

## Graphics Evolutionary Computations in Higher Order Parametric Bezier Curves

Monday Eze<sup>1,\*</sup>, Charles Okunbor<sup>2</sup>, Deborah Aleburu<sup>3</sup>, Olubukola Adekola<sup>1</sup>, Ibrahim Ramon<sup>4</sup>,  
Nneka Richard-Nnabu<sup>5</sup>, Oghenetega Avwokuruaye<sup>6</sup>, Abisola Olayiwola<sup>3</sup>, Rume Yoro<sup>7</sup> and  
Esomu Solomon<sup>8</sup>

<sup>1</sup>Department of Computer Science, Babcock University, Ogun, Nigeria

<sup>2</sup>Department of Computer Science, Admiralty University, Delta, Nigeria

<sup>3</sup>Department of Computer Science, Caleb University, Imota, Lagos, Nigeria

<sup>4</sup>Department of Computer Science Education, Federal College of Education (Tech), Akoka, Lagos, Nigeria

<sup>5</sup>Department of Computer Science, Alex Ekwueme Federal University, Ebonyi, Nigeria

<sup>6</sup>Department of Cyber Security, Admiralty University, Delta, Nigeria

<sup>7</sup>Department of Computer Science, School of ICT, Delta State Polytechnic, Ogwashi-Uku, Delta, Nigeria

<sup>8</sup>Department of Computer Science, National Open University, Nigeria

\*Corresponding Author: Monday Eze. Email: ezem@babcock.edu.ng

Received: 10 June 2021; Accepted: 14 July 2021

**Abstract:** This work demonstrates in practical terms the evolutionary concepts and computational applications of Parametric Curves. Specific cases were drawn from higher order parametric Bezier curves of degrees 2 and above. Bezier curves find real life applications in diverse areas of Engineering and Computer Science, such as computer graphics, robotics, animations, virtual reality, among others. Some of the evolutionary issues explored in this work are in the areas of parametric equations derivations, proof of related theorems, first and second order calculus related computations, among others. A Practical case is demonstrated using a graphical design, physical hand sketching, and programmatic implementation of two opposite-faced handless cups, all evolved using quadratic Bezier curves. The actual drawing was realized using web graphics canvas programming based on HTML 5 and JavaScript. This work will no doubt find relevance in computational researches in the areas of graphics, web programming, automated theorem proofs, robotic motions, among others.

**Keywords:** Parametric curve; computer graphics; bezier curve; blending function; robotics

### 1 Introduction

The importance of parametric curves in Computer Graphics [1], Computer Aided Design (CAD), Animation, Robotics and a number of other areas of computing cannot be overemphasized. The focus of this research is on a special parametric curve known as Bezier Curves [2]. The concept of Bezier Curves originated from the works of Piere Biezier [3] who is renowned to have applied it in design of car chassis. A Bezier curve is a smooth parametric curve, controlled by the shape of a defining polygon. The



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

defining polygon consists of a number of points, known as control points [4], which make up an enclosure called Convex Hull [5]. The order of a Bezier curve is equivalent to the number of control points.

For a Bezier Curve of control points  $P_0, P_1, \dots, P_N$ , the points  $P_0$  and  $P_N$  are called end points, and the rest of the points from  $P_1$  to  $P_{N-1}$  may lie outside the curve. Given that a Bezier curve is of order  $X$ , then its degree is defined as  $X - 1$ . For instance, a quadratic Bezier curve is of degree two, a cubic Bezier curve is of degree three, while a quartic Bezier curve is of degree four respectively. Computations involving Bezier curves of higher degree are said to be very expensive [6], thus, many researches tend to be limited to quadratic and cubic curves. The concept of degree elevation [7] has to do with the conversion of a Bezier curve of degree  $X$  to an entirely new one of degree  $X + 1$  with the same shape. The De Casteljau's algorithm [8] is one of the recursive techniques used for Bernstein polynomial evaluations.

Recent advances in Bezier curves have witnessed its applications in robotics, especially in the area of path planning and generation [9]. Engineering researches in the area of multi-material flow [10] make use of the parametric curves. The Bezier parametric curves are also utilized in creating aesthetic automobile surfaces and in evaluating reflected images in ocean research [11]. Among other applications, some of the most common are in Computer Graphics, Animation [12], and Virtual reality [13]. These and many other real-life applications of Bezier curves make adequate use of the foundational concepts. Thus, this work demystifies evolutionary concepts that touch on theorems, proofs, algorithms, and computations in parametric Bezier curves relevant to computer graphics.

This work is organized into ten sections. First is an introduction, and then a computational exploration of parametric functions. This is followed by studies on blending function derivations, Bezier matrix, Bezier curve sketching, curve join and foundational proofs of two important theorems. The concluding part of this work focused on derivatives, system implementations, and a conclusion.

## 2 Parametric Exploration

The general parametric equation of a Bezier curve is given by Eq. (1).

$$Z(x) = \sum_{k=0}^n P_k J_{N,k}(x), \quad 0 \ll x \ll 1 \quad (1)$$

where,  $Z(x)$  is any point on the curve,  $P_k$  is the  $k$ th Control point of the Bezier Curve,  $J_{N,k}(x)$  is the Bernstein or Blending function,  $N$  is the degree of the Bezier Curve in question. The Blending function is the control function, also referred to in a number previous researches as the Bezier basis functions. The components of  $Z(x)$  are further defined as follows:

$$J_{N,k}(x) = C_k^N x^k (1-x)^{N-k} \quad (2)$$

$$C_k^N = \frac{N!}{k!(N-k)!} \quad (3)$$

$$N! = (1).(2).(3) \dots N \quad (4)$$

It can be deduced from the "Eqs. (1)–(4)" that the Parametric equation  $Z(x)$  is the sum of the products of Blending functions  $J$  and the corresponding control points  $P$ .

## 3 Blending Function Derivation

It is necessary at this stage to demonstrate the derivation of higher order blending functions. A 5th order will be used as a case study.

Given a Bezier Curve of 5th Order, one important task is to derive all the blending functions [14]. Since the Order of a Bezier Curve is same as the number of control points, it implies that a Bezier curve of 5th order has a total of five (5) control points. The relationship between the Order and the Degree of a curve is given by Eq. (5):

$$\text{Degree} = \text{Order} - 1 \tag{5}$$

Thus, for a 5th order case, the degree is  $N = 5 - 1$ , which is 4. The Blending functions are therefore computed as follows:

$$J_{N,k}(x) = C_k^N x^k (1-x)^{N-k} = \frac{N!}{k!(N-k)!} x^k (1-x)^{N-k}$$

$$J_{4,0}(x) = C_0^4 x^0 (1-x)^{4-0} = \frac{4!}{0!(4-0)!} x^0 (1-x)^{4-0}$$

Therefore,

$$J_{4,0}(x) = \frac{4!}{0! 4!} x^0 (1-x)^{4-0} = (1-x)^4 \tag{6}$$

$$J_{4,1}(x) = C_1^4 x^1 (1-x)^{4-1} = \frac{4!}{1!(4-1)!} x^1 (1-x)^{4-1}$$

Therefore,

$$J_{4,1}(x) = \frac{4!}{1! 3!} x^1 (1-x)^3 = 4x(1-x)^3 \tag{7}$$

$$J_{4,2}(x) = C_2^4 x^2 (1-x)^{4-2} = \frac{4!}{2!(4-2)!} x^2 (1-x)^{4-2}$$

Therefore:

$$J_{4,2}(x) = \frac{4!}{2! 2!} x^2 (1-x)^2 = 6x^2(1-x)^2 \tag{8}$$

$$J_{4,3}(x) = C_3^4 x^3 (1-x)^{4-3} = \frac{4!}{3!(4-3)!} x^3 (1-x)^{4-3}$$

Therefore:

$$J_{4,3}(x) = \frac{4!}{3! 1!} x^3 (1-x)^1 = 4x^3(1-x) \tag{9}$$

$$J_{4,4}(x) = C_4^4 x^4 (1-x)^{4-4} = \frac{4!}{4!(4-4)!} x^4 (1-x)^{4-4}$$

Therefore:

$$J_{4,4}(x) = \frac{4!}{4! 0!} x^4 (1-x)^0 = x^4 \tag{10}$$

Substituting all the Blending Functions into the general Eq. (1), the exact Parametric equation for a 4-degree Bezier Curve is therefore given by:

$$Z(x) = P_0J_{4,0}(x) + P_1J_{4,1}(x) + P_2J_{4,2}(x) + P_3J_{4,3}(x) + P_4J_{4,4}(x) \quad (11)$$

Therefore,

$$Z(x) = P_0(1-x)^4 + P_14x(1-x)^3 + P_26x^2(1-x)^2 + P_34x^3(1-x) + P_4x^4 \quad (12)$$

#### 4 Derivation of Bezier Matrix

The Bezier Matrix derivation will be demonstrated using a 4th degree case. First, the Blending functions are written in full polynomial format, and then converted to twisted coefficient formats as follows:

$$\begin{aligned} J_{4,0}(x) &= (1-x)^4 = (1-x)(1-x)(1-x)(1-x) = (1-2x+x^2)(1-2x+x^2) \\ &= 1-4x+6x^2-4x^3+x^4 \end{aligned}$$

In twisted coefficient format, this gives

$$J_{4,0}(x) = x^4 - 4x^3 + 6x^2 - 4x + 1 \quad (13)$$

$$\begin{aligned} J_{4,1}(x) &= 4x(1-x)^3 = 4x(1-x)(1-2x+x^2) = 4x(1-2x+x^2-x+2x^2-x^3) \\ &= 4x(1-3x+3x^2-x^3) = 4x-12x^2+12x^3-4x^4 \end{aligned}$$

In twisted coefficient format, this gives:

$$J_{4,1}(x) = -4x^4 + 12x^3 - 12x^2 + 4x + 0 \quad (14)$$

$$J_{4,2}(x) = 6x^2(1-x)^2 = 6x^2(1-2x+x^2) = 6x^2-12x^3+6x^4$$

In twisted coefficient format, this gives:

$$J_{4,2}(x) = 6x^4 - 12x^3 + 6x^2 + 0x + 0 \quad (15)$$

$$J_{4,3}(x) = 4x^3(1-x) = 4x^3-4x^4$$

In twisted coefficient format, this gives:

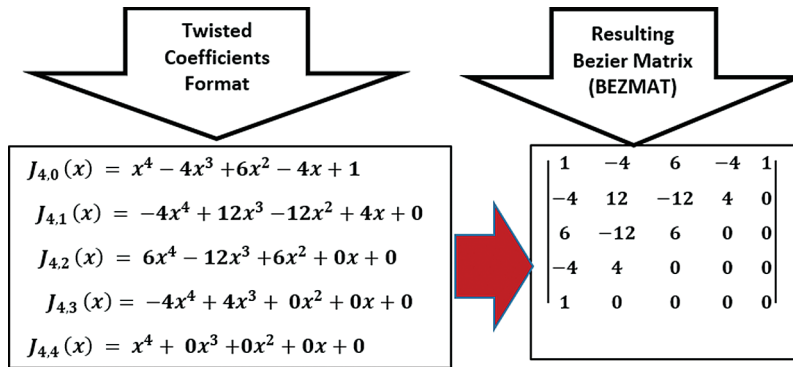
$$J_{4,3}(x) = -4x^4 + 4x^3 + 0x^2 + 0x + 0 \quad (16)$$

$$J_{4,4}(x) = x^4$$

In twisted coefficient format, this gives:

$$J_{4,4}(x) = x^4 + 0x^3 + 0x^2 + 0x + 0 \quad (17)$$

The Bezier Matrix (BEZMAT) is generated by collating all the twisted coefficients of the Blending Functions as shown in Fig. 1:



**Figure 1:** Sample of generated bezier matrix

Thus, the Parametric Equation in Bezier Matrix format is given by:

$$P(x) = [x^4 x^3 x^2 x 1] \cdot BEZMAT \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \tag{18}$$

where, BEZMAT is the Bezier Matrix.

### 5 Bezier Curve Sketching

Suppose a Bezier Curve has four Control Points given by B0 [1, 1], B1 [4, 4], B2 [7, 3] and B3 [6, 1]. The first computational task is to determine six points lying in the curve. The second task is to sketch the curve. Since there are four control points, it implies that the curve is of Order = 4, and thus, of Degree = 3, which is a Cubic Bezier Curve. The Parametric Equation is given by:

$$Z(k) = B_0(1 - k)^3 + B_1 3k(1 - k)^2 + B_2 3k^2(1 - k) + B_3 k^3 \tag{19}$$

To determine the points lying on the curve, first we select six random points (RANPO) which satisfy the inclusion condition  $0 << k << 1$ .

$$Let RANPO = [0, 0.2, 0.4, 0.5, 0.8, 1.0] \tag{20}$$

Substituting the values of the Control Points B0, B1, B2, B3 and B4 in the Parametric equation, we have:

$$Z(k) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - k)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3k(1 - k)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3k^2(1 - k) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} k^3 \tag{21}$$

For Parameter  $k = 0$ :

$$Z(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 0)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(0)(1 - 0)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(0)^2(1 - 0) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0)^3 = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$$

For Parameter  $k = 0.2$ :

$$\begin{aligned} Z(0.2) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 0.2)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(0.2)(1 - 0.2)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(0.2)^2(1 - 0.2) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.2)^3 \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.512) + \begin{bmatrix} 4 \\ 4 \end{bmatrix} (0.384) + \begin{bmatrix} 7 \\ 3 \end{bmatrix} (0.096) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.008) \\ &= \begin{bmatrix} 0.512 \\ 0.512 \end{bmatrix} + \begin{bmatrix} 1.536 \\ 1.536 \end{bmatrix} + \begin{bmatrix} 0.672 \\ 0.288 \end{bmatrix} + \begin{bmatrix} 0.048 \\ 0.008 \end{bmatrix} = \begin{bmatrix} 2.768 \\ 2.344 \end{bmatrix} \end{aligned}$$

For Parameter  $k = 0.4$

$$\begin{aligned} Z(0.4) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 0.4)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(0.4)(1 - 0.4)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(0.4)^2(1 - 0.4) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.4)^3 \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.216) + \begin{bmatrix} 4 \\ 4 \end{bmatrix} (0.432) + \begin{bmatrix} 7 \\ 3 \end{bmatrix} (0.288) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.064) \\ &= \begin{bmatrix} 0.216 \\ 0.216 \end{bmatrix} + \begin{bmatrix} 1.728 \\ 1.728 \end{bmatrix} + \begin{bmatrix} 2.016 \\ 0.864 \end{bmatrix} + \begin{bmatrix} 0.384 \\ 0.064 \end{bmatrix} = \begin{bmatrix} 4.344 \\ 2.872 \end{bmatrix} \end{aligned}$$

For Parameter  $k = 0.5$

$$\begin{aligned} Z(0.5) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 0.5)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(0.5)(1 - 0.5)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(0.5)^2(1 - 0.5) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.5)^3 \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.125) + \begin{bmatrix} 4 \\ 4 \end{bmatrix} (0.375) + \begin{bmatrix} 7 \\ 3 \end{bmatrix} (0.375) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.125) \\ &= \begin{bmatrix} 0.125 \\ 0.125 \end{bmatrix} + \begin{bmatrix} 1.500 \\ 1.500 \end{bmatrix} + \begin{bmatrix} 2.625 \\ 1.125 \end{bmatrix} + \begin{bmatrix} 0.750 \\ 0.125 \end{bmatrix} = \begin{bmatrix} 5.000 \\ 2.875 \end{bmatrix} \end{aligned}$$

For Parameter  $k = 0.8$

$$\begin{aligned} Z(0.8) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 0.8)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(0.8)(1 - 0.8)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(0.8)^2(1 - 0.8) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.8)^3 \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.008) + \begin{bmatrix} 4 \\ 4 \end{bmatrix} (0.096) + \begin{bmatrix} 7 \\ 3 \end{bmatrix} (0.384) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (0.512) \\ &= \begin{bmatrix} 0.008 \\ 0.008 \end{bmatrix} + \begin{bmatrix} 0.384 \\ 0.384 \end{bmatrix} + \begin{bmatrix} 2.688 \\ 1.152 \end{bmatrix} + \begin{bmatrix} 3.072 \\ 0.512 \end{bmatrix} = \begin{bmatrix} 6.152 \\ 2.056 \end{bmatrix} \end{aligned}$$

For Parameter  $k = 1.0$

$$\begin{aligned} Z(1.0) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1 - 1)^3 + \begin{bmatrix} 4 \\ 4 \end{bmatrix} 3(1)(1 - 1)^2 + \begin{bmatrix} 7 \\ 3 \end{bmatrix} 3(1)^2(1 - 1) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (1)^3 \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.0) + \begin{bmatrix} 4 \\ 4 \end{bmatrix} (0.0) + \begin{bmatrix} 7 \\ 3 \end{bmatrix} (0.0) + \begin{bmatrix} 6 \\ 1 \end{bmatrix} (1.0) = \begin{bmatrix} 6 \\ 1 \end{bmatrix} \end{aligned}$$

The sketch of the curve is shown in Fig. 2. The control points are  $B_0, B_1, B_2,$  and  $B_3$ . The curve is shown in red colour, and the six points estimated to lie on the curve are  $E_0, E_1, E_2, E_3, E_4$  and  $E_5$ . The enclosure  $B_0 B_1 B_2 B_3 B_0$  constitutes the convex hull.

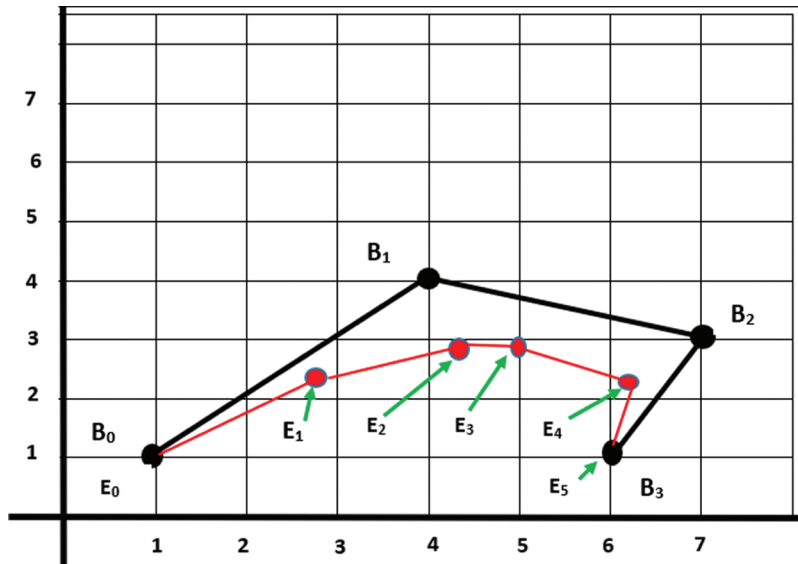


Figure 2: A sample cubic bezier curve

### 6 Curve Join Condition

In a number of scientific applications, there are cases where two or more parametric curves need to be joined to form a single one. This process is tagged curve continuity [15]. Imagine two separate Bezier Curves P and Q shown in the left-hand side (LHS) of Fig. 3, with  $P_0$  and  $Q_0$  as starting points of the curves, and  $P_1$  and  $Q_1$  as the end points respectively. The two separate curves are joined to form a single curve shown in the right-hand side (RHS) of the figure.

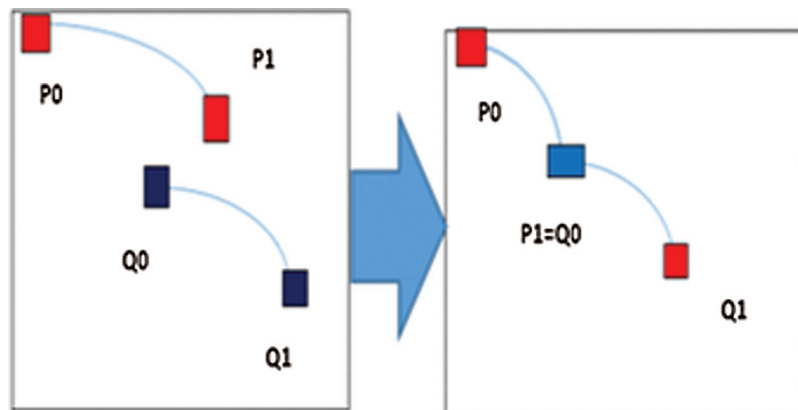


Figure 3: Demonstration of curve joining

For curve continuity to hold, some computational conditions must be met. These are known as zero-order, first order, and third order parametric continuities respectively as will be discussed next.

First and foremost, for any two curves to P and Q to be joined, such that P is followed by Q, then the end point of curve P must be the beginning point of curve Q. This is known as the zero-order parametric continuity condition [16] as demonstrated in the RHS of Fig. 3. The equation is given by

$$P(1) = Q(0) \quad (22)$$

Secondly, suppose the two curves P and Q are of degree N, then for the join to be smooth, the first-order parametric condition must hold. The conditional equation is a derivative [17] given by:

$$P^1(1) = Q^1(0) \quad (23)$$

Thus, the first-order parametric continuity condition [18] ensures the smoothness of a joint curve. The implication of this rule is that the tangent at the end point of first curve P is the same as the tangent at the first point of the second curve Q. As earlier proved in this research, the first derivatives of P at end points are given by the equations:

$$P^1(0) = n(P_1 - P_0) \quad (24)$$

$$P^1(1) = n(P_N - P_{N-1}) \quad (25)$$

Finally, second order parametric continuity condition [19] states that second derivatives at the end point of the first curve P is the same as that of the beginning point of second curve Q. Mathematically:

$$P^{II}(1) = Q^{II}(0) \quad (26)$$

Again, the formula for the second derivatives at the beginning and end points are given by:

$$P^{II}(0) = N(N - 1)(P_2 - 2P_1 + P_0) \quad (27)$$

$$P^{II}(1) = N(N - 1)(P_N - 2P_{N-1} + P_{N-2}) \quad (28)$$

Research has shown that whenever a curve satisfies the second order parametric continuity, it also automatically satisfies the first order continuity [20].

## 7 Foundational Proofs

A number of foundational theorems of Bezier Curves are of key applications to Computer Graphics. Two of these theorems to be proved in this work are the partition of unity theorem and the symmetry theorem respectively.

### 7.1 Partition of Unity Theorem

The partition of unity theorem [21] states that the sum of the Bernstein Polynomial of degree N is equal to unity. Mathematically:

$$\sum_{k=0}^N B_{k,N}(t) = 1, \quad \text{where } t \in [0, 1]. \quad (29)$$

Proof:

Obviously, the trivial statement  $[(1 - t) + t] = 1$  is always true for every parameter t. Substituting this statement into a general Binomial Eq. (29), we have:



$$1 = (1)^N = [(1 - t) + t]^N = \sum_{k=0}^N \binom{N}{k} (1 - t)^{N-k} (t)^k = \sum_{k=0}^N B_{k,N}(t)$$

which completes the proof.

### 7.2 Bernstein Symmetry Theorem

The Bernstein Symmetry Theorem [22] states that every Bernstein Polynomial follows the mathematical symmetry as follows:

$$B_{N-k,N}(x) = B_{k,N}(1 - x) \text{ for } k \in [0, 1] \tag{30}$$

Expanding the LHS of the symmetry Eq. (30) gives:

$$B_{N-k,N}(x) = \frac{N!}{(N - (N - k))!(N - k)!} (1 - x)^{(N-(N-k))} x^{N-k}$$

Therefore,

$$B_{N-k,N}(x) = \frac{N!}{k!(N - k)!} (1 - x)^k x^{N-k} \tag{31}$$

Expanding the RHS of the symmetry Eq. (30) gives:

$$B_{k,N}(1 - x) = \frac{N!}{(N - k)! k!} (1 - (1 - x))^{N-k} (1 - x)^k$$

Therefore,

$$B_{k,N}(1 - x) = \frac{N!}{(N - k)! k!} x^{N-k} (1 - x)^k \tag{32}$$

By simple comparison, it is clear that “Eqs. (31) and (32)” are same. Thus, the theorem is proved.

### 8 Derivation of Derivatives

Derivatives of parametric curves is of key importance, due to its application in Computer Graphics and other areas of computing. Research has shown that the derivative of Bezier Curve is also in itself, another Bezier curve [23]. An unambiguous demonstration of the derivation process beginning with product rule of Calculus is as follows:

$$\begin{aligned} \frac{d}{dx} B_{k,N}(x) &= \frac{N!}{(N - k)! k!} \frac{d}{dx} (1 - x)^{N-k} (x)^k \tag{33} \\ &= \frac{N!}{(N - k)! k!} (-(N - k)(1 - x)^{N-k-1} (x)^k + (1 - x)^{N-k} .k.x^{k-1}) \\ &= \frac{-(N - k)N!}{(N - k)! k!} (1 - x)^{N-k-1} (x)^k + \frac{N!k}{(N - k)! k!} (1 - x)^{N-k} .x^{k-1} \\ &= \frac{-N(N - 1)!}{(N - k - 1)! k!} (1 - x)^{N-1-k} (x)^k + \frac{N(N - 1)!}{(((N - 1) - (k - 1))!(k - 1)!)} (1 - x)^{(N-1)-(k-1)} .x^{k-1} \\ &= -N(B_{k,N-1}(x)) + N(B_{k-1,N-1}(x)) \end{aligned}$$

Therefore,

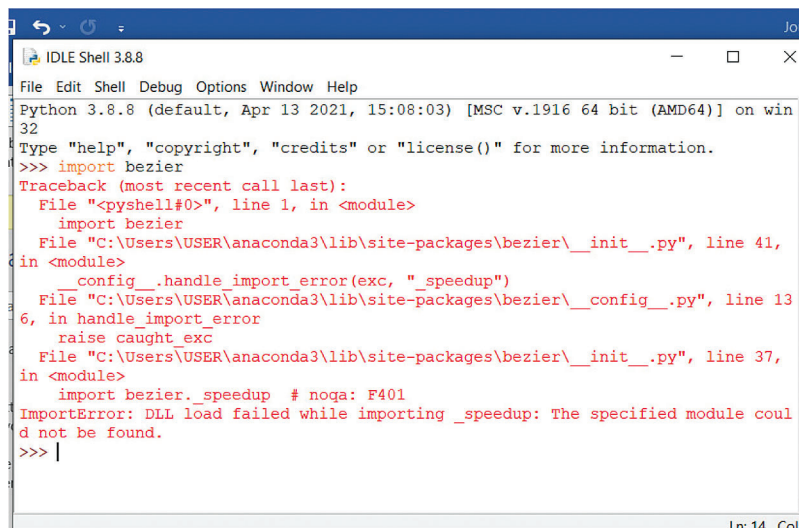
$$\frac{d}{dx}B_{k,N}(x) = N(B_{k-1,N-1}(x) - B_{k,N-1}(x)) \quad (34)$$

which is the formula for first derivative.

## 9 System Implementation

Before concluding this work, it is necessary to demonstrate the generation of Bezier Curves and applications in computer graphics in programmatic terms. A number of programming languages have application programming interface (API) or re-usable modules [24] specifically available for Bezier Curves related graphical designs and implementations. For instance, in Python the Bezier module is installed using the command `pip install Bezier` [25], and can be invoked into the programming environment using the command `import Bezier` [26].

However, Python versions 3.7 to 3.8 may bring “DLL Load Failed while importing `_Speedup`” error message in attempt to import Bezier as shown in Fig. 4, especially due to failure in pip installation process.



```

IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import bezier
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import bezier
  File "C:\Users\USER\anaconda3\lib\site-packages\bezier\_init_.py", line 41,
in <module>
    _config_.handle_import_error(exc, "_speedup")
  File "C:\Users\USER\anaconda3\lib\site-packages\bezier\_config_.py", line 13
6, in handle_import_error
    raise caught_exc
  File "C:\Users\USER\anaconda3\lib\site-packages\bezier\_init_.py", line 37,
in <module>
    import bezier._speedup # noqa: F401
ImportError: DLL load failed while importing _speedup: The specified module coul
d not be found.
>>> |

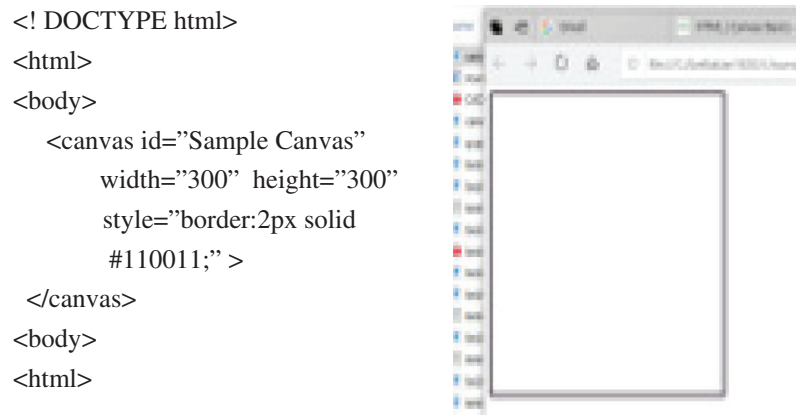
```

**Figure 4:** Load failed error message in python

This anomaly is common in a number of Python integrated development environment (IDEs) such as Spider, Jupyter and PyCham used within Anaconda [27].

One very feasible alternative is the use of Canvas tools [28] in web graphics. The hypertext markup language (HTML) canvas can be used in conjunction with JavaScript [29] to build complex graphical objects in a bottom-up approach, beginning from simple parametric Bezier curves as components. The canvas element is a graphical container, which is driven by Javascripts to generate graphical kernels such as paths, boxes, circles, text, images, images, and many others.

By its nature, a canvas is borderless and empty until it is enabled with Javascript. This is achieved by specifying an `<id attribute>` tag to refer it in a script. Furthermore, the width and height attributes are used to define the size of the canvas, and then the style attribute is used to add a border. The sample code snippet shown in Fig. 5 was saved as a HTML file called `canvas.html` and executed in a browser to give the output on the right-hand side.



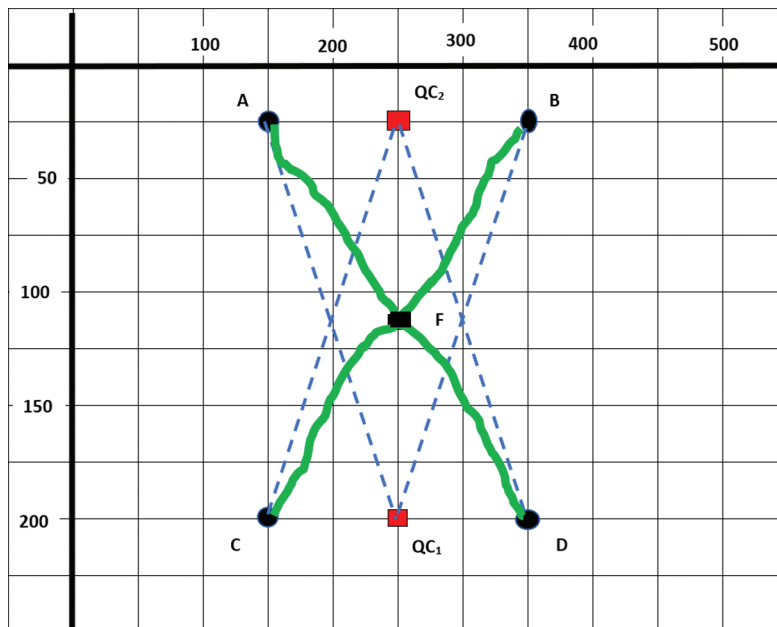
**Figure 5:** Boarder drawing canvas code

The canvas functions for drawing quadratic and cubic Bezier curves are *quadraticCurveTo* and *bezierCurveTo* respectively.

The syntax for creating a quadratic Bezier is *quadraticCurveTo (ctx, cty, a, b)*; where (ctx, cty) represent the coordinate of the control point, (a, b) is the coordinate of the end point, whereas the current point is achieved using the *moveTo* command.

In a similar way, since the cubic Bezier has two control points, the syntax for creating it is given by *bezierCurveTo(ctx1, cty1, ctx2, cty2, a, b)*; where (ctx1, cty1) and (ctx2, cty2) represent the coordinates of the two control points.

The practical demonstration in this work uses HTML 5 Canvas in conjunction with Javascript for programming to implement the physical design shown in Fig. 6. The curve implementation is limited to use of quadratic Bezier curves only. The aim is to use quadratic Bezier curves to create handless twin-cups. One of the cups is standing up, while the second is turned upside down.

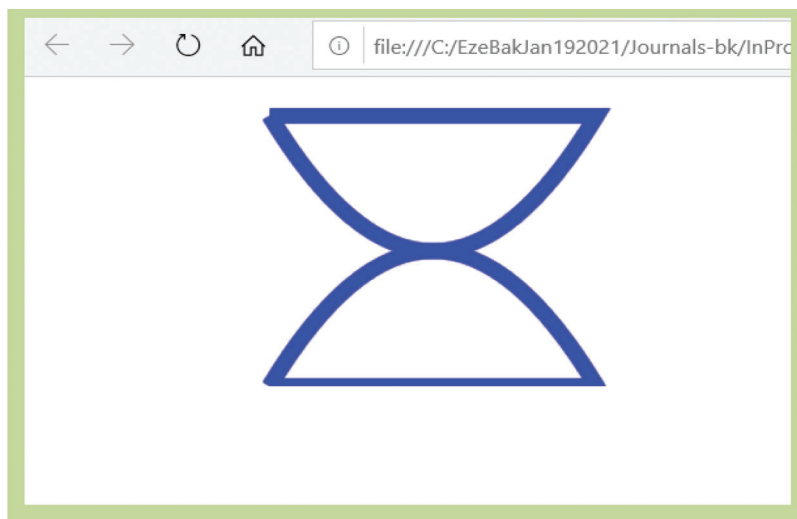


**Figure 6:** Physical design of bezier curves

A careful look at the diagram shows that the design space is of dimension  $500 \times 200$ . The coordinate points visibly marked are A (150, 25), B (350, 26), C (150, 200), D (350, 200), F (250, 125), QC1 (250, 200) and QC2 (250, 25).

The points QC1 and QC2 which are marked in RED colour are the control points-QC1 is the control point for the upper quadratic Bezier curve while QC2 is the control point for the lower quadratic curve. The point A and B are the end points of the upper quadratic curve, while C and D are the end points of the lower curve.

The upper cup is ABF while the lower cup turned up-side down is CFD. Both have been sketched using a handheld pencil, all of which are to be generated using Bezier curves. Both cups touch at point F. The program output is shown in Fig. 7, while the canvas program source code is shown in Section 9.1.



**Figure 7:** Output of program runs

### 9.1 Source Code

The source code is as follows:

```
<!DOCTYPE HTML>
<! Bezier Curve Design Sample Project. Implemented in Canvas>
<html>
<head>
<style>
  body {
    margin: 0px;
    padding: 0px;
  }
</style>
</head>
<body>
```

```
<canvas id="BezierCurveDesign" width = "500" height = "200"></canvas>
<script>
var canvas = document.getElementById('BezierCurveDesign');
var context = canvas.getContext('2d');

context.beginPath();
context.moveTo(150, 25);
context.lineTo(350, 25);
// First Quadratic curve
context.quadraticCurveTo(250, 200, 150,25);

context.moveTo(150, 200);
context.lineTo(350, 200);
// Second Quadratic curve
context.quadraticCurveTo(250, 25, 150, 200);

context.lineWidth = 10;
context.strokeStyle = 'blue';
context.stroke();
</script>
```

## 10 Conclusion

This research has presented a very unambiguous study of Bezier Parametric curves. It lays theoretical foundation of Bezier curves, and presents relevant theorems and corresponding proofs. The importance of Bezier curves in Computer Graphics, and in other fields of computing was touched. The work was concluded by showing how Bezier curves could be programmed using HTML 5 canvas and Java Script.

While Bezier parametric curves have a lot of possibilities, there are however some limitations. One limitation is the obvious computational complexity, especially for handling higher order cases [30]. This is why a number of applications tend to focus their interests on cubic and quadratic cases. Secondly, it is also difficult to get exact points on the curve. This issue however, has been resolved using a number of algorithms, for instance de Casteljau [31,32].

As a means of evaluation, it is necessary to mention other works or tools that could be comparable to the use of canvas technology, as this current study. The Open GL has been successfully explored in Bezier curves implementations in a research by [33]. However, the focus was limited to cubic cases.

A number of other comparable works are [34], which is uses Adobe Illustrator, [35] which is uses 3D Maya, based on animation techniques, both of which are mainly used for drawing the required objects. On the other hands, [36] is a computational work, though the main focus was on the control points of Bezier curves, unlike this research which covered evolutionary computational issues, as well as the grammatical implementation.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

- [1] H. Osama, O. Ruba, A. Rizik and A. Bashar, "Impact of computer graphics on the engineering product design: conceptual analysis," *Scientific Research and Essays*, vol. 8, no. 25, pp. 1553–1561, 2013.
- [2] B. Senay and K. Bulent, "Defining a curve as a bezier curve," *Journal of Taibah University for Science*, vol. 13, no. 1, pp. 522–528, 2019.
- [3] Y. Ayse, T. Tunahan and O. Gozde, "On geometry of rational bezier curves," *Honam Mathematical Journal*, vol. 43, no. 1, pp. 88–99, 2021.
- [4] V. Peter, A. Richard and A. Olabisi, "Solution of systems of disjoint Fredholm–Volterra integro-differential equations using bezier control points," *Science World Journal*, vol. 15, no. 4, pp. 25–40, 2020.
- [5] B. Amir, P. Edouard and S. Shoham, "The cyclic block conditional gradient method for convex optimization problems," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2024–2049, 2015.
- [6] P. Rafael, G. Juan, C. Faouzi, O. Joaquín and E. Ole, "High-performance computation of bézier surfaces on parallel and heterogeneous platforms," *International Journal of Parallel Programming*, vol. 46, no. 6, pp. 1035–1062, 2018.
- [7] M. R. Abedallah, L. Byung-Gook and Y. Jaechil, "Multiple degree reduction and elevation of bézier curves using Jacobi-Bernstein basis transformations," *Numerical Functional Analysis and Optimization*, vol. 28, no. 9, pp. 1179–1196, 2007.
- [8] C. Fuhua, K. Anastasia and L. Alice, "Beta-bezier curves," *Computer-Aided Design & Applications*, vol. 18, no. 6, pp. 1265–1278, 2021.
- [9] V. Guillaume, G. Valerie and B. Marie, "Cubic bézier local path planner for nonholonomic feasible and comfortable path generation," in *Proc. IEEE ICRA*, Xian, China, pp. 1–7, 2021.
- [10] C. Vincent, G. Celine, L. Eva, P. Clair, R. Magali *et al.*, "Curved interface reconstruction for 2d compressible multi-material flows," *ESAIM: Proceedings and Surveys*, vol. 67, pp. 178–190, 2020.
- [11] Y. Norimasa and S. Takafumi, "Planar curves based on explicit bézier curvature functions," *Computer-Aided Design and Applications*, vol. 17, no. 1, pp. 77–87, 2020.
- [12] J. Read, "Animating the real: Illusions, musicality and the live dancing body," *The International Journal of Screendance*, vol. 11, pp. 59–75, 2020.
- [13] G. Mariana-Daniela and A. Emilio, "Implications of virtual reality in arts education: Research analysis in the context of higher education," *Education Sciences*, vol. 10, no. 225, pp. 1–19, 2020.
- [14] S. Ahmet, Y. Nurullah and K. Gulden, "A novel modeling and smoothing technique in global optimization," *Journal of Industrial Management*, vol. 15, no. 1, pp. 113–130, 2019.
- [15] A. Mustafa and Z. Omar, "Geometric piecewise cubic bézier interpolating polynomial with C2 continuity," *SPIIRAS Proceedings*, vol. 20, no. 1, pp. 133–159, 2021.
- [16] L. Fenhong, H. Gang, A. Muhammad and M. Kenjiro, "The generalized H-bézier model: Geometric continuity conditions and applications to curve and surface modeling," *Mathematics*, vol. 8, no. 924, pp. 1–21, 2020.
- [17] S. Richard, M. Brandon and M. Kalvin, "Pictorial integration in the calculus," *International Journal of Mathematical Education in Science and Technology*, vol. 52, no. 1, pp. 144–154, 2021.
- [18] L. Joseph, L. Tong and M. John, "Non-linear least squares fitting of bézier surfaces to unstructured point clouds," *AIMS Mathematics*, vol. 6, no. 4, pp. 3142–3159, 2021.
- [19] X. Han and J. Yang, "Multi-degree reduction of bézier curves with distance and energy optimization," *Journal of Applied Mathematics and Physics*, vol. 4, no. 1, pp. 8–15, 2016.
- [20] E. Ghomanjani, M. H. Farahi, A. Jçman, A. Kamyad and N. Pariz, "Bezier curves based numerical solutions of delay systems with inverse time," *Mathematical Problems in Engineering*, vol. 602641, pp. 1–16, 2014.
- [21] D. Jorge and J. M. Pena, "Geometric properties and algorithms for rational q-bézier curves and surfaces," *Mathematics*, vol. 8, no. 541, pp. 1–15, 2020.

- [22] K. Khalid, D. K. Lobiyal and K. Adem, "A de Casteljaou algorithm for Bernstein type polynomials based on (p, q)-integers," *Applications and Applied Mathematics-an International Journal*, vol. 13, no. 2, pp. 997–1017, 2018.
- [23] J. Sanchez-Reyes, "Comment on defining a curve as a bezier curve," *Journal of Taibah University for Science*, vol. 14, no. 1, pp. 849–850, 2020.
- [24] K. Artur and S. Jakub, "Application programming interface for the cloud-based management of gamified eGuides," *Information-an International Interdisciplinary Journal*, vol. 11, no. 6, pp. 2–11, 2020.
- [25] S. John and S. Alan, *Python All-in-One for Dummies*. New Jersey: John Wiley & Sons, Inc, pp. 20–110, 2019.
- [26] B. Sebastian, "A primer on python for life science researchers," *PLOS Computational Biology*, vol. 3, no. 11, pp. 2052–2057, 2007.
- [27] M. Everton, R. Raimundo and S. Christine, "The ecology of human-anaconda conflict: A study using internet videos," *Open Access Journal-Tropical Conservation Science*, vol. 9, no. 1, pp. 43–77, 2016.
- [28] D. John, *Web Programming with HTML5, CSS and Javascript*. Burlington: Jones & Bartlett Learning, pp. 1–40, 2019.
- [29] S. Laini, N. Kalam, S. Anjelina and S. Alfat, "The design of learning media using javascript and its implementation in the local web server," in *Proc. ICOLSSTEM*, University of Jember, East Java, Indonesia, pp. 1–10, 2021.
- [30] T. Popiel and L. Noakes, "Bézier curves and c2 interpolation in Riemannian manifolds," *Journal of Approximation Theory*, vol. 148, no. 2, pp. 111–127, 2007.
- [31] J. Li, Y. Ji and C. Zhu, "De Casteljaou algorithm and degree elevation of toric surface patches," *Journal of Systems Science and Complexity*, vol. 34, no. 1, pp. 21–46, 2021.
- [32] Z. Sir and B. Juttler, "On de Casteljaou-type algorithms for rational bézier curves," *Journal of Computational and Applied Mathematics*, vol. 288, no. 1, pp. 244–250, 2015.
- [33] Y. Kumar, S. K. Srivastava, A. K. Bajpai and N. Kumar, "Development of computer aided design algorithms for bezier curves/surfaces independent of operating system," *WSEAS Transactions on Computers*, vol. 11, no. 6, pp. 159–169, 2012.
- [34] H. Lieng, F. Tasse, J. Kosinka and N. A. Dodgson, "Shading curves: Vector-based drawing with explicit gradient control," *Computer Graphics Forum*, vol. 34, no. 6, pp. 228–239, 2015.
- [35] R. Kushwaha, "Procedure of animation in 3d autodesk maya: Tools and techniques," *International Journal of Computer Graphics & Animation*, vol. 5, no. 4, pp. 15–27, 2015.
- [36] S. Baydas and B. Karakas, "Defining a curve as a bezier curve," *Journal of Taibah University for Science*, vol. 13, no. 1, pp. 522–528, 2019.