Web Course on Computer Aided Geometric Design

L. Fernández-Jambrina and R. López-Pulido

Abstract. In this contribution we show a web-oriented course for learning Computer Aided Geometric Design at university level. The topics that are covered are Bézier, rational and spline curves and surfaces. The contents and notation are mainly grounded on reference [1].

To create this course an application package for Maple has been written, which allows symbolic calculation of most common procedures in CAGD. This has enabled us to enclose animated graphics in our course. Interactive surface graphics have also been included by using JavaviewLib (http://www.javaview.de). Finally interactive applets in Java can also be found in the book. They depict simple examples of spline, rational and Bezier curves. They have been programmed with the interactive geometry program Cinderella (http://www.cinderella.de). More details about the course can be found at http://debin.etsin.upm.es/~leonardo.

§1. Introduction

Computer-Aided Geometric Design (CAGD) is the basis for modern design in most branches of industry, from naval and aeronautic to textile industry. Therefore this subject should be included in most Schools and Faculties of Engineering. In principle, students should just get familiar with some specific design applications, such as Rhino3D, Maxsurf, Foran,... but a thorough knowledge of their capabilities comes from learning at least the algorithms that lie behind the application, even if the students are not to become developers themselves. This is the purpose of CAGD, the geometric description of algorithms for design.

Nowadays industrial design is mainly grounded on Non-Uniform Rational B-Splines (NURBS). These do not require great mathematical developments, just knowledge of polynomial and rational functions and a bit of

xxx and xxx (eds.), pp. 1–4.

ISBN 0-9728482-x-x

differential geometry of curves and surfaces. Therefore a course of CAGD could be included in the first semesters of most engineering studies.

One of the major drawbacks of a course on CAGD is that most design applications are too complicated for a basic course, since they usually include more possibilities than can be covered in it. It would be advisable then either to provide simple tools that implement the basic algorithms of CAGD or to allow the students to write their own codes in a programming language familiar to them (C, Java, Matlab, Maple...).

In this contribution we aim to summarize the technical contents of a CAGD course that has been developed by the authors as an optional subject for all sort of students at the Universidad Politécnica de Madrid (U.P.M.). Considering that the U.P.M. campus is not concentrated, but its premises are spread over a vast region of the city and its outskirts, e-learning is surely the best choice for such a heterogeneous group of students and schools. Lessons were not given in any classroom, but the students could follow the course from a website of the university (http://www.gate.upm.es), which is specifically devoted to undergraduate and postgraduate courses in the web for students of Spain and the Americas. The contents of the course in English and Spanish may be checked at http://debin.etsin.upm.es/~leonardo.

§2. Applications

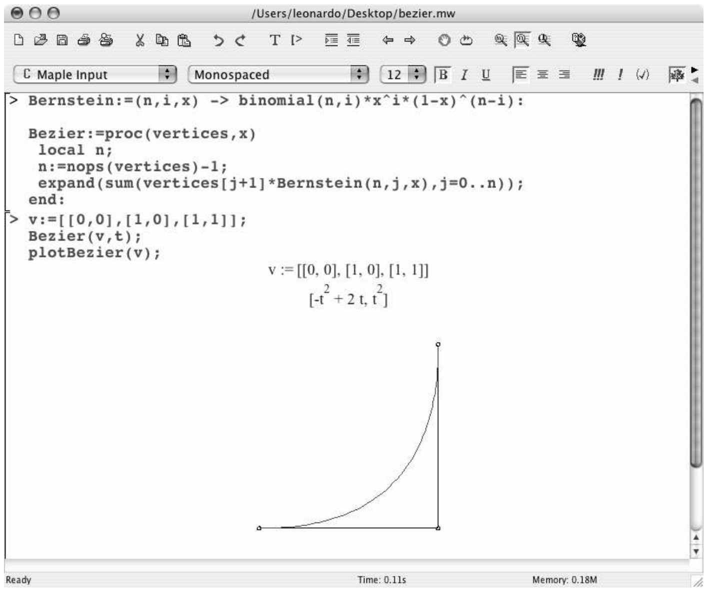
2.1. HTML

Since the course is intended for the web, it is written mainly in HTML code, although we have provided some notes for the chapter as PDF files. Obviously raw mathematical stuff is quite involved to be read in a web page and therefore proofs and details have been removed from the HTML files and are included just in the PDF notes, which can be read as an independent book [2].

2.2. Maple

In order to generate graphics and animations we have made use of Maple programming language (http://www.maplesoft.com/). It may be argued that other applications could suit better our purpose. But we have chosen it for its versatility for producing plots and movies easily in almost every common format (bmp, gif, jpeg, postscript...), for its duality in working with both analytical and numerical data, and, most important, for being the most popular Symbolic Calculus application at Spanish universities, since almost every university has a campus licence for it.

For several months we have built a Maple library of NURBS that comprises the most popular algorithms for curves and surfaces as can be found, for instance, in [1]. This library is not fully operational by now and therefore has not been offered to our students so far, but it



```
> Bernstein:=(n,i,x) -> binomial(n,i)*x^i*(1-x)^(n-i):

  Bezier:=proc(vertices,x)
   local n;
   n:=nops(vertices)-1;
   expand(sum(vertices[j+1]*Bernstein(n,j,x),j=0..n));
  end:
> v:=[[0,0],[1,0],[1,1]];
  Bezier(v,t);
  plotBezier(v);
```

$$v := [[0, 0], [1, 0], [1, 1]]$$

$$[-t^2 + 2\,t, t^2]$$

Fig. 1. Maple code worksheet

serves well for producing examples, exercises, graphics and animations. We expect to complete the Maple library soon. As an example of a simple Maple worksheet we show in Figure 1 the Maple code for a function that calculates the i-th Bernstein polynomial of degree n in the variable x.

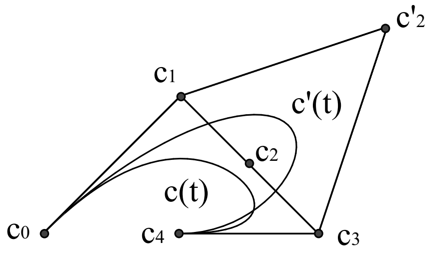


Fig. 2. Two shots of an animation

```
Bernstein:=(n,i,x) -> binomial(n,i)*x^i*(1-x)^(n-i);
```

And another function that uses `Bernstein` for calculating the Bézier parametrization in a variable `x` of a polynomial curve of control polygon given by the list `vertices`,

```
Bezier := proc(vertices, x)
  local n ;
  n := nops(vertices)-1;
  expand(
     sum(vertices[j+1]*Bernstein(n, j, x), j=0..n));
end ;
```

This procedure is used to compute the parametrization of a parabola of control polygon $[[0,0],[1,0],[1,1]]$ in the variable t, $[2t-t^2,t^2]$. Finally, a third procedure is invoked to draw the plot of the parabola and its control polygon.

Or, for instance, we may show the code for the blossom of a B-spline polygon calculated on a sequence of values of the parameter,

```
 dblossom:=proc(vertices,knot,ts)
   local n,i;
   n:=nops(vertices)-1;
   if (nops(ts) <> n) then
     ERROR(`different number of vertices and iterations`) fi;
   if (nops(knot) <> 2*n) then
     ERROR(`erroneous number of knots`) fi;
   if (n<1) then
     ERROR(`erroneous number of iterations`) fi;
   if (knot[n]=knot[n+1]) then
     RETURN() fi; #gets rid of empty intervals
   if (n=1) then
     RETURN(dinterpol(vertices,ts[1],knot)) fi;
   dblossom([dinterpol(vertices,ts[1],knot)],
     [seq(knot[i],i=2..2*n-1)],[seq(ts[i],i=2..n)]);
 end:
```

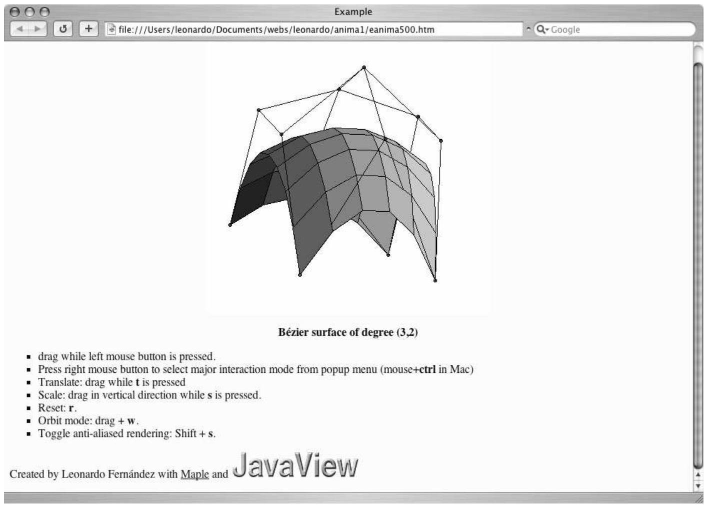
which makes use of an auxiliary function which returns a B-spline polygon of degree $n-1$ after interpolating the vertices of a polygon of degree n at a value u using a sequence of knots.

```
 dinterpol:=proc(vertices,u,knot)
   local n;
   n:=nops(vertices)-1;
   if (nops(knot) <> 2*n) then
     ERROR(`error in number of knots`) fi;
   seq(inter(knot[i+n],knot[i],u)*vertices[i]+
     inter(knot[i],knot[i+n],u)*vertices[i+1],i=1..n);
```

```
end:

inter:=proc(a,b,u)
  if (a=b) then 1 else (u-a)/(b-a) fi;
end:
```

Graphics can be easily concatenated in Maple in order to produce animations, as it is shown in Figure 2, which can be exported as animated GIF files and have been extensively used in the HTML code of the course.


Example
file:///Users/leonardo/Documents/webs/leonardo/anima1/eanima500.htm
Google

Bézier surface of degree (3,2)

- drag while left mouse button is pressed.
- Press right mouse button to select major interaction mode from popup menu (mouse+**ctrl** in Mac)
- Translate: drag while **t** is pressed
- Scale: drag in vertical direction while **s** is pressed.
- Reset: **r**.
- Orbit mode: drag + **w**.
- Toggle anti-aliased rendering: Shift + **s**.

Created by Leonardo Fernández with Maple and JavaView

Fig. 3. A Maple JavaViewLib graphic

2.3. Javascript

Maple graphics and animations are nice, but for an undergraduate course they lack interactivity. For instance, when a surface is drawn, one would request not just to define the best view, the one in which its most remarkable features are shown, but to allow the student to rotate, animate, rescale the surface in order to learn its details, a task that cannot be accomplished with a single view or even with a non-interactive animation.

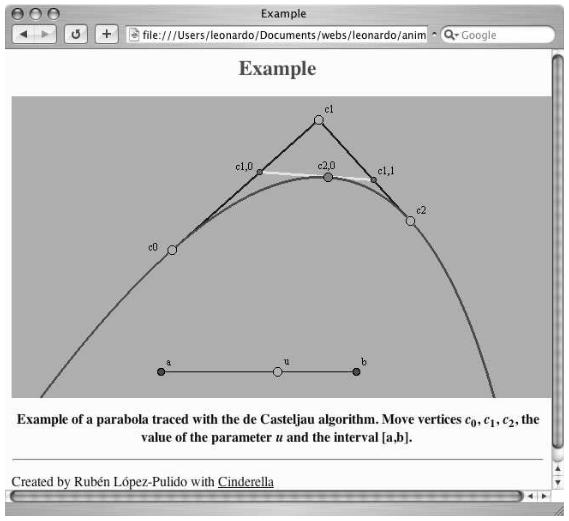
This inconvenience can be circumvented using the JavaViewLib package for Maple, which allows exportation of Maple graphics as Java applets which can be viewed, animated, rescaled, rotated, translated ad libitum by the student (http://www.javaview.de/). In future versions we expect to implement the full JavaView for incorporating interactive graphics for surfaces.

An example of a Maple JavaViewLib graphic is shown in Figure 3.

2.4. Cinderella

The possibility of modification of the views of a 3D graphic is interesting, but still the information for a CAGD course would be too static, since the student is not allowed yet to create or modify anything, but to view what has been done before. Since, as it has already been mentioned, we wished to provide simple examples that could reduce the overwhelming amount of information of any design application, we decided to create our own applets in a language that could be implemented universally in web pages.

Java programming language was obviously our first choice, but since we did not want to build new applications from the beginning, we resorted to Cinderella [4].



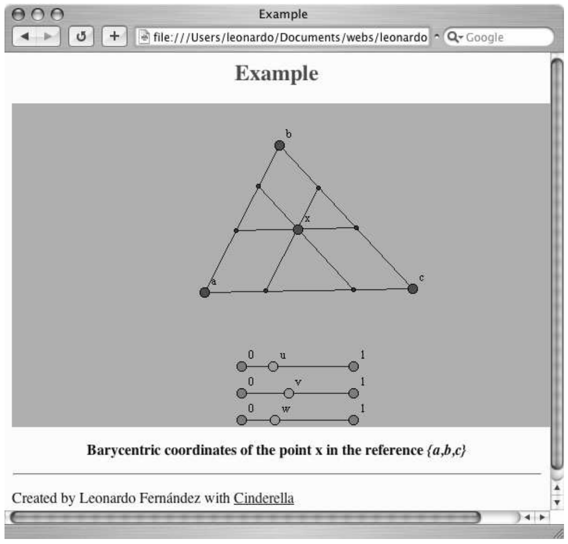
Example of a parabola traced with the de Casteljau algorithm. Move vertices c_0, c_1, c_2, the value of the parameter u and the interval [a,b].

Created by Rubén López-Pulido with Cinderella

Fig. 4. A Cinderella example

Although it is not a common application out of Faculties of Mathematics, Cinderella offers most of what we might ask. Basically Cinderella works as a Maths teacher would do in front of a blackboard. It allows to draw points, lines, circles, parallel and perpendicular lines, calculation of areas and intersections of lines and curves, animations, ac in order to produce mathematical constructions. And, most important of all, Cinderella is capable of exporting the results as Java applets that may be inserted in web pages (http://www.cinderella.de/).

We have made use of these capabilities to produce several interactive



Barycentric coordinates of the point x in the reference *{a,b,c}*

Created by Leonardo Fernández with Cinderella

Fig. 5. Example of calculation of barycentric coordinates with Cinderella

constructions that depict some aspects of CAGD. In Figure 4 you may see a construction made with Cinderella, which draws a quadratic curve from its control polygon with the help of the de Casteljau algorithm. Every point can be modified.

The major drawback that we found in Cinderella was that it cannot deal with analytical expressions, just with synthetical constructions, but it has been surmounted in a recent beta version.

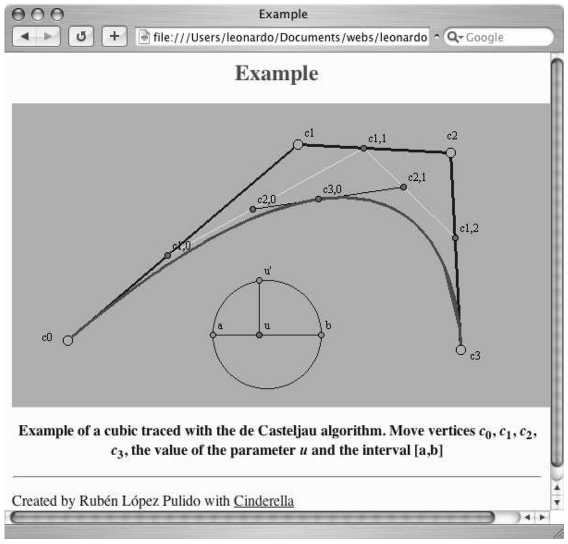
§3. Contents

The contents of the course are structured in six chapters: one of them is devoted to generalities, three are devoted to curves and two are devoted to surfaces.

3.5. Review of Mathematics

The first chapter is a self-contained short review of mathematical concepts that are necessary to follow the main part of the course. Part of the contents of this chapter are surely well known by the students: Cartesian coordinates, ratio of three points, double ratio of four points, whereas other concepts may be new to them: homogeneous coordinates, barycentric coordinates, projective transformations.

In Figure 5 we see an example of calculation of the barycentric coordinates of a point in the plane done with Cinderella.



Example

Example of a cubic traced with the de Casteljau algorithm. Move vertices c_0, c_1, c_2, c_3, the value of the parameter u and the interval [a,b]

Created by Rubén López Pulido with Cinderella

Fig. 6. Drawing a Bézier cubic with Cinderella

3.6. Bézier curves

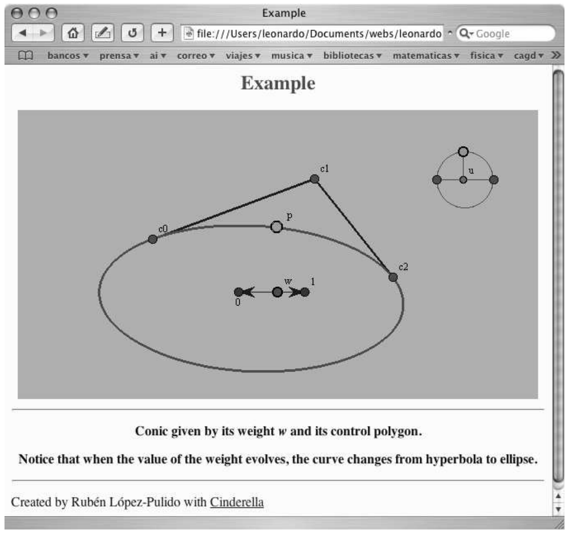
The second chapter introduces actual CAGD with polynomial curves in terms of Bernstein polynomials and control polygons. The main topics of Bézier curves are displayed as it can be found in every CAGD textbook [1], [3], [5]. The de Casteljau algorithm is described and the blossom of a Bézier curve is introduced to obtain control polygons for curves. How to link curves polynomial curves is described and applications to interpolation and approximation problems are discussed.

In Figure 6 we show an example done with Cinderella of the de Casteljau algorithm. With this applet the user may define the control polygon and trace the curve at each value of the parameter.

3.7. Rational curves

As a first step of involvement weights are added to Bézier curves in the third chapter in order to construct rational curves. The main features and problems of rational curves are briefly discussed, such as points at infinity, degree elevation and derivatives. A separate section is devoted to conics as rational curves in order to provide a meaning to weights.

In Figure 7 you may see a construction made with Cinderella, which draws a rational quadratic curve from its control polygon and weight. The weight may be raised or lowered to show the curve changing from an ellipse to a hyperbola.



Example

Conic given by its weight *w* and its control polygon.

Notice that when the value of the weight evolves, the curve changes from hyperbola to ellipse.

Created by Rubén López-Pulido with Cinderella

Fig. 7. Drawing a conic with Cinderella

3.8. Spline Curves

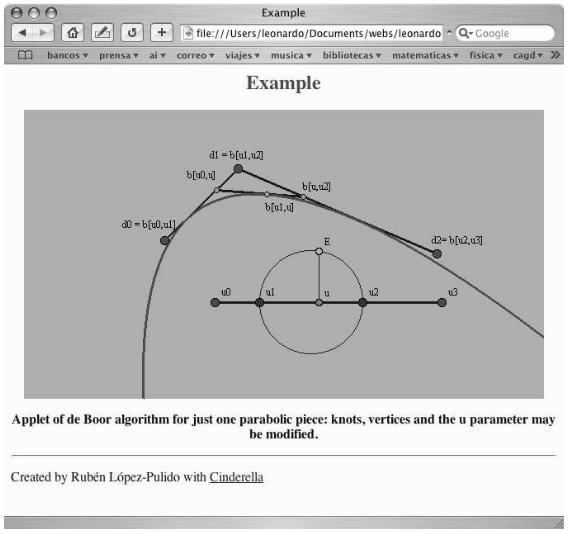
The final topic about curves is piecewise polynomial curves. After reviewing several examples of construction of spline curves, the procedure is generalized via the de Boor algorithm in terms of B-spline polygons and knots. The new features introduced by spline curves are explored and the algorithm for insertion of new knots is explained and applied to convertion of B-spline polygons to control polygons. A quick reference is made to rational B-spline curves. Finally, B-spline functions are defined and their features are detailed.

In Figure 8 we may see the procedure of construction of a segment of a B-spline curve. The user may choose the points of the B-spline polygon and the list of knots.

3.9. Bézier Surfaces

Polynomial surfaces are introduced in a similar way to Bézier curves thanks to the concept of control nets. Generalization to rational and spline surfaces is nearly straightforward. Topics that were discussed for curves are extended to surfaces: blossoming, calculation of control nets, degree elevation, expressions for derivatives... Interpolation of surfaces is discussed at the end of the chapter. Conditions for joining Bézier patches in terms of their control nets are analyzed.

Only product surfaces are considered in this chapter, since Bézier and B-spline triangles could be too involved for undergraduate students. The



Example

Applet of de Boor algorithm for just one parabolic piece: knots, vertices and the u parameter may be modified.

Created by Rubén López-Pulido with Cinderella

Fig. 8. A segment of a spline curve drawn with Cinderella

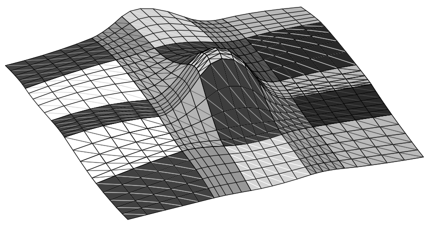


Fig. 9. Spline surface

study of Bézier triangles is deferred to another chapter.

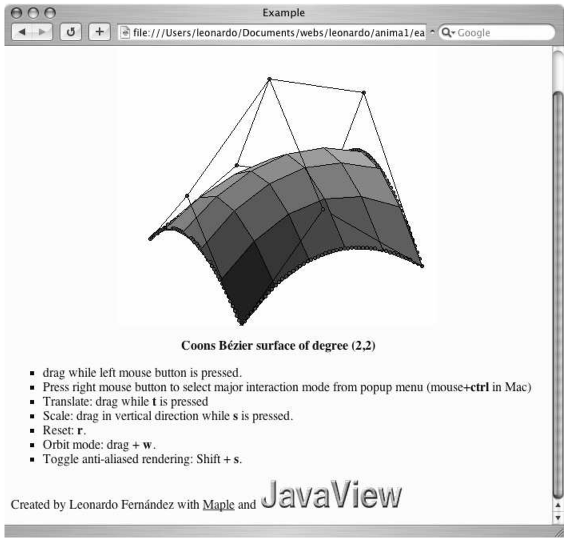
In Figure 9 the local effects of moving a point of the B-spline net of a spline surface are shown.

3.10. Generation of Surfaces

The last chapter is miscellanea of different ways of generating surfaces. Translational, ruled, developable, Coons surfaces are introduced. An extended section is devoted to surfaces of revolution, both as Bézier and as spline surfaces.

In Figure 10 we may see the construction of a Coons surface through four curves done with Maple and JavaViewLib.

§4. Conclusions and Future Developments



Coons Bézier surface of degree (2,2)

- drag while left mouse button is pressed.
- Press right mouse button to select major interaction mode from popup menu (mouse+**ctrl** in Mac)
- Translate: drag while **t** is pressed
- Scale: drag in vertical direction while **s** is pressed.
- Reset: **r**.
- Orbit mode: drag + **w**.
- Toggle anti-aliased rendering: Shift + **s**.

Created by Leonardo Fernández with Maple and JavaView

Fig. 10. Maple JavaViewLib applet for a Coons surface

In this talk we have shown how different applications may be combined to provide an efficient e-course on CAGD. The course has been successfully included this year in the Universidad Politécnica de Madrid studies as an optional subject.

As future developments we wish to reinforce the use of Maple by incorporating Maple applets, maplets, and by exploring the possibilities of the recently released version of Maple 9 for web applications. The new version of Cinderella includes analytical calculations and thereby should make easier the task of creating new applets.

We aim to include new chapters in the course that may make it interesting for postgraduate students. We have in mind a separate chapter on developable surfaces, an issue that is of major importance for naval industry and other branches of engineering that depend on steel sheets. We also wish to include a chapter on Bézier triangles as a different way of constructing surfaces.

§5. Acknowlegments

The authors wish to thank the G.A.T.E. of the Universidad Politécnica de Madrid for supporting the course in their website and technical assistance. Leonardo Fernández-Jambrina wishes to thank Universidad Politécnica de Madrid for financial support. Rubén López-Pulido wishes to thank the E.T.S.I. Navales, Fundación Marqués de Suanzes and the Departamento de Enseñanzas Básicas de la Ingeniería Naval for their financial support. All graphics were made with Maple and Cinderella. Details were modified with Adobe Illustrator 10.

§6. References

1. Farin, G., *Curves and Surfaces for CAGD: a Practical Guide*, Morgan Kaufmann Publishers, San Francisco 2002.
2. Fernández-Jambrina L., *Notas sobre Diseño Geométrico Asistido por Ordenador*, F.E.I.N., Madrid, submitted.
3. Hoscheck, J. and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley Massachussets, 1993.
4. Richter-Gebert, J. and U.H. Kortenkamp, *The Interactive Geometry Software Cinderella*, Springer Verlag, Berlin, 1999.
5. Salomon, D., *Computer Graphics and Geometric Modeling*, Springer Verlag, Berlin, 1999.

L. Fernández-Jambrina, R. López-Pulido
ETSI Navales, Universidad Politécnica de Madrid
Madrid, Spain
`lfernandez@etsin.upm.es`
`http://debin.etsin.upm.es/ilfj.htm`