Automatic Deduction in (Dynamic) Geometry: Loci Computation

Francisco Botana

Departamento de Matemática Aplicada I, Universidad de Vigo, Campus A Xunqueira, 36005 Pontevedra, Spain

Miguel A. Abánades

CES Felipe II, Universidad Complutense de Madrid, 28300 Aranjuez, Spain

Abstract

A symbolic tool based on open source software that provides robust algebraic methods to handle automatic deduction tasks for a dynamic geometry construction is presented. The prototype has been developed as two different worksheets for the open source computer algebra system *Sage*, corresponding to two different ways of coding a geometric construction, namely with the open source dynamic geometry system *GeoGebra* or using the common file format for dynamic geometry developed by the *Intergeo* project. Locus computation algorithms based on Automatic Deduction techniques are recalled and presented as basic for an efficient treatment of advanced methods in dynamic geometry. Moreover, an algorithm to eliminate extraneous parts in symbolically computed loci is discussed. The algorithm, based on a recent work on the Gröbner cover of parametric systems, identifies degenerate components and extraneous adherence points in loci, both natural byproducts of general polynomial algebraic methods. Several examples are shown in detail.

Keywords: Automatic Deduction in Geometry, Locus, Sage, GeoGebra, Gröbner Cover

Email addresses: fbotana@uvigo.es (Francisco Botana), abanades@ajz.ucm.es (Miguel A. Abánades)

1. Introduction

Automated Deduction in Geometry, understood as the study and development of computer programs designed to prove geometry theorems, can be traced back almost five decades to the influential work of Gelernter in the field of Artificial Intelligence [1]. However, the real flourishing of the field came in the early 1980's with the development by Wu of an algebraic method based on Ritt's characteristic set for proving a restricted set of geometry theorems [2]. Impressive results by several authors using Wu's method (e.g. [3, 4, 5, 6]) encouraged researchers to consider other algebraic methods, among which those based on Gröbner bases [7] proved to be the most relevant (see [8, 9, 10]).

Directly related to automatic proving is the concept of automatic *discovery*, more closely related to the work presented in this article. While automatic proving deals with verifying geometric statements, automatic discovery tries to find complementary hypotheses for statements to become true or, as stated in [11], to "finding the missing hypotheses so that a given conclusion follows from a given incomplete set of hypotheses". A systematic use of algebraic methods based on Gröbner bases to address the question of automatic discovery in geometry was thoroughly developed in [12].

From the appearance of the very first prototypes of automatic provers it was clear the need to develop accompanying graphic interfaces. Almost at the same time appeared the first stable dynamic geometry systems (DGS) with great acceptance, specially in the field of education. Let us recall that a DGS is a computer application that allows the exact on-screen drawing of geometric diagrams and their interactive manipulation by mouse dragging. Also, computer algebra systems (CAS), whose core functionality is the manipulation of mathematical expressions in symbolic form, reached a very high level of development and started becoming available in most teaching and research institutions. Several authors since then have postulated a deeper cooperation between DGS and CAS in order to enhance the abilities of dynamic geometry programs with algebraic proving and discovering algorithms. In particular, in [12] one can read that their method can be regarded as "the core of a future program [...] that allows, when linked simultaneously with a tool for displaying geometric constructions and a symbolic computation package, the interactive exploration of geometric properties".

Considerable attention and efforts have been dedicated since then to these emerging new tools which can be termed symbolic - dynamic geometry envi-

ronments, a synthesis of DGS and CAS. In particular, two main approaches have been followed. Some DGS incorporate their own code to perform symbolic computations (e.g. [13]), while others choose to reuse existing Computer Algebra Systems (e.g. [14, 15, 16, 17]).

In this article we present an efficient solution in this second direction strictly based on open source software applications. In particular, the developed prototype symbolically computes equations of geometric loci and determines the validity of geometric statements specified in an open source DGS. Moreover, an extension of the prototype has been developed that accepts, as input, the description of a geometric locus specified using a common DGS file format developed in an European project and accepted by most European DGS. Finally, complementing the symbolic computation of geometric loci, a tool to detect degenerate components as well as extraneous adherence points in the returned loci has been developed based on the recent Gröbner Cover algorithm for solving parametric systems in [18]. It is illustrated with several examples of loci and envelopes. To the best of our knowledge, the described approach is the first to implement in a completely automatic way (i.e. with no user interaction at all) the removal of extraneous parts (degenerate components in particular) from objects computationally generated in a DGS. We feel this is an important step in the solution of one of the main current bottlenecks in the development of dynamic geometry environments. We are confident that this method will be integrated in coming versions of these environments.

Following are a few words on the main building blocks of our system, namely the open source applications GeoGebra and Sage and the DGS common file format Intergeo.

1.1. GeoGebra

GeoGebra is an open source DGS with algebraic capabilities, establishing a direct relationship between the objects in the different windows: graphics, algebra, and spreadsheet. It was created in 2001 by Markus Hohenwarter as part of his Master in Mathematics Education at the University of Salzburg, Austria, winning him the 2002 European Academic Software Award (EASA) in the category of Mathematics. Since then, the tool, using the word of mouth and Internet, spread rapidly throughout the world, and has become a collaborative project with impressive figures: users in 190 countries, versions in 62 languages and almost a million visitors to its web site every month.

It is worth mentioning the ongoing creation of a network of International GeoGebra Institutes (IGI) that serves as a platform from which teachers and researchers from around the world work together to promote activities and research related to GeoGebra and its applications. Currently there are 180 official IGIs in 80 countries.

1.2. Sage

Sage (`http://sagemath.org`) is a free open source CAS designed to be a viable multi-platform free open source alternative to proprietary - and expensive - systems such as Mathematica, Maple or Matlab. Besides being open source, the integration of multiple tools and the possibility of remote access via the Internet make their most notable features.

Built out of nearly 100 open-source packages (including Singular, Axiom, Maxima,...), Sage features a unified interface that takes the form of a notebook in a web browser or the command line. Using the notebook, Sage connects either locally to your own Sage installation or to a Sage server on the network. Inside the Sage notebook you can create embedded graphics, beautifully typeset mathematical expressions, add and delete input and share your work.

Sage was created in 2004 by William Stein (`http://wstein.org`), professor at the University of Washington, motivated, among other things, by several disagreements over some accessibility issues with the developers of Magma, a highly specialized commercial CAS in whose development he had collaborated.

1.3. Intergeo

The *Intergeo* (i2g) file format is an XML-based specification designed to describe any construction created with a DGS. It is based on OpenMath [19], a standard for representing mathematical objects with their semantics and was developed as part of the Intergeo project (`http://i2geo.net`), an eContentplus European project in which the authors took part dedicated to the sharing of interactive geometry constructions across systems.

An i2g file takes the form of a compress file package. The file intergeo.xml provides a textual description of the construction in two parts, one with the elements of the construction that describes the (static) initial configuration and one where the geometric constraints among the elements are included. A detailed specification of the file format can be found at `http://i2geo.net/xwiki/bin/view/I2GFormat/`, where several examples are provided.

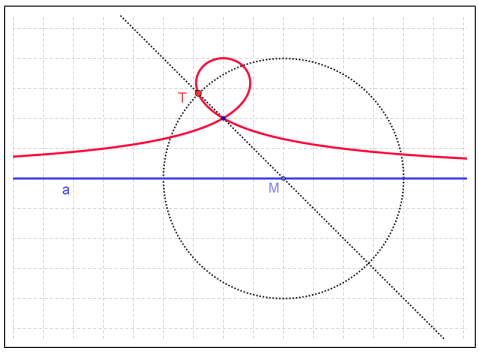

Figure 1: (one branch of) The conchoid of Nichomedes as the locus set traced by T as M runs along the line a.

2. Automatic Discovery and locus equations

Given a dynamic geometry construction, a locus generally refers to the set of points determined by the different positions of a point, the tracer or tracing point, as a second point in which the tracer depends on, called the mover or moving point, runs along the one dimensional object to which the mover is restrained (see Figure 1).

Some DGS allow higher dimensional tracing elements, making possible the construction of more involved geometric objects such as envelopes (curves that are tangent to a family of lines). Being the algebraic treatment of envelopes very similar to that of loci determined by points, almost everything in this work can be easily generalized towards the computation of envelope equations.

Most systems implement loci generation just from a graphic point of view, returning a locus as a set of points in the screen with no algebraic information. Two main strategies have been followed to determine the implicit equation of a locus in a DGS. On one hand, there is the numerical approach, based on a set of sample points in the locus. This is the method followed by the commercial systems Cabri and Cinderella that apply direct interpolation. It has the advantage of being relatively easy to implement within the system but its numerical nature makes it prone to inaccuracies (see [20] for an in-depth study of this standard approach). A slightly different method is presented in [21] where an algorithm based on the analysis of eigenvalues and eigenvectors to determine the degree and coefficients of the "closest" algebraic curve to a set of sample points of a locus constructed by ruler and compass is analyzed. Although impressively precise in certain situations, the method shows some

weaknesses (*Our algorithm makes a false degree guess more often for curves of relatively high degree. They may be recognized as curves of lower degree* [21, p. 63]), making it a promising, but still open approach to automated loci determination.

On the other hand there is the symbolic approach followed in this work. Given a dynamic geometry construction with algebraic (i.e. polynomial) elements, a locus can be viewed as the solution of a parametric polynomial system where the parameters correspond to the semi–free symbolic coordinates of the moving point (recall that the moving point is constrained to a linear object, thus having one degree of freedom). In other words, the goal is discovering the algebraic properties (i.e. in terms of equations) that the (coordinates of the) tracing point must satisfy in order for the construction to keep its constraints while the moving point varies its position along its path. In this sense, the determination of loci can be viewed as a particular instance of automatic discovery in geometry.

The algorithm for automatic discovery of loci followed in this work was first proposed by the first author [20] and it derives from an earlier proposal for automatic discovery in elementary geometry via algorithmic commutative algebra and algebraic geometry using Gröbner bases in [12]. This same algorithm has recently been implemented by the DGS JSXGraph to determine the equation of a locus set using remote computations on a server [22], an idea previously developed by the authors in [17].

A different symbolic approach to loci discovery based on Wu's method [2] has been suggested by Chou [23] and implemented by Roanes-Macías and Roanes-Lozano in some particular situations [24]. However their implementation is purely algebraic with no graphical environment for diagram construction.

Roughly speaking, the procedure for automatic discovery goes as follows. A statement is considered where the conclusion does not follow from the hypotheses. Two types of points are considered in a construction: free and bounded points. A free point has two degrees of freedom while a bounded point is one subject to constraints. Depending on its number of constraints, the degree of freedom of a bounded point will be 0 or 1. Symbolic coordinates are assigned to the points of the construction (where every free point gets two new free variables u_i, u_{i+1}, and every bounded point gets up to two new dependent variables x_j, x_{j+1}) so the hypotheses and thesis are rewritten as polynomials $h_1, \ldots, h_n$ and t in $\mathbb{Q}[u, x]$. Eliminating the dependent variables in the ideal $(hypotheses, thesis)$, the vanishing of every element in the

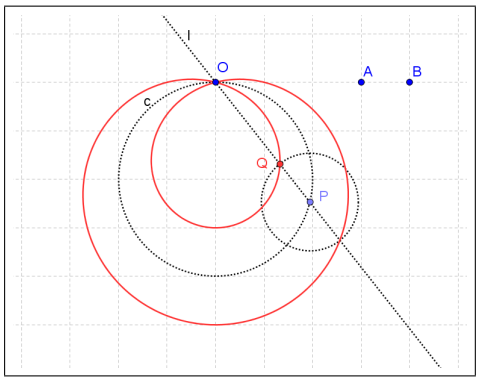

Figure 2: Limaçon of Pascal as the locus set traced by Q as P runs along the circle c.

elimination ideal $(hypotheses, thesis) \cap \mathbb{Q}[u]$ is a necessary condition for the statement to hold.

The problem of finding the equation of a locus for a particular geometric construction can be viewed as a particular case of this general setting with numerical coordinates for the free points, free variables corresponding to the tracing point and dependent variables to all other bounded points. The elimination (using Gröbner bases) of the dependent variables in the polynomial ideal obtained as translation of the construction leaves us with a set of polynomials in the independent variables. The zero set of these polynomials is a superset of the sought locus set. This algebraic set may contain extra components due sometimes to the algebraic nature of the method (it returns only Zariski closed sets) and due sometimes to some degenerate positions in the construction.

For instance, let us consider the limaçon of Pascal, a locus set that can be described as a conchoid as follows. Let O be a fixed point on the circle c and l be a line passing through O and P (any point on c). Let Q be a point on l such that $distance(P, Q) = k$, where k is a constant. The limaçon of Pascal is the locus set traced by Q as P moves along c as shown in Figure 2 (where k is given by the length of segment AB).

Treating the moving point P as the parameter for the locus construction we make the following assignment of coordinates: $P(a, b)$, $Q(x, y)$. If we consider that O is the point $(0, 2)$ and $distance(A, B) = 1$ we get the ideal $I = (a^2 + b^2 - 4, (x-a)^2 + (y-b)^2 - 1, x(b-2) - a(y-2))$ whose polynomials correspond, respectively, to the following geometric constraints: P is in the circle of center $(0, 0)$ and radius 2, $distance(P, Q) = 1$ and $Q \in Line(P, O)$.

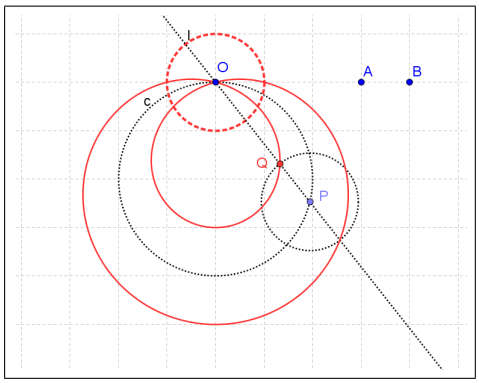

Figure 3: Limaçon of Pascal with extra circle.

Eliminating variables (parameters) a and b we obtain the following product of two polynomials $(x^4+2x^2y^2+y^4-9x^2-9y^2+4y+12)(x^2+y^2-4y+3)$. While the first factor provides the implicit equation for the actual limaçon, the second factor corresponds to a spurious circle corresponding to the degenerate case for which $P = O$ when the line l ceases to exist (see Figure 3).

The relevance of degeneracy conditions was pointed out from the very first works in Automatic Deduction in Geometry (see, for instance, [25, 26]). While an automatic procedure to identify general degeneracy conditions is still to be done, a method to efficiently treat the case of degeneracy conditions in the particular case of geometric loci is presented in section 4.

3. Connecting GeoGebra and Sage: an Automatic Deduction Environment

Besides communication through a standard command window, one can interact with Sage (either locally or remotely) using the Sage notebook working on a worksheet. In a Sage worksheet, one can write code using Sage, Python, and any other software included in Sage. Since the notebook presentation is done through a standard web browser, HTML code is also accepted.

The developed automatic deduction environment prototype consists of a Sage worksheet in which two different tasks are performed over GeoGebra constructions. Algebraic automatic deduction techniques based on Gröbner bases are used to either compute the equation of a geometric locus in the case of a locus construction or to determine the truth of a general geometric statement included in the GeoGebra construction as a Boolean variable. The Sage

worksheet includes a GeoGebra applet that allows the direct construction of a diagram or the upload of a previously designed GeoGebra construction.

More precisely, once a geometric diagram has been input in the applet, a JavaScript method provided by GeoGebra is used to obtain the GeoGebra XML description of the construction as a string inside a JavaScript variable. However, Sage does not include any standard way to transfer the content of a JavaScript variable to a Sage variable. The solution to this critical point comes from *tampering* with the code of Sage itself, using the following code (inside HTML) to control the cell flow.

```
code = "myxml = " + " \'\'\'" + document.applets[0].getXML()
  + " \'\'\' ";
\$("#cell_input_17").val(code);
evaluate_cell(17);
```

Once the XML description of the GeoGebra construction is available to Sage as a string, a first parsing process takes place in which the different GeoGebra elements in the construction are read, creating an algebraic counterpart for each element. This is done with an ad-hoc programming in Python involving several hundred lines of code (see [27] for details). As an example, the following snippet generates the polynomial equation of the line through two points.

```
def Line(n,p,q):
  """Constructs the line $n$ that goes through the points
    $p$ and $q$."""
  vdir=(x(q) - x(p), y(q) - y(p))
  eq=Set([(absc - x(p))*(y(q) - y(p)) -
    (orde - y(p))*(x(q) - x(p))])
```

After all the algebraic structures in the construction have been set up, the appropriate variables are initialized and the ideal corresponding to the task (locus or proof) is generated. Singular (a CAS with special emphasis on commutative algebra included in Sage) is then called to basically compute a Gröbner basis for this ideal. Each generator is factored and a process of logical expansion is performed on the conjunction of the generators in order to remove repeated factors.

Finally, the answer is presented to the user. In the case of a locus task, besides providing the locus equation, the graph corresponding to this equation is included in the GeoGebra applet together with the original locus construction.

Let us remark that the use of JavaScript to establish the direct communication between Sage and the GeoGebra applet is the key step in making the user experience completely automatic. This has made possible to circumvent the question/answer nature of Sage to generate what amounts to a one-click add-on for GeoGebra.

The system is presented in a prototype state and currently accepts a limited (easy to expand) set of construction primitives, namely Free points, Midpoint (point-point), Point (on Circle and on Line), Segment (point-point), Line (point-point, point-parallelLine), OrthogonalLine, Circle (center-radius, center-point, center-radiusAsSegment), Intersect (object-object), Locus and RelationBetweenTwoObjects (parallelism, perpendicularity).

Let us underline the fact that the system is solely based on Open Source software in contrast with other DGS-CAS implementations based on proprietary applications (e.g. [15] with Mathematica, [16] with Maple, [28] with Maple and/or Mathematica).

We illustrate the use of the prototype with two examples: the proof of a basic theorem specified in GeoGebra using a boolean variable and the computation of the equation of a classical locus. Both examples are presented as teaching scenarios where the idea is to complete the sequence construct/experiment/conjecture with a completely reliable answer, helping a student understand the difference in reliability between a numerical and a symbolic computation based on deep algebraic structures.

(only for electronic version) The following video shows a demonstration of use of the Sage worksheet SymbComp_GeoGebra.sws.

(only for electronic version) **[link to video 1]**

Access to the full code is available by request to the authors or by downloading the virtual machine (431 MB) available in [29], adapted from [30]. Once installed the virtual appliance, it is accessible through `https://localhost:8000`.

3.1. A GeoGebra example of proof

We consider the basic theorem that states that the three altitudes of a triangle meet at the orthocenter. This result can be easily reproduced in

terms of a boolean variable in a GeoGebra construction. Given triangle ABC, it suffices to consider the intersection point P of the altitudes a and b (through vertices A and B respectively). The boolean statement encompassing the theorem is $line(C, P) \perp line(A, B)$ as shown in Figure 4 (see boolean variable g in algebra window).

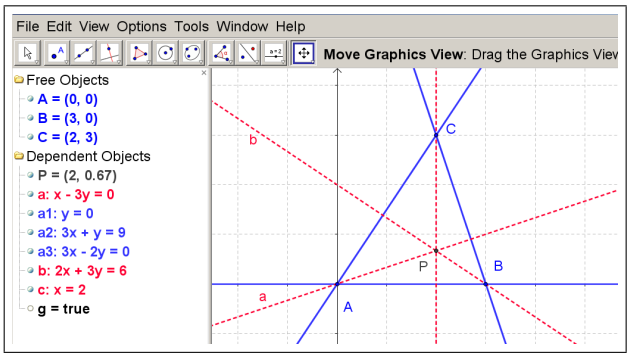

Figure 4: (numerically) True: the three altitudes of a triangle meet at the orthocenter.

The *true* provided by GeoGebra based on numerical computations is symbolically corroborated by Sage as shown in Figure 5.

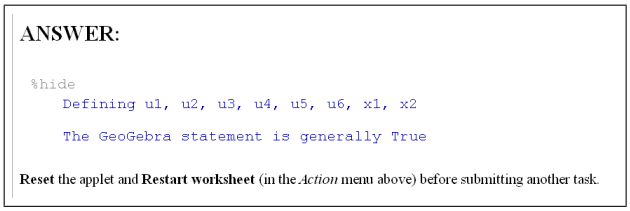

Figure 5: (symbolically) True: the three altitudes of a triangle meet at the orthocenter.

Notice that, unlike the answer provided by GeoGebra, the answer provided by Sage is not only based on symbolic computations but also completely general. Symbolic variables (u_1 to u_6) are used as generic coordinates for the three vertices, what makes the answer a general statement about any generic triangle. Degenerate situations (such as that of two vertices being the same) are excluded, hence the words "generally true" in the statement.

3.2. A GeoGebra example of locus

In the second example we consider the construction of the classical conchoid of Nichomedes. In Figure 6 we can see how GeoGebra plots only part of the curve in many instances.

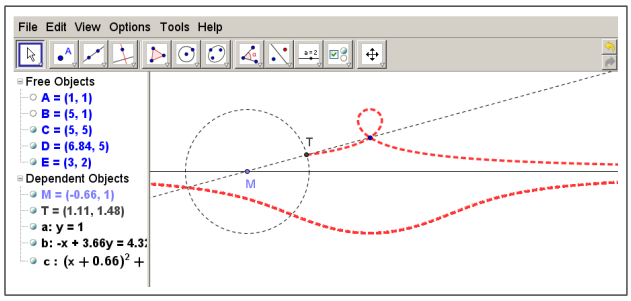

Figure 6: (partial) Current representation of the conchoid of Nichomedes in GeoGebra.

Figure 7 shows the equation of the locus set as given by Sage.

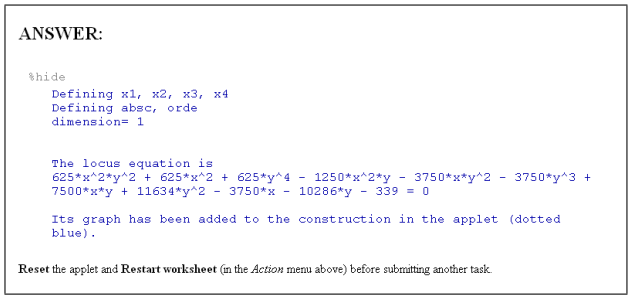

Figure 7: Equation of locus set as given by Sage.

Moreover, the complete graph of the given equation is automatically included in the GeoGebra applet together with the original (partial) representation of the locus set (figure 8).

3.3. Opening the System to any Intergeo Compliant System

The system presented above is specifically designed for a particular DGS (GeoGebra). To make it available to other systems a variation of the prototype has been developed that admits locus constructions specified using the Intergeo common file format. In this case, no graphic interface is provided and the (Intergeo) XML-description of the locus has to be directly included in a text area. After that, the system proceeds as the one described in Section 3 providing the equation of the locus and its graph.

Among the list of elements covered by the i2g format, we find several definitions of locus. However, no analogue of the GeoGebra boolean statement is considered (yet) by the format. This is the reason why the developed Sage worksheet does not accept true/false queries.

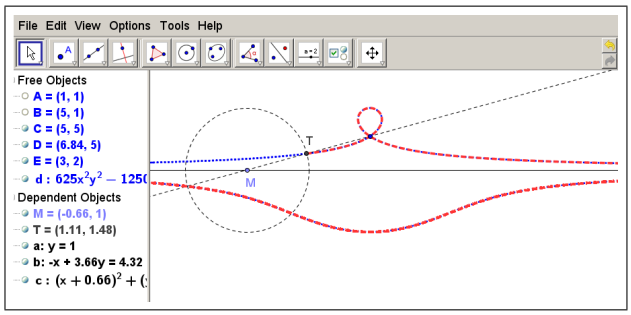

Figure 8: (complete) Representation of the conchoid of Nichomedes in GeoGebra.

Other works related to interactive web mathematics and automatic provers using OpenMath include [31, 32]. As noted in section 1.3, OpenMath is the standard for representing mathematical objects with their semantics that was taken as basis for the i2g file format.

To generate an i2g locus description, one can use any of the DGS supporting the i2g format (see [33] for a list of softwares and their current implementation status). As an example, we consider the construction provided in the Intergeo web site as an example of a construction with the `locus_defined_by_point_on_circle` element. It is the construction, in the DGS JSXGraph (`http://jsxgraph.de`), of a cardioid as the locus set traced by the point P as the point X runs along a circle (see [34]).

As a sample, the following is the encoding of the locus element in the i2g description of the construction:

```
<locus_defined_by_point_on_circle>
    <locus out="true">L</locus>
    <point>X</point>
    <point>P</point>
    <circle>c</circle>
</locus_defined_by_point_on_circle>
```

Figure 9 shows the equation and graph of the locus as provided by Sage.

Let us observe that although JSXGraph has implemented a remote tool to determine the equation of a locus [22], it is based on a remote server not readily available to the general user.

(only for electronic version) The following video shows a demonstration of use of the Sage worksheet SymbComp_Intergeo.sws.

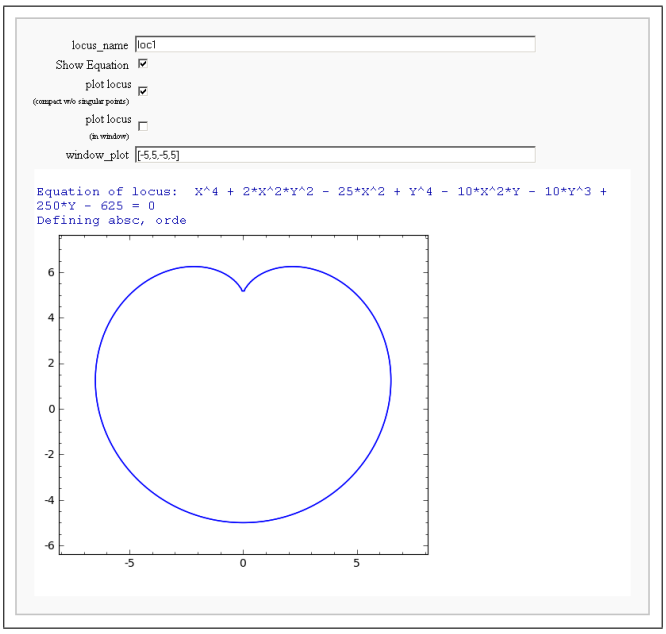

Figure 9: Equation and graph of cardioid as given by Sage.

(only for electronic version) **[link to video 2]**

Access to the full code is available by request to the authors or by downloading the virtual machine available in [29].

4. Using Gröbner Systems to Detect Degenerate Components

As recalled in section 2, the numerical approximation to the generation of loci has limitations such as the lack of adequate heuristics for an efficient interpolation of sample points and the impossibility of knowing a reliable algebraic description. The latter makes loci (in general, thus including envelopes) a special data structure in DGS that requires specific treatment. For example, construction of points in a locus is only possible in sophisticated versions of software (possible in GeoGebra only now in recent version 4.0). Another example, discussed below, is the computation of tangents to loci: Since you do not have an algebraic description, the system may not, in general, draw the tangent to a locus.

The symbolic approach solves these problems providing a polynomial characterization of a locus. However, it introduces two new problems. On

the one hand, the translation to polynomial equations of the geometric description of the locus can introduce extra algebraic constraints unrelated to the original geometric construction. On the other hand, extraneous solutions can also be introduced due to the elimination procedure. We will address both kind of problems in this section.

Recalling the limaçon of Pascal in section 2, eliminating a, b in the ideal generated by the hypotheses we get the equation of the limaçon and an extra circle due to the coincidence of P and O. This circle, although meaningful from an algebraic point of view (since the locus of Q when P and O coincide is constrained just by the condition $distance(Q, P) = 1$, so giving the circle), is clearly a rejectable solution. No current DGS would return such a locus component and a common user would not expect it. This degenerate condition could be explicitly excluded by the user (as in GDI [35]) stating that points P and O must be different. In such a case, the ideal would have an extra generator $(a^2 + (b-2)^2)t - 1$, and after eliminating a, b and t the true equation of the limaçon would be returned. Nevertheless, relying on the user to exclude degenerate conditions is an unsafe approach, since these conditions can be extremely difficult to find. Moreover, the educational uses of DG software advise against this approach.

Regarding extraneous solutions introduced through the elimination procedure, a simple illustration is the locus in figure 10.

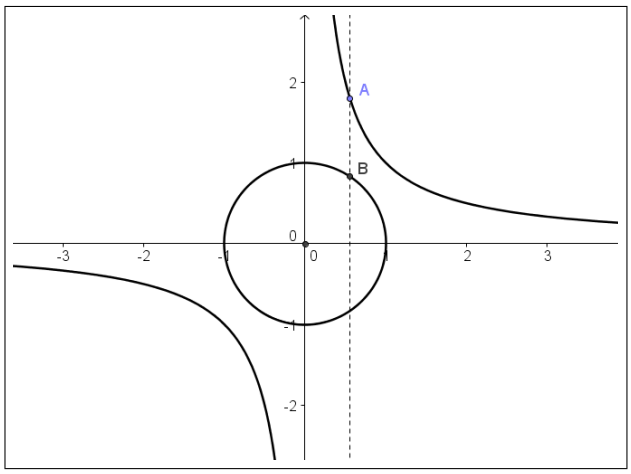

Figure 10: Given a moving point A on the hyperbola $xy = 1$, draw a perpendicular line to the x-axis and find the locus of points B on this line and on the unit circle.

The elimination procedure returns the unit circle $x^2 + y^2 = 1$, whereas

the true locus is the projection of the hyperbola on the unit circle, that is, the circle without the points $(0, \pm 1)$. Being I the ideal generated by the construction polynomials, $V = \mathbf{V}(I)$ the variety generated by I, $I_{x,y}$ the ideal that eliminates all variables but those of the locus point, and $\pi_{x,y}(V)$ the sought projection, we have that $\pi_{x,y}(V) \subseteq \mathbf{V}(I_{x,y})$. So, through elimination we get the smallest variety containing the locus, or, in others words, the Zariski closure of the projection. In order to handle this adherence in an automatic way, we tried the approach developed in [36], where the author gives alternative proofs to the Extension and Closure theorems of elimination theory for affine varieties. His proofs are algorithmic in essence, so they could be easily integrated in our system. And, with our implementation, removing the spurious points in the projection of the hyperbola was immediate, returning the locus as a constructible set, $\mathbf{V}(\langle x^2 + y^2 - 1 \rangle) \backslash \mathbf{V}(\langle x, y^2 - 1 \rangle)$. Nevertheless, with most elementary constructions our code was not able to remove the extraneous parts. For instance, computing the pedal of the ellipse $36x^2 + 100y^2 = 225$ with respect to $(0, 0)$ (see figure 11), the locus returned is $4x^4 + 8x^2y^2 - 25x^2 + 4y^4 - 9y^2 = 0$. Since this variety contains the origin, it cannot be the sought pedal. It contains some extra part to be removed that was not possible to determine with this method.

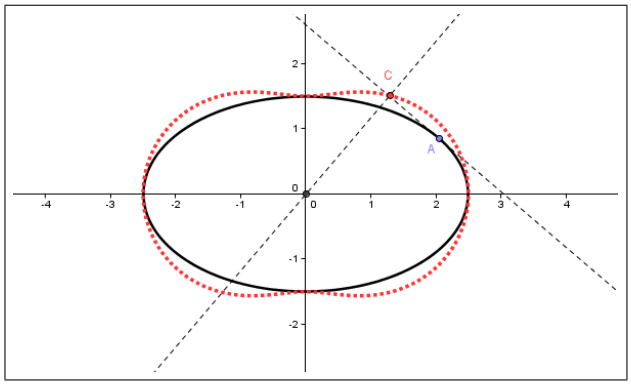

Figure 11: Pedal of the ellipse $36x^2 + 100y^2 = 225$ with respect to the origin.

The pedal example deserves more attention. It illustrates a current limitation of standard DGS. If we want to compute in GeoGebra the pedal of the pedal with respect to a given point, we need to compute the tangent by this point to the first pedal. Since for GeoGebra this first pedal is just a list of pixels, no internal method is available and the Locus command does not trigger. The symbolic approach will also fail in this case, since the polynomial

description of the pedal is erroneous.

Both kind of situations (removing degenerated parts and returning 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 (see [37] for a description of related issues in the context of automatic discovery in geometry). The variables occurring in the equations which describe a construction can be naturally divided into a set of parameters and a set of unknowns. The parameters correspond to the coordinates of the bounded points (recall that, since we are searching for the locus equation, the coordinates of the free points are their actual numerical values), while the unknowns (variables from now on), correspond to the coordinates of the locus point. Thus, the parametric polynomial system is formed by the polynomial geometric constraints. We seek then solutions of the parametric system in terms of the variable values, and the structure of the solution space.

The study of parametric polynomial systems via Gröbner bases was initiated by [38]. Since then, many improvements to this theory have been made (e.g. [39], [40], [41], [18], [42], to mention a few). Although most of the proposed algorithms can be used to solve the problems raised here, in this paper we primarily use the GröbnerCover algorithm (GC) proposed in [18], deferring a thorough study of other algorithms to a future work. Previous works where related theoretical approaches have been proposed are [43] and [44] in the contexts of automatic proving and discovery, respectively.

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 $a \in \mathbb{C}^m$.

We briefly recall here the main characteristics of the GC algorithm using $\mathbb{Q}$ as field and $\mathbb{C}$ as its algebraically closed extension. Given a parametric ideal $I \subset \mathbb{Q}[\bar{a}, \bar{x}]$ generated by $p_1, \ldots, p_r$ and homogeneous in the $\bar{x}$ variables, and fixed a monomial order $\succ_{\bar{x}}$ on the variables, let $I_a \subset \mathbb{C}[\bar{x}]$, for $a \in \mathbb{C}$, be the ideal generated by all $p(a, \bar{x}) \in \mathbb{C}[\bar{x}]$ for $p \in I$. The canonical *Gröbner cover* is a set $\{(S_1, B_1), \ldots, (S_s, B_s)\}$ of pairs such that

- The S_i's are pairwise disjoint, locally closed subsets of $\mathbb{C}^m$ with $\mathbb{C}^m = \bigcup S_i$.
- For $a, b \in \mathbb{C}^m$, $\mathrm{lpp}(I_a)$=$\mathrm{lpp}(I_b)$ if and only if there exists an i such that $a, b \in S_i$.

- The B_i's are finite subsets of monic polynomials of $\mathcal{O}(S_i)[\bar{x}]$, where $\mathcal{O}(S_i)$ denotes the ring of regular functions on S_i.
- For $a \in S_i$, $\mathrm{lpp}(B_i)$ is the minimal generating set of $\mathrm{lpp}(I_a)$, and evaluating every element of B_i at $a \in S_i$ we get the reduced Gröbner basis of I_a with respect to $\succ_{\bar{x}}$.

If the ideal I is not homogeneous, GC will return a similar result but with the second condition not always satisfied. It can be proved that the Gröbner cover of a system does not depend on the algorithm to compute it. The cover only depends on the ideal and the monomial order.

Concerning the geometric aspects of the GC algorithm, it should be noted that a Gröbner cover groups together all the values of the parameters for which the system of equations has the same type of complex solutions (counted with multiplicities) [18, p. 1393]. Discriminant varieties have been successfully used to study parametric polynomial systems (cf. [41]). They deal with the implicit description of the variables as a function of the parameters. Roughly speaking, a discriminant variety determines a partition of the space of parameters $\mathbb{C}^m$ such that over each open subset not intersecting the discriminant variety, the system has a constant number of solutions. While Gröbner covers depend on arbitrary choices, such as monomial orderings, discriminant varieties are intrinsic objects [41, p. 638]. The authors of this work define the notion of minimal discriminant variety, propose an algorithm to compute it, and discuss the state of the art for these varieties and parametric polynomial systems solving. One of the listed approaches consists of using comprehensive Gröbner bases, where the GC algorithm can be classified. Besides discriminant varieties, the notion of border polynomial should be cited as an alternative way to discuss parametric systems (see [45] for the relations between both concepts for triangular systems).

The GC algorithm puts the emphasis on obtaining a canonical description of the parametric system, while other algorithms using comprehensive Gröbner systems focus on effectiveness and speed. Using GC authors' own words, *the main focus [...] is not on the efficiency of the algorithm but on computing a Gröbner system that has as few segments as possible and satisfies some additional nice properties, so that the compact output allows an easy interpretation and the algorithm is easy to use in applications* [18, p. 1392]. Although the algorithm in [42] (which substantially improves the Suzuki-Sato algorithm (SS)) is generally more efficient, GC offers a compact discussion of the system, in general canonical. However, compactness in the

description comes at the price of using sometimes regular functions instead of polynomials (see [18, p. 1394]). A satisfactory approach in these situations has not been found so far, and thus in such cases we fall back to using SS. This limits the applications to just the detection of degenerate components in these cases.

There are two practical reasons for the use of GC (and SS). Since GC is a very sophisticated algorithm, the availability of an implementation is a compelling reason. Moreover GC's implementation is written in Singular, which provides an efficient incorporation in Sage, our main development tool. The SS algorithm, although mainly developed in Risa/Asir, also has an implementation in Singular available in [46]. Furthermore, its open source character will allow its installation on a web server to provide free remote processing of DGS tasks.

Our use of GC (and occasionally SS) exchanges the roles commonly assigned to parameters and variables. Since in this section we assume that the algebraic description of the locus or envelope is known (either by the algorithm in [35] or the implementation for envelopes in [47]), we focus on the use of the algorithms to remove degenerate components and spurious parts in the adherence.

4.1. Degenerate components

Rougly speaking, a degenerate component in a locus is one whose dimension is greater than that of the ideal defined by the values of the parameters that generate that component. For example, in the case of the limaçon in section 2, the circle $x^2+y^2-4y+3=0$ (variety of dimension 1) is obtained associated to $a=0$ and $b=2$ (parameter set of dimension 0), so we declare this component as degenerate. Our use of GC departs from the traditional way: we study the structure of the space of variables (x, y, the coordinates of the locus) so we exchange the roles of parameters and variables. GC determines in this case four segments:

- Segment 1
  - segment: $\mathbb{C}^2 \setminus (\mathbb{V}(x^2+y^2-4y+3) \cup \mathbb{V}(x^4+2x^2y^2-9x^2+y^4-9y^2+4y+12))$
  - basis: $\{1\}$
- Segment 2

  - segment: $(\mathbb{V}(x^2+y^2-4y+3) \setminus (\mathbb{V}(2y-3, 4x^2-3)) \cup (\mathbb{V}(x^4+2x^2y^2-9x^2+y^4-9y^2+4y+12) \setminus (\mathbb{V}(2y-3, 4x^2-3) \cup \mathbb{V}(y-2, x))$
  - basis: $\{(2x^2+2y^2-4y)b+(-x^2y-2x^2-y^3+2y^2-3y+6), (2x^2+2y^2-4y)a+(-x^3-xy^2+4xy-3x)\}$

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

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

Our strategy is to examine, for each locus factor, the dimension of the ideal generated by the values of the parameters for the segment in which that factor appears as minuend. Thus, for the circle $x^2+y^2-4y+3=0$ the parametric ideal is $\langle a, b-2\rangle$, of dimension 0, so that component is declared as degenerate. For the second factor of the locus the dimension of the parametric ideal is 1, equal to the dimension of the component of the locus which is determined then to be non-degenerate.

The following is a detailed scheme of the algorithm:

- **Algorithm deg-comp**
  - Input: factloc - the factors of the locus; pols - polynomials describing the construction; vars - the locus coordinates; paras - the mover coordinates.
  - Output: a list deg containing the degenerated components

- **Code**
  - begin
  - gc = GröbnerCover(pols)
  - for f in factloc:
    - df = dimension($\langle$f$\rangle$)

        * gbf = the Gröbner basis of the segment where f appears as minuend
        * If = ⟨gbf,f⟩
        * SIf = saturation(If, ⟨subtrahends of expressions with f in the segment⟩)
        * de = dimension(elimination(SIf, paras))
        * if de < df then append(deg, f)
    - return deg
    - end

(only for electronic version) The following document, offered as supplementary material, shows the trace of the execution of the program for the detection of degenerate components in the Sage worksheet DetectingDeg.sws.

(only for electronic version) **[link to supplementary document DetectingDeg.pdf]**

Access to the full code is available by request to the authors or by downloading the virtual machine available in [29].

4.2. Removing extraneous parts from the adherence

The GC algorithm can also be used to remove spurious parts of the loci introduced by the algebraic process of variable elimination. We have seen above how neither the numerical nor the symbolic approach to computing loci is adequate to determine the pedal of a pedal of an ellipse. The numeric approach does not allow the computation of tangent lines to the first pedal and the standard symbolic approach returns an extra point, also preventing the correct computation of the second pedal. A Gröbner cover for this construction, where (a, b) are the coordinates of the moving point on the ellipse, is composed of three segments

- Segment 1
    - segment: $\mathbb{C}^2 \setminus \mathbb{V}(4x^4 + 8x^2y^2 - 25x^2 + 4y^4 - 9y^2)$
    - basis: $\{1\}$

- Segment 2
  - segment: $\mathbb{V}(4x^4 + 8x^2y^2 - 25x^2 + 4y^4 - 9y^2) \setminus \mathbb{V}(y, x)$
  - basis: $\{(4x^2 + 4y^2)b - 9y, (4x^2 + 4y^2)a - 25x)\}$
- Segment 3
  - segment: $\mathbb{V}(y, x) \setminus \mathbb{V}(1)$
  - basis: $\{1\}$

The basis $\{1\}$ for the third segment tells us that the point $(0, 0)$ can not be described in terms of any parameter values so it must be removed from the locus whose algebraic description is then given by $\mathbb{V}(4x^4+8x^2y^2-25x^2+4y^4-9y^2) \setminus \mathbb{V}(y, x)$. The computation of the second pedal is now possible. It is given by the following equation that again includes the extra point $(0, 0)$ (see both pedals together with the original ellipse in Figure 12).

$262144x^{12} + 1572864x^{10}y^2 + 3932160x^8y^4 + 5242880x^6y^6 + 3932160x^4y^8 + 1572864x^2y^{10} + 262144y^{12} - 1472512x^{10} - 7759872x^8y^2 - 16314368x^6y^4 - 17108992x^4y^6 - 8951808x^2y^8 - 1869824y^{10} - 1010556x^8 + 2991888x^6y^2 + 13457944x^4y^4 + 13898000x^2y^6 + 4442500y^8 - 164025x^6 - 1366875x^4y^2 - 3796875x^2y^4 - 3515625y^6 = 0.$

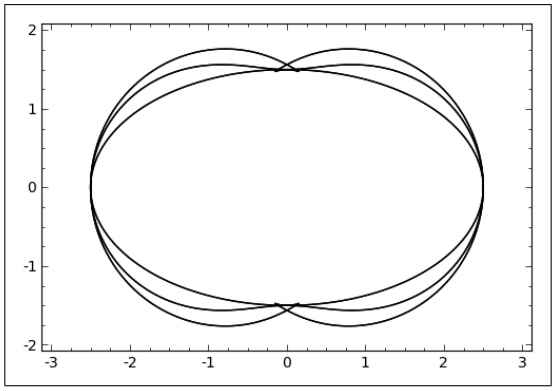

Figure 12: Ellipse $36x^2 + 100y^2 = 225$, pedal of the ellipse with respect to the origin and pedal of the pedal with respect to the origin.

This example shows how for the effective parametric treatment of dynamic geometry constructions, description in terms of constructible sets, rather than equations, has to be adopted.

Putting together this observation that allows to remove spurious elements in loci that are detected when the Gröbner basis for the segment is 1, with the suppression of degenerate components seen above, a simple procedure can be defined to perform both tasks. We have included a test to detect when a segment belonging to the true locus is contained in another segment. The first segment in then eliminated.

4.3. An example of failure

It was said above that the GC algorithm provides, among the ones that compute Gröbner systems, a compact description for a parametric system and to do that it sometimes has to replace ordinary polynomials by regular functions. The following example illustrates this situation, showing two possible solutions. We consider the deltoid as the envelope of the Simson lines. More concretely, given the triangle $A(-1,0), B(1,0), C(0,1)$ and a point $D(a,b)$ on its circumcircle, we consider the orthogonal projections of D to the sides AB and BC. The line joining these projections is a Simson line, and when D moves along the circumcircle the envelope of this family of lines generates the Steiner deltoid, $x^4+2x^2y^2+10x^2y-x^2+y^4-6y^3+12y^2-8y=0$. However, when X coincides with B the Simson line is undefined, introducing in the equation of the envelope a degeneracy condition, namely $x+y-1=0$. Similarly to what has been done for the limaçon above, an inequality condition can be defined to avoid this situation, as in [47]. If that condition is not detected (or is not made available in the option list provided to the user), the approach described here fails due to the existence of regular functions in the Gröbner basis corresponding to the following segment

$$(\mathbb{V}(x+y-1)\setminus(\mathbb{V}(y,x-1)\cup\mathbb{V}(4y^2-4y-1,x+y-1)))\cup(\mathbb{V}(x^4+2x^2y^2+10x^2y-x^2+y^4-6y^3+12y^2-8y)\setminus(\mathbb{V}(y,x-1)\cup\mathbb{V}(4y^2-4y-1,x+y-1)\cup\mathbb{V}(4y+1,16x^2-27)\cup\mathbb{V}(y-2,x)))$$

For a correct description of the reduced Gröbner basis of this segment we need a regular function given by $(4xy+x+4y^2-3y-1)b+(-x^3-x^2y+x^2-xy^2-3xy+x-y^3-2y^2+4y-1)$ when $x+y+1\neq 0$ and $(2x^2y+x^2+2y^3-y^2-y-1)b+(4x^2y+x^2-4y^3+7y^2-2y-1)$ otherwise, so the basis is given by

$$\{\{(4xy+x+4y^2-3y-1)b+(-x^3-x^2y+x^2-xy^2-3xy+x-y^3-2y^2+4y-1),\\ (2x^2y+x^2+2y^3-y^2-y-1)b+(4x^2y+x^2-4y^3+7y^2-2y-1)\},\\ (2x^2+4xy+x-2y^2+5y-2)a+(-x^3-x^2y-x^2-xy^2-3xy+x-y^3+4y^2-4y)\}$$

(A simple example with regular functions can be found in [18], p. 1393.)

A first solution consists of obtaining a GröbnerCover of the space of parameters a, b corresponding to the moving point, rather than the approach followed so far studying the structure of the space of coordinates of the locus. That is, we simply exchange the roles of coordinates of the tracing and moving points. For this deltoid, the segments are

- Segment 1
  - segment: $\mathbb{C}^2 \setminus \mathbb{V}(a^2 + b^2 - 1)$
  - basis: $\{1\}$
- Segment 2
  - segment: $\mathbb{V}(a^2 + b^2 - 1) \setminus \mathbb{V}(b, a - 1)$
  - basis: $\{(y + (-b^2 - b), y + (a * b - a)\}$
- Segment 3
  - segment: $\mathbb{V}(b, a - 1) \setminus \mathbb{V}(1)$
  - basis: $\{x + y - 1\}$

Applying the algorithm deg-comp to this GröbnerCover it follows that the component of the locus given by $x + y - 1 = 0$ is degenerate (it is one-dimensional while the segment is just one point) and the segment 2 corresponds to a non-degenerate component (the deltoid described above) of dimension 1, the same as the segment, which is the circumcircle minus the point $(1, 0)$.

We do not know the relationship, if any, between the structures of the spaces of variables and parameters, so it can not be determined which should be studied first. Since this article uses the solution of parametric systems to remove unwanted elements in loci whose description is previously known, preference has been given to the space of variables, but the alternative study of the parameter space has, in our view, similar doses of plausibility.

However, although infrequently, there are geometric constructions in which GC does not provide any solution (or does not do so in a reasonable time). For these cases we use the algorithm proposed in [48]. It generally performs

with a higher speed but the returned segments are generally not disjoint. This loss of compactness is the cause that our approximation is limited to the detection of degenerate components (which may appear several times) and does not consider the problem of removing spurious parts in loci.

For the curve $y^2 = x^3$, its 1–offset (envelope of the circles with radius 1 and centered on the curve) is a curve of degree 14 which factors into

$$(x^2+y^2-1)^3(729x^8+729x^6y^2+216x^7-1458x^5y^2-1458x^3y^4-2900x^6-2619x^4y^2+729x^2y^4-2376x^5-4892x^3y^2-4158xy^4+3870x^4-297x^2y^2+4072x^3+5814xy^2-1188x^2+729y^6-1685y^4+427y^2-1656x+529)$$

The GC algorithm is not able to find a partition for the space in a reasonable time, so we use a variation of the deg-comp algorithm on the partition into segments returned by SS. The component x^2+y^2-1 (provided by the singular point of the base cubic) is found to be degenerate. Figure 13 shows both the initial cubic and the (correct) offset as provided by Sage.

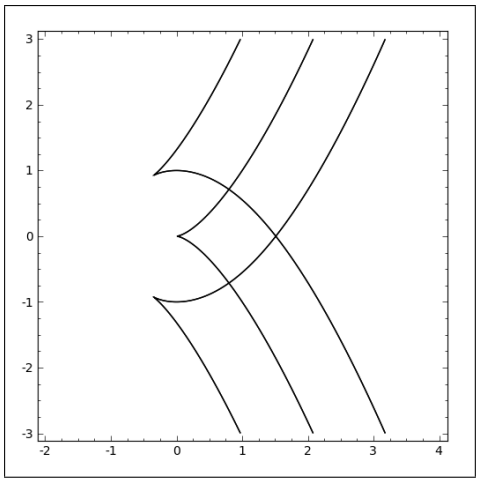

Figure 13: 1–offset for the cubic $y^2 = x^3$.

Concerning time, and with the exception of the above 1–offset, all the examples considered in this paper are computed in a few seconds (see the DetectingDeg supplementary document for exact times). For instance, our algorithm finds that the component x^2+y^2-1 of the 1–offset is degenerated in 0.2 seconds. It should be noted that these times are estimated using the canonical Sage server sagenb.org. The corresponding times using a Pentium 4 machine could be incremented by a factor of 10, what can result in a

significant loss of interactivity, intrinsic to dynamic geometry environments. The complexity of both the algorithms and systems involved in the prototype call for an implementation through a client-server architecture where optimal user experience can be achieved. In this case, the times of this approach make it suitable for an interactive use.

5. Conclusions

This article presents an interactive environment joining a dynamic geometry system and a computer algebra system in which automatic tasks can be processed following algorithms of automatic discovery based on Gröbner bases. This initial idea, also found in other authors, has in this presentation two main innovations. On the one hand, at the technical level, the system consists of an efficient complete integration of two different open source applications via a bidirectional communication between the applications. On the other hand, at the theoretical level, an algorithm to automatically remove unwanted elements in an automatically computed locus is presented. The described method, effective in most situations, removes degenerate components and extra adherence points in a locus, tackling two subtle problems inherent to the nature of algebraic polynomial methods. The algorithm is based on the study of parametric systems.

Although the system is based on readily available tools, we understand that a general DGS user can feel intimidated by the use of different new applications. To make the system more accessible it is our idea to develop a dedicated web application based on the remote Sage server capabilities in which all tasks are processed away from the user.

Acknowledgments

This research was partially supported by the project MTM2011-25816-C02-(01,02) funded by the Spanish *Ministerio de Economía y Competitividad* and the European Regional Development Fund (ERDF). The authors also want to thank Radoslaw Kirov for his hint on the use of JavaScript to control the cell flow in a Sage worksheet and Antonio Montes and Michael Wibmer for their personal comments on the GC algorithm. Finally, we want to acknowledge the work and suggestions of the referees, that have contributed to substantially improve our work.

References

[1] H. Gelernter, Realization of a geometry-theorem proving machine, in: E. Feigenbaum, J. Feldman (Eds.), Computers and thought, McGraw-Hill, New York, 1963, pp. 134 – 152.

[2] W. T. Wu, Mechanical Theorem Proving in Geometries, Springer, Vienna, 1994.

[3] W. Wu, Some recent advances in mechanical theorem proving of geometries, in: W. W. Bledsoe, D. W. Loveland (Eds.), Automated Theorem Proving: After 25 Years, volume 29 of *Contemporary Mathematics*, AMS, 1984, pp. 235–241.

[4] W. Wu, Basic principles of mechanical theorem proving in geometries, Journal of Automated Reasoning 2 (1986) 221–252.

[5] D. Wang, X. S. Gao, Geometry theorems proved mechanically using Wu's method - Part on Euclidean geometry, MM Research Preprints 2 (1987) 75–106.

[6] S. C. Chou, Mechanical Geometry Theorem Proving, D. Reidel Publishing Company, Dordrecht, Netherlands, 1988.

[7] B. Buchberger, Gröbner bases: an algorithmic method in polynomial ideal theory, in: N. K. Bose (Ed.), Multidimensional Systems Theory, Reidel, Dordrecht, Netherlands, 1985, pp. 184–232.

[8] D. Kapur, Geometry theorem proving using Hilbert's Nullstellensatz, in: B. W. Char (Ed.), SYMSAC'86: Proceedings of the fifth ACM symposium on Symbolic and algebraic computation, ACM Press, New York, NY, USA, 1986, pp. 202–208.

[9] S. C. Chou, W. F. Schelter, Proving geometry theorems with rewrite rules, Journal of Automated Reasoning 2 (1986) 253–273.

[10] B. Kutzler, S. Stifter, Automated geometry theorem proving using Buchberger's algorithm, in: B. W. Char (Ed.), SYMSAC'86: Proceedings of the fifth ACM symposium on Symbolic and algebraic computation, ACM Press, New York, NY, USA, 1986, pp. 209 – 214.

[11] D. Kapur, J. L. Mundy, Wu's method and its application to perspective viewing, Artificial Intelligence 37 (1988) 15–26.

[12] T. Recio, M. P. Vélez, Automatic discovery of theorems in elementary geometry, Journal of Automated Reasoning 23 (1999) 63–82.

[13] X. S. Gao, J. Z. Zhang, S. C. Chou, Geometry Expert, Nine Chapters, Taiwan, 1998.

[14] D. Wang, Geother: A geometry theorem prover, in: M. A. McRobbie, J. K. Slaney (Eds.), Automated Deduction - Cade-13, volume 1104 of *Lectures Notes in Computer Science*, Springer, 1996, pp. 166–170.

[15] F. Botana, J. L. Valcarce, A dynamic-symbolic interface for geometric theorem discovery, Computers and Education 38 (2002) 21–35.

[16] E. Roanes-Lozano, E. Roanes-Macías, M. Villar, A bridge between dynamic geometry and computer algebra, Mathematical and Computer Modelling 37 (2003) 1005–1028.

[17] J. Escribano, F. Botana, M. A. Abánades, Adding remote computational capabilities to dynamic geometry systems, Mathematics and Computers in Simulation 80 (2010) 1177–1184.

[18] A. Montes, M. Wibmer, Gröbner bases for polynomial systems with parameters, Journal of Symbolic Computation 45 (2010) 1391–1425.

[19] M. Dewar, OpenMath: An overview, ACM SIGSAM Bulletin 34(2) (2000) 2–5.

[20] F. Botana, Interactive versus symbolic approaches to plane loci generation in dynamic geometry environments, in: P. M. Sloot, A. Hoekstra, C. K. Tan, J. J. Dongarra (Eds.), Computational Science - ICCS 2002, volume 2330 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2002, pp. 211–218.

[21] P. Lebmeir, J. Richter-Gebert, Recognition of computationally constructed loci, in: F. Botana, T. Recio (Eds.), Automated Deduction in Geometry, volume 4869 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2007, pp. 52 –67.

[22] M. Gerhäuser, A. Wassermann, Automatic calculation of plane loci using Gröbner bases and integration into a dynamic geometry system, in: P. Schreck, J. Narboux, J. Richter-Gebert (Eds.), Automated Deduction in Geometry, volume 6877 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 68–77.

[23] S. C. Chou, Proving elementary geometry theorems using Wu's algorithm, in: W. W. Bledsoe, D. W. Loveland (Eds.), Automated Theorem Proving: After 25 years, volume 29 of *Contemporary Mathematics*, American Mathematical Society, 1984, pp. 243–286.

[24] E. Roanes-Macías, E. Roanes-Lozano, Automatic determination of geometric loci. 3d-Extension of Simson-Steiner theorem, in: J. A. Campbell, E. Roanes-Lozano (Eds.), Artificial Intelligence and Symbolic Computation, volume 1930 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2001, pp. 157–173.

[25] D. Kapur, Using Gröbner bases to reason about geometry problems, Journal of Symbolic Computation 2/4 (1986) 399–408.

[26] F. Winkler, Gröbner bases in geometry theorem proving and simplest degeneracy conditions, Mathematica Pannonica 1 (1990) 15–32.

[27] F. Botana, On the parametric representation of dynamic geometry constructions, in: B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, B. Apduhan (Eds.), Computational Science and Its Applications - ICCSA 2011, volume 6785 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 342–352.

[28] P. Todd, Geometry expressions: a constraint based interactive symbolic geometry system, in: F. Botana, T. Recio (Eds.), Automated Deduction in Geometry, 6th International Workshop, ADG 2006, volume 4689 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, Heidelberg, New York, 2007, pp. 189–202.

[29] Virtual Box server for Automatic Deduction, `http://193.146.36.205/VBServer4AutomaticDeduction.ova`, Last accessed January 2013.

[30] Sage Lite Server2, `http://modular.math.washington.edu/home/wstein/www/home/emil/doc/html/en/relase-notes-slvms-b.htm`, Last accessed January 2013.

[31] O. Caprotti, A. M. Cohen, On the role of OpenMath in interactive mathematical documents, Journal of Symbolic Computation (2001) 351–364.

[32] H. Barendregta, A. M. Cohen, Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants, Journal of Symbolic Computation 32 (2001) 3–22.

[33] Intergeo, `http://i2geo.net/xwiki/bin/view/I2GFormat/ImplementationsTable`, Last accessed January 2013.

[34] JSXGraph, `http://http://i2geo.net/xwiki/bin/download/I2GFormat/WebHome/locuscardioid.xml`, Last accessed January 2013.

[35] F. Botana, J. L. Valcarce, A software tool for the investigation of plane loci, Mathematics and Computers in Simulation 61(2) (2003) 139–152.

[36] P. Schauenburg, A Gröbner-based treatment of elimination theory for affine varieties, Journal of Symbolic Computation 42 (2007) 859–870.

[37] T. Recio, M. P. Vélez, An introduction to automated discovery in geometry through symbolic computation, in: U. Langer, P. Paule (Eds.), Numerical and Symbolic Scientific Computing, volume 1 of *Texts and Monographs in Symbolic Computation*, Springer Vienna, 2012, pp. 257–271.

[38] V. Weispfenning, Comprehensive Gröbner bases, Journal of Symbolic Computation 14 (1992) 1–29.

[39] A. Suzuki, Y. Sato, An alternative approach to comprehensive Gröbner bases, Journal of Symbolic Computation 36 (2003) 649 – 667.

[40] A. Montes, A new algorithm for discussing Gröbner basis with parameters, Journal of Symbolic Computation 33 (2002) 183–208.

[41] D. Lazard, F. Rouillier, Solving parametric polynomial systems, Journal of Symbolic Computation 42 (2007) 636–667.

[42] D. Kapur, Y. Sun, D. Wang, A new algorithm for computing comprehensive Gröbner systems, in: S. M. Watt (Ed.), ISSAC '10 Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation, ACM New York, 2010, pp. 29 – 36.

[43] X. Chen, P. Li, L. Lin, D. Wang, Proving geometric theorems by partitioned–parametric Gröbner bases, in: H. Hong, D. Wang (Eds.), Automated Deduction in Geometry, 5th International Workshop, ADG 2004, volume 3763 of *Lectures Notes on Artificial Intelligence*, pp. 34–43.

[44] A. Montes, T. Recio, Automatic discovery of geometry theorems using minimal canonical comprehensive Gröbner systems, in: F. Botana, T. Recio (Eds.), Automated Deduction in Geometry, 6th International Workshop, ADG 2006, volume 4689 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, Heidelberg, New York, 2007, pp. 113–139.

[45] M. Moreno Maza, B. Xia, R. Xiao, On solving parametric polynomial systems, Mathematics in Computer Science 6 (2012) 457–473.

[46] A. Suzuki, `http://kurt.cla.kobe-u.ac.jp/~sakira/CGBusingGB/`, Last accessed January 2013.

[47] F. Botana, J. L. Valcarce, Automatic determination of envelopes and other derived curves within a graphic environment, Mathematics and Computers in Simulation 67 (1-2) (2004) 3–13.

[48] A. Suzuki, Y. Sato, A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases, in: Proceedings of the 2006 international symposium on Symbolic and algebraic computation, ISSAC '06, ACM, New York, NY, USA, 2006, pp. 326 – 331.