, j)$ is*

$$\tilde{\alpha}_{n-j-1,j} = \alpha_{n-j-1,j} + \kappa \cdot (n-j)\alpha_{n-j,j} + \bar{\kappa} \cdot (j+1)\alpha_{n-j-1,j+1}. \quad (11)$$

As a consequence, if $\alpha_{n-j,j} \neq 0$ and $\alpha_{n-j-1,j} = 0$ then for almost all $\kappa \in \mathbb{C}$, we have $\tilde{\alpha}_{n-j-1,j} \neq 0$.

Since similarities form a group, \mathcal{C}_1 and \mathcal{C}_2 are similar if and only if $\tilde{\mathcal{C}}_1 = T(\mathcal{C}_1)$ and \mathcal{C}_2 are similar. Therefore, there is no problem in applying a translation T on \mathcal{C}_1 . Furthermore, by Lemma 4.1 a random translation will suffice. Now after replacing \mathcal{C}_1 by $T(\mathcal{C}_1)$ if necessary, we can write \mathbf{a} and λ in terms of \mathbf{b} and $\bar{\mathbf{b}}$. Thus the system \mathcal{S} can be written in the variables are $\mathbf{b}, \bar{\mathbf{b}}$. Then the curves are similar if and only if this system has solution with $\mathbf{a} \neq 0$.

Hence, we have the following algorithm **Impl-Sim-Spec** to check similarity in this case.

5 Experimentation.

Algorithms **Impl-Sim-General** and **Impl-Sim-Spec** have been implemented in Maple 2016 ([15]). The code is available in [9].

Algorithm Impl-Sim-Spec

Require: Two square-free, implicit algebraic curves $f(x, y) = 0$, $g(x, y) = 0$, neither lines nor circles, satisfying the hypotheses of the Δ -special case.

Ensure: Whether or not they are related by an orientation-preserving similarity.

- 1: Check whether or not $\alpha_{n-1,0} \neq 0$.
 - 2: If $\alpha_{n-1,0} = 0$ then find a translation $T(z) = z + k$ such that $\tilde{\alpha}_{n-1,1} \neq 0$, and replace $\mathcal{C}_1 \rightarrow T(\mathcal{C}_1)$.
 - 3: Write \mathbf{a} and λ in terms of $\mathbf{b}, \bar{\mathbf{b}}$.
 - 4: Compute the system \mathcal{S} from Eq. (5) and substitute the above expressions into \mathcal{S} .
 - 5: Check if \mathcal{S} has solution with $\mathbf{a} \neq 0$.
 - 6: In the affirmative case, return “The curves are related by an orientation-preserving similarity”. Otherwise, return “The curves are not related by an orientation-preserving similarity”.
-

5.1 Example I

Let $\mathcal{C}_1, \mathcal{C}_2$ be defined by $f(x, y)$ and $g(x, y)$,

$$f(x, y) = 15x^2y - 40xy^2 - 15y^3 + 5x^2 + 5xy - 35y^2 + 5x - 5y + 2$$

and

$$g(x, y) = y^3 + 2xy^2 - x^2y - xy - 2x^3 + 1.$$

Both curves are shown in Figure 2. Using Lemma 3.2, one can check that we fall into the generic case. By applying the algorithm **Impl-Sim-General**, we get an orientation-preserving similarity $h(z) = \mathbf{a}z + \mathbf{b}$ with

$$P(\omega) = -125(\omega^3 + 2\omega^2 - \omega - 2)(448\omega^6 + 4416\omega^5 + 8880\omega^4 - 1920\omega^3 - 8880\omega^2 + 4416\omega - 448).$$

Adding the equation $P(\omega) = 0$ to the system in r, ω obtained from \mathcal{S} , we get

$$\omega = -2, r = 1, \lambda = \left(-\frac{11}{125} - \frac{2i}{125}\right) r^3 (1 + i\omega)^3 = 1,$$

$$\mathbf{a} = r(1 + i\omega) = 1 - 2i, \quad \mathbf{b} = \frac{17r}{50} - \frac{19\omega r}{50} - \frac{1}{10} + \left(\frac{71\omega r}{100} + \frac{11r}{50} + \frac{1}{5}\right)i = 1 - i.$$

The ratio of this similarity is equal to $\sqrt{5}$.

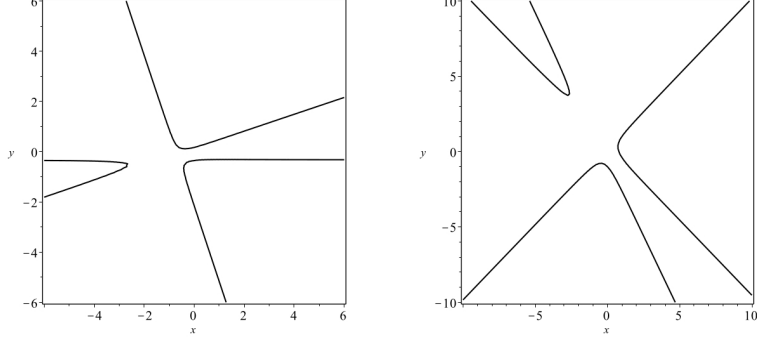


Figure 2: The curve \mathcal{C}_1 (left) and a related curve \mathcal{C}_2 obtained by similarity (right).

5.2 Example II

Let $\mathcal{C}_1, \mathcal{C}_2$ be defined by $f(x, y)$ and $g(x, y)$,

$$\begin{aligned} f(x, y) = & (13950\sqrt{3} - 677538)x^3 + (194892 - 696664\sqrt{3})x^2y - (699966\sqrt{3} + 23192)x^2 \\ & - (630524 + 208136\sqrt{3})xy^2 + (215376 - 413926\sqrt{3})xy - (222558 + 22108\sqrt{3})x \\ & - (68990 + 75910\sqrt{3})y^3 + (-301776 + 6722\sqrt{3})y^2 + (43396 - 48880\sqrt{3})y - 4314\sqrt{3} - 20756, \end{aligned}$$

and

$$\begin{aligned} g(x, y) = & (-589 + 235\sqrt{3})x^3 + (-584 + 178\sqrt{3})x^2y - (566 + 187\sqrt{3})x^2 - (372 + 218\sqrt{3})xy^2 \\ & + (932 - 693\sqrt{3})xy + (904 - 322\sqrt{3})x - (405 + 345\sqrt{3})y^3 + (231 + 792\sqrt{3})y^2 \\ & + (397 + 80\sqrt{3})y + 232 - 121\sqrt{3}. \end{aligned}$$

Both curves are shown in Figure 3. Using Lemma 3.2, one can check that we fall into the generic case. By applying the algorithm `Impl-Sim-General`, we get an orientation-preserving similarity $h(z) = \mathbf{a}z + \mathbf{b}$ with $P(\omega)$ of degree equals 9. Adding the equation $P(\omega) = 0$ to the system in r, ω obtained from \mathcal{S} , we get

$$\omega = \frac{5}{6}, r = 6, \lambda = \frac{1}{2}, \mathbf{a} = r(1 + i\omega) = 6 + 5i, \mathbf{b} = 2 + 2i.$$

The ratio of this similarity is equal to $\sqrt{61}$.

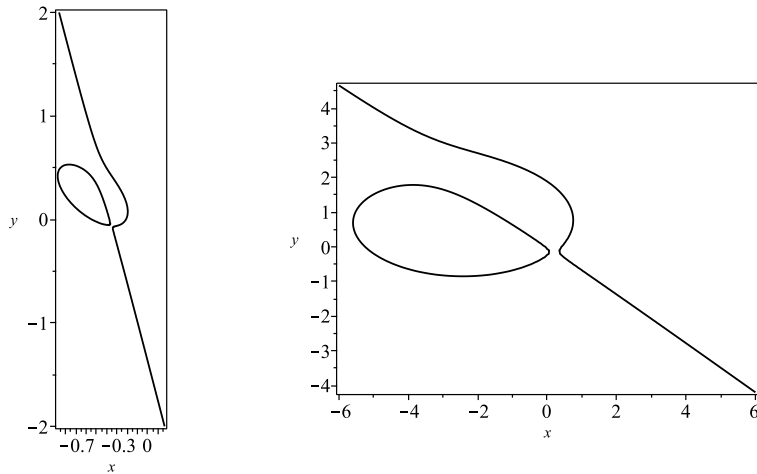


Figure 3: The curve \mathcal{C}_1 (left) and a related curve \mathcal{C}_2 obtained by similarity (right).

5.3 Performance and comparison with a straightforward method

Algorithms `Impl-Sim-General` and `Impl-Sim-Spec` can be unified into one algorithm that first checks the case we are in, and then executes Algorithms `Impl-Sim-General` or `Impl-Sim-Spec` accordingly. We have tested the performance of our algorithm on 10 classical curves, shown in Table 1. In each case, we consider the curve shown in Table 1, and another curve obtained from the first one by applying a similarity. Additionally, we compared the performance of our algorithm with a straightforward (ST) approach, using Eq. (4) on the algebraic equations of the curves to derive a polynomial system in the unknown parameters of the similarity, plus the constant λ in Eq. (4). In Table 1 we provide the timing of our algorithm (t_{our}) and the timing of the algorithm using the straightforward approach ($t_{\text{ST-DEFAULT}}$). In this case, we used Maple’s generic symbolic solver `SolveTools:-PolynomialSystem`, with the default option, for solving polynomial systems. In all the cases, our algorithm could beat the straightforward approach.

It is interesting to observe that if one specifies `engine=groebner`, which is not the default option, in Maple’s `SolveTools:-PolynomialSystem` solver, the straightforward method beats our method for degree ≤ 8 ; we have included the timings corresponding to this option in $t_{\text{ST-GROEBNER}}$. However, this is not the case when we consider higher degrees, in particular for the











Curve	Descartes' folium	Bernoulli's lemniscate	Epitrochoid	Cardioid offset	Hypocycloid
					
degree	3	4	4	8	8
t_{our}	0.46	0.39	0.46	2.62	2.37
$t_{\text{ST-DEFAULT}}$	9.37	1.64	33.0	> 1800	> 1800
$t_{\text{ST-GROEBNER}}$	0.0623	0.1345	0.07575	0.5632	0.9185
Curve	4-leaf rose	8-leaf rose	12-leaf rose	16-leaf rose	20-leaf rose
					
degree	6	10	14	18	22
t_{our}	0.75	0.79	1.56	5.67	9.85
$t_{\text{ST-DEFAULT}}$	> 1800	> 1800	> 1800	> 1800	> 1800
$t_{\text{ST-GROEBNER}}$	0.2499	1.41155	7.189750	18.9005	54.5491

Table 1: Average CPU time (seconds) of the algorithms for well-known curves.

roses of 8, 12, 16 and 20 petals. In particular, one can see that the increase in the timing is more uniform in our method, while for the straightforward method there seems to be a jump in the computation time, even when the option `engine=groebner` is used. For each curve, we have highlighted the best time among t_{our} , $t_{\text{ST-DEFAULT}}$, $t_{\text{ST-GROEBNER}}$, in blue. Funny enough, for the last four roses our symbolic method does better than the numeric methods of Maple’s package `RootFinding [Isolate]`.

Notice that the curves considered in Table 1 are curves with symmetries, so the number of similarities between each one of the given curves and the curve which results after applying a similarity, is higher than one. We tested several examples of dense curves of high degrees with 0 or 1 similarities between them, and we observed that the straightforward approach, using `engine=groebner` in Maple’s `SolveTools:-PolynomialSystem` solver, was very fast. However, this is not the case when the number of similarities is higher, i.e. when we consider curves with symmetries.

6 Conclusions and Future Work

We have presented a formal algorithm for checking whether or not two algebraic curves are similar. When it is the case, the similarities relating the curves are obtained. Our algorithm uses complex representations, also employed by other previous approaches to the problem, as well as several usual tools in the field of computer algebra. The resulting algorithm has been implemented and tested in the computer algebra system Maple 2016, and experiments confirm that the algorithm is efficient and fast.

Our method works when the coefficients of the curve are rational or belong to an algebraic extension. One can wonder if the method can be adapted to the case of approximate, non-exact coefficients (e.g. floating point coefficients). Some first experiments in this direction suggest that the answer is affirmative, but a complete study requires further investigation. Another interesting potential line of work is the detection of implicit algebraic curves related by affine, non-necessarily orthogonal, transformations. Some of our ideas seem to be applicable in that problem, although the question needs further considerations. Finally, another challenging problem is moving to higher dimensions, particularly algebraic surfaces.

Acknowledgements. Juan G. Alcázar and Gema M. Díaz Toca are supported by the Spanish Ministerio de Economía y Competitividad and by the European Regional Development Fund (ERDF), under the project MTM2017-88796-P. Additionally, Juan G. Alcázar and Carlos Hermoso are members

of the Research Group ASYNACS (Ref. CCEE2011/R34). The authors are grateful to the reviewer, whose comments allowed to improve an earlier version of the paper.

References

- [1] Alcázar J.G., Hermoso C., Muntingh G. (2014), *Detecting Similarity of Plane Rational Plane Curves*, Journal of Computational and Applied Mathematics, vol. 269, pp. 1–13.
- [2] Alcázar J.G., Díaz Toca G.M., Hermoso C. (2015), *On the problem of detecting when two implicit plane algebraic curves are similar*, ArXiv arXiv:1505.06095v1.
- [3] Alcázar J.G., Lavička M., Vršek J. (2018), *Symmetries and similarities of planar algebraic curves using harmonic polynomials*, ArXiv 1801.09962.
- [4] Boutin M. (2000), *Numerically Invariant Signature Curves*, International Journal of Computer Vision 40(3), pp. 235–248.
- [5] Calabi E., Olver P.J., Shakiban C., Tannenbaum A., Haker S. (1998), *Differential and Numerically Invariant Signature Curves Applied to Object Recognition*, International Journal of Computer Vision, 26(2), pp. 107–135.
- [6] Cox D., Little J. O’Shea D. (1992), *Ideals, varieties and algorithms*, Springer.
- [7] Coxeter, H. S. M. (1969), *Introduction to geometry*, Second Edition, John Wiley & Sons, Inc., New York-London-Sydney.
- [8] Gal R., Cohen-Or D. (2006), *Salient geometric features for partial shape matching and similarity*. ACM Transactions on Graphics, Volume 25(1), pp. 130–150.
- [9] G.M. Díaz Toca. (2015), <http://webs.um.es/gemadiaz/miwiki/doku.php?id=papers>.
- [10] Fischer G. (2001), *Plane algebraic curves*, American Mathematical Society, USA.
- [11] Hauer M., Jüttler B. (2018), *Projective and affine symmetries and equivalences of rational curves in arbitrary dimension*, Journal of Symbolic Computation Vol. 87, pp. 68–86.

- [12] Huang Z., Cohen F.S. (1996), *Affine-Invariant B-Spline Moments for Curve Matching*, IEEE Transactions on Image Processing, vol. 5, No. 10, pp. 1473–1480.
- [13] Lebmair P., Richter-Gebert J. (2008), *Rotations, Translations and Symmetry Detection for Complexified Curves*, Computer Aided Geometric Design 25, pp. 707–719.
- [14] Lei Z., Tasdizen T., Cooper D.B. (1998), *PIMs and Invariant Parts for Shape Recognition*, Proceedings Sixth International Conference on Computer Vision, pp. 827–832.
- [15] Maple (2015). Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario.
- [16] Mishra R., Pradeep K. (2012), *Clustering Web Logs Using Similarity Upper Approximation with Different Similarity Measures*, International Journal of Machine Learning and Computing vol. 2, no. 3, pp. 219–221.
- [17] Monagan M.B., Wittkopf A. (2000). *On the design and implementation of Brown’s algorithm over the integers and number fields*, Proceedings ISSAC (International Symposium on Symbolic and Algebraic Computation) 2000, pp. 225–233.
- [18] Mozo-Fernández J., Munuera C. (2002), *Recognition of Polynomial Plane Curves Under Affine Transformations*, Applicable Algebra in Engineering, Communications and Computing, vol. 13, pp. 121–136.
- [19] Mozo-Fernández J., Munuera C. (2002), *Recognition of Polynomial Plane Curves Under Affine Transformations*, Applicable Algebra in Engineering, Communications and Computing, vol. 13, pp. 121–136.
- [20] Sener S., Unel M. (2005), *Affine invariant fitting of algebraic curves using Fourier descriptors*, Pattern Analysis and Applications vol. 8, pp. 72–83.
- [21] Suk T., Flusser J. (1993), *Pattern Recognition by Affine Moment Invariants*, Pattern Recognition, vol. 26, No. 1, pp. 167–174.
- [22] Tarel, J.-P., Cooper, D.B., (1998). A new complex basis for implicit polynomial curves and its simple exploitation for pose estimation and invariant recognition. Conference on Computer Vision and Pattern Recognition (CVPR’98), pp. 111–117.

- [23] Tarel J.P., Cooper D.B. (2000), *The Complex Representation of Algebraic Curves and Its Simple Exploitation for Pose Estimation and Invariant Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, No. 7, pp. 663–674.
- [24] Taubin G., Cooper D.B. (1992), *Object Recognition Based on Moments (or Algebraic) Invariants*, Geometric Invariance in Computer Vision, J.L. Mundy and A.Zisserman, eds., MIT Press, pp. 375–397, 1992.
- [25] Unel, M., Wolovich, W.A. (1998). *Complex representations of algebraic curves*. International Conference on Image Processing ICIP 1998, pp. 272–276.
- [26] Unel M., Wolowich W.A. (2000), *On the Construction of Complete Sets of Geometric Invariants for Algebraic Curves*, Advances in Applied Mathematics, vol. 24, pp. 65–87.
- [27] Weiss I. (1993), *Noise-Resistant Invariants of Curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, No. 9, pp. 943–948.
- [28] Wolowich W., Unel M. (1998), *The determination of implicit polynomial canonical curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20(10), pp. 1080–1089.
- [29] Wolowich W., Unel M. (1998), *Vision-Based System Identification and State Estimation*. In: “The Confluence of Vision and Control, Lecture Notes in Control and Information Systems”, New York, Springer-Verlag.

Appendix I: Proofs of some technical results.

Proof. (of Proposition 3.1) Let

$$p(y) = \prod_{i=1}^n (y - m_i), \quad q(y) = \prod_{i=1}^n (y - \tilde{m}_i).$$

Hence

$$Q(\omega, y) = \frac{1}{(1 - yt)^n} \prod_{i=1}^n (y + \omega - \tilde{m}_i + y\omega\tilde{m}_i),$$

which implies

$$\text{num}(Q(\omega, y)) = \prod_{i=1}^n (y(1 + \omega\tilde{m}_i) + \omega - \tilde{m}_i).$$

Since $p(y)$ and $\text{num}(Q(\omega, y))$ have both positive degree in y , $P(\omega) = 0$ if and only if they have a common factor with positive degree in y (see Chapter 3 in [6]). In turn, this happens if and only if there exists $r \in \{1, \dots, n\}$ such that $(y - m_r)$ divides $\text{num}(Q(\omega, y))$. As a consequence $\text{num}(Q(\omega, m_r)) = 0$. Then there exists $j \in \{1, \dots, n\}$ with $m_r + \omega - \tilde{m}_j + \omega m_r \tilde{m}_j = 0$, and so $m_r = \tilde{m}_j$, $1 + \tilde{m}_j m_r = 0$ hold. Since $p(y)$ and $q(y)$ are real polynomials, this implies that $y^2 + 1$ divides both polynomials. ■

Proof. (of Lemma 3.2)

(\Leftarrow) Since $|\beta_{n-j,j}| = \binom{n}{j} \cdot |\beta_{n-j,j}|$ for $j = 0, \dots, n-1$, we have

$$\Delta_j = \left[(n-j)^2 \binom{n}{j}^2 - (j+1)^2 \binom{n}{j+1}^2 \right] \cdot |\beta_{n-j,j}|^2.$$

Since $\frac{\binom{n}{j+1}}{\binom{n}{j}} = \frac{n-j}{j+1}$, we can easily see that $\Delta_j = 0$. Furthermore, since by definition $\beta_{n-j,j} \neq 0$, we have $\beta_{n-j,j} \neq 0$ for $j = 0, \dots, n-1$. Therefore, $\alpha_{n-j,j} \neq 0$ for $j = 0, \dots, n-1$.

(\Rightarrow) By induction over k , one can prove that $|\beta_{n-(j+k),j+k}| = \binom{n}{j+k} \cdot |\beta_{n-j,j}|$ for $0 \leq k \leq n-j$. Thus, when $k = n-j$ we deduce that $|\beta_{0,n}| \neq 0$. Therefore, $\beta_{0,n} \neq 0$ and by Lemma 3.1, $\beta_{n,0} \neq 0$. Hence $j = 0$. Furthermore, since by definition $\beta_{n-j,j} \neq 0$, we also deduce that all the $\alpha_{n-j,j}$'s are nonzero. ■

Proof. (of Corollary 3.1) If $\mathcal{C}_1, \mathcal{C}_2$ are similar then from Equation (5), we have that

$$\mathbf{a}^{n-j} \cdot \bar{\mathbf{a}}^j \cdot \beta_{n-j,j} = \lambda \cdot \alpha_{n-j,j} \quad (12)$$

for $j = 0, \dots, n$. Furthermore, by taking the absolute value in (9) and fixing $j = \mathbf{j}$, we get that

$$|\lambda| = \frac{|\beta_{n-\mathbf{j},\mathbf{j}}| \cdot |\mathbf{a}|^n}{|\alpha_{n-\mathbf{j},\mathbf{j}}|}. \quad (13)$$

So from (12) and (13), we get

$$|\alpha_{n-\mathbf{j},\mathbf{j}}| = \frac{|\alpha_{n-\mathbf{j},\mathbf{j}}| \cdot |\beta_{n-\mathbf{j},\mathbf{j}}|}{|\beta_{n-\mathbf{j},\mathbf{j}}|}.$$

Then the statement follows. ■

Proof. (of Lemma 4.1) The coefficient of $\tilde{\mathcal{C}}_1 = T(\mathcal{C}_1)$ of bidegree $(n-j-1, j)$ comes from the terms of F with bidegrees $(n-j, j)$, $(n-j-1, j)$ and $(n-j-1, j+1)$. Each of these terms provides, in turn, the terms of (11): the term of F of bidegree $(n-j, j)$ provides the first term of (11), the term of bidegree $(n-j-1, j)$ provides the term in κ , and the term of $(n-j-1, j+1)$ provides the term in $\bar{\kappa}$. As for the second part of the statement, assume that $\alpha_{n-j,j} \neq 0$, $\alpha_{n-j-1,j} = 0$, and suppose by contradiction that $\tilde{\alpha}_{n-j-1,j}$ vanishes for all κ . By substituting $\kappa = 1$ in (11), we have $(n-j)\alpha_{n-j,j} + (j+1)\alpha_{n-j-1,j+1} = 0$; by substituting $\kappa = i$ in (11), we get that $(n-j)\alpha_{n-j,j} - (j+1)\alpha_{n-j-1,j+1} = 0$. Putting these two equations together, we deduce that $\alpha_{n-j-1,j+1} = \alpha_{n-j,j} = 0$, contradicting that $\alpha_{n-j,j} \neq 0$. ■