KETpic-Matlab Toolbox for LaTeX High-Quality Graphical Artwork in Educational Materials on Bézier Curve Algorithms at a Master Level

Akemi Gálvez[1,2](✉), Setsuo Takato[1], Masataka Kaneko[3], Javier Del Ser[4,5,6], and Andrés Iglesias[1,2]

[1] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan

[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, 39005 Santander, Spain
galveza@unican.es

[3] Faculty of Pharmaceutical Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan

[4] Tecnalia Research & Innovation, 48160 Derio, Bizkaia, Spain

[5] University of the Basque Country UPV/EHU, Bilbao, Bizkaia, Spain

[6] Basque Center for Applied Mathematics (BCAM), Bilbao, Bizkaia, Spain

Abstract. This paper introduces a new KETpic-Matlab toolbox to generate LaTeX-embedded high-quality graphical artwork about the main algorithms for Bézier curves and related topics. The package has been implemented by the authors as a supporting middleware tool to generate educational materials for a computer-aided geometric design (CAGD) course at an advanced level (typically a Master/PhD course; even a senior course in some engineering degrees). Its primary goal is to instill geometric intuition in our students and help them develop critical thinking based on geometric reasoning for problem solving. We also aim at providing the instructors of such courses with a computer library to produce high-quality graphics for educational materials with a seamless integration into LaTeX. In this regard, all graphical objects are encoded as plain LaTeX-readable source code that yields nice pictures after standard LaTeX compilation. The application of this package to generate educational materials is discussed through six illustrative examples of interesting properties of the Bézier curves. They show that our package is very easy to use, supports many different graphical options, and fosters students' creative geometric thinking with very little effort.

Keywords: Bézier curve · De Casteljau's algorithm · Educational materials · Graphical artwork · Matlab toolbox

1 Introduction

During the last few years, the authors have been deeply involved in the development of high-quality educational materials for teaching and learning at various University levels. Among them, a course about CAGD (the acronym for

Computer-Aided Geometric Design) is currently taught in the Master of Mathematics and Computer Science of the University of Cantabria. Similar courses are also delivered in many other Master and PhD programs, even in advanced senior year of some engineering degrees (e.g., mechanical engineering, aerospace engineering, ship building engineering). Although it also explores some remarkable applications in many fields (such as computer-aided design and manufacturing - CAD/CAM, and geometric modeling and processing for medical imaging in computer tomography (CT) and magnetic resonance imaging (MRI), to mention just two of them), the course is mainly focused on the mathematical and geometrical aspects of curves and surfaces and their properties.

In our experience, the biggest challenge of this course is how to instill a geometric intuition about the CAGD algorithms [1] in our students so that they can properly apply geometric reasoning to problem solving. Very often, we find that our students do solve problems by following an strictly algebraic approach: they focus exclusively on the equations and perform algebraic manipulations on them to get a proper solution to a given problem. However, they fail to provide a geometric interpretation to this process and typically do not related the obtained solution to any particular geometric configuration or whatever. This is a critical indication that they are operating on a purely algebraic basis, without any mental relationship with the real geometry of the problem. Clearly, there is a need for a less mechanical, "automatic-pilot" approach for this kind of courses.

In addition to other issues, part of the problem is related to the classroom materials our students typically use. Many textbooks and other materials rely on equations and algebraic manipulations more often than advisable, and usually skip the geometric interpretation. This approach, albeit undesirable, can somehow be justified: producing good graphical materials is cumbersome and time-consuming, and requires some expertise about computer tools for graphical editing, which are also usually expensive and difficult to use. In this context, the geometric flavor of our course poses a particularly challenging question:

How to provide our students with an affordable set of classroom materials with high-quality graphical artwork?

In our opinion, such materials must fulfill the following conditions:

- *they must have a strong focus on geometrical intuition:* in this regard, they should allow our students to grasp the subtle details of the most common algorithms in CAGD in a geometrical fashion rather than the (most standard but less effective) equation-based approach.
- *they must be accurate:* accuracy is essential in many CAGD geometric algorithms; otherwise, they do not work properly and students will find troubles in understanding them at full extent.
- *they must have a high visual quality:* classroom materials must include all elements required to fully understand the algorithms. Generally, they include axis labels and ticks, text annotations, legends, colors, varying types of lines (solid, dashed, dotted) with varying sizes for line thickness, and many other

graphical and text elements playing the role of visual clues for better understanding.
- *they must be affordable to produce:* this means that materials must be not only meaningful in terms of contents but also somehow optimized regarding the time and effort required to produce them. Instructors time is limited, so issues such as a long learning curve for the programs or applications used to generate the classroom materials or a large production time should be avoided. Typically, such programs should provide a number of valuable features on a user-friendly ready-to-use approach.
- *they must be easy to deploy:* this implies that the storage size of the graphical elements must be kept to the minimum. Typically, high-resolution artwork demands large file sizes, thus preventing the materials to be smoothly deployed over the web or attached to email messages. Vector graphics can be the solution for this problem, but most vector-graphics programs are intended for professional use in printing and publishing business; as a result, they tend to be quite expensive and difficult to use. A good solution should ideally overcome these drawbacks.
- *they must be LaTeX-compatible:* LaTeX is a standard de facto tool for high-quality scientific publishing and editing. Therefore, it is advisable for the tools used to generate educational materials to have a seamless integration into LaTeX for optimal quality and performance.

Unfortunately, it is not easy to find computer tools providing all the features mentioned above. A pioneering effort in this regard was launched a few years ago under the KETpic project (see Sect. 2 for details). In this paper, we propose to tackle the issue by using a KETpic toolbox for Bézier curves developed by the authors on the popular scientific computer program *Matlab* [8].

The structure of this paper is as follows: in Sect. 2 we describe the main features of our KETpic-Matlab toolbox for Bézier curves. The application of this toolbox to generate educational materials is discussed in Sect. 3 through six illustrative examples of interesting properties of the Bézier curves. This section also illustrates the potential of our toolbox as an educational supporting tool for scientific visualization and critical geometric thinking. The paper closes with the main conclusions and some ideas for future work in the field.

2 KETpic-Matlab Toolbox for Bézier Curve Algorithms

This section describes the main features of our KETpic toolbox on Matlab. For the sake of completeness, we also present briefly KETpic to those readers not familiar with this interesting software.

2.1 About KETpic

KETpic (an acronym for *K*isarazu *E*ducational *Tpic*) is a middleware software released in 2006 for high-quality mathematical drawing in LaTeX [9]. The software, conceived at the "Kisarazu National College of Technology" for educational purposes, has roots in *Tpic* system, a terminal driver that supports the LaTeX `picture` environment. Originally developed for the computer algebra system *Maple* in April 2006, it was designed to provide LaTeX end-users with extensive support for visualization and printing of high-quality graphical objects. As such, it is an ideal package to generate printed materials (books, slides, reports, etc.) with graphical content for educational purposes and professional printing. See [3–5,7,10,11] for several examples and additional information about this software.

Briefly speaking, KETpic is a library of macros and scripts to generate LaTeX-readable source code for high-quality scientific artwork. Such macros can be implemented on different computer algebra systems and other scientific programs, thus yielding different versions of the package (used as plug-ins or add-ons). Depending on the scientific system they are based on, these plug-ins might typically run internally in a quite different way, but the process is usually transparent to end-users, thus minimizing the time required to get accustomed to the program. Once KETpic is loaded, users are simply requested to execute commands in the system of their choice to plot graphical data. Specialized embedded KETpic commands generate additional LaTeX source code and files, which are subsequently compiled in LaTeX in the usual manner. As a result, accurate graphical figures are readily obtained from small plain text files, an optimal solution for efficient data storage and file transfer. A first KETpic version for standard mathematical curves in Matlab is described in [2], then extended in [6].

2.2 KETpic-Matlab Toolbox Pipeline

To use the KETpic-Matlab toolbox, we have to proceed according to the following pipeline. The process starts up by opening Matlab for a new session and loading the KETpic toolbox through the following command:

```
>> Ketinit
```

Generation of figures in LaTeX from the original pictures created in *Matlab* requires to carry out some steps. As mentioned above, any graphical object is encoded as a plain text file to be compiled in LaTeX. The following script executes this process:

```
1   >> Openfile('filename.tex');
2   Beginpicture('');
3         graphical functions here (see Sect. 3 for details)
4   Endpicture(#picture);
5   Closefile();
```

This script stores all data associated with the graphical object into the file *filename.tex* for subsequent use in LaTeX. Line 1 of this script opens such a file in the folder indicated in the namepath. Second line defines the units of length for the final picture (with default value 1 cm). The command `Beginpicture` is also used to create the `\begin{picture}...\end{picture}` environment in

LaTeX. The graphical functions in line 3 convert the data points into a sequence of LaTeX-readable commands to be inserted into the `picture` environment created in line 2 for standard compilation. The command `Endpicture()` in line 4 performs two different actions: on one hand, it closes the `picture` environment in the TeX file. On the other hand, it allows us to set up the display of cartesian axes, according to its value: `1` (empty value is also feasible) if axes are to be displayed and `0` otherwise. Finally, Line 5 closes the file.

The final output of this process is the file *filename.tex* in our workspace folder. It contains a description of the graphical objects created in *Matlab* in terms of LaTeX and *Tpic* commands. The file can be embedded into a standard LaTeX file for compilation. The following code will yield a printout of the graphical objects within the file:

```
\documentclass[11pt]{article}
     ...
\newlength{\Width}
\newlength{\Height}
\newlength{\Depth}
     ...
\begin{document}
     ...
\input{ filename.tex }
     ...
\end{document}
```

It is the typical LaTeX code with a `documentclass` declaration and the `document` environment. The only difference are three lines in the preamble (between the start of the file and the `\begin{document}` command) that specify new directives for the length units, and the `\input` command in the main body of source code that causes the indicated file to be read and processed, exactly as if its contents had been inserted in the current file at that point. Compilation of the code above generates the graphical objects described in *filename.tex* in the form of a DVI file.

3 Illustrative Examples

In this section, some illustrative examples about the application of the KETpic-Matlab toolbox described in previous section to different algorithms for Bézier curves are described. In what it follows, we assume that the reader is familiar with the main concepts about free-form parametric curves [1]. A *free-form parametric Bézier curve* $\mathbf{C}(t)$ *of degree* n is defined as:

$$\mathbf{C}(t) = \sum_{j=0}^{n} \mathbf{P}_j B_j^n(t) \qquad (1)$$

where $\mathbf{P}_j$ are vector coefficients (usually referred to as the *control points*), $B_j^n(t)$ are the *Bernstein polynomials of index j and degree n*, given by:

$$B_j^n(t) = \binom{n}{j} \, t^j \, (1-t)^{n-j} \tag{2}$$

and t is the *curve parameter*, defined on the finite interval $[0, 1]$. Note that in this paper vectors are denoted in bold. By convention, $0! = 1$.

3.1 Example 1: De Casteljau's Algorithm for Evaluation

The *De Casteljau's algorithm* is a recursive algorithm developed by French mathematician Paul de Casteljau in 1959 (while working for automotive company Citroën) to evaluate a Bézier curve at a point given by the parameter value $t = t_0$. Such a point $\mathbf{C}(t_0)$ can be obtained by the recurrence relation:

$$\begin{cases} \mathbf{P}_j^{(0)} = \mathbf{P}_j & j = 0, \ldots, n \\ \mathbf{P}_j^{(k)} = \mathbf{P}_j^{(k-1)}(1-t_0) + \mathbf{P}_{j+1}^{(k-1)} t_0 & j = 0, \ldots, n-k \, ; \, k = 0, \ldots, n \end{cases} \tag{3}$$

In this case, $\mathbf{C}(t_0)$ is evaluated in n steps of this algorithm as: $\mathbf{C}(t_0) = \mathbf{P}_0^{(n)}$.

A very exciting feature about the De Casteljau's algorithm is the fact that it has a straightforward and appealing geometric interpretation. The process is shown graphically in Fig. 1. We start with an initial Bézier curve with 8 control points, displayed in Fig. 1 (top-left). As the reader can see, the control points are connected by a control polygon comprised of linear segments between consecutive control points. Then, we subdivide each line segment of the control polygon according to the ratio $t_0 : (1 - t_0)$ and connect the resulting points to get a new control polygon with one fewer control point (and hence, one fewer segment) for each recursion step, until eventually reaching a single point. The different steps of this process are shown in Fig. 1: the figure in top-right corresponds to the first step, where seven control points (displayed in red) are obtained. Next step is shown in the second row for six control points (displayed in green on the left) and five control points (in blue on the right). The process ends after seven steps, with a final point corresponding to the value $t = 0.4$ (displayed in brown in Fig. 1 (bottom-right)).

It is important to remark that the pictures in Fig. 1 (and actually all pictures in this paper) do not come from files in any graphical format, such as EPS, JPG, GIF, or the like. Instead, they are a simple collection of LaTeX-readable instructions from a plain text file, generated as described in previous section. As remarked above, this has the clear advantage of a very small file size (about only a few KB). In addition, no graphical editor is needed to produce or manipulate our artwork; just a few simple functions from our KETpic-Matlab toolbox are required. In this particular example, the function `DeCasteljauAlgorithm` is invoked. Its syntax is as follows:

`DeCasteljauAlgorithm(`*cp*, t_0, *th*, *iter*, *lgopts*`)`

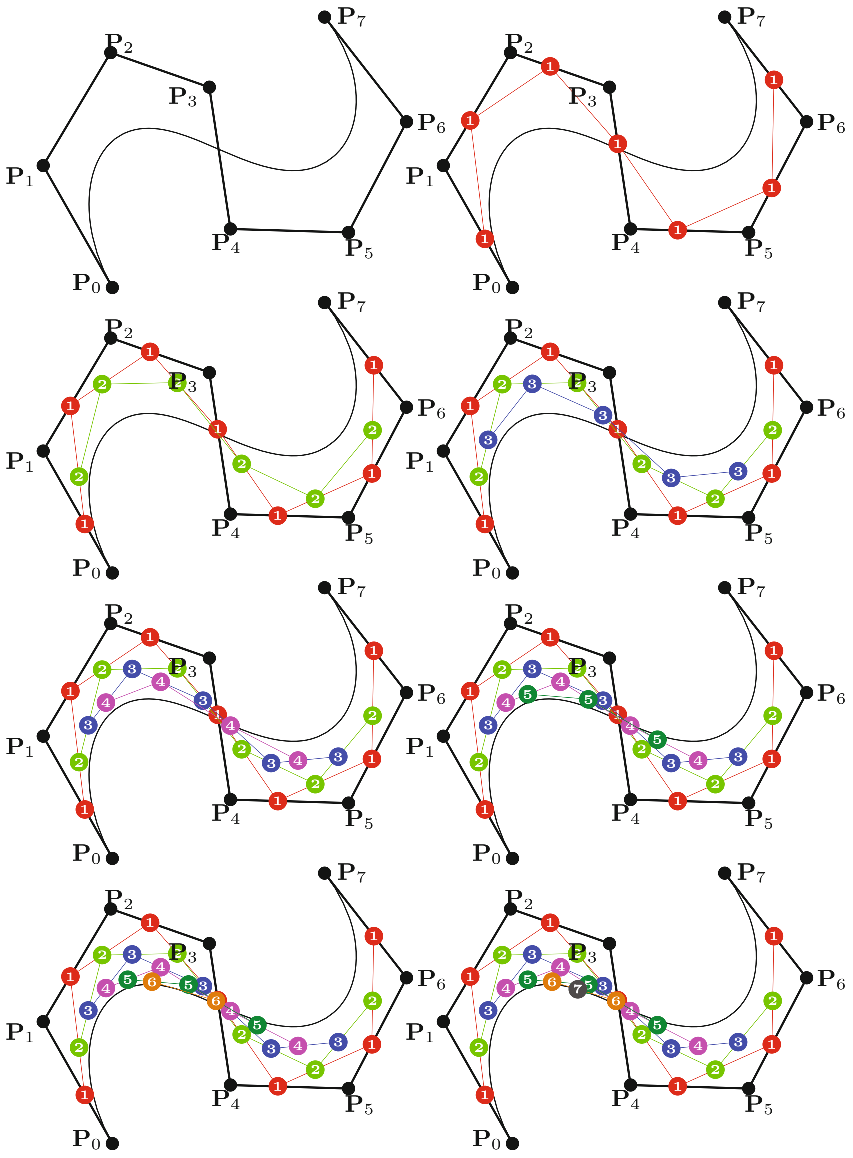

Fig. 1. (left-right, top-bottom) Graphical representation obtained with our KETpic-Matlab toolbox of all steps of the De Casteljau's algorithm for evaluation of a Bézier curve at the parametric value $t_0 = 0.4$. (Color figure online)

where *cp* indicates the list of control points for the Bézier curve, t_0 is the parameter value at which the curve is evaluated, *th* means the line thickness for the curve, *iter* means the number of steps for the De Castellau's algorithm (by default, this number is the degree of the curve, but it can be changed to any lower value), and *lgopts* is a list of graphical options available to improve our artwork. These options include the point size and color for the control points, line width, line type (e.g., solid, dashed, dotted, etc.), and color for the control polygon, type of symbols used for the control points (e.g., diamond, star, bullet, triangle, filled circle, empty circle, etc. as well as any TEX symbol), and other related options, such as the font size and font family for legends and labels, options for axis labels and ticks, options for grids, bounding boxes, and many others. For instance, the new control points obtained at each step of the De Casteljau's algorithm in Fig. 1 are displayed as filled circles of a different color for each step. Also, the iteration number is displayed in white within the filled circle for a given font size. All these features can be modified at will by using the list of options of the `DeCasteljauAlgorithm` function. This remarkable set of available options allows us to increase notably the expressive power of the resulting figures. Our students can now grasp all the details about how does this algorithm actually work, and what is the meaning of each control point step by step. As a result, the students' focus is shifted from the mathematical equations to the geometric intuition about the algorithm. Furthermore, the process displayed in Fig. 1 can be executed in real time to generate an animation, which can be displayed on a web browser and/or exported to a PDF file, for instance.

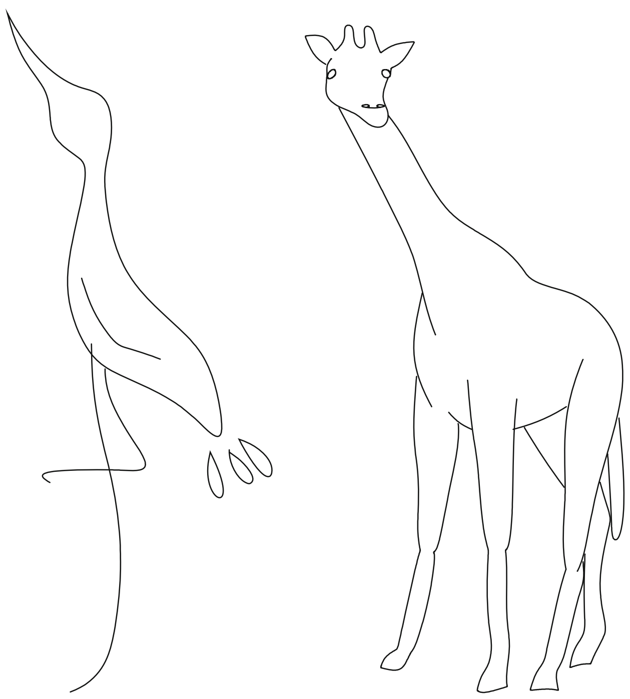

Fig. 2. Two examples of Bézier curves generated with our KETpic-Matlab toolbox.

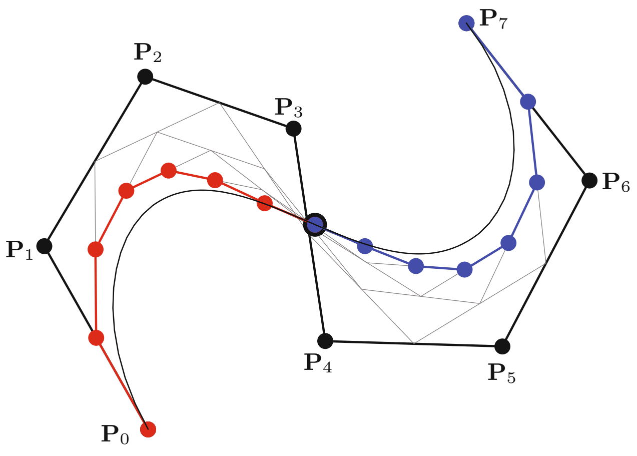

Fig. 3. Application of the De Casteljau's algorithm for curve subdivision: the original Bézier curve with 8 control points in Fig. 1 (top-left) is subdivided at the parametric value $t_s = 0.5$. As a result, the curve is subdivided into two new Bézier curves with 8 control points each (displayed in red and blue color, respectively). (Color figure online)

Figure 2 shows two examples of artistic shapes generated with our KETpic-Matlab toolbox and corresponding to the silhouette of a crane and a giraffe, respectively. In this case, the function `BezierCurve(`*cp, lgopts*`)` is used.

3.2 Example 2: De Casteljau's Algorithm for Subdivision

The De Casteljau's algorithm can also be applied without any modification to curve subdivision, i.e., to split a Bézier curve of degree n into two Bézier curves of the same degree n joined together at an arbitrary parameter value $t = t_s$. To this aim, it is enough to apply the recursive process described in Sect. 3.1 and then consider the two Bézier curves of degree n and control points:

$$\left\{\mathbf{P}_0^{(0)}, \mathbf{P}_0^{(1)}, \dots, \mathbf{P}_0^{(n)}\right\} \text{ and } \left\{\mathbf{P}_0^{(n)}, \mathbf{P}_1^{(n-1)}, \dots, \mathbf{P}_n^{(0)}\right\} \tag{4}$$

respectively.

The geometric interpretation of this process is graphically shown in Fig. 3. The subdivision process of the original Bézier curve with 8 control points in Fig. 1 (top-left) at the parametric value $t_s = 0.5$ yields two new Bézier curves with 8 control points each. The first one, displayed in red in Fig. 3, is obtained by taking the initial control point of the control polygon for each step of the De Casteljau's algorithm, displayed in each picture of Fig. 1 (i.e., first control point $\mathbf{P}_0$ (top-left), then first control point with label 1 (top-right), then first one with label 2 (second row-left) and so on). The second curve, displayed in blue, corresponds to a similar process but taking the last control point at each step instead. The `SubdivideCurve(`$cp, t_s, options$`)` function in our toolbox returns this graphical output in one step.

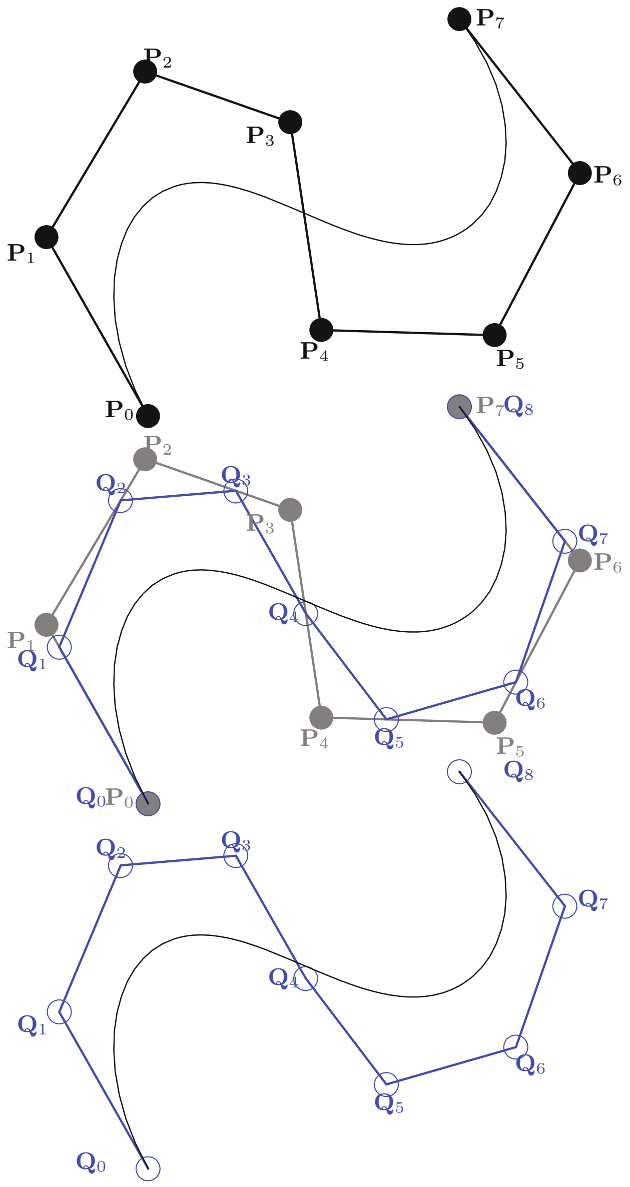

Fig. 4. Modified De Casteljau's algorithm for degree raising: (top): original Bézier curve of degree 7; (middle) original (in gray) and new control polygon (in blue) for degree raising; (bottom) resulting Bézier curve of degree 8. (Color figure online)

3.3 Example 3: Modified Algorithm for Degree Raising

From Eqs. (1)–(2), it is evident that the degree of curve $\mathbf{C}(t)$ depends on the number of control points. In general, increasing the degree makes the curve more complex and requires longer computation time, but it also gives the curve higher flexibility and increases the curve ability to design more complicated shapes. Because of this reason, it might be very helpful to be able to increase the degree of a Bézier curve without changing its shape. This process, usually called degree raising (also degree elevation), can be performed with a procedure that is not exactly the De Casteljau's algorithm, but quite similar to it.

Suppose a Bézier curve of degree n with control points $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n\}$ defined as above. We want to increase the degree of the curve to $n+1$ while still preserving the same shape of the curve. The problem is now how to compute the new control points $\{\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{n+1}\}$. Since the Bézier curve always interpolate the end control points, we can take $\mathbf{Q}_0 = \mathbf{P}_0$ and $\mathbf{Q}_{n+1} = \mathbf{P}_n$. The remaining control points can be computed as:

$$\mathbf{Q}_j = \frac{j}{n+1}\mathbf{P}_{j-1} + \left(1 - \frac{j}{n+1}\right)\mathbf{P}_j \quad ; \quad j = 1, \ldots, n \tag{5}$$

The geometric interpretation of this process is shown in Fig. 4. Starting with a Bézier curve of degree 7 (top), for each segment of the control polygon connecting the points $\mathbf{P}_{j-1}$ and $\mathbf{P}_j$ we consider the point $\mathbf{Q}_j$ on that segment that divides it in a ratio $\left(1 - \frac{j}{n+1}\right) : \frac{j}{n+1}$. In this particular example, we have $n+1 = 8$ and obtain the new control polygon in blue in Fig. 4 (middle). Note that the original curve is now displayed in gray to emphasize visually the new control polygon. The resulting Bézier curve of degree 8 is shown in Fig. 4 (bottom). Note that this curve and the original one have identical shape. They are obtained by using the `DegreeRaising(`*cp, options*`)` function in our toolbox.

3.4 Example 4: Variation Diminishing Property

An interesting feature of the Bézier curves is the variation diminishing property, which in short states that the curve is smoother than its control polygon. Geometrically, this means that if a line is drawn through the curve, the number of intersections with the curve will be less than or equal to the number of intersections with the control polygon. In our Master course on CAGD, we challenge our students to put this property in practice by using some educational materials generated with the KETpic-Matlab toolbox described in this paper. Figure 5 shows an illustrative example. Top picture displays a geometric configuration of a control polygon defined by the sequence of 5 control points in the picture and the three lines depicted in the figure. Then, we ask the students to:

– draw the control polygon
– determine the maximum number of times that the curve can intersect each of those lines
– draw the Bézier curve
– determine how many times the curve does really intersect each of those lines
– explain the results.

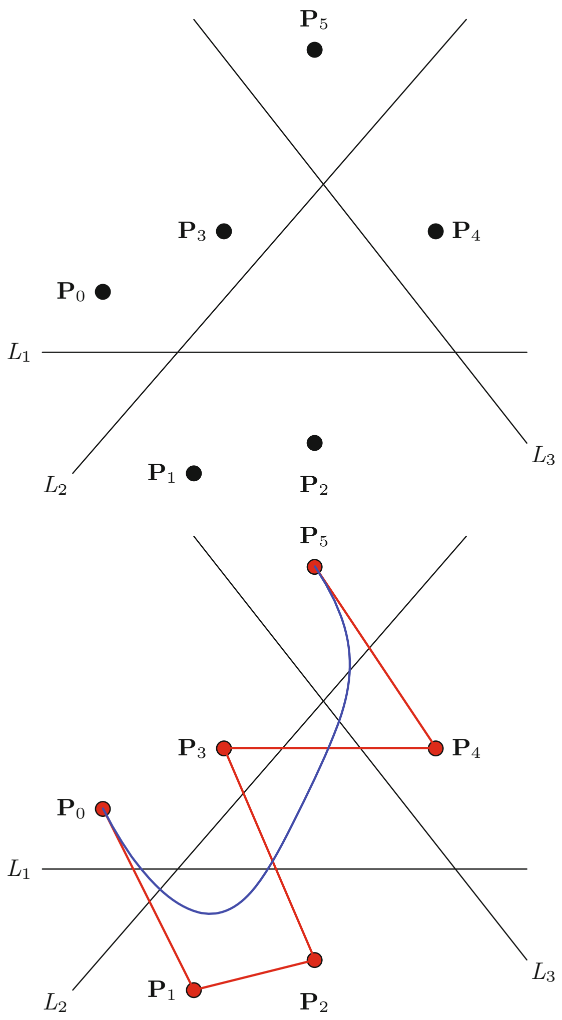

Fig. 5. Example of the variation diminishing property of Bézier curves: (top) initial configuration of control points and lines; (bottom) graphical solution to the problem.

Then, the students can check their results by using our toolbox to yield the solution to this problem, shown in bottom picture of Fig. 5. We found that many students were surprised by this graphical result, as they did expect the curve to replicate more faithfully the shape of the control polygon. Of course, any other geometric configuration for the control points and lines can readily be arranged with our toolbox, so even more intriguing situations can be discussed and analyzed. This shows the potential of our toolbox as a learning tool, as it encourages the student to think twice about this property and the expected behavior of the Bézier curve.

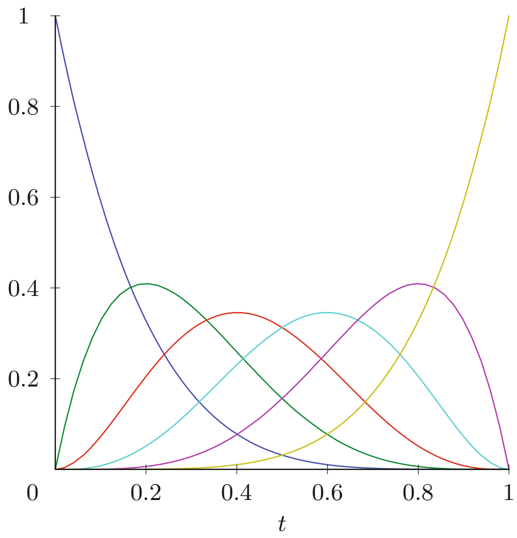

Fig. 6. Graphical representation of the Bernstein polynomials given by Eq. (2).

3.5 Example 5: Basis Functions and Partition of Unity Property

An interesting feature of the basis functions of a Bézier curve is the so-called partition of unity property, which says that:

$$\sum_{j=0}^{n} B_j^n(t) = 1 \quad ; \quad \forall t \in [0,1] \tag{6}$$

In this example, we show the basis functions in Fig. 6 associated with a Bézier curve and ask our students to:

– determine the number of control points of the curve
– determine the degree of the curve
– check the partition of unity property for different parametric values.

While the first two questions can easily be answered from the figure, the last one is more difficult to check. A remarkable feature of our toolbox in this regard is the fact that all figures generated with it are *metric*, which means that all distances are referred to a certain unit length, so the students can actually measure distances on the figures. For instance, the reader can easily check that the unit length in the vertical and horizontal axes is exactly the same, as opposed to many other graphical representations where a golden ratio or other non-unit value is applied to the axes ratio. In fact, our students were surprised to see the basis functions displayed with equal sizes in both axes, as all textbooks show a "more stylized" version. Owing to this feature, the students can check the partition of unity directly on the figure without any size distortion.

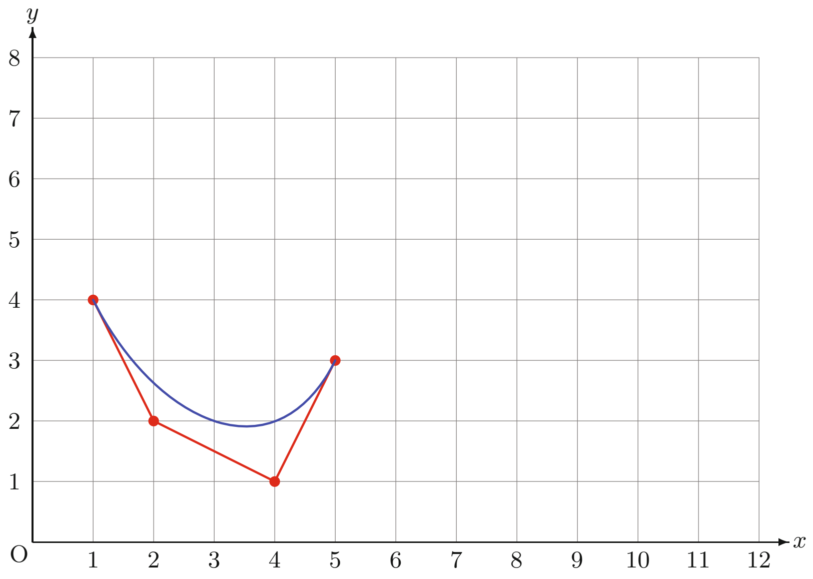

Fig. 7. Our readers are challenged to solve the problem described in Sect. 3.6 by using this figure and their geometrical reasoning.

3.6 Example 6: Connecting Bézier Curves with a Given Continuity

A very interesting (and useful) property of the control polygon of a Bézier curve is that not only the curve does interpolate the first and last control points but also the end segments of the control polygon determine the tangency of the curve at such control points. In particular, the tangent vector of the curve at the first (respectively, last) control point is given by the first (resp., last) segment of the control polygon. This property is very useful in order to connect several Bézier curves with a certain order of continuity.

To emphasize this, we give Fig. 7 to our students and ask them to determine (without using the mathematical equation of the displayed Bézier curve) some feasible control points for a second Bézier curve of the same degree as the curve shown in the figure so that:

- they are not connected at any point
- they are connected at the leftmost control point with C^0-continuity
- they are connected at the rightmost control point with C^0-continuity
- they are connected at the leftmost control point with C^1-continuity
- they are connected at the rightmost control point with C^1-continuity

After reading all paper contents so far, we think that you, our reader, are ready to solve this problem by using solely your geometrical reasoning, similar to our students. And hopefully, you will also have the same feeling of satisfaction as our own students when you realize that you grasped the main concepts about

Bézier curves with little effort taking advantage of the toolbox described here. We strongly encourage you to try. *Good luck and let's go!*.

4 Conclusions and Future Work

In this paper we introduced a new KETpic-Matlab toolbox to generate LaTeX-embedded high-quality graphical artwork about the main algorithms for Bézier curves and related topics. The package was implemented by the authors as a supporting middleware tool to generate educational materials for a computer-aided geometric design (CAGD) course at an advanced level. Its primary goal is to instill geometric intuition in our students and help them develop critical thinking based on geometric reasoning for problem solving. We also aim at providing the instructors of such courses with a computer library to produce high-quality graphics for educational materials with a seamless integration into LaTeX. In this regard, all graphical objects are encoded as plain LaTeX-readable source code that yields nice pictures after standard LaTeX compilation. The application of this package to generate educational materials is discussed through six illustrative examples of interesting properties of the Bézier curves. They show that our package is very easy to use, supports many different graphical options, and fosters students' creative geometric thinking with very little effort.

Our KETpic-Matlab toolbox has already been applied during the last academic year for two Master courses on CAGD with very encouraging results. The feedback from our students has been very positive; they praised the strong geometric orientation of the teaching materials generated with this software, its ease of use, and its smooth integration into LaTeX. However, a deeper and more formal analysis is still needed to determine its advantages and limitations at full extent. This process will require larger groups of students and a more rigorous statistical analysis based on questionnaires designed with the support of pedagogy and education sciences experts. We also plan to extend our package to the (more difficult) case of Bézier surfaces. A user's manual and some support material for teachers and instructors would also be needed for wider dissemination of this software and its adoption as a standard tool in CAGD courses worldwide. All these tasks will be part of our plans for future work in this field.

Acknowledgments. This work was kindly supported by Japanese Society for Promotion of Science (JSPS) KAKENHI Grant Numbers 15K01037, 15K00944, and 16K01152, the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project Ref. #TIN2012-30768, Toho University, the University of Cantabria, Tecnalia Research & Innovation, the University of the Basque Country, and the Basque Center for Applied Mathematics (BCAM).

References

1. Farin, G.: Curves and Surfaces for CAGD, 5th edn. Morgan Kaufmann, San Francisco (2002)
2. Gálvez, A., Iglesias, A., Takato, S.: New Matlab-based KETpic plug-in for high-quality drawing of curves. In: International Conference on Computational Science and its Applications-ICCSA 2009, CA, pp. 123–131. IEEE Computer Society Press, Los Alamitos (2009)
3. Gálvez, A., Iglesias, A., Takato, S.: Matlab-based KETpic add-on for generating and rendering IFS fractals. Commun. Netw. **56**, 334–341 (2009)
4. Gálvez, A., Iglesias, A., Takato, S.: KETpic Matlab binding for efficient handling of fractal images. Int. J. Future Gener. Commun. Netw. **3**, 1–14 (2010)
5. Gálvez, A., Kitahara, K., Kaneko, M.: *IFSGen4LATEX* : interactive graphical user interface for generation and visualization of iterated function systems in LATEX. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 554–561. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44199-2_84
6. Iglesias, A., Gálvez, A.: Computer software program for representation and visualization of free-form curves through bio-inspired optimization techniques. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 570–577. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44199-2_86
7. Kaneko, M., Izumi, H., Kitahara, K., Abe, T., Fukazawa, K., Sekiguchi, M., Tadokoro, Y., Yamashita, S., Takato, S.: A simple method of the TeX surface drawing suitable for teaching materials with the aid of CAS. In: Bubak, M., Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008. LNCS, vol. 5102, pp. 35–45. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69387-1_5
8. The Mathworks Inc: Using Matlab, Natick, MA (1999)
9. Sekiguchi, M., Yamashita, S., Takato, S.: Development of a maple macro package suitable for drawing fine TeX pictures. In: Iglesias, A., Takayama, N. (eds.) ICMS 2006. LNCS, vol. 4151, pp. 24–34. Springer, Heidelberg (2006). doi:10.1007/11832225_3
10. Takato, S., Gálvez, A., Iglesias, A.: Use of ImplicitPlot in drawing surfaces embedded into LaTeX documents. In: International Conference on Computational Science and its Applications-ICCSA 2009, CA, pp. 115–122. IEEE Computer Society Press, Los Alamitos (2009)
11. Yamashita, S., Abe, T., Izumi, H., Kaneko, M., Kitahara, K., Sekiguchi, M., Fukazawa, K., Takato, S.: Monochrome line drawings of 3D objects due to the programmability of KETpic. In: International Conference on Computational Science and its Applications-ICCSA 2008, CA, pp. 277–283. IEEE Computer Society Press, Los Alamitos (2008)