An Algorithm for Automatic Discovery of Algebraic Loci
Extended Abstract

Francisco Botana, Antonio Montes and Tomas Recio

Depto. Matemática Aplicada I, Universidad de Vigo, Spain
Depto. Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Catalunya
Depto. Matemáticas, Estadística y Computación, Universidad de Cantabria, Spain

1 Introduction

A defining characteristic of dynamic geometry (DG) systems is that unconstrained parts of a construction can be moved, and, as they do, all elements automatically self–adjust, preserving dependent relationships and constraints (see [1]). As a natural consequence, DG software allows users to keep track on the path of an object that depends on another one while this last object is dragged. These constructive loci are perhaps one of the most appealing abilities in DG. This constraint approach led to a simple strategy: the locus of a *tracer* object, depending somehow on a *driver* object with a predefined path, is drawn by sampling the driver path and plotting the tracer position for each sample. Most DG programs use segments to join these positions in order to suggest a continuous curve. Nevertheless, the uniform division of the path can produce anomalous loci when a small variation of the driver object produces important changes on the tracer position, as illustrated in Figure 1 by the curve returned in The Geometer's Sketchpad [2] as a conchoid when the focus O is almost on the path of the driver point P.

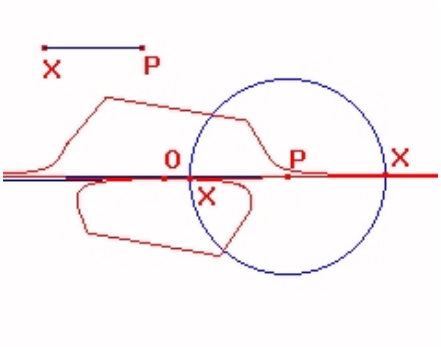

Fig. 1. An aberrant conchoid of Nicomedes in The Geometer's Sketchpad.

It must be noted that in order to get both branches of the conchoid the user must compute two loci, corresponding to the two possible intersections of a circle and the driver's path.

A relevant issue when finding locus consists of the knowledge the system has about the new object. The above approach describes a locus just as a list of points, thus forbiding a posteriori computations (consider, for instance, computing a tangent to a curve obtained as locus). Cabri [3] includes an option to compute the equations of loci based on polynomial interpolation. It uses a sample of points on the locus to generate equations (with degree not greater than 6), so giving just approximate results. Figure 2 shows the equations for an astroid (note again that the astroid is obtained as two loci). Although the algorithm is not public, it seems that it is very unstable, and the returned results are frequently erroneous even for simple cases.

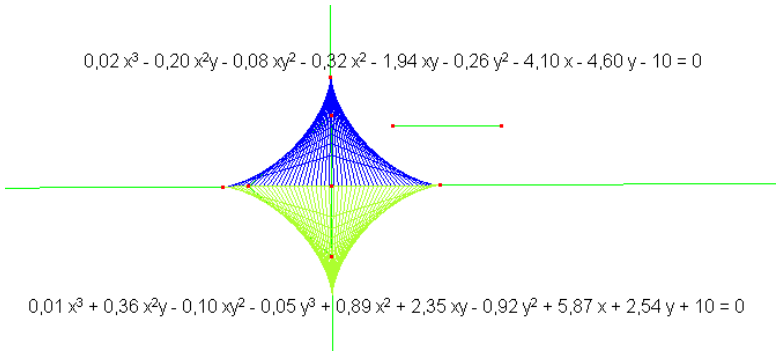

Fig. 2. Equations returned by Cabri for the upper and lower halves of an astroid.

2 Automatic Loci Discovery via Groebner Bases

In [4] a further development of the well–known approach to automatic theorem proving in elementary geometry via algorithmic commutative algebra and algebraic geometry is discussed. Rather than confirming/refuting geometric statements or deriving geometric formulae, the issue of automatic discovery of statements is considered. The method has been specialized in [5] to deal with discovery of standard loci. A rough description of the method is as follows: Given a geometric construction with a point whose locus is the one we are looking for, the procedure begins by translating the geometric properties into algebraic expressions. We use the field of rational numbers $\mathbb{Q}$ and $\mathbb{C}$, the field of complex numbers, as an algebraically closed field containing the former. The collection of construction properties is then expressed as a set of polynomial equations

$$p_1(x_1,\ldots,x_n)=0,\ldots,p_r(x_1,\ldots,x_n)=0,$$

where $p_1,\ldots,p_r \in \mathbb{Q}[x_1,\ldots,x_n]$. Thus, the affine variety defined by $V=\{p_1=0,\ldots,p_r=0\}\subset\mathbb{C}^n$ contains all points $(x_1,\ldots,x_n)\in\mathbb{C}^n$ which satisfy the construction requirements, that is, the set of all common zeros of $p_1,\ldots,p_r$ in the n-dimensional affine space of $\mathbb{C}$ describe all the possible positions of the construction points. In particular, the positions of the locus point define the locus we are searching for. Thus, supposing that the locus point coordinates are x_{n-1}, x_n, the projection

$$\pi_{n-2}: V\subset\mathbb{C}^n\rightarrow\mathbb{C}^2$$

gives an extensional definition of the locus in the affine space $\mathbb{C}^2$. This projection can be computed via the $(n-2)$th elimination ideal of $\langle p_1,\ldots,p_r\rangle$, I_{n-2}. The Closure theorem states that $V(I_{n-2})$ is the smallest affine variety containing $\pi_{n-2}(V)$, or, more technically, that $V(I_{n-2})$ is the Zariski closure of $\pi_{n-2}(V)$. So, except some missing points that lie in a variety strictly smaller than $V(I_{n-2})$, we can describe the locus computing a basis of I_{n-2}. This basis is computed as follows: given the ideal $\langle p_1,\ldots,p_r\rangle\subset\mathbb{Q}[x_1,\ldots,x_n]$, let G be a Groebner basis of it with respect to lex order where $x_1>x_2>\ldots>x_n$. The Elimination theorem states that $G_{n-2}=G\cap\mathbb{Q}[x_{n-1},x_n]$ is a Groebner basis of I_{n-2}. Unfortunately, we will find in many cases that $G_{n-2}=\emptyset$, that is, $V(\langle G_{n-2}\rangle)=\mathbb{C}^2$, which only allows as conclusion the irrelevant statement that "the locus is contained in the plane". Nevertheless, we do not want to eliminate all variables except those of the locus point, but simply the variables stemming from dependent points. So, the construction properties are translated into a set of polynomial equations

$$p_1(x_1,\ldots,x_s,u_1,\ldots,u_t)=0,\ldots,p_r(x_1,\ldots,x_s,u_1,\ldots,u_t)=0,$$

where $x_1,\ldots,x_s$ are the dependent point coordinates, and $u_1,\ldots,u_t$ are those of free points. Note that the coordinates of the locus point are included in this $\{u_-\}$ set (thus also allowing the study of loci of points not constructible in the environment). The elimination of $x_1,\ldots,x_s$ in the ideal $\langle p_1,\ldots,p_r\rangle$ returns another ideal $\langle q_1,\ldots,q_m\rangle$ whose affine variety $V(q_1,\ldots,q_m)\subset\mathbb{C}^t$ contains the locus.

Despite the cost of computing Groebner bases, this method has been proved as successful for automatically determining loci in DG environments (see a prototype in [5]). It has been incorporated into JSXGraph [6] under a web–based access, and is currently being incorporated into GeoGebra [7, 8]. Apart from the structural algebraic limitation of this method, its main drawback deals with the inclusion of special/degenerate components of the sought loci. If a geometric constraint becomes undefined for some instance of the construction, the corresponding polynomial will not be taken into account during elimination, and a spurious part of the locus will be included in the final answer. As an illustration of the case, consider a limaçon of Pascal where the focus lies on the base curve.

Fig. 3. An extra circle when computing a limaçon of Pascal with JSXGraph.

In such a case, an extra circle centered at the focus will be returned as part of the limaçon, as computed by JSXGraph in Figure 3.

Another source of imprecision comes from the variety–like description of loci. A variety describing a locus can contain extra points not satisfying the geometric constraints of the problem. For instance, the pedal of an ellipse with respect to its center will be described by a variety including the center, if following the above approach. Since this point is not a part of the pedal, any subsequent computation using its polynomial description will be wrong.

3 Automatic Loci Discovery via Parametric Groebner Bases

The remotion of special/degenerated parts and the computation of loci as constructible sets, rather than varieties can be efficiently solved in the field of dynamic geometry by using the theory of parametric polynomial systems. Here, we propose using the Groebner Cover (GC) algorithm [9]. The variables occurring in the equations which describe a locus construction can be divided into a set of parameters and a set of unknowns. The parameters (a, b) correspond to the locus point, while the unknowns $(x, y, \ldots)$ (variables from now on), correspond to the remaining points of the geometric construction. Finding the locus is equivalent to obtain the set of values of the parameters for which it exists a finite number of values of the variables. The values of the parameters for where it does not exist any solution do not form part of the locus. Moreover, the parameter values for which it exists an infinite number of solutions of the variables correspond to a degenerate construction and must also be excluded from the "Normal" locus.

Thus, we look for solutions of the parametric system in terms of the parameter values, and the structure of the solution space. More formally, a parametric polynomial system over $\mathbb{Q}$ is a finite set of polynomials $p_1, \ldots, p_r \in \mathbb{Q}[\bar{a}, \bar{x}]$ in the variables $\bar{x} = x_1, \ldots, x_n$ and parameters $\bar{a} = a_1, \ldots, a_m$. The goal is studying the solutions of the algebraic systems $\{p_1(a, \bar{x}), \ldots, p_r(a, \bar{x})\} \subset \mathbb{Q}[\bar{x}]$ which are obtained by specializing the parameters to concrete values $a \in \mathbb{C}^m$.

The GC algorithm emphasizes the obtention of a canonical description (as compact as possible) of the parametric system. Before sketching our algorithmic use of GC, we discuss in some detail the obtention of a limaçon of Pascal. Let $O(0, 2)$ be a fixed point on the circle $c : x^2 + y^2 = 4$ and l be a line passing through O and $P(x, y)$ (any point on c). Let $Q(a, b)$ be a point on l such that $distance(P, Q) = 1$. We seek for the locus of Q when P moves along the circle c. The system of equations is:

$$S = x^2 + y^2 - 4, (b-2)x - ay + 2a, (a-x)^2 + (b-y)^2 - 1.$$

The standard elimination procedure of the preceding section returns the variety

$$V = \mathbb{V}(a^4 + 2a^2b^2 + b^4 - 9b^2 - 9b^2 + 4b + 12) \cup \mathbb{V}(a^2 + b^2 - 4b + 3),$$

where the former corresponds to the sought conchoid, whereas the last one comes from a degeneracy stemming from the coincidence of P and the focus O. GC returns four segments for this parametric system:

- Segment 1
  - segment:
    $\mathbb{Q}^2 \setminus (\mathbb{V}(a^2 + b^2 - 4b + 3) \cup \mathbb{V}(a^4 + 2a^2b^2 - 9a^2 + b^4 - 9b^2 + 4b + 12))$
  - basis: $\{1\}$
- Segment 2
  - segment:

    $$(\mathbb{V}(a^2 + b^2 - 4b + 3) \setminus (\mathbb{V}(2b - 3, 4a^2 - 3))$$
    $$\cup(\mathbb{V}(a^4 + 2a^2b^2 - 9a^2 + b^4 - 9b^2 + 4b + 12) \setminus (\mathbb{V}(2b - 3, 4a^2 - 3) \cup \mathbb{V}(b - 2, a))$$

  - basis:

    $$\begin{cases} (2a^2 + 2b^2 - 4b)y + (-a^2b - 2a^2 - b^3 + 2b^2 - 3b + 6), \\ (2a^2 + 2b^2 - by)x + (-a^3 - ab^2 + 4ab - 3a). \end{cases}$$

- Segment 3
  - segment: $\mathbb{V}(b - 2, a) \setminus \mathbb{V}(1)$
  - basis: $\{4y - 7, 16x^2 - 15\}$
- Segment 4
  - segment: $\mathbb{V}(2b - 3, 4a^2 - 3) \setminus \mathbb{V}(1)$
  - basis: $\{x + 2ay - 4a, y^2 - 3y + 2\}$

These segments must be understood as follows. Segment 1 states that any general point in the plane does not satisfy the locus conditions (there is no solution of the parametric system, since the basis is $\{1\}$), unless the point lies on the circle $a^2+b^2-4b+3=0$ or the curve $a^4+2a^2b^2-9a^2+b^4-9b^2+4b+12=0$ (note that these factors were previously obtained with the standard elimination approach). Segment 2 declares that the points lying on the circle and the limaçon satisfy the required constraints, and the variables x, y can be expressed in terms of the parameters a, b by the given base and have a single solution. Nevertheless, some of these points correspond to other parametric values, as described by segments 3 and 4, where the system has two solutions in the variables.

A locus–oriented procedure to interpret this canonical segment decomposition is as follows:

Add all the segments corresponding to a finite number of solutions (in the variables), i.e. segments 2, 3 and 4 in this example. This gives exactly the same result as with the previous method, namely the variety V. Nevertheless, if we specialize the basis over the component $\mathbb{V}(a^2+b^2-4b+3)$ we obtain $\{y-2, x\}$ as basis. This shows that this curve of the locus corresponds to a single point of the variables (the point P), and thus it should be declared as a special component of the locus. An automatic procedure that takes into account the above discussion when it is applied to the GC gives the following output for the locus:

$$[[\mathbb{V}(a^2+b^2-4b+3), \text{"}Special\text{"}], [\mathbb{V}(a^4+2a^2b^2-9a^2+b^4-9b^2+4b+12)]]$$

so that one can distinguish between the "Normal" components of the locus and the "Special" components if they exist.

A live demo of loci computations by using a Singular [10] webservice [11] with the described approach will be given during the talk. This web–based resource could be used to enhance DG systems abilities, as GeoGebra is currently doing for symbolic proving.

Acknowledgements

First and third authors supported by the Spanish "Ministerio de Economía y Competitividad" and by the European Regional Development Fund (ERDF), under the Project MTM2011–25816–C02–(01, 02). Second author partly supported by ESF EUROCORES programme EuroGIGA – ComPoSe IP04 – MICINN Project EUI–EURC–2011–4306.

References

1. King, J., Schattschneider, D.: *Geometry Turned On.* MAA, Washington (1997)
2. Jackiw, N.: *The Geometer's Sketchpad.* Key Curriculum Press, Berkeley (1997)
3. Laborde, J.M., Bellemain, F.: Cabri Geometry II. Texas Instruments, Dallas (1998)
4. Recio, T., Vélez, M.P.: Automatic Discovery of Theorems in Elementary Geometry. J. Autom. Reasoning 23, 63–82 (1999)

5. Botana, F., Valcarce, J.L.: A Software Tool for the Investigation of Plane Loci. Math. Comput. Simul. 61(2), 139–152 (2003)
6. JSXGraph, http://jsxgraph.uni-bayreuth.de
7. GeoGebra, http://www.geogebra.at
8. GeoGebra Locus Line Equation, http://www.geogebra.org/trac/wiki/LocusLineEquation
9. Montes, A., Wibmer, M.: Groebner Bases for Polynomial Systems with Parameters. J. Symb. Comput. 45, 1391–1425 (2010)
10. Decker, W., Greuel, G.M., Pfister, G., Schönemann, H.: Singular 3–1–3 — A computer algebra system for polynomial computations, http://www.singular.uni-kl.de (2011)
11. Singular WebService, http://code.google.com/p/singularws