Math.Comput.Sci. (2019) 13:131–141
https://doi.org/10.1007/s11786-018-0390-0

Mathematics in Computer Science

Achievements and Challenges in Automatic Locus and Envelope Animations in Dynamic Geometry

Zoltán Kovács

Received: 18 November 2017 / Accepted: 20 May 2018 / Published online: 13 November 2018

Abstract A survey on the speed of real-time animation of loci and envelopes in the dynamic geometry software GeoGebra is presented.

Keywords Locus · Envelope · Computer graphics · Animation

Mathematics Subject Classification Primary 97G40; Secondary 14H50

1 Introduction

Dynamic geometry experiments on loci and envelopes may be fruitful activities in learning and research contexts. In particular, studying the behavior of certain geometry constructions when changing some initial parameters of them, may lead to new conjectures or to new kind of visualization of well-known theorems. Some recent papers on modeling planar linkages and proving some properties on their moving parts [15,17] show just one possible way to use locus animation in modern classrooms.

In the paper [14] we reported that non-trivial locus and envelope animations can be performed quickly enough by using today's computers with implicit computer algebra use. In the background, without notifying the user on the fact that thousands of symbolic computations are performed, a pure dynamic geometry experience can be relived, making geometry more geometrical than before.

Six examples were shown in [14] in details when benchmarking them in four different computer algebra systems. In this technical paper we continue investigating the computation speed in three systems, all used directly in the dynamic mathematics tool GeoGebra [12]. It is remarked that the obtained data is a result of continuous improvements of many "ingredients" of the underlying system, including GeoGebra, its embedded computer algebra system Giac, and the construction steps that describe the mathematical content in a faster way than in some earlier attempts. In this sense the focus is on measuring if the assembled parts of the investigated system is capable of supporting classroom and research use of animating locus and envelope curves.

Z. Kovács (✉)
The Private University College of Education of the Diocese of Linz, Salesianumweg 3, 4020 Linz, Austria
e-mail: zoltan@geogebra.org

Z. Kovács
Johannes Kepler University, Linz School of Education, Altenberger Straße 69, 4040 Linz, Austria

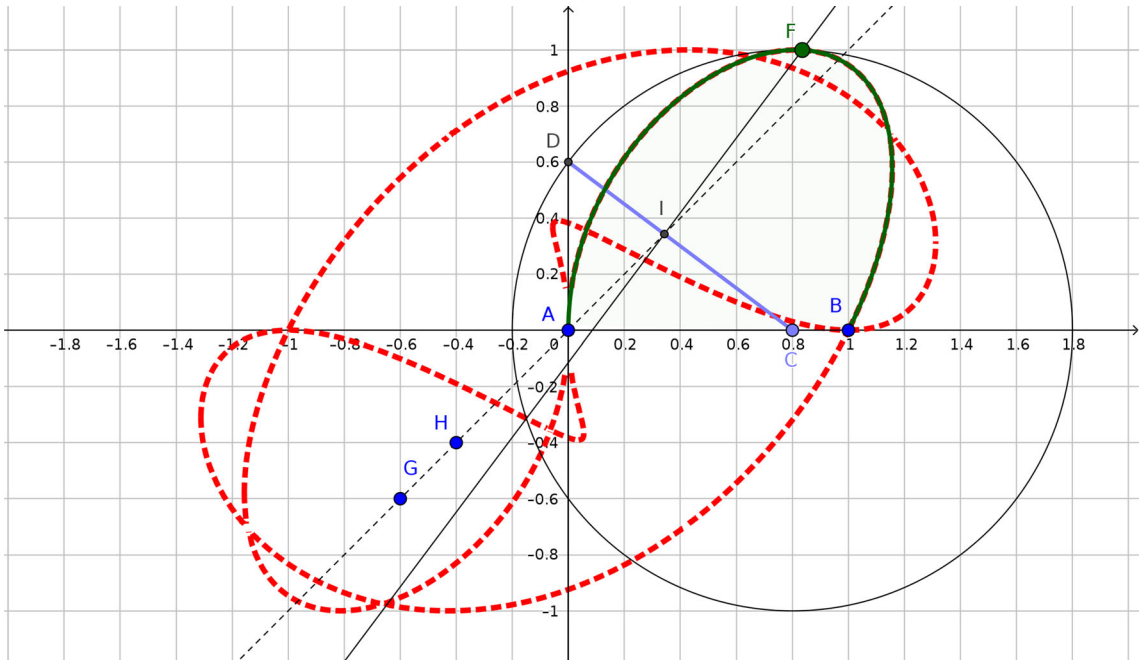

Fig. 1 An example of a beautiful algebraic curve

Quick computations may be beneficial in both a learning or a scientific context. When a learner or a researcher can obtain visual information of a locus being changed on the initial parameters (that is, on the given free points), the pieces of information can be compared quickly enough by experimenting. Conjectures can be found faster when the animation is quicker, because more special cases can be overviewed during the same time. In addition, the speed of animation can be classified into several categories—some of them are acceptable for the learning/researching process, but some are not.

Therefore, the speed of displaying or animating locus/envelope curves can be crucial when using a dynamic geometry tool for this purpose. Our report summarizes how GeoGebra's capabilities in this direction are evaluated. By studying the output of the benchmarking system we will confirm that the symbolic dynamic geometry approach is useful in investigating several non-trivial research problems, not only well-known classroom exercises.

Last, but not least we emphasize that by using this tool beautiful "new" algebraic curves can be created surprisingly quickly. In Fig. 1 a sliding ladder CD is leaning against the wall $x = 0$, $y \geq 0$, whilst $y = 0$, $x \geq 0$ is the base. The line GH is fixed. A circle with center C and radius CD is drawn. Let the point I be the intersection of CD and GH. Also, point F is created as any of the intersection points of the circle and a perpendicular to CD at point I. Here the continues thick curve shows the real locus of point F. By using fast elimination via Gröbner basis computations it turns out that this curve is a part of the dashed curve which is $4\,x^8 + 8\,y^8 + 16\,x^2\,y^6 + 8\,x^3\,y^5 + 12\,x^4\,y^4 + 16\,x^5\,y^3 + 8\,x^6\,y^2 + 8\,x^7\,y - 12\,x^6 - 8\,y^6 - 4\,x\,y^5 - 44\,x^2\,y^4 - 20\,x^3\,y^3 - 24\,x^4\,y^2 - 32\,x^5\,y + 12\,x^4 + y^4 + 32\,x^2\,y^2 + 24\,x^3\,y - 4\,x^2 = 0$, an irreducible algebraic curve that is the Zariski closure of the real locus.

On the structure of this paper: In Sect. 2 the benchmarking system is described. Section 3 depicts the items of the used database. Section 4 shows a test case in detail. In Sect. 5 some possible steps for the future work are sketched up.

2 The Benchmarking System

In the lifecycle of a new algorithm there is a step when it is integrated into a software tool as an experimental feature. Later on, the feature has to be foolproof to be applicable in a production environment. Schools are especially good examples for production environments where the implemented algorithms must work reliable.

In the paper [14] the new algorithms for the `LocusEquation` and `Envelope` commands in GeoGebra have been explained. We set up 152 test cases in order to test the reliability and speed of the discussed algorithms. The test cases are GeoGebra files: each file consists of a geometric construction and a final output which is a certain planar algebraic curve with integer coefficients. The benchmarking system, available at https://dev.geogebra.org/trac/browser/trunk/geogebra/test/scripts/benchmark/art-plotter, is written in Unix shell, connected with an SQLite database to collect the results of various test runs based on the actual codebase of GeoGebra. An automatic run is performed on a daily basis initiated by a local installation of the Jenkins continuous integration system at https://prover-test.geogebra.org since March 2017.

The benchmarking system runs each test on three different computer algebra environments. In two cases the Giac computer algebra system is used, once as a Java Native Interface, and once as a JavaScript system [20]. The latter method was tested by using the PhantomJS system [11], primarily created for headless website testing. The third case calls Singular [6] as its backend.

While the performing algorithm in the Giac tests are the same, for Singular a slightly different algorithm has been programmed, namely the one described in [3], based on Singular's `grobcov` library. It sometimes results in a more sophisticated output, but also requires more time to finish.

Classification of a run is somewhat subjective, mainly because there is no official classification standard of animation speeds. On one hand, 24 FPS is the normal framerate considered "good enough" in movies as it became standard for 35 mm sound film in the 1930s [22] and is still used frequently. Thomas Edison however recommended using at least 46 FPS, "anything less will strain the eye" [9]. On the other hand, cartoons have usually much less frames per second, e.g. 12 or even less [23]. Also, over a certain frame rate human eyes cannot even distinguish between them, although visual evidence can be seen at [1] that 15, 30 and 60 FPS result differently. Actually, [22] considers animations over 5 FPS as *motions*, so we also accept results above this value as satisfactory.

A test run is classified as follows.

1. "Fast" if the frame rate is at least 12 FPS.
2. "Moderate" between 5 and 12 FPS.
3. "Jerky" between 1 and 5 FPS.
4. "Heavy" below 1 FPS.
5. "Infeasible" if the computation has timed out, or was unsuccessful because of lack of resources. By default the timeout is 60 s.
6. "Incorrect" if the result differs from the expected value. In the database there can be multiple correct output results for the same input because some points can be considered also correct from different approaches.

Table 1 shows an output of a full test run—it is a hardcopy of the web page at https://prover-test.geogebra.org/job/GeoGebra-art-plottertest/268/artifact/test/scripts/benchmark/art-plotter/html/all.html. Here we refer to some colors that may be fully checked on the web page, namely, fast case runs are colored green, moderate runs are light blue, jerky runs are orange, heavy runs are red, infeasible runs are white and incorrect runs are black. The results for the `grobcov` library are sometimes incorrect because of some technical issues—they should be addressed in the future. For each run "deg" shows the degree of the output polynomial and FPS reports the measured speed.

All expected algebraic curves are stored with their coefficients in a textual database in the folder `tests`, file `expected_results`. Each entry has the following form like this example:

```
Agnesi-witch 3,3,-64,0,16,0,0,0,4,4,1
```

This means that the expected curve, which is defined in the file `tests/Agnesi-witch.ggb` in this particular example, is

$$\sum_{j=1}^{3}\sum_{k=1}^{3} A_{j,k}x^{j-1}y^{k-1} = 0$$

Table 1 An output of the benchmarking system after a full run on 2017-11-02, on an Intel(R) Core(TM) i7 CPU 860 @ 2.80 GHz machine, Ubuntu Linux 16.04.3, 64 bit

Test	DesktopInternal		DesktopGrobcov		Web	
	deg	FPS	deg	FPS	deg	FPS
Agnesi-witch	4	50.00	4	4.36	4	2.89
Arbeo-A01-conchoid-Nicomedes	4	33.33	4	3.09	4	3.13
Arbeo-A02	2	100.00	2	7.81	2	5.12
Arbeo-A03	2	52.63	2	5.10	2	3.73
Arbeo-A04-linkage-Watt	6	33.33	6	2.30	6	2.45
Arbeo-A05-limacon	6	35.71	4	1.46	6	1.95
Arbeo-A06-conchoid-Nicomedes	4	47.61	4	3.10	4	3.27
Arbeo-A07	2	47.61	2	1.90	2	3.13
Arbeo-A08	2	25.00	0	1.46	2	2.39
Arbeo-A09	4	27.02	4	0.64	4	1.71
Arbeo-A10	2	47.61	2	1.69	2	3.19
Arbeo-B02	3	45.45	3	2.06	3	3.05
Arbeo-B03	2	37.03	2	5.84	2	4.23
Arbeo-B04	2	55.55	2	4.69	2	3.48
Arbeo-B05	2	58.82	2	5.88	2	3.78
Arbeo-B06	2	71.42	2	6.75	2	4.40
Arbeo-B07	16	3.27		t/o		t/o
Arbeo-B08	8	33.33		t/o	8	1.81
Arbeo-B10	6	23.80		t/o	6	1.64
Arbeo-B11	1	62.50	1	4.62	1	4.04
Arbeo-B12	1	31.25	1	4.58	1	3.38
Arbeo-B13	1	83.33	1	7.51	1	5.05
Arbeo-B14	1	83.33	1	9.25	1	5.10
Arbeo-B15	1	71.42	1	9.17	1	4.85
Arbeo-B16-limacon	6	41.66	4	1.59	6	2.12
BKovacs-robot	16	19.60		t/o		t/o
BKovacs-robot-simplified	6	43.47	6	1.31	6	2.70
Blazek-example1	3	15.87	6	0.88	3	0.94
Blazek-example1-difficult		t/o				t/o
Blazek-example1-easy	2	0.28		t/o		t/o
Blazek-example1-integer	2	0.27		t/o		t/o
Blazek-example1-moderate		t/o				t/o
Botana-ladder-Lissajous-1	8	22.72		t/o	8	1.39
Botana-ladder-Lissajous-2	8	15.87		t/o	8	0.88
Botana-ladder-Lissajous-3	8	20.83	8	0.17	8	1.19
Botana-ladder-ellipses	4	29.41	4	0.93	4	1.96
Euler-ineq-2	18	0.28		t/o		t/o
Euler-ineq-2.01	18	0.16		t/o		t/o
Euler-ineq-2.1	18	0.22		t/o		t/o
Excalibur487	2	45.45	2	3.86	2	2.84
HartAFrame-difficult	20	0.22		t/o		t/o
HartAFrame-integer	15	1.53		t/o		t/o
HartAFrame-moderate	20	0.33		t/o		t/o
HartInversor1	7	12.34	0	0.22	7	0.83
HartInversor2	7	13.33	0	0.37	7	0.92
HartInversor3	7	14.28	0	0.35	7	0.85
Hasek-Apollonius-converse	3	22.72	2	1.05	3	1.79
Holfeld-24	4	45.45	4	1.85	4	3.18
Holfeld-24-absfact	4	47.61	4	2.23	4	3.19
Holfeld-35	6	20.83	6	0.49	6	1.66
ICME-square	2	29.41	2	4.29	2	1.57

Table 1 continued

Test	DesktopInternal		DesktopGrobcov		Web	
	deg	FPS	deg	FPS	deg	FPS
ICME-square-double	4	15.15	4	3.95	4	1.46
Lambert-2-parallels	2	47.61	2	5.07	2	3.32
Lambert-Thales	2	125.00	0	8.84	2	6.62
Lambert-angle-bisector	2	23.80	2	5.07	2	2.49
Lambert-bisector	1	125.00	0	10.98	1	5.37
Lambert-circle-1	2	76.92	0	8.47	2	6.32
Lambert-circle-2	2	62.50	0	7.93	2	4.67
Lambert-circle-3	2	76.92	0	8.62	2	6.09
Lambert-circle-4	2	100.00	0	8.00	2	4.56
Lambert-line	1	142.85	0	10.86	1	7.75
Lambert-parallel-1	1	76.92	0	9.61	1	6.17
Lambert-parallel-2	1	71.42	0	7.46	1	4.50
Oldenburg-Lego4094	6	35.71	6	1.27	6	2.08
Oldenburg-Lego4094-2	6	35.71	6	1.26	6	2.15
Oldenburg-Lego4094-simplified	6	40.00	6	1.26	6	2.49
Peaucellier	2	7.46		t/o	2	0.48
Peaucellier-cell	2	18.18	2	0.87	2	1.56
Peaucellier-integer	1	16.94	1	1.49	1	1.12
Pech-Blazek	2	23.80	2	3.19	2	1.04
Ptolemy-1	4	16.12	4	2.78	4	1.49
Ptolemy-2	2	13.33	2	3.93	2	1.58
Ptolemy-3	2	16.39	2	3.81	2	1.50
Ptolemy-4	2	9.61	2	3.74	2	1.49
Ptolemy-5	6	13.69	6	2.48	6	0.59
Ptolemy-6	2	17.54	2	3.80	2	1.45
Recio-competence1	5	6.36	0	1.57	5	0.43
Roberts-mechanism	12	14.70	0	0.37	12	0.94
Roberts-mechanism-puzzle	16	3.06	16	2.08		t/o
Roberts-mechanism-puzzle-zoom	16	2.40	16	2.11		t/o
Simson-Wallace	2	40.00	2	3.87	2	2.25
Steiner-Lehmus	12	1.53		t/o		t/o
Steiner-deltoid	5	2.45			5	0.57
Surynkova-kinematic-1	4	20.00	4	0.76	4	1.26
Surynkova-kinematic-1-integer	4	19.60	4	0.76	4	1.61
Surynkova-kinematic-2	12	1.19				t/o
Surynkova-kinematic-2-integer	12	1.47				t/o
Velez-rectangle-area-perimeter	8	22.72	8	2.40	8	1.71
Zanoli-conchoid-deSluze1	3	10.10	3	0.48	3	0.89
always	0	90.90			0	4.03
angle-bisector-discrete	4	28.57	4	5.05	4	1.34
astroid	6	3.96	6	0.80	6	2.64
cardioid	4	3.96	6	1.47	4	2.28
circle-limacon	4	55.55	4	3.63	4	3.17
circle-limacon-integer	2	62.50	0	4.40	2	3.40
diagonals-equal	2	71.42	0	7.93	2	4.50
diagonals-halve	2	66.66	2	7.46	2	5.23
ellipse-as-envelope	2	3.96	2	0.97	2	2.69
equilateral-triangle-sums	4	23.25	4	2.62	4	2.03
implicit-intro-1	2	43.47	2	5.64	2	2.77
implicit-intro-2	2	41.66	2	5.58	2	2.83
inscribed-angle-theorem	2	1.36	0	4.76		t/o
limacon	6	23.80	4	0.94	6	2.18
limacon-trisectrix	4	3.92	4	1.44	4	1.97
line-reflection-point	1	111.11	1	8.69	1	5.71
linkage-3-bar	6	32.25	6	1.70	6	2.28

Table 1 continued

Test	DesktopInternal		DesktopGrobcov		Web	
	deg	FPS	deg	FPS	deg	FPS
linkage-4-bar	12	9.34		t/o	12	0.60
linkage-Chebyshev	6	32.25	6	1.34	6	2.38
linkage-Chebyshev-lambda	6	24.39	6	1.21	6	1.72
linkage-Chebyshev-puzzle	6	20.83	6	1.35	6	1.45
linkage-Hoecken	12	6.84		t/o		t/o
linkage-Hoecken-zoom	12	3.84		t/o		t/o
linkage-Russell	3	38.46	3	3.33	3	2.73
midline-theorem-discrete	2	58.82	2	8.13	2	4.09
mirror-tangents-ellipse-hyperbola1	2	16.12	0	1.86	2	1.42
mirror-tangents-ellipse-hyperbola2	2	17.24	0	1.86	2	1.65
mirror-tangents-ellipse-hyperbola3	13	0.15		t/o		t/o
nephroid-concurrent	6	7.87	6	0.50		t/o
nephroid-parallel	8	4.95				t/o
never	0	100.00	0	20.83	0	4.42
orthic-isosceles	2	34.48	2	4.27	2	2.04
orthocenter-locus	2	35.71	2	5.74	2	3.86
parabola	2	62.50	2	5.81	2	3.59
parabola-dynamic-coords	2	55.55	2	5.88	2	3.49
perpendicular-angle-bisectors	1	5.95	0	2.02	1	0.29
point-minus-vector	1	71.42	1	6.06	1	4.65
point-plus-vector	1	71.42	1	6.25	1	4.76
product3	6	43.47	6	5.49	6	2.95
product4	8	34.48	8	5.23	8	2.23
product5	10	25.00	10	4.36	10	1.83
product6	12	19.60	12	3.58	12	1.39
product7	14	9.61	14	2.50	14	0.42
remove-contradictory-constraints	4	34.48	4	1.63	4	1.60
right-triangle-altitude-theorem	4	34.48	4	4.46	4	1.72
sawmill	1	32.25	1	2.75	1	2.73
sliding-ladder-anypoint	4	25.00	4	1.63	4	1.81
sliding-ladder-anypoint-rational	4	28.57	4	1.26	4	2.04
sliding-ladder-midpoint	2	50.00	2	4.25	2	3.53
string-art	5	3.35			5	1.90
string-art-simple	2	3.77	2	2.02	2	2.89
strophoid	4	43.47	3	2.53	4	3.35
variations01	2	111.11	0	8.77	2	4.52
variations02	1	58.82	1	8.77	1	3.83
variations03	2	37.03	2	6.99	2	3.54
variations04	1	90.90	0	10.75	1	6.71
variations05	2	41.66	2	7.19	2	3.74
variations06	1	35.71	1	7.87	1	2.99
variations07	2	27.77	2	6.89	2	2.80
variations08	2	43.47	2	6.45	2	3.03
variations09	4	52.63	4	6.49	4	2.59
variations10	2	5.71	0	7.46	2	0.16
variations11	6	47.61	6	6.94	6	2.81
Total (of 152)	**150**		**100**		**129**	

where

$$A = \begin{bmatrix} -64 & 0 & 16 \\ 0 & 0 & 0 \\ 4 & 4 & 1 \end{bmatrix},$$

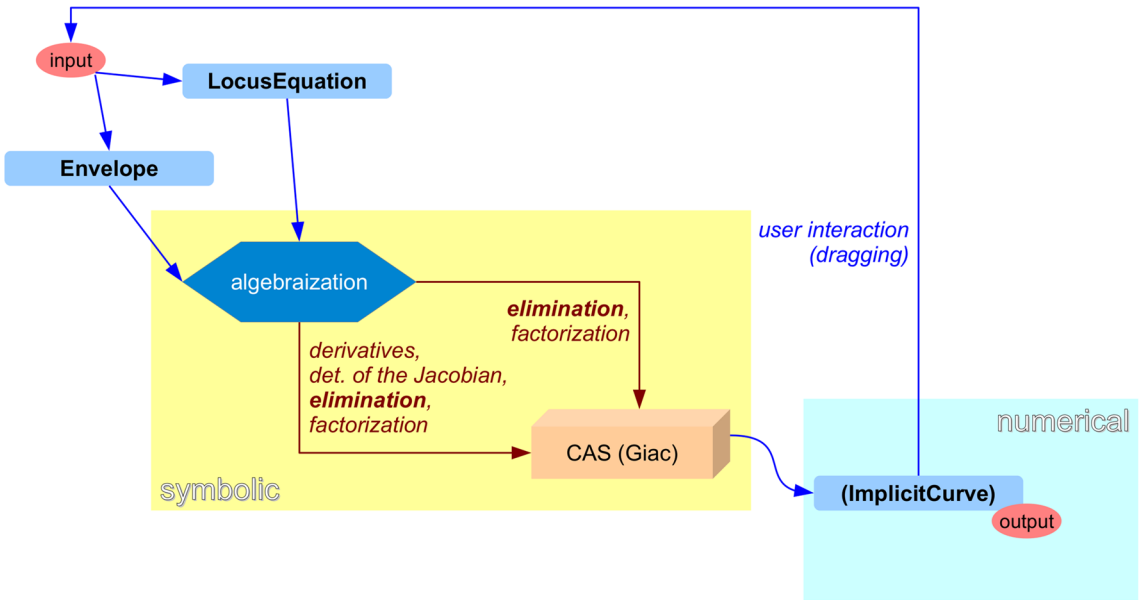

Fig. 2 GeoGebra's workflow of locus and envelope animations, taken from [14]

that is, $-64 + 16y^2 + 4x^2 + 4x^2y + x^2y^2 = 0$.

The results for the web runs are somewhat inaccurate, because the obtained time includes the first initialization of the computer algebra system as well. On the user's intervention (that is, on dragging the free points) the later computations will be somewhat faster, therefore these results show worse data than they are in reality.

The table was obtained with full automatization, that is, without any human modification of the measured data.

Finally we highlight that the benchmark contains only one part of the computation for one frame. Some other steps are also required to visualize the plot, and to maintain user interaction (see Fig. 2), but those steps are computationally much easier than eliminating variables from a polynomial ideal (see [14] for more details).

3 The Database

The test database being used consists of various kinds of tests. It focuses on covering former or present bugs and inaccuracies of the implemented algorithms. The first implementation of computing locus equations in GeoGebra goes back to 2010, that time, in the frame of the Google Summer of Code project Sergio Arbeo programmed the prototype. He created several GeoGebra files which were uploaded to his project blog https://dev.geogebra.org/trac/wiki/LocusLineEquation. One part of the benchmark database consists of Arbeo's examples (A01–A10 and B02–B16: 25 tests).

During the past years the system has been improved, extensively rewritten and tested by several researchers and end users. As a result, another part of the database, 11 tests were created by Noël Lambert, who was one of the primary testers of the implicit locus equation subsystem. 8 tests (named `linkage-...`) come from direct use of loci in GeoGebra's Automated Reasoning system [18], concentrating on the possible role of planar linkages in STEAM education—23 additional tests are also related to linkages [17]. 11 tests are a demonstration of the implicit locus subsystem for triangles in an elementary context (named `variations-...`, [13]). The other tests are different experiments, including well known results (like the trace of the midpoint of a sliding ladder) or completely new ones (like the Pech–Blažek theorem, [4]).

This also means that many of the tests were created or inspired by various people, including (alphabetically) Miguel Abánades, Sergio Arbeo, Jiří Blažek, Francisco Botana, Jesús Escribano, Roman Hašek, Markus Hohen-

Table 2 Comparison of FPS results on Blažek's first example if different input parameters are given

B	E	F	G	FPS
(0, 2)	(2, 0)	(2, 4)	(4, 2)	0.28
(−3, 2)	(2, −3)	(7, 2)	(7, 2)	0.27
(−3, 2)	(2, −3)	(7, 2)	(2.75, 6.94)	0.03
(**−3.25**, **1.96**)	(**2**, **−3.25**)	(**2.56**, **7.22**)	(**7.25**, **2**)	**t/o**

warter, Damjan Kobal, Benedek Kovács, Noël Lambert, Antonio Montes, Reinhard Oldenburg, Pavel Pech, Tomás Recio, Chris Sangwin, Csilla Sólyom-Gecse, Petra Surynková, Róbert Vajda, M. Pilar Vélez, Florian Wakolbinger, Jan Zahradník and Carla Zanoli. See [2,4,5,7,10,19,21] for some more references.

An immediate remark that the database has entries of different difficulty levels from the computational point of view. Some tests (e.g. `Arbeo-B13`, `Arbeo-B14` and `Arbeo-B15`) are reasonably fast on all computer algebra systems, others are fast on the DesktopInternal backend but infeasible on the other two. The DesktopInternal backend is seemingly the fastest one, and just a few tests are reported to be slow on it. The most difficult one is Blažek's first example [4] which cannot be computed in 60 s if the input parameters are not integers.

It is remarkable that the output may heavily depend on the input parameters. Table 2 shows how the FPS value changes when the coordinates of the free points are changed.

This behavior can be observed in other tests, too. For example, when checking Euler's inequality [19], the case $R = 2r$ results in 0.28 FPS, the $R = 2.1r$ case gives 0.22 FPS, and the $R = 2.01r$ case reaches only 0.16 FPS, when $A = (0, 0)$, $B = (1, 0)$. For $R = 2r$ and a randomly chosen A only 0.01 FPS can be achieved.

As a conclusion, dragging is not recommended on these "heavy" examples, because the user can quickly get into a situation that the software tool is keeping computing the next frame without success, and the program seems to be "freezing". For a next step it is suggested to improve the software to avoid such situations.

Finally we note that the degree of the output polynomials are between 1 and 20. It is not uncommon that the coefficients of the polynomials (that is, the elements of matrix A) are "big" integers. For example, in the test case `Blazek-example1` the top-left constant is

1303228239500621445754 8753

(and the other coefficients are of similar magnitudes). The example with the biggest integers is `HartAFrame-difficult` which is stored with more than 30,000 decimal digits (this means about 70 digits as an average for each coefficient).

4 An Example

As a real life example we refer to the nephroid curve as caustic of a circle (Fig. 3). This curve can be seen, among others, in a coffee cup or a mopping bucket every day. By using GeoGebra's desktop version one can animate the curve in real time, depending on different positions of the source of the light. The name of the example is `nephroid-concurrent` which provides a motion of 7.87 FPS. (The Gröbner cover algorithm can also compute the envelope equation, but it is definitely slower by reaching only 0.50 FPS.)

Figure 3 can be obtained by performing the following ideas and steps. The point D is the source of the light. The light ray collides with the perimeter of the circle in point C. The mirrored ray continues its way further in another direction. When D is fixed, the envelope of the mirrored rays is a nephroid curve. While D is dragged from an external source into an internal point of the circle, several curves will be drawn as different frames of an animation. The final one here is $45095\,x^6 + 36995\,y^6 - 10800\,x\,y^5 + 119085\,x^2\,y^4 - 21600\,x^3\,y^3 + 127185\,x^4\,y^2 - 10800\,x^5\,y + 43320\,x^5 + 86640\,y^5 + 43320\,x\,y^4 + 173280\,x^2\,y^3 + 86640\,x^3\,y^2 + 86640\,x^4\,y + 35580\,x^4 + 93540\,y^4 + 77280\,x\,y^3 + 129120\,x^2\,y^2 + 77280\,x^3\,y + 20800\,x^3 + 56960\,y^3 + 48960\,x\,y^2 + 51840\,x^2\,y +$

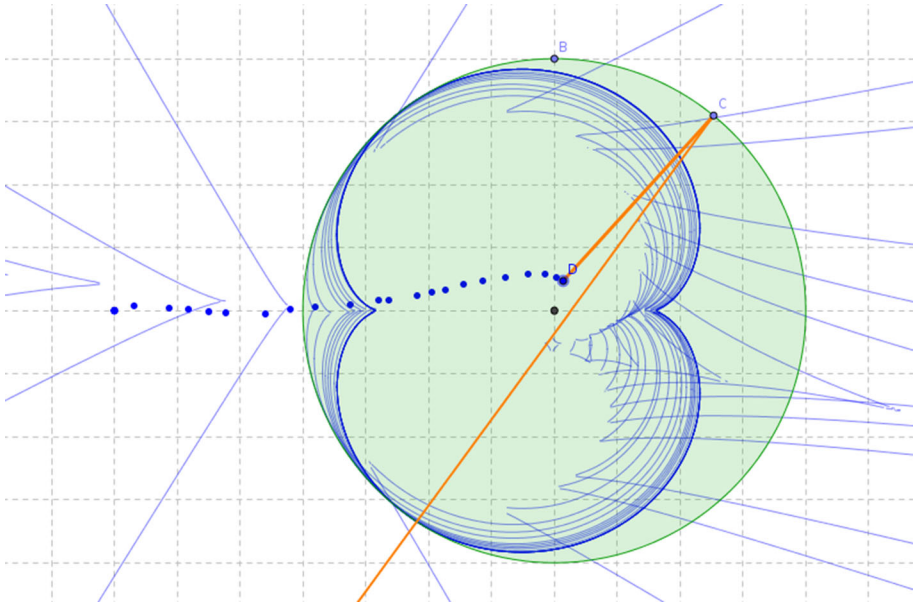

Fig. 3 Caustics of a circle

$8832\ x^2 + 20028\ y^2 + 14928\ x\ y + 1920\ x + 3840\ y + 320 = 0$ which depends on inputs $B = (0, 4)$ and $D = (0.2, 0.4)$.

By experimenting with the dynamic applet one can realize that the envelope equation is usually a sextic, but in some special cases it falls back to a quartic, as it is the case for the point $D = (-4, 0)$. Namely, the computed curve is $(x + 4)\ \left(27\ x^4 + 54\ x^2\ y^2 - 288\ x^2 + 512\ x + 27\ y^4 - 288\ y^2 + 256\right) = 0$, which means that the envelope here is a union of a line (the tangent to the circle at D) and a quartic. Also in the special case $D = (0, 0)$ the output collapses to $x^2 + y^2 = 0$.

By obtaining several curves with their algebraic equations in a short response time the user can make a conjecture for the general case. For the caustics it may be difficult to actually *prove* this conjecture rigorously—the relevance of the presented approach is to obtain a valid *conjecture*.

Let us remark that in this example a *CAS step* was required to better understand the result, namely, *factorization*. There are other examples where the locus or envelope equation is a product of various polynomials, and sometimes the geometric content is a subset of the factors. This is also the case in the tests `HartAFrame-...` and `HartInversor...` that describe linear motions in some planar linkages, and those linear motions can be observed only through one of the factors.

5 Conclusion and Future Work

We presented a survey of GeoGebra's abilities in animating locus and envelope curves. It turned out that the desktop version of GeoGebra is capable of visualizing 150 test cases out of 152, while its web version is able to do that in 129 cases. The alternatively available method, via Singular's `grobcov` library was successful in 100 cases.

On one hand, these capabilities could be successfully used in teaching introductory or more advanced planar Euclidean geometry in classrooms. Even if the output of a computation is a simple geometric object, namely a line or a circle, the fact that such a result must be the output, may be non-trivial for the student, and, from the computational point of view, it may require heavy computations. For example, constructions containing angle bisectors (see test case `Lambert-anglebisector`), or some special setup of Ptolemy's theorem (see test case `Ptolemy-4`) are surprisingly difficult to compute. We also highlight that easy questions can lead to difficult geometry. For instance,

in [13] simple formulas may result in quartic curves, or a detailed study [19] on Euler's inequality leads to two irreducible octic polynomials (see test case `Euler-ineq-2`).

On the other hand, such experiments may be interesting in research to obtain new kind of curves with interesting properties.

Clearly, there is still room for improving the underlying algorithms, particularly in the web version, by speeding up computations, and avoid that the software is not responding because of the heavy or infeasible amount of remaining operations.

Futher speedups can be achieved via using various techniques:

- one possibility is to harness `asm.js` [24] support in modern browsers (this feature, as of October 2017, is already working in Microsoft Edge, but it might be solved for Firefox as well),
- the other one is running native code as it is already implemented in GeoGebra Classic 6 via Electron/Node builds (as of October 2017), see [16] for a simple application based on Giac.
- Finally, another promising possibility is to use WebAssembly [8] (but this kind of technology is not yet available in all browsers).

We lastly remark that obtaining the exact identification of the locus by using the `grobcov` library should be solved as a future work, including the test cases which currently do not work yet.

Acknowledgements Open access funding provided by Johannes Kepler University Linz.

References

1. Allen, B.: 15 FPS vs. 30 FPS vs. 60 FPS (HTML5). http://boallen.com/fps-compare-html5.html (2014)
2. Abánades, M.Á., Botana, F., Kovács, Z., Recio, T., Sólyom-Gecse, C.: Development of automatic reasoning tools in geogebra. ACM Commun. Comput. Algebra **50**, 85–88 (2016)
3. Abánades, M.Á., Botana, F., Montes, A., Recio, T.: An algebraic taxonomy for locus computation in dynamic geometry. Comput.-Aided Des. **56**, 22–33 (2014)
4. Blažek, J., Pech, P.: Geometric problems and benefits of DGS. http://mintlinz.pbworks.com/w/file/fetch/118189569/Blazek%20-%20Pech.pdf
5. Botana, F., Kovács, Z.: Teaching loci and envelopes in GeoGebra. GeoGebraBook. http://tube.geogebra.org/m/R8nUbEQV (2014)
6. Botana, F., Kovács, Z.: A singular web service for geometric computations. Ann. Math. Artif. Intell. **74**(3), 359–370 (2015)
7. Botana, F., Kovács, Z.: New tools in GeoGebra offering novel opportunities to teach loci and envelopes. arXiv:1605.09153 [cs.CG] (2016)
8. Bright, P.: The web is getting its bytecode: webassembly. Ars Technica Condé Nast (2015)
9. Elsaesser, T.: Early cinema: space, frame, narrative. BFI Publishing, London, p. 284, ISBN 0-85170-244-9 (1990)
10. Hašek, R., Kovács, Z., Zahradník, J.: Contemporary interpretation of a historical locus problem with the use of computer algebra. In: Kotsireas, I., Martínez-Moro, E. (eds.) Applications of Computer Algebra. ACA 2015. Springer Proceedings in Mathematics & Statistics, vol. 198, pp. 191–2015. Springer, Cham (2017)
11. Hidayat, A.: PhantomJS. http://phantomjs.org (2017)
12. Hohenwarter, M.: GeoGebra: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene. Master's thesis, Paris Lodron University, Salzburg, Austria (2002)
13. Kovács, Z.: Variations on implicit loci in a triangle. A GeoGebraBook. https://www.geogebra.org/m/gH3b2Vwa (2016)
14. Kovács, Z.: Real-time animated dynamic geometry in the classrooms by using fast Gröbner basis computations. Math. Comput. Sci. http://link.springer.com/article/10.1007/s11786-017-0308-2 (2017)
15. Kovács, Z., Kovács, B.: A compilation of LEGO Technic parts to support learning experiments on linkages. arXiv:1712.00440 [math.HO] (2018)
16. Kovács, Z.: Giac electron example. https://github.com/kovzol/giac-electron-example (2017)
17. Kovács, Z., Recio, T., Vélez, M.P.: Reasoning on linkages. A GeoGebraBook. https://www.geogebra.org/m/sDCTVGrg (2017)

18. Kovács, Z., Recio, T., Vélez, M.P.: GeoGebra automated reasoning tools. A tutorial. A GitHub Project. https://github.com/kovzol/gg-art-doc (2017)
19. Kovács, Z., Vajda, R.: A note about Euler's inequality and automated reasoning with dynamic geometry. arXiv:1708.02993v2 (2017)
20. Kovács, Z., Parisse, B.: Giac and GeoGebra–improved Gröbner basis computations. In: Computer Algebra and Polynomials, Vol. 8942 of the series Lecture Notes in Computer Science, pp. 126–138 (2015)
21. Oldenburg, R.: FeliX—mit Algebra Geometrie machen. In: Computeralgebra Rundbrief, Sonderheft zum Jahr der Mathematik. http://www.fachgruppe-computeralgebra.de/data/JdM-2008/Sonderheft.pdf (2008)
22. Read, P., Meyer, M.-P.: Restoration of motion picture film. Conservation and Museology. Butterworth-Heinemann, Oxford, pp. 24–26. ISBN 0-7506-2793-X (2000)
23. Wikipedia Contributors, *Frame rate*, Wikipedia, The Free Encyclopedia. Accessed 25 November 2016 (2016)
24. Zakai, A.: Asm.js. http://asmjs.org (2017)