CADGME 2009 - Session DEIM

Integrating DG and CAS abilities under a common framework.

Jesús Escribano (U. Complutense, Madrid),
Franciso Botana (U. Vigo),
Miguel Abánades (CES Felipe II, Aranjuez).

Connexion between DGS and CAS.

CAS = Computer Algebra System

- Computer application with capacities on symbolic, numeric and graphical computations.
- Examples: Mathematica, Maple, SAGE.

DGS = Dynamic Geometry System

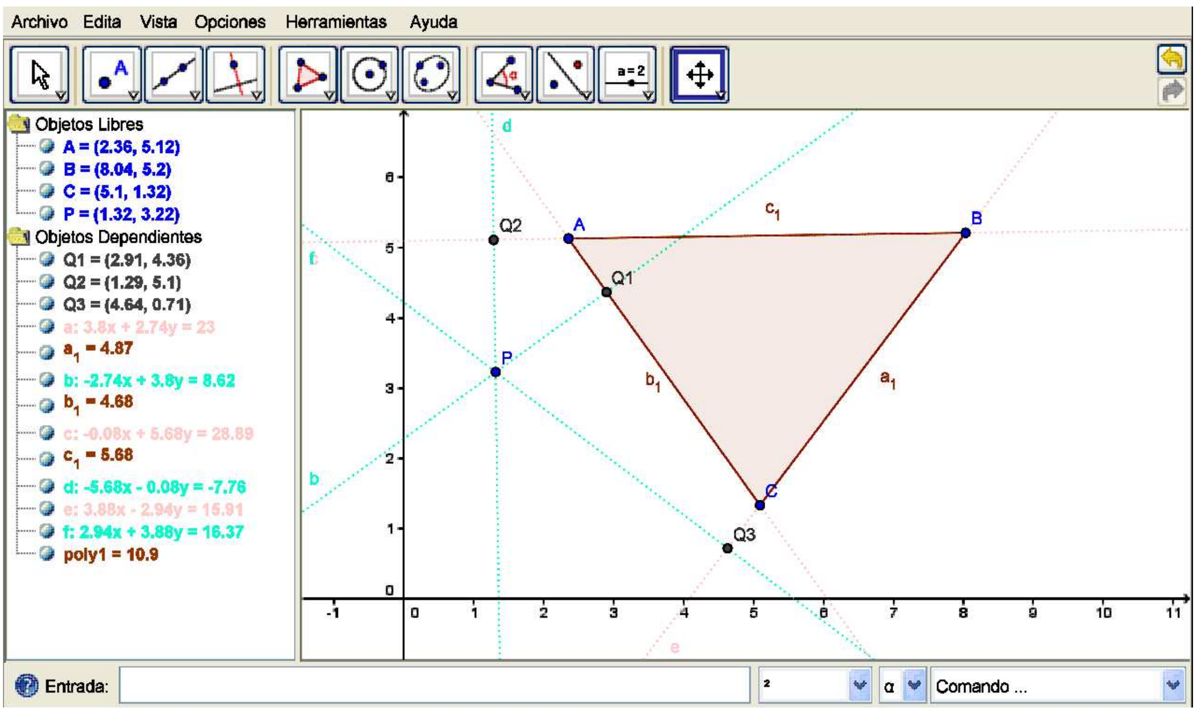
- Computer program that allows to draw a geometric construction, and to manipulate it in a **interactive** way.
- When you move the elements, all the construction moves dinamically.
- Examples: Cabri (Cabri II Plus), Cinderella, Geometer's Sketchpad, GeoGebra

Why we need to connect DGS and CAS?

- Search for complete and correct results, from the algebraic point of view (equations, graphics)
- Solving problems that can be easily described with a DGS, but whose solutions can not be represented (generic locus).

Example/question

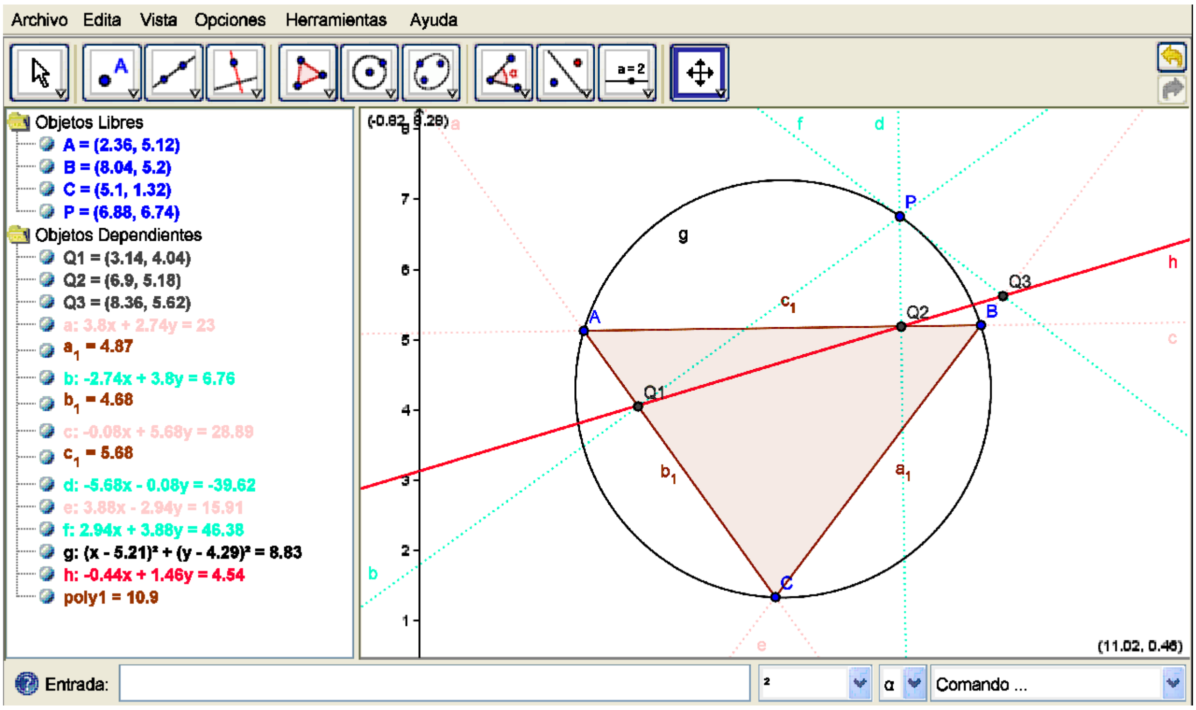
Consider the triangle ABC and a point P.
Let Q1,Q2,Q3 be the orthogonal projections of P onto the sides of ABC.
Are Q1, Q2 and Q3 aligned?



NOT in general.
What is the locus set of points P such that Q1, Q2 and Q3 are aligned?

SOLUTION

```
%hide
html('<applet code="geogebra.GeoGebraApplet" archive="http://www.geogebra.org/webstart/geogebra.jar"
width=850 height=500  MAYSCRIPT><param name="showMenuBar" value="true"/><param name="showAlgebraInput"
value="true"/><param name="showToolBar" value="true"/><param name="filename"
value="http://kimba.mat.ucm.es/~jesus/lugarTriangulosol.ggb"/><param name="framePossible"
value="false"/></applet>')
```



We describe two cases of successful integration of DG resources and CAS abilities through SAGE.

Why SAGE?

SAGE (***Software for Algebra and Geometry Experimentation***), is a **free** mathematical system, that combines several open source mathematical packages, with a common ***interface*** based in Python.

- **SAGE** is a free open source system developed as a viable alternative to expensive and opaque systems such as Mathematica, Maple, ...
- The **Notebook** interface of SAGE allows user to experiment the mathematical features, just using Internet, without installing anything to the personal computer.
- Main author: William Stein (University of Washington) + 150 active developers.
- First version: Feb. 2006.

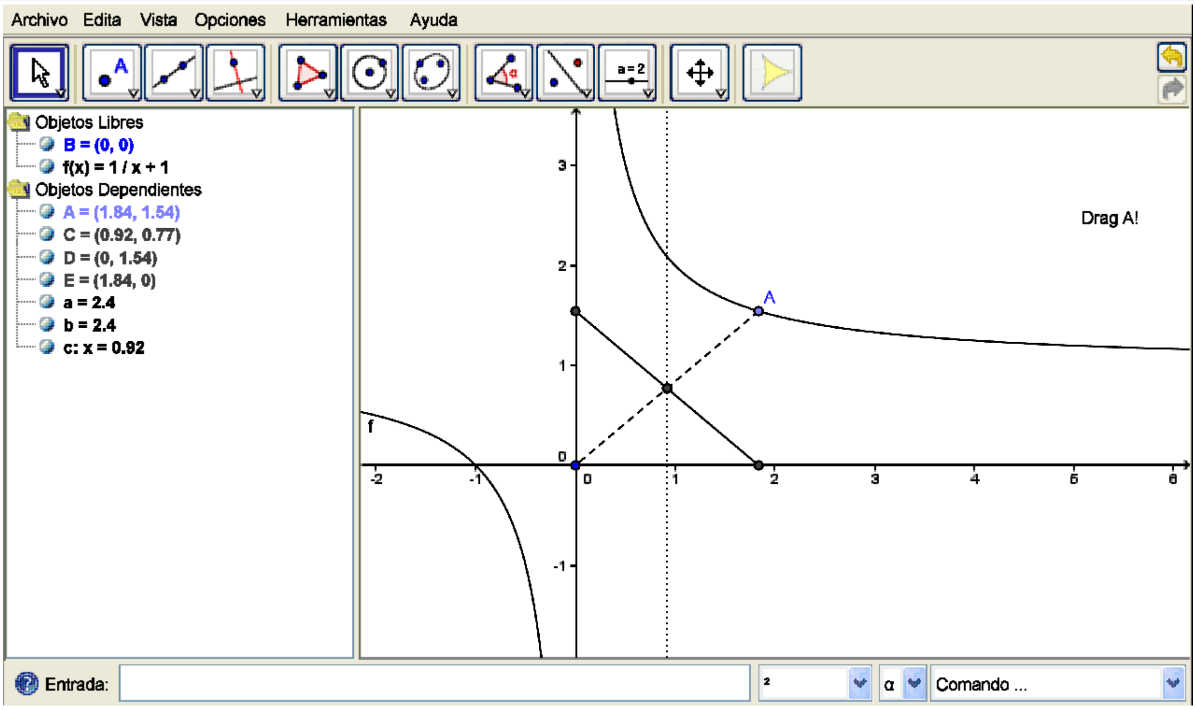
LINK: http://www.sagemath.org/

Remote use: www.sagenb.org

Generalized trammel of Archimedes

- Standard trammel: ellipse traced by a point on a sliding ladder.
- Generalized trammel: ladder of variable length.
- **Construction**
  1. enter function $y = g(x)$ determining length of ladder
  2. take a point on graph of g(x)
  3. with the **tool**, select the graph, the origin and the point on the graph.
  4. Setting the trace on for the ladder its envelope can be visualize.

```
%hide
html('<applet code="geogebra.GeoGebraApplet" archive="http://www.geogebra.org/webstart/geogebra.jar"
width=850 height=500  MAYSCRIPT><param name="showMenuBar" value="true"/><param name="showAlgebraInput"
value="true"/><param name="showToolBar" value="true"/><param name="filename"
value="http://webs.uvigo.es/fbotana/FlexibleLadder.ggb"/><param name="framePossible" value="false"/>
</applet>')
```



We use **Singular** to compute the **equation** of the ladder's envelope.

- $R = \mathbb{Q}[x, y, u, v]$, **polynomial ring** over the rational numbers with variables x, y, u, v:

```
R = singular.ring(0, '(u,v,x,y)', 'dp')
```

- Polynomial for the **floor** where the ladder is sitting (always the same):

```
f=singular("v*x+u*y-u*v")
```

- Polynomial for function g in terms of u and v (changing x by u and y by v).
- Example: if $y = 1/x$ we enter v*u-1; if $y = x^2/3$ entonces 3v-u^2,...

```
g=singular("(v-2)*u -1")
```

- Third polynomial necessary for computation:

```
h=singular(singular.diff(f,'u')*singular.diff(g,'v')-singular.diff(f,'v')*singular.diff(g,'u'))
```

- **Ideal** generated by the three polynomials:

```
I=singular.ideal(f,g,h);I
```

```
-u*v+v*x+u*y,
u*v-2*u-1,
-v*x+u*y-2*u+2*x
```

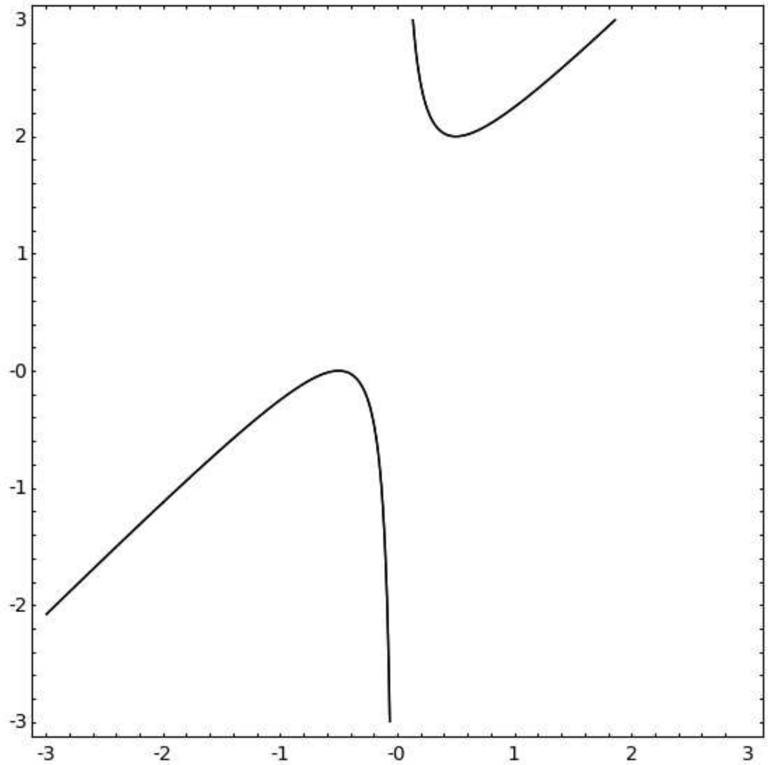
- **Equation** of envelope (eliminating variables u and v):

```
t=singular.eliminate(I,'uv')
if t.sage_polystring()=="0":
   upol=singular("u")
   vpol=singular("v")
   J=singular.ideal(upol,vpol)
   H=singular.sat(I,J)
   t=singular.eliminate(H[1],'uv')
t
```

```
4*x^2-4*x*y+4*x+1
```

- **Graph** of the envelope:

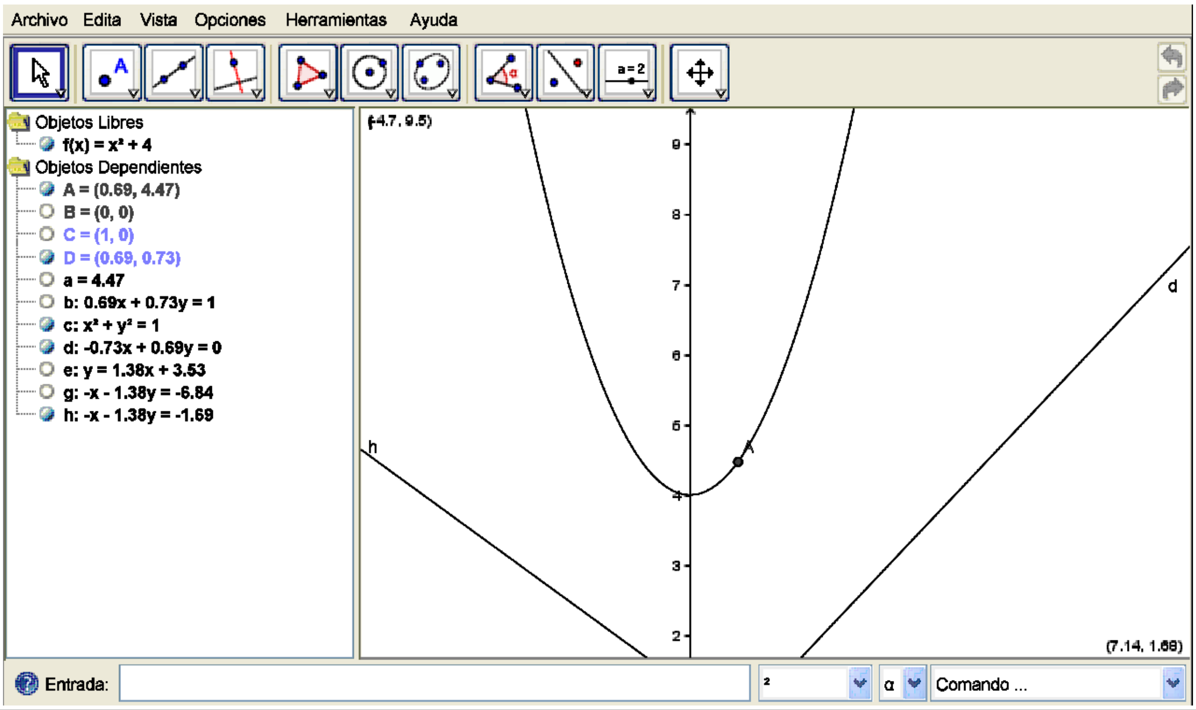
```
var('x,y')
graph=implicit_plot(eval(t.sage_polystring()),(x,-3,3),(y,-3,3),plot_points=200)
graph.show(aspect_ratio=1)
```



Lagrange Multipliers.

Find the maximum and minimum temperatures on the circle $x^2 + y^2 = 1$ if the temperature of a point (x, y) is $T(x, y) = 4 + x^2 - y$.

```
%hide
html('<applet code="geogebra.GeoGebraApplet" archive="http://www.geogebra.org/webstart/geogebra.jar"
width=850 height=500  MAYSCRIPT><param name="showMenuBar" value="true"/><param name="showAlgebraInput"
value="true"/><param name="showToolBar" value="true"/><param name="filename"
value="http://webs.uvigo.es/fbotana/grad1.ggb"/><param name="framePossible" value="false"/>
</applet>')
```




```
var('x,y')
T(x,y)=4+x^2-y
g(x,y)=x^2+y^2-1
```

Calculate the Lagrangian $L(x,y) = T(x,y) - \lambda * g(x,y)$. We'll use "lam" for λ because we can't use the word "lambda" in python/Sage (it means something different than what we want).

```
var('lam')
L(x,y,lam)=T(x,y)-lam*g(x,y)
```

```
gradL=L.gradient()(x,y,lam)
gradL
```

```
(-2*lam*x + 2*x, -2*lam*y - 1, -x^2 - y^2 + 1)
```

We get four solutions to the three equations given by $\nabla L = \vec{0}$.

```
solve([gradL[0]==0,gradL[1]==0,gradL[2]==0], [x,y,lam])
```

```
[[x == 0, y == -1, lam == (1/2)], [x == 0, y == 1, lam == (-1/2)],
[x == 1/2*sqrt(3), y == (-1/2), lam == 1], [x == -1/2*sqrt(3), y ==
(-1/2), lam == 1]]
```

We can reason through these solutions from the three equations $2x - 2\lambda x = 0$, $-2\lambda y - 1 = 0$, and $-y^2 - x^2 + 1 = 0$ as follows:

From the first equation, $2\lambda x = 2x$, we see that $2\lambda x - 2x = 0$, or $2x(\lambda - 1) = 0$. This means that either $x = 0$ or $\lambda = 1$.

- Case $\lambda = 1$: Then the second equation gives us $-2y - 1 = 0$, so $y = 1/2$. When $y = -1/2$, then the third equation gives us $x^2 + 1/4 = 1$, or $x = \pm\sqrt{3/4}$. So we get two points: $(\sqrt{3/4}, -1/2)$ and $(-\sqrt{3/4}, -1/2)$.
- Case $x = 0$: Then the third equation gives us $y^2 = 1$, or $y = \pm 1$. Both of these solutions also make sense in the second equation. Thus we get two more points, $(0, 1)$ and $(0, -1)$.

Now to find the maximum and minimum, we plug each of the four points into our temperature function.

```
T(0,1)
```

```
3
```

```
T(0,-1)
```

```
5
```

```
T(sqrt(3/4), -1/2)
```

```
21/4
```

```
T(-sqrt(3/4), -1/2)
```

```
21/4
```

The maximum is therefore at both $(\sqrt{3/4}, -1/2)$ and $(-\sqrt{3/4}, -1/2)$ (max value $21/4$) and the minimum is at $(0, 1)$ (min value 3).

Here is what the circle looks like, if we imagine the z-value representing the temperature. To get this graph, I've parameterized the boundary like we can do on the problems in the 2nd derivative section.

```
var('t')
parametric_plot3d([cos(t), sin(t), T(cos(t), sin(t))], (t, 0, 2*pi))
```