Teaching loci and envelopes in GeoGebra*

Francisco Botana[1] and Zoltán Kovács[2]

[1] Dept. Applied Mathematics I, University of Vigo,
Campus A Xunqueira,
E-36005 Pontevedra, Spain
`fbotana@uvigo.es`
[2] Dept. Mathematics Education, Johannes Kepler University,
Altenberger Strasse 54,
A-4040 Linz, Austria,
`zoltan@geogebra.org`

Abstract. GeoGebra is open source mathematics education software being used in thousands of schools worldwide. Since version 4.2 (December 2012) it supports symbolic computation of locus equations as a result of joint effort of mathematicians and programmers helping the GeoGebra developer team. The joint work, based on former researches, started in 2010 and continued until present days, now enables fast locus and envelope computations even in a web browser in full HTML5 mode. In conclusion, classroom demonstrations and deeper investigations of dynamic analytical geometry is ready to use on tablets or smartphones as well.

In our talk we consider some typical secondary school topics when investigating loci is a natural way of defining mathematical objects. Such topics include definition of a parabola and other conics in different situations like synthetic definitions or points and curves associated with a triangle. In most secondary schools, however, no other than quadratic curves are discussed, but generalization of some exercises and also every day problems will introduce higher order algebraic curves. Thus our talk will mention the cubic curve "strophoid" as locus of heights of a triangle when one of the vertices moves on a circle. Also quartic "cardioid" and sextic "nephroid" can be of every day interest when investigating mathematics in a coffee cup.

We will also focus on GeoGebra specific tips and tricks when constructing a geometric figure to be available for getting the locus equation. Among others, simplification and synthetization (via the intercept theorem) will be mentioned.
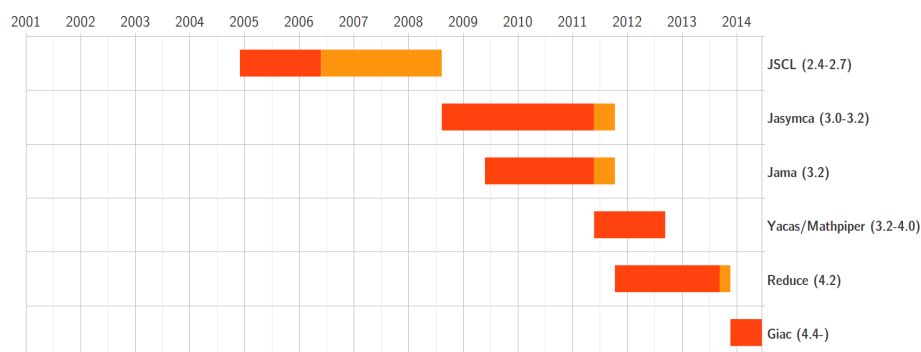
1 Introduction

1.1 Overview

GeoGebra [1] is open source mathematics education software being used by millions of users worldwide. It is mainly used to visualize mathematical relations in a dynamic way by supporting reading correlations off not only visually but also numerically.

---

* Original version of this paper has been published as a GeoGebraBook at `http://www.geogebratube.org/student/b128631`

This approach has been continuously extended since 2004 by using an embedded computer algebra system (CAS) in GeoGebra in version 2.4. In those days CAS *JSCL* was used which has been changed to *Jasymca* [2] (2008, GeoGebra 3.0), *Jama* [3] (2009, GeoGebra 3.2), *Yacas/Mathpiper* [4] (2011, GeoGebra 3.2-4.0), *Reduce* [5] (2011, GeoGebra 4.2) and *Giac* [6] (2013, GeoGebra 4.4).

2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	

JSCL (2.4-2.7)

Jasymca (3.0-3.2)

Jama (3.2)

Yacas/Mathpiper (3.2-4.0)

Reduce (4.2)

Giac (4.4-)

Red period means continuous development phase in GeoGebra. Orange period shows stable phase with no longer development.

Interest in using CAS support in GeoGebra started a more specialized interest in computing the algebraic equation of more general geometric objects than lines and circles, namely locus equations. A team of mathematicians (including the first author) located in Spain offered scientific partnership and collaboration for the GeoGebra team (including the second author) located in Austria, and their joint work was funded by the *Google Summer of Code* program in 2010 by supporting a Spanish university student *Sergio Arbeo* to implement the computation of algebraic locus equations for GeoGebra 4.0. Arbeo programmed the computations by using an extra CAS *JAS* [7] in GeoGebra, but his code was later modified by the second author to use Reduce (and even later Giac) instead.

As a result, locus equation computations are already present in GeoGebra since version 4.2 and because of the numerous user feedback the newer versions (including GeoGebra 5) contain some additional enhancements and bug fixes as well. Also many users found the introduced **LocusEquation** command useful and easy to use in education as well. In Section 2 we consider some possible classroom uses for the **LocusEquation** command.

However Arbeo covered a wide set of classroom problems, meanwhile new mathematical methods appeared to handle some problematic situations. The joint work [8] of *Montes*, *Recio, Abánades*, *Botana* and the *Singular* CAS team yielded to have a powerful method to compute locus equations by using the Gröbner cover (grobcov) package in Singular. In this way GeoGebra has been extended to outsource computations to *SingularWS* [9,10], an external web service computing locus equations for

GeoGebra (among other computations). This method has been found extensible to compute not only locus equations but envelopes as well. In Section 3 we show some of these envelopes, pointing out the possibility of introducing them also in secondary schools.

2 Loci

2.1 An example

According to Wikipedia, locus is a *set of points whose location satisfies or is determined by one or more specified conditions*. Being more specific, in GeoGebra locus is the set of output points P' constructed by given steps while the input point P is running on a certain path. In other words, let point P be an element of a path, and let point P' is the output point for the chosen input P after some transformations of P into P'.

In general the locus is a curve as the output set of points P', since also the input points P build up a curve. For example, let the input curve be circle c and P is a perimeter point of c. Let the center of the circle be C. Now let us construct point P' such that P' is the midpoint of PC. Clearly, the locus curve here is also a circle described by center C and half of the radius of c.

This example can be entered into GeoGebra either by using the graphical user interface with the mouse, or by the keyboard in the Algebra Input (here we put point C into $(2,3)$ and use radius 4):

- **C=(2,3)**
- **c=Circle[C,4]**
- **P=Point[c]**
- **P'=Midpoint[P,C]**
- **Locus[P',P]**

Now by using **LocusEquation[P',P]** instead of **Locus[P',P]** we can also check the result algebraically: an implicit curve is displayed with the equation $x^2 - 6x + y^2 - 4y = -9$. It is also possible to see how the equation changes dynamically when point C is dragged.

How can this equation be computed mathematically? Let us define coordinates x_C, y_C, x_P, y_P, $x_{P'}$ and $y_{P'}$ for the points defined above. Now the following equations are valid:

1. $x_C = 2$
2. $y_C = 3$
3. $(x_P - x_C)^2 + (y_P - y_C)^2 = 4^2$
4. $x_{P'} = \frac{x_P + x_C}{2}$
5. $y_{P'} = \frac{y_P + y_C}{2}$

What we need is to convert this equation system to a single equation containing only coordinates of point P'. In algebra this computation is called *elimination*, i.e. eliminating all variables except $x_{P'}$ and $y_{P'}$.

In GeoGebra this computation is achieved by the Giac CAS in the background, but it can also be computed directly by using a GeoGebra command (which calls the appropriate Giac statement):
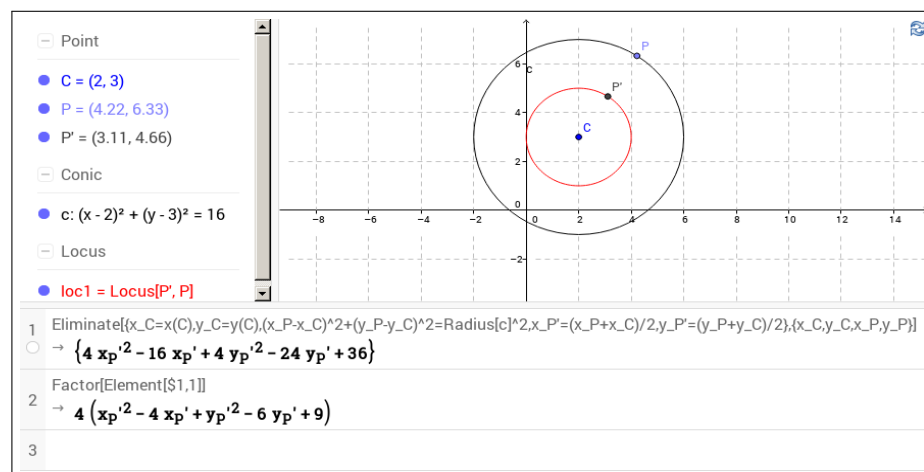
Here the user can add command **LocusEquation[loc1]** to make GeoGebra compute the equation automatically. By clicking on the marble to the left of the input line GeoGebra will also display the geometric form of the equation, i.e. another circle will be drawn (the same as loc1).

### 2.2 Classroom examples

Definition of a parabola. Maybe the most natural example to illustrate how useful loci are is definition of a parabola. As Wikipedia writes it is *the set of points equidistant from a single point (the focus) and a line (the directrix)*. This definition is for beginners, however, not easy to handle. There is at least one abstract step inbetween, namely that to find the distance from a line we may need a perpendicular being drawn.

Thus, when focus F and directrix d are given, constructing one point P of parabola p is as follows:

1. Choose an arbitrary point D of d.
2. Construct a perpendicular line to d on D.
3. Construct the bisector b of D and F.
4. Let the intersection point of the perpendicular line and b be P.
5. Now P is a point of parabola p since PD=PF (because bisector b is actually the axis for the mirroring of point D to F).

In fact drawing bisector b is also a hidden step since we implicitly used some basic properties of the reflection.

This kind of definition of the parabola is usual in many secondary schools, however the equivalence of this definition and the analytical one (that is, the usual formula for a parabola is $y = ax^2 + bx + c$ for some constants a, b and c) is not obvious.

Equation of the parabola: $x^2 - 4x - 4y = -4$
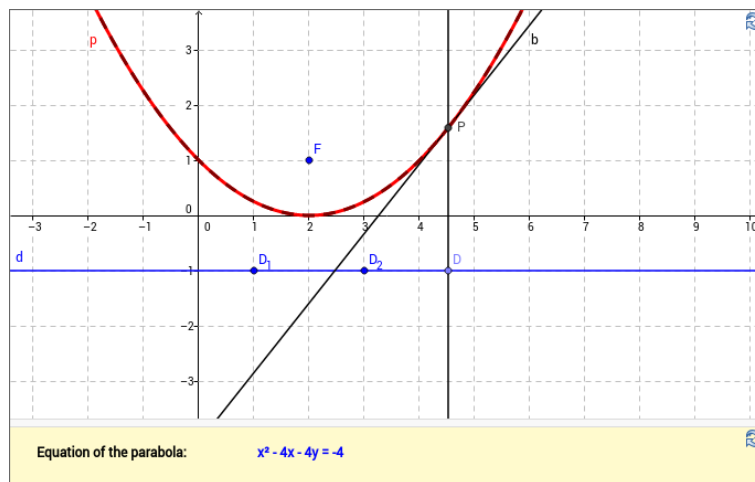
In the figure we can see a dark red dashed parabola which is the real locus drawn by GeoGebra numerically. The lighter red curve has been computed symbolically by the **LocusEquation** command. In this special case these two curves are exactly the same. We will see some examples below where it is not the case. The reason comes from the algorithm we use to compute the equation. (In many cases the symbolical result can be improved by using extended algorithms.)

Locus of the orthocenter. A parabola can also be obtained by getting the orthocenter points of a triangle if two vertices are fixed and the third one moves on a line which is parallel to the opposite side.

Equation of the locus: x² - x + 3y = 3
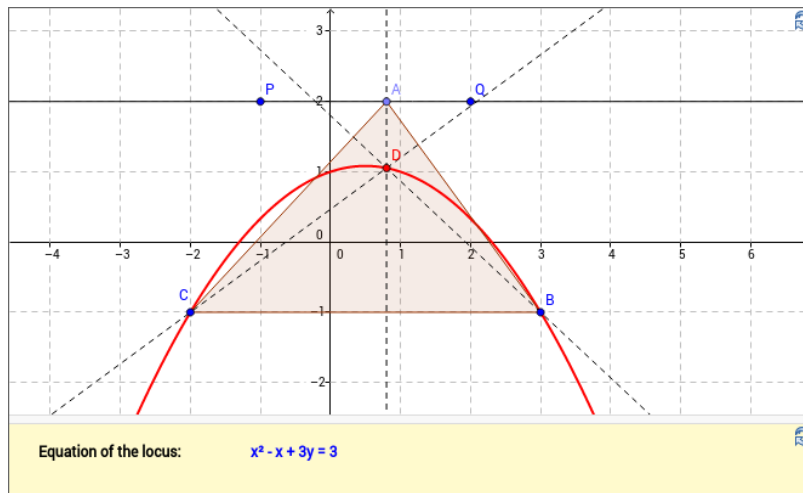
In this figure point A is constrained by line PQ. Students can drag point A on line PQ and see how the orthocenter D is changing meanwhile. It is not obvious to prove that the locus here is a parabola, but the students are able at least to get experience by changing points B and C by preserving parallelism of lines PQ and BC.

On the other hand, the locus equation will not be essentially different on other positions of P, Q, B and C: it will be quadratic in most sets of positions. Some of these positions seem easy to investigate, for example when PQ is perpendicular to BC (here the result will be a linear equation). Others, for example by putting P to $(1,1)$ and not changing anything else in the set, the locus result is a hyperbola, namely $x^2 + xy + y = 5$. First, this formula is hard to analyze in secondary school since it is not in explicit form like a function $y = f(x)$. Second, this formula is still a quadratic implicit equation and thus it can open horizons of generalization to cover all kind of conics.

In case P=$(1,1)$, Q=$(2,0)$, B=$(3,1)$, C=$(3,1)$ the computed locus equation is $-xy - x + y^2 + 3y = -2$, i.e. $-xy - x + y^2 + 3y + 2 = 0$ whose left hand side is the product of $(y+1)$ and $(y - x + 2)$, two lines, namely $y = -1$ and $y = x - 2$ written in the usual explicit form. In this constellation height of side AB always lies on line CQ since it is perpendicular to PQ. Thus point D will also lie on line CQ, so it seems sensible that the locus equation is CQ in this case. Unfortunately, GeoGebra's **LocusEquation** shows an extra line here, not only CQ (which has the explicit equation $y = x - 2$) but also another one. This example shows that the real locus may be a subset of the result of the **LocusEquation** command.

One further step forward is to constrain point A on a circle, not a line. In this way we can obtain non-quadratic locus equations like the strophoid formula which is a cubic one.

Equation of the locus: $x^4 - 4x^3 + x^2 y^2 + 8x^2 y + 5x^2 + 2x y^2 - 8x y + 2x + y^2 - 16y = 8$
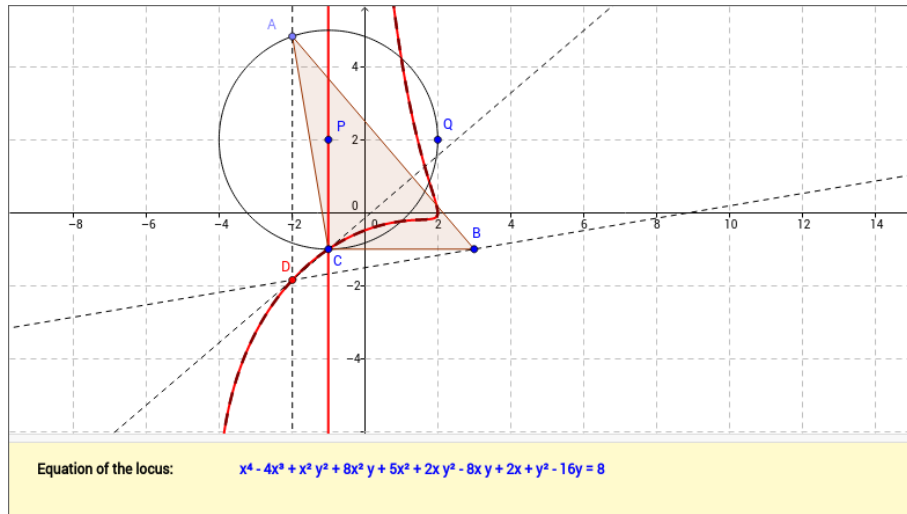
Here points P and Q define a circle which has perimeter point A as a constraint.

In this figure a quartic equation is shown, but the real locus is a cubic curve. That is an extra component is shown (here line $x = -1$) as in many other cases when dragging points P, Q, B or C. On the other hand, by moving point P down, for example into $(-1, 1)$ or $(-1, 0)$, the locus is the same as **LocusEquation** computes: in these cases the locus is a real quartic curve.

A beautiful side case is when B=P and C=Q. In this case the real locus is a right strophoid [10] curve and an extra line component is drawn on points C and Q.

2.3   Technical details

Computational background. We think that most Readers of this papers are neither technicians, nor mathematicians, but teachers. Nevertheless, it is good to know some computational details how the **LocusEquation** command works.

Computing a locus equation can be time consuming even for fast computers. Basically, a set of equations has to be created in the background: the more objects we have in our construction, the more variables and equations we need. After setting up algebraic equations, they have to be solved symbolically in an efficient way. For this task we use *Gröbner bases* [11].

GeoGebra uses the Giac computer algebra system to compute Gröbner bases as efficiently as possible, but the general method is still double exponential in the number of variables. On the other hand, Giac runs in a web browser in today's computers, and this slows down computations by almost one magnitude. (This means that computing locus equation in the desktop version of GeoGebra is still much faster than observing the construction in a web browser. For the future, however, there are plans to speed up JavaScript computations by substituting them with native instructions.)

Giac is a powerful CAS, but it can slow down if extreme input must be processed. That is why it is desired to solve equations only having integer coefficients. To achieve this, it is suggested to use so-called dynamic coordinates in GeoGebra: to create free points A', B', C', ... first and then define point **A=DynamicCoordinates[A',round(x(A')),round(y(A'))]**, then use similar definitions for points B and C and so on.

Also using "easy" coordinates will speed up computations. For example, putting A into the origin, B on the x-axis and using small integers instead of larger numbers may decrease computation time significantly.

Supported construction steps. Since the Gröbner basis computation assumes algebraic (polynomial) equations, there are restrictions for the available construction steps for the **LocusEquation** command. First of all, only Euclidean construction steps are supported. Even if a step could be converted into a Euclidean construction, some non-trivial way of wording are not supported, for example, when the user defines a parabola by entering its explicit formula, then it cannot be discovered by the **LocusEquation** command at the moment. Instead, the parabola must be constructed by using the appropriate GeoGebra tool.

Most Euclidean two dimensional construction steps are already supported. Javadoc at the GeoGebra Developer Wiki [12] provides with a full list of them.

If a geometry problem is described fully or partially by formulas, it may be difficult to translate it to a purely Euclidean construction. Below we will see an example how this can be achieved.

Agnesi's witch. Here we provide two examples to implement Agnesi's witch [13] in GeoGebra. The first approach will be a general way which will result in slow computation and some extra components. The second approach will be much faster and results only in one extra component.

For the first approach we simply consider the formula $y = \frac{1}{x^2+1}$. Here we need to define the unit (1) and compute the square of x based on this unit, then add these two lengths. Then we need to compute the reciprocal of the result, and translate the final length y to the correct position of the coordinate system.

In this figure we can see a numerical locus in red and a symbolical locus in blue. In fact Agnesi's curve is just a part of these curves since the conversion of its formula will introduce extra components. Let us follow the steps we made in this figure:

1. Point A is created (origin).
2. Point B is created, $(0, 1)$.
3. Line a is the y-axis.
4. Line b is the x-axis.
5. Point C lies on the x-axis, it will be the projection of a point of the curve to the x-axis, i.e. its abscissa will be x.
6. We will use the intercept theorem [14] to construct x^2. So we create a triangle with sides having length 1 (AB) and x (AC). This triangle will be right, but this property is not necessary. The third side of the triangle will be line c.
7. For the intercept theorem we prepare length x also on line AB by drawing circle d.
8. Point D is intersection of line a and circle d. (In fact there are two intersection points here, but we use the "upper" one.) Now AD=x.
9. Line e is parallel to D and lies on D.
10. Point E is intersection of line b and d. By using the intercept theorem obviously AE=x^2.
11. We are preparing addition, thus we draw another circle f around the origin having unit radius.
12. Point F is intersection of line b and circle f. (In fact there are two intersection points here, but we use the "left" one.) Now EF=$x^2 + 1$.
13. We would like to copy the unit length, so we create segment g as the unit (i.e., AB).
14. Circle h is around point E with unit radius.
15. Line i is perpendicular to line b and lies on point E.

16. Point G is intersection of line i and circle h. (In fact there are two intersection points here, but we use the "upper" one.) Now GE is a copy of the unit. This is a preparation for applying another intercept theorem.
17. We will use triangle EFG for the intercept theorem, thus we draw line FG as line j.
18. Point H is again an intersection of circle h and line b. (The "left" one.) Now we copied the unit as EH, too.
19. Drawing line k as a parallel one with line j through H.
20. Intersection of lines i and k is point I.
21. Now applying the intercept theorem for length IE we obtain IE=$\frac{1}{x^2+1}$. This will be y.
22. Now we will copy this length to point C orthogonally. Thus we draw a perpendicular with line b through C. This will be line m.
23. We copy length IE to point C upwards, thus we draw circle p around C with radius IE.
24. "Upper" intersection point of m and p will be point J.
25. Locus of point J while C is moving on the x-axis is what we search for. In fact, only those points J are proper which have positive abscissa.
26. Finally, **LocusEquation** shows a 12 degree polynomial, $x^8y^4 - 2x^4y^4 - 2x^4y^2 + y^4 - 2y^2 + 1$, which is a product of the cubic Agnesi curve and its reflection to the x-axis, and two other cubics (reflections of each other), namely $x^2y - y - 1$, $x^2y - y + 1$, $x^2y + y - 1$ and $x^2y + y + 1$.

After finishing this construction it is clear that we almost surely obtain extra components since it is impossible to exclude the "right" intersection point in steps 12 and 18, for example. The construction process described above illustrates the weakness of the Euclidean (i.e. in fact algebraic) method.

Finally we refer to a more simple definition of Agnesi's curve, also used by a Google "doodle" on the 296th anniversary of Maria Agnesi's birth on 16 May 2014.
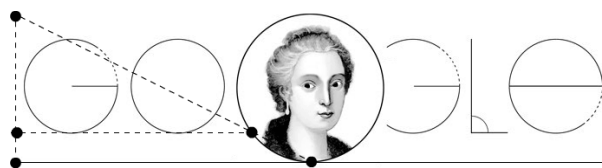
This "easy" definition allows GeoGebra to show Agnesi's witch much faster than above and make it computationally possible to drag the input points even in a web browser. Of course, in many cases such a simplification is an intellectual challenge by searching for algebraic or geometric simplifications to result in less variables in the Gröbner basis computations.
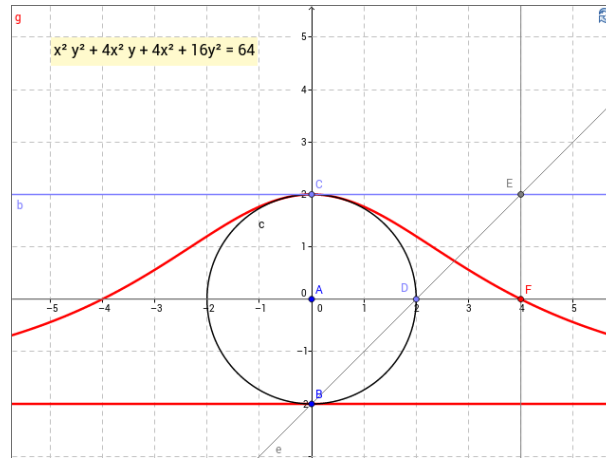
As an exercise, we leave to the Reader to prove that the trace of point F is $y = \frac{1}{x^2+1}$ if B is in the origin, A=$(0, 1/2)$ and C=$(0, 1)$. Also an exercise in GeoGebra to improve this figure: use **DynamicCoordinates** instead of point capturing to grid points (which yields non-continuous motion for point D). Another improvement can be to put the equation text into a fix position, preferably in the second Graphics View.

3 Envelopes

3.1 Motivation

A definition of a circle, according to The Free Dictionary is *a plane curve everywhere equidistant from a given fixed point, the center*. Now by constructing the tangent line in a point of the circle we find that the tangent is perpendicular to the radius. When considering the trace of the tangent lines, we find that the union U of the tangents of a circle is the whole plane except the disc inside the circle.
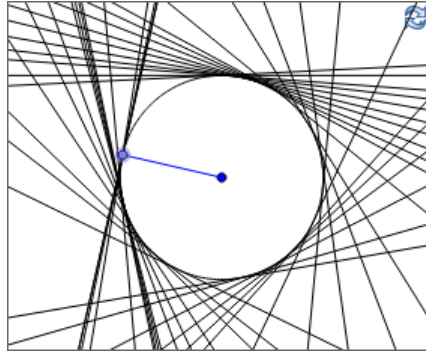
Now let us consider the same figure from another point of view. Having the set U we may be interested of a curve such that its tangents are the lines of U. Such a curve can be the given circle, but that it is the only possible curve (that is, the question has a unique answer) is not straightforward.

On the other hand, each element of U is equidistant from the center of the circle because distance is defined by measuring orthogonal projection. This idea leads us to define a parabola by considering the set U' of lines being equidistant from a given point F and a given line d. To measure the distance from line d we consider each point D of line d and take the perpendicular bisector of points F and D.
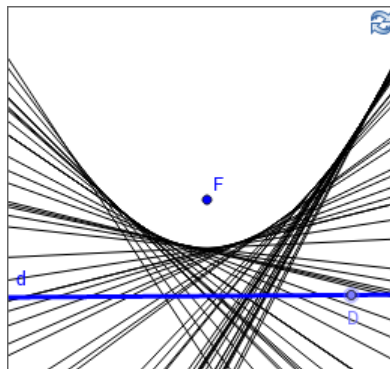
We have already mentioned in the previous section that usual definition of a parabola p' is that it is the locus of points P' which are equidistant from focus F' and directrix d'. Now let D' be an arbitrary point of d', line b the bisector of segment F'D' and P' the intersection of the perpendicular to d' in D' and b. A well-known property of b that it is the tangent of parabola p' in point P'. To prove that, consider the following figure.
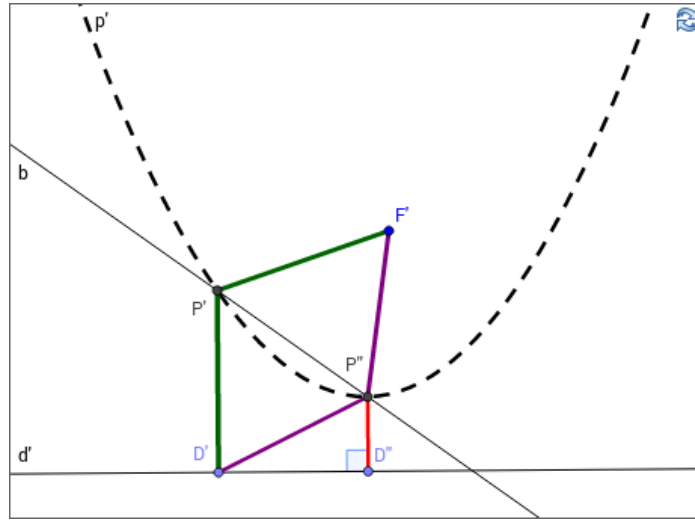
Let us assume that b is not a tangent of parabola p' in point P'. Then there is another point P" on b, also element of the parabola, that is b is a secant line of p'. Assumably this point P" was created by foot point D" (element of d'), for which D"P"=P"F'. Since b is the bisector of D'F', also D'P"=P"F' holds. But this means that D'P"=D"P", that is in triangle D'D"P" (which is a right triangle) hypothenuse D'P" and cathetus D"P" have the same length, which is impossible. This contradiction ensures that b is a tangent, not a secant.

Thus we proved that these two different definitions of a parabola (i.e. the classical one by using locus, and this second one which uses the concept of trace of the bisector) are equivalent, that is they define the same parabola.

3.2 A detailed example

String art, or pin and thread art, according to Wikipedia [15], is characterized by an arrangement of colored thread strung between points to form abstract geometric patterns or representational designs such as a ship's sails, sometimes with other artist material comprising the remainder of the work.
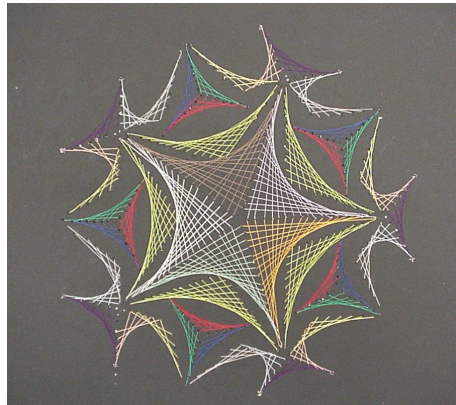
In the figure above strings play the same role as tangents in the previous examples. As Markus Hohenwarter, inventor of GeoGebra refers in paper GeoGebra: Vom Autodesign zur Computerschriftart (2008) [16] and in the 8th chapter of the GeoGebra 4.4 Introductory Book [17] (2013), the segments we can see are tangents to a quadratic Bézier curve.

In the following applet [18] one can do a similar experiment by using GeoGebra, and eventually use its **Envelope** command to check whether the resulted contour curve is quadratic.


YouTube video at https://www.youtube.com/watch?v=V-Cq2VMsiZw
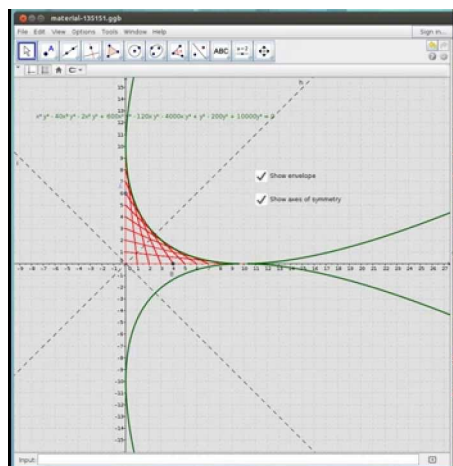
By no mean the activity to draw the segments AB while A is dragged on grid points between $(0,0)$ and $(0,10)$—and meanwhile B is moved between $(10,0)$ and $(0,0)$—is an easy task for many types of school pupils. Also the result as getting the contour of the segments can be expected to be a straightforward way of the next step of understanding. However, obtaining the envelope equation is at a different step of difficulty level.

First of all, the obtained equation is of 5th grade, containing not only the curve itself, but its reflection to the x-axis, and also the x-axis itself. This equation is $x^4y - 40x^3y - 2x^2y^3 + 600x^2y - 120xy^3 - 4000xy + y^5 - 200y^3 + 10000y = 0$ which is an implicit equation, but it can be factorized into three factors. Unfortunately, factorization is not discussed at secondary level, so we need to find another approach to go into the very details.

Fortunately, all these problems can be managed in secondary school by changing the construction in some sense. On one hand, we will rotate the axes by 45 degrees to obtain an explicit equation: in this case one of the parabolas can be written in form $y = ax^2 + bx + c$. On the other hand, we will use magnification of 10, so that we will obtain parabola $y = \frac{x^2}{2} + \frac{1}{2}$ which can be described with directrix $y = 0$ and focus $(0,1)$.
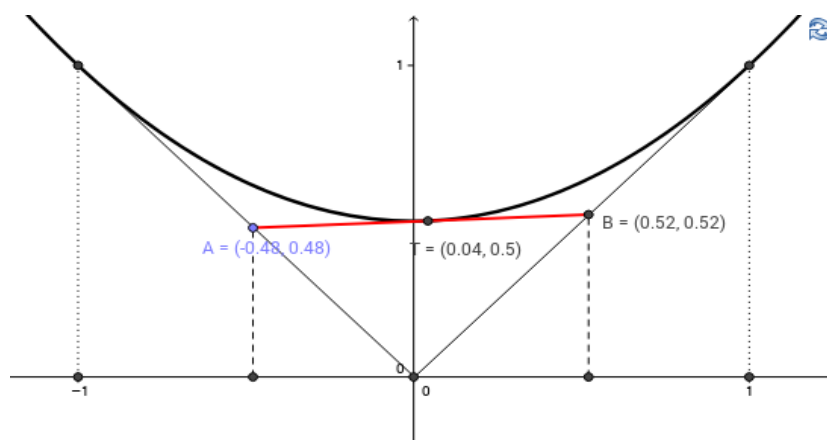
Now we prove that segment AB is always a tangent of the parabola described above. We will use only such methods which can be discussed in a secondary school as well. We would like to compute the equation of line AB to find the intersection point T of AB and the parabola.

So first we recognize that if point A=$(-d, d)$, then point B=$(1-d, 1-d)$. Since line AB has an equation in form $y = ax + b$, we can set up equations for points A and B as follows: $d = a \cdot (-d) + b$ (1) and $1 - d = a \cdot (1 - d) + b$ (2). Now (1) − (2) results in $a = 1 - 2d$ and thus, by using (1) again we get $b = 2d - 2d^2$.

Second, to obtain intersection point T we consider equation $ax + b = \frac{x^2}{2} + \frac{1}{2}$ which can be reformulated to search the roots of quadratic function $\frac{x^2}{2} - ax - b + \frac{1}{2}$. If and only if the discriminant of this quadratic expression is zero, then AB is a tangent. Indeed, the determinant is $(-a)^2 - 4 \cdot \frac{1}{2} \cdot (-b + \frac{1}{2}) = a^2 + 2b - 1$ which is, after expanding a and b, obviously zero.

Despite this is an analytical proof, by computing the x-coordinate of T (which is $a = 1 - 2d$) the Reader may think of finding a synthetic proof as well.

### 3.3 Mathematics in the coffee cup

The nephroid curve (see inside the coffee cup on Stuart Levy's photo [19] above) is a 6th order algebraic curve defined by the envelope of a set of mirrored light rays as a family of curves. Unfortunately, computationally it is rather complex to solve the corresponding equation system, thus it is inconvenient to use the **Envelope** command with the recent version of GeoGebra.

From the optical point of view, there are two approaches. One possibility is to assume that the source of the light is a point. In this case the rays will be concurrent. The other possibility is to assume that the source is infinitely distant, in this case the rays will be parallel. Clearly, the second case is the mathematical "limit" of the first one since if the point converges to infinity, the models will be closer and closer to each other.

The first approach is computationally easier. In the following figure we can investigate the model of the concurrent rays by using the Java desktop version of GeoGebra.
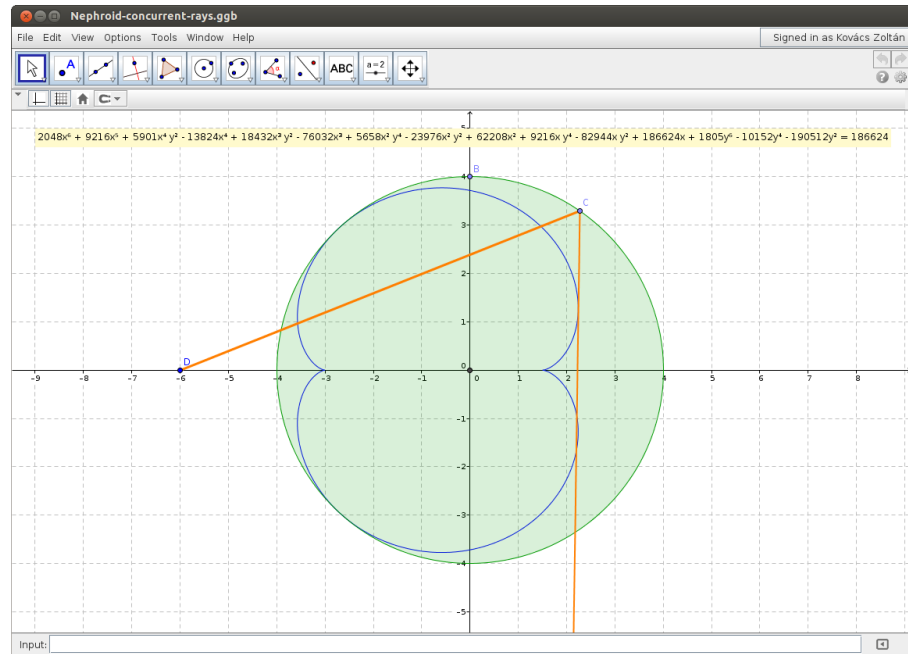
GeoGebra applets in GeoGebraBooks use Giac, but newest versions of GeoGebra can be configured to use faster methods than Giac has. In the following video we can learn how the Java desktop version can be started to use or not use the external computation machine SingularWS with the embedded Gröbner cover algorithm.
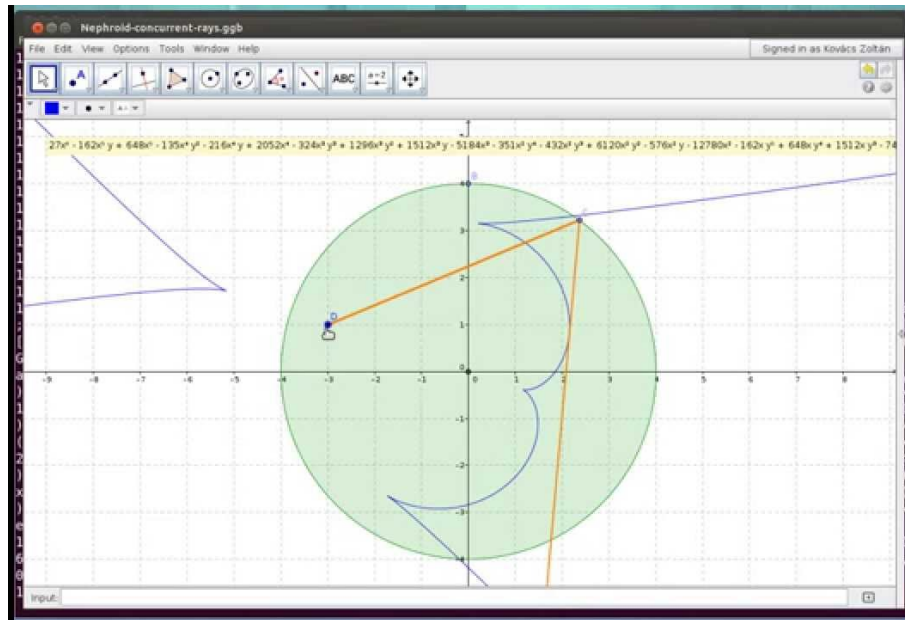
YouTube video at https://www.youtube.com/watch?v=nV_C4N7mWGs

Finally, the approach of the parallel rays is computationally the most difficult one. It is impossible for Giac to compute the envelope equation in a reasonable time, thus we have to force using SingularWS and the Gröbner cover method by using the command line:

YouTube video at `https://www.youtube.com/watch?v=-mGTaJR2zyw`
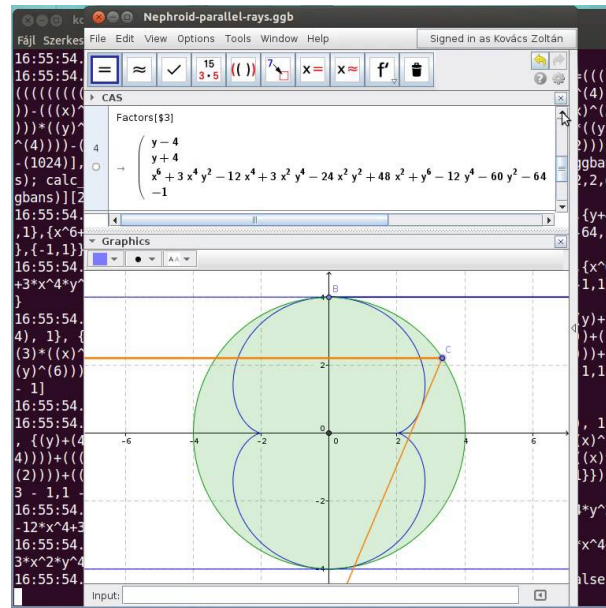
As we can see in the above videos, in the parallel case there are two extra components which can be separated by factorization. But also in the concurrent case when the source of the light is a perimeter point of the circle there is an extra component.

Conclusion. As Sander Wildeman [20] remarks, *once you have written an article about caustics you start to see them everywhere*. His illustration is mathematics on the bottom of an empty water bucket caused by a light bulb on the ceiling. We can only agree: mathematics is everywhere, not only in geometric forms of basic objects but various loci and envelopes. Mathematics is indeed everywhere—so maths teachers can build motivation on emphasizing these not well known facts in the modern era of education.

References

1. http://www.geogebra.org
2. http://webuser.hs-furtwangen.de/~dersch/jasymca2/indexEN.html
3. http://math.nist.gov/javanumerics/jama/
4. http://www.mathpiper.org/
5. http://www.reduce-algebra.com/
6. http://www-fourier.ujf-grenoble.fr/~parisse/giac.html
7. http://krum.rz.uni-mannheim.de/jas/
8. M. Á. Abánades, F. Botana, A. Montes, T. Recio. An algebraic taxonomy for locus computation in dynamic geometry. *Computer-Aided Design* (56), November 2014, p. 22–33 (to appear)
9. https://code.google.com/p/singularws/
10. F. Botana, Z. Kovács, S. Weitzhofer. Implementing theorem proving in GeoGebra by using a Singular webservice, *Proceedings EACA 2012. Libro de Resúmenes del XIII Encuentro de Álgebra Computacional y Aplicaciones*, p. 67–70. Univ. of Alcalá (Spain), 2012
10. http://mathworld.wolfram.com/RightStrophoid.html
11. B. Buchberger. *An Algorithm for Finding the Bases Elements of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal* (German). PhD thesis, Univ. of Innsbruck (Austria), 1965
12. http://dev.geogebra.org/trac/wiki
13. http://en.wikipedia.org/wiki/Witch_of_Agnesi
14. http://en.wikipedia.org/wiki/Intercept_theorem
15. http://en.wikipedia.org/wiki/String_art
16. M. Hohenwarter. GeoGebra: Vom Autodesign zur Computerschriftart (German). *Computeralgebra Rundbrief, Sonderheft zum Jahr der Mathematik, April 2008*, p. 9–10
17. www.geogebra.org/book/intro-en.pdf
18. http://geogebratube.org/student/m135151
19. http://www.geom.uiuc.edu/~fjw/calc-init/nephroid/
20. http://www.phikwadraat.nl/huygens_cusp_of_tea/