Bölüm 6

The Evolution of Bézier Curves in Computer-Aided Geometric Design (CAGD): *A Systematic Review*

Seda Karateke[1]

Rumeysa Akalin[2]

Mehmet Gümüş[3]

Abstract

Computer Aided Geometric Design (CAGD) is a field of science that grows with the wind of technology behind it, developing day by day and enchanting us. It is an undeniable fact that the development process of this hot topic is based on mathematical foundations. At the core of this mathematical foundation lies the theory of Bézier curves. We conducted a study aiming to present the birth of Bézier curves, their historical development and especially their applications in the field of computer-aided geometric design with a rich spectrum. Another emerging field that we would like to draw your attention to here is *reverse engineering*. Discovering the function, structure and working principle of an object or system with an inferential reasoning method is included in the field of reverse engineering. At this point, it would be logical to say that the concepts of CAGD and reverse engineering support each other. We foresee that future researchers will contribute to the development and application process of Bézier curves, which are very useful in Computer Aided Design (CAD), CAGD and engineering, with different artificial intelligence-based algorithms and new tools. We see CAGD and other related fields as very bright and fruitful interdisciplinary fields that will culminate mathematician-engineer collaboration. This chapter presents a systematic review work referring to

[1] Dr. Öğr. Üyesi, Tez Eş Danışmanı, İstanbul Topkapı Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, sedakarateke@topkapi.edu.tr, Orcid: 0000-0003-1219-0115.

[2] Yüksek Lisans Öğrencisi, Zonguldak Bülent Ecevit Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Ana Bilim Dalı, rum.akalin@gmail.com

[3] Doç. Dr., Tez Eş Danışmanı, Zonguldak Bülent Ecevit Üniversitesi, Fen-Edebiyat Fakültesi, Matematik Bölümü, m.gumus@beun.edu.tr, Orcid: 0000-0001-7938-2918.

https://doi.org/10.58830/ozgur.pub165.c677

practice of Bézier curves theory and its solutions to the real-life problems in CAGD.

Introduction

Computer-aided geometric design (CAGD) is a branch using computers to create, modify, optimize, and analysis of a design. Computer-aided drafting (CAD) and computer-aided design and drafting (CADD) are the other terms to be used in literature. CAD is used throughout the engineering process from the conceptual design of 2-dimensional (2D) and 3-dimensional (3D) models and products to their assembly. It is also ideal for designing many objects such as automobiles, jewellery, furniture, and also household appliances. In addition, today, many CAD applications are made use of open and commercial sources to obtain images with advanced animation and rendering capabilities [1-5].

If we go back to the beginning of its mathematical roots, Casteljau came up with an idea to solve the problems which he faced at Citroën in 1958. He based the result of his work on a new interpolation theory in 1959. In this theory, interpolation is generalized in a different way according to both the Hermitian method and Lagrangian-style points. The approximation of the arcs of the polynomials, regardless of their degree, is provided and some polynomials are reconstructed to a certain degree using a sufficient number of points. In addition, a new smoothing method has been found and this method has been more effective than the one described in terms of the classical least squares method. This theory of Casteljau has been modified for use in computer-aided design/computer-aided manufacturing (CAD/CAM) and has succeeded in pushing the limits of flexibility and adjustability, especially in human-machine interaction. After this pioneering work, contributions to the literature by engineers and researchers have continued [6].

Indeed, the most important discoveries in the history of CAGD are Bézier curves and surfaces. These theories were developed independently by P. Bézier at Renault and P de Casteljau at Citroën. Whole theory of polynomial curves and surfaces in the Bernstein form is named after Bézier. Also, after a conference at the University of Utah in 1974, CAGD became a discipline [1,7].

Bézier curves have been used widely in computer graphics (CG), geometric design (GD), and also image processing and other related areas. Bézier curves need to be replaced with polygons (see Fig.1.1) first for some applications such as collision detection, rendering, optimization, and curve fitting etc [8].

Since Bézier curves can be defined by a recursive algorithm developed by P de Casteljau, we need to give a certain representation in the sense of theoretical background based on Bernstein polynomials.

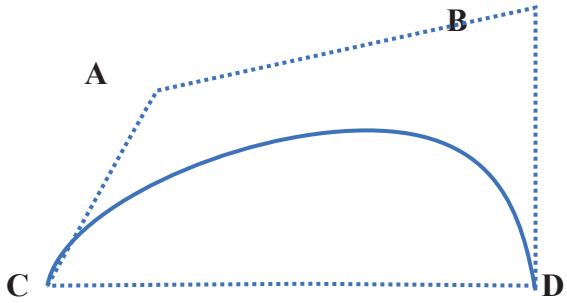

Figure 1. 1. A Bézier curve and its defining polygon ([9]).

Our work provides a systematic review related to use of Bézier curves theory and its CAGD applications to the real-life phenomenon.

This book chapter consists of five parts. In the introduction part, the historical process and also the process related to the first starting point of the Bézier theory are explained and the basic definitions and concepts to be used throughout the chapter are presented. In the second part, the relations between Bernstein polynomials and Bézier curves are examined. In the third part, which forms the core of the section, studies that have important contributions to the literature, including various applications of these curves to the field of computer-aided geometric design, are compiled. In section 4, result of the study that guides the reader is given. In the appendix A part; Python 3.10 programming language (64 bit) codes of Figure 1.2. have been shared with the readers.

1. Bernstein Polynomials

1.1. Definition ([1])

Bézier curves are defined by Bernstein polynomials as follows:

$$\mathcal{B}_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{1.1}$$

where the binomial coefficients are given by:

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!\,(n-i)!}, & \text{if } 0 \le i \le n \\ 0, & \text{else.} \end{cases} \tag{1.2}$$

Before we focus on the importance of Bernstein polynomials (see Fig.1.2 for the cubic case) to Bézier curves, we give some important properties fulfilling the followings:

- *Recursion*:

$$\mathcal{B}_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{1.3}$$

$$= \binom{n-1}{i} t^i (1-t)^{n-i} + \binom{n-1}{i-1} t^i (1-t)^{n-i} \tag{1.4}$$

$$= (1-t)\mathcal{B}_i^{n-1}(t) + t\mathcal{B}_{i-1}^{n-1}(t). \tag{1.5}$$

thus

$$\mathcal{B}_i^n(t) = (1-t)\mathcal{B}_i^{n-1}(t) + t\mathcal{B}_{i-1}^{n-1}(t) \tag{1.6}$$

is verified with $\mathcal{B}_0^0(t) = 1,\ \mathcal{B}_j^n(t) = 0, \text{for } j \notin \{0, \dots, n\}$.

- *Partition of unity*:

Using the well-known binomial theorem,

$$\sum_{j=0}^{n} \mathcal{B}_j^n(t) = \sum_{j=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} = [1 + (1-t)]^n = 1$$

$$\sum_{j=0}^{n} \mathcal{B}_j^n(t) = 1 \tag{1.7}$$

is obtained.

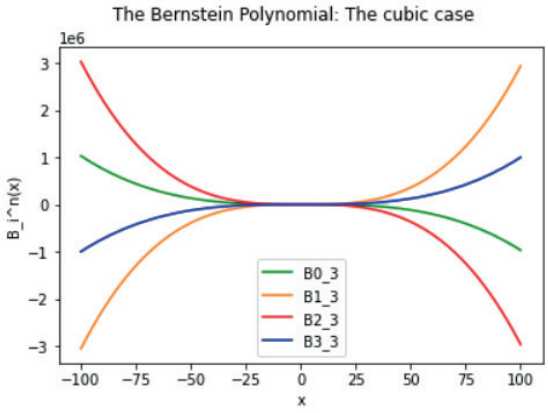

Figure 1. 2. The Bernstein polynomials: the cubic case ([33]-[34]).

- *Affine invarience:* while Bézier curves are invariant under affine transformations, they are not invariant under projective transformations. So, (1.7) verifies this property algebraically.
- *Convex hull property:* for the values $t \in [0,1]$, the Bernstein polynomials are nonnegative. This yields (1.7). However, for the values t outside of the interval $[0,1]$, the convex hull property does not work.

$$\sum_{i=0}^{n} \mathbf{b}_i \mathcal{B}_i^n(t) = \sum_{i=0}^{n} \mathbf{b}_i \mathcal{B}_i^n\left(\frac{u-a}{b-a}\right), \quad a, b \in \mathbb{R} \tag{1.8}$$

$$\mathbf{b}[\alpha_1 r + \alpha_2 s, t_2] = \alpha_1 \mathbf{b}[r, t_2] + \alpha_2 \mathbf{b}[s, t_2]; \ \alpha_1 + \alpha_2 = 1, \tag{1.9}$$

where t_2 is a blossom argument, the blossom $\mathbf{b}$ is affine with respect to its first argument, but it is also affine for the second one thanks to the symmetry property in (1.9) for any points $r, s \in \mathbb{R}$.

- *Endpoint interpolation:* let $\delta_{i,j}$ be the Kronecker delta function. As the results of the equations (1.7), $\mathcal{B}_i^n(0) = \delta_{i,0}$, and $\mathcal{B}_i^n(1) = \delta_{i,n}$.
- *Symmetry:* in words: there is no difference between labeling Bézier points as two different ordering such as $p_0, p_1, \ldots, p_n$ or

$p_n, p_{n-1}, \ldots, p_0$. Only the directions of these two curves make difference. In other words, we describe this situation using the below mathematical formula:

$$\sum_{j=0}^{n} \mathbf{b}_j \mathcal{B}_j^n(t) = \sum_{j=0}^{n} \mathbf{b}_{n-j} \mathcal{B}_j^n(1-t), \text{ for every } t \in [0,1]. \quad (1.10)$$

and using (1.1)

$$\mathcal{B}_j^n(t) = \mathcal{B}_{n-j}^n(1-t) \quad (1.11)$$

holds, so it is concluded that Bernstein polynonials are symmetric with respect to the points t and $1-t$.

- *Invariance under combinations:* for $\alpha_1 + \alpha_2 = 1$, we get linearity property as below:

$$\sum_{j=0}^{n}(\alpha_1 \mathbf{b}_j + \alpha_2 \mathbf{c}_j)\mathcal{B}_j^n(t) = \alpha_1 \sum_{j=0}^{n} \mathbf{b}_j \mathcal{B}_j^n(t) + \alpha_2 \sum_{j=0}^{n} \mathbf{c}_j \mathcal{B}_j^n(t). \quad (1.12)$$

1.1.1. Why Bernstein polynomials matter for the construction of Bézier curves?

Let us remember that Bézier curves can be written as blossom form (see Chapter 3.4 in [1]).

Since we can rewrite $t = (1-t).0 + t.1$,

$$\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i \mathcal{B}_i^n(t) \quad (1.13)$$

is obtained.

In a similar manner, the intermediate P de Casteljau points $\mathbf{b}_i^r$ can be considered with reference to Bernstein polynomials of order r:

$$\mathbf{b}_i^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} \mathcal{B}_j^r(t). \quad (1.14)$$

As we see from (1.14), the points $\mathbf{b}_i^r$ build upon the given Bézier points $\mathbf{b}_i$ and also these points create the Bézier curves by themselves. So, we can write a Bézier curve as below:

$$\mathbf{b}^n(t) = \sum_{i=0}^{n-r} \boldsymbol{b}_i^r(t) \mathcal{B}_i^{n-r}(t). \quad (1.15)$$

2. On Bézier curves and more

Based on the information we have given so far, it is not difficult to predict the shape of a curve formed by a Bézier polygon. Now we are ready to give the mathematical formula of a Bézier curve.

2.1. Definition ([9])

A Bézier curve $\wp(t)$ of degree n is defined as follows:

for every $t \in [0,1]$,

$$\wp(t) = \sum_{i=0}^{n} p_i J_{n,i}(t), \tag{2.1}$$

where

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{2.2}$$

is the Bézier or Bernstein basis or blending function.

We know that $J_{n,i}(t)$ is the ith nth-order Bernstein basis function. The degree of the Bernstein basis function is denoted by n, and also p_i are the control points.

There exists an other basis, called as B-spline basis (from Basis spline). Bernstein basis is included in this one as a spesific case. In [10], Gordon and Riesenfeld have implemented B-spline basis to the definition of curve.

Now, we give the definition of B-spline curve.

2.2. Definition ([9, 10])

Let $P(t)$ be the position vectors along the curve function with the parameter t, a B-spline curve is defined by

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t), \; t_{min} \le t < t_{max}, \; 2 \le k \le n+1 \tag{2.3}$$

where B_i, $N_{i,k}(t)$ are the position vectors of $n+1$ defining polygon vertices, and normalized B-spline basis functions, respectively.

2.3. Definition ([11], [17])

Cubic Bézier curves are described with four control points like p_0, p_1, p_2 and p_3 in a three-dimensional or higher dimensional space.

Let us write $\wp_{p_j,p_k,p_l}(t)$ to denote quadratic Bézier curve constructed with any three points $p_j, p_k,$ and p_l. Acting similarly, cubic Bézier curve denoted by $\wp_c(t)$ can be defined as an affine combination of the quadric ones as followings:

$$\wp_c(t) = (1-t)\,\wp_{p_0,p_1,p_2}(t) + t\wp_{p_0,p_1,p_2}(t), \quad 0 \le t \le 1. \tag{2.4}$$

The explicit form is:

$$\wp_c(t) = (1-t)^3\wp_0 + 3(1-t)^2 t\wp_1 + 3(1-t)t^2\wp_2 + t^3\wp_3, \quad 0 \le t \le 1. \tag{2.5}$$

3. Previous Reviews on Bézier Curves in Computer-Aided Geometric Design (CAGD)

In the literature, there are many systematic and wide-ranging studies dealing with "Bézier curves in CAGD" from different perspectives.

The theory of Bézier curves; not just computer graphics, animation, Computer-Aided Geometric Design (CAGD); it is also a promising comprehensive mathematical theory that is prominent in many popular areas of state of the art technology, such as shape preservation of curves and surfaces, robotics, image compression or font design, highway-railway design, satellite path planning, and even creation of 3D tensor product surface models ([11]-[14], [36]).

Here, we aim to focus on mostly CAGD applications of Bézier curves and their related modifications.

In CAGD systems, it is an effective way that fitting a Bézier curve to a set of data. Rushi and Yahya have used cubic Bézier curve function for curve fitting employing the least square method, which optimizes the parameters of the curve function. The Bézier curve is the most ideal of all other polynomial-based curves used to represent the complex and piecewise polynomial curve. Therefore, it is indeed entitled to a loan at CAGD. Authors have presented a curve fitting scheme relying on cubic Bézier curve with four steps and also they develop an effective algorithm to make approximation to the bitmap images (see Fig.3.1) ([15]-[16]).

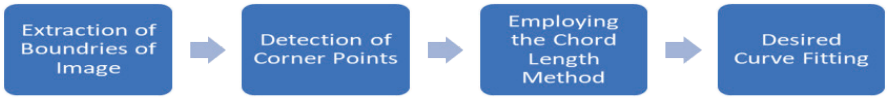

Figure 3. 2. A diagram for the curve fitting algorithm ([16]).

According to Bashir et al. [13], creating a curve or surface has a counterpart in CAGD. Cubic B-splines and Bézier curves are widely used in CAD and CAGD, but in some cases there are difficulties in obtaining the desired shape due to the structure of their polynomials. To overcome this challenge, shape parameters have come into play.

Sarfraz [18] has focused on the problem of fitting curves and surfaces for CAD. He has used a piecewise rational cubic interpolation to provide shape control of parametric curves. Here, his main idea is to be able to generalize Gordon's [19] blending-function method.

Chen and Wang [20] have extended a significant method standing on the Progressive Iterative Approximation (PIA) property of the univariate Normalized Totally Positive (NTP) bases. They have done this progress using triangular Bézier basis, triangular and rational Bézier surfaces since these kinds of surfaces play a vital role in CAGD in the sense of reverse engineering.

Wang and Rosen [21] have focused on CAD methods for constructing fabricated lattice structures to strengthen mechanical properties. Here, the authors have described the overall design process for the truss structure and discuss how solid modelling techniques and approaches such as parametric modelling, finite element methods and optimization have been applied to design the truss structure.

Mineur et al. have presented a new approach in their work [22]. According to this approach, a curve of appropriate shape is fitted to a set of data points. This technique is basically rest on the classical curve concept. The curve fitting techniques given here have made an important contribution to the literature as they are mathematically based on the preliminary shape characterization of the curve to be modelled and using a typical curve definition. In addition, a modification method has been developed to preserve the characteristic shape of the curve. According to this modification, an automatic fitting procedure is developed from a point set of data, based on the concepts of curve and arc length. Thus, it can be

implemented on surface modelling software and gives very effective results in car body shape design.

Li et al. [23] have introduced the method of approximating Bézier curve by the least square polygon (LSP), which is derived by optimizing the mean error between the approximating polygon and the Bézier curve.

Wang et al. [24], impressed by Mineur et al., have introduced a 3D Bézier curves and a 3D affine transformation of a uniform and scaling and a rotation. In other words, a kind of generalization of planar typical curves developed by Mineur et al. is also proposed. This contribution has germinated design and automotive applications supported with effective examples.

In [25], a cubic trigonometric Bézier curve (T-Bézier) -akin to the classical cubic Bézier curve- with two shape parameters has been introduced by Han et al. They have made the shape of the curve adjustable by changing the values of its parameters, while the control polygon remain unaffected. Thus, it is easy to adapt T-Bézier curves to CAD/CAGD/CAM systems using cubic Bézier curves.

In [26], a more practical definition is given for Beta-Bézier curves, and with this new definition, the properties of Beta-Bézier curves are more easily studied. For example, the convex hull property of Beta-Bézier curves has been provided. Thus, it has been shown that Beta-Bézier curves are not only carry all the essential properties of Bézier curves, such as the convex body property, recursive subdivision, B-spline conversion, and interpolation, but also have the ability to change the shape of a Bézier curve segment.

Moustafa et al. [27] have extended the thought of tension control proposed in [26] from curves to surfaces. Together with extending the work done to surfaces, an efficient rectangular mesh interpolation scheme using suggested patch types is also given. Thus, the proposed surface patches can be reshaped without moving the control points.

Fortes and Medina [28] have proposed a novel fitting method. This method aims to improve the previous fitting models from two ways: (1) obtaining the wireframe on which the fitting patch rests, by means of Bézier techniques, allowing the shape of the surface to be mounted to be expanded more harmoniously inside the hole; (2) adapting the fitting patch to a sufficiently general and non-cartesian wireframe to be able to securely collect information of the surface to be fitted.

As a recent work in [29], Astonkar has recommended a CAD generative network lean on the transformer, rested on an analogy between CAD processes and spoken language. Also, in this study, the author aims to create a learning-based deep engineering sketch as an important step towards parametric CAD model synthesis.

Again in a very recent paper in [30], Ramanantoanina and Hormann have concentrated on a generalization of Sánchez-Reyes's idea [31] related to periodic Bézier curves. They have created a difference about this generalization of Bézier curves to the rational medium by exploring the use of trigonometric binocular form in the concept of curve design. Further, it has been shown to provide more direct control over the shape of the curve and complements the usual shape control tools given by the periodic rational Bézier form (weights and control points).

The authors in [32] have aimed to find the minimum radius of a Bézier curve consisting of sixteen cubic Bézier curves on a bike path in Mălini, Romania. They have preferred to compare the procedure between hand-drawn horizontal curves and approach the trace using cubic Bézier curves and Python.

In the current study [35], Korotkiy at al. have been able to close a gap that existed in the literature of contemporary architecture with this study. In the architecture, smooth curves from nature and non-linear shapes have been extensively used. Thanks to state of the art calculation methods and digital modelling, architectural structures based on such nature curves can be replanned. From this point of view, certain conditions must be met for the practical use of a graphically defined irregular curve. It has been made possible to satisfy these necessary conditions by using a composite cubic Bézier curve. Moreover, a graphic-based analytical algorithm has been developed and in this way desired curves can be created.

4. Conclusion

The literature on Bézier curves in CAGD is summarized and synthesized in this book chapter. Our study contributes to the existing literature and knowledge repository and paves the way for further research. In conclusion, the theory of Bézier curves offers a long-term and promising future to all designers and researchers working in the fields of technology, geometry, science and industry, and CAGD.

REFERENCES

1. Farin, G. (2011). *Curves and Surfaces for CAGD: A Practical Guide (The Morgan Kaufmann Series in Computer Graphics) 5th Edition*: Morgan Kaufmann.
2. Narayan, K. L, K. Rao, M.K., and Sarcar, M.M.M. (2008). *Computer Aided Design and Manufacturing*. New Delhi: Prentice Hall of India.
3. Duggal, V. (2000). *Cadd Primer: A General Guide to Computer Aided Design and Drafting-Cadd, CAD*: Mailmax Pub.
4. Madsen, D. A. (2012). *Engineering Drawing & Design*. Clifton Park, New York: Delmar. p. 10.
5. Farin, G., Hoschek, J., and Kim, M.-S. (2002). *Handbook of computer aided geometric design [electronic resource]*: Elsevier.
6. Casteljau, d. F. d. P. (1986). *Mathematics and CAD-Shape mathematics and CAD*: Paris, Hermes Publishing.
7. Barnhill, R. and Riesenfeld, R. F. (1974). (Ed.). *Computer Aided Geometric Design*: Academic Press.
8. Li, Y., Zhang, M., Jin, W. *et al.* (2023). Approximating Bézier curves with least square polygons. *Vis Comput*. https://doi.org/10.1007/s00371-023-02806-0.
9. Rogers, D. F. and Adams, J. A. (1989). *Mathematical Elements for Computer Graphics*: McGraw-Hill.
10. Gordon, W. J. and Riesenfeld, R. (1974). *B-Spline Curves and Surfaces*: Academic Press, 5-126.
11. Raseli, S.S., Faisal, N.A.M.K and Mahat, N. (2022). The Construction of Cubic Bézier Curve, *Journal of Computing Research and Innovation (JCRINN)*, 7(2), 111-120.
12. Kilicoglu, S. and Senyurt, S. (2020). On the involute of the cubic Bézier curve by using matrix representation in E^3. *European Journal of Pure and Applied Mathematics*, 13(2), 216–226.
13. Bashir, U., Abbas, M., and Ali, J. M. (2013). The G2 and C2 rational quadratic trigonometric Bézier curve with two shape parameters with applications. *Applied Mathematics and Computation*, 219(2013), 10183–10197.
14. Abbas, M., Jamal, E., and Ali, J. M. (2011). Bézier curve interpolation constrained by a line. *Applied Mathematical Sciences*, 5(37), 1817–1832.
15. Zain, S.A.A.A.S.M., Misro, M. Y., and Miura K.T. (2022). Curve Fitting Using Generalized Fractional Bézier Curve. *Proceedings of CAD'22, Beijing, China, 77-8*.
16. Rusdi, N. A., and Yahya, Z. R. (2015). Reconstruction of generic shape with cubic Bézier using least square method. *AIP Conference Proceedings*, (pp. 050004-1-050004-6).

17. Burkardt, J. "*Forcing Bézier Interpolation*". Archived from the original on 2013-12-25.
18. Sarfraz, M. (1995). Curves and Surfaces for Computer Aided Design using C^2 Rational Cubic Splines, *Engineering with Computers*.11:94-102.
19. Gordon, W.J. (1971) Blending function methods of bivariate and multivariate interpolation and approximation, *SIAM J. Num. Anal*., 8, 158-177.
20. Chen, J. and Wang, G.-J. (2011). Progressive iterative approximation for triangular Bézier surfaces, *Computer-Aided Design*, 43(8), 889-895.
21. Wang, H. V., & Rosen, D. W. (2001). *Computer-aided design methods for the additive fabrication of truss structure* (Master's thesis, School of Mechanical Engineering, Georgia Institute of Technology).
22. Mineur Y., Lichah T., Castelain J.M., and Giaume, H. A. (1998). Shape controled fitting method for Bézier curves. *Comput Aided Geom Design*. 15(9):879–91.
23. Li, Y., Zhang, M., Jin, W. *et al.* (2023). Approximating Bézier curves with least square polygons. *Vis Comput* .
24. Wang, A., He, C., Zheng, J., and Zhao, G. (2023). 3D Class A Bézier curves with monotone curvature, *Computer-Aided Design*. 159-103501.
25. Han, X., Ma, Y., and Huang, X. (2009). The cubic trigonometric Bézier curve with two shape parameters, *Applied Mathematics Letters*. 22, 226–231.
26. Cheng, F., Kazadi, A., and Lin, A. (2020). *Beta-Bézier curves*. CAD'20. https://doi.org/10.14733/cadconfp.2020.343-347.
27. Moustafa, S., Kazadi, A, Cheng, F., Lai, S., and Lin, A.J., *Beta-Bézier Surfaces*, CAD'23. doi: 10.14733/cadconfP.2023.xxx-yyy.
28. Fortes, M.A. and Medina, E. (2022). Fitting missing data by means of adaptive meshes of Bézier curves. *Mathematics and Computers in Simulation*. 191.33-48.
29. Astonkar, D. V. (2023). For Computer- Aided Design, a Deep Engineering Sketch Generative Network, *Eur. Chem. Bull*.1703-1712.
30. Ramanantoanina, A. and Hormann, K. (2023). Shape control tools for periodic Bézier curves, *Computer Aided Geometric* Design, 103.
31. Sánchez-Reyes, J. (2009). Periodic Bézier curves. *Comput. Aided Geom. Des*. 26, 989–1005.
32. ŞOMÎTCĂ, I.-A., DEACONU, Ş.-E., and ŞOMÎTCĂ, S.-A. (2023). A new Procedure of Computing the Minimum Radii of Bézier Curves and Applications in Designing a Bike Trail. *Journal of Applied Computer Science & Mathematics*. 1(17), No. 35, Suceava.
33. Öğrekçi, S. (2023, June). Retrieved from https://sogrekci.com/ders-notu/python-ve-bilimsel-hesaplama/42-grafik-cizimi/, (2023, June).
34. Retrieved from https://pybilim.wordpress.com/2014/01/27/matplotlib-3-bitisik-sekiller-yazi-ve-diger-suslemeler/, (2023, June).

35. Korotkiy, V.A., Usmanova, E.A., Khmarova, L.I. (2023). The Design of Architectural Forms Based on Irregular Curves. In: Radionov, A.A., Ulrikh, D.V., Timofeeva, S.S., Alekhin, V.N., Gasiyarov, V.R. (eds) Proceedings of the 6th International Conference on Construction, Architecture and Technosphere Safety. ICCATS 2022. Lecture Notes in Civil Engineering, vol 308. Springer, Cham. https://doi.org/10.1007/978-3-031-21120-1_29.

36. Akalin, R., Karateke, S. , Gümüş, M. , and Zontul, M. (02.06.2022). A Review on Bézier Curves Applications in Computer Aided Geometric Design (CAGD). INTERNATIONAL GRADUATE RESEARCH SYMPOSIUM IGRS'22. (Oral Presentation), (Publication No:7719997), file:///C:/Users/CarryUp/Downloads/igrs22-program-day-2_v5.htm

Appendix A. Cubic Bernstein Polynomials Python 3.10 (64 bit) Programming Language Codes:

In [76]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
#Bi_n=sum(i,n,x)(n,i)*(1-t)**(n-i)*t**i
x = np.linspace(-100, 100, 100)
B0_3=(1-x)**3
plt.plot(x,B0_3,"g", label="B0_3")

x = np.linspace(-100, 100, 100)
B1_3=3*(1-x)**2*x
plt.plot(x,B1_3,"tab:orange", label="B1_3")

x = np.linspace(-100, 100, 100)
B2_3=3*(1-x)*x**2
plt.plot(x,B2_3,"r", label="B2_3")

x = np.linspace(-100, 100, 101)
B3_3=x**3
plt.plot(x,B3_3,"b", label="B3_3")


plt.xlabel("x")
plt.ylabel("B_i^n(x)")
plt.suptitle("The Bernstein Polynomial: The cubic case")
plt.legend(loc="best")
plt.show()
```