Int. j. adv. multidisc. res. stud. 2026; 6(1):1428-1435

Received: 07-01-2025
Accepted: 17-01-2026

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

The Comparison Between Bezier and B-splines Curves Using MATLAB

[1] Taha Gabaireldar Elradi, [2] Ahmed MohamedNour MohamedAli

[1] Department of Mathematics, Faculty of Science, University of Albutana, Sudan

[2] Department of Mathematics, Faculty of Science, Sudan University of Sciences and Technology, Sudan

DOI: https://doi.org/10.62225/2583049X.2026.6.1.5686
Corresponding Author: **Taha Gabaireldar Elradi**

Abstract

Curves are characterized by their multiple uses in various fields, especially in mathematics, engineering, and computer science. This study aimed to find a comparison between Bezier and B-spline curves using MATLAB. The study followed the applied analytical approach. The study reached the most important results, which are that when a point in the B-spline curve changes, the curve changes locally around that point, while when a point in the Bezier curve changes, the curve changes completely. And the number of control points in a B-spline curve is one-third less than the number of control points in a Bezier curve. Furthermore, the control points in a B-spline curve are not tangent to the curve's path, whereas in a Bezier curve, the control points are tangent to the curve's path. Based on the results, we found that the B-spline curve performs better than the Bezier curve. The study also utilized MATLAB to drive both Bezier and B-spline curves, allowing for a more precise observation of local and global effects. We recommend that researchers conduct further observations regarding the local and global effects of both types of curves and utilize these findings in mathematical, engineering and medical fields.

Keywords: Bezier, B-splines, MATLAB

Introduction

OPC (Optical Proximity Correction) technology has been evolving at the same pace as the lithography technology in order to achieve higher resolutions at every new technology node. One approach for further extending lithography resolution is ILT (Inverse Lithography Technology). This technology is based on applying an inverse model to the target design to determine which structure should be present on the mask [1]. In order to compensate the lithography process errors, the ILT creates curves from the design layouts [2]. There were three main limiting factors for the adoption of ILT: modeling runtime, mask-writing time, and data volume. Regarding the runtime, the inverse lithography models would take a long time for producing a satisfying solution. Currently, advances in algorithms, such as AI (Artificial Intelligence), as well as on computing power with latest CPUs and GPUs, have mitigated this issue [3]. The second limiting factor was the mask writing time. The standard of the industry was writing masks using VSB (variable-shaped-beam) writers. This type of equipment is suitable for fast-writing regular shapes (rectangles and trapezoids), but not for curvilinear shapes. With the adoption of multibeam mask writers, the characteristics of the shapes no longer impact the mask writing time [4]. At the same time, such curvilinear shapes are currently stored as polygons which contain lots of vertices, hence an increase of the layout file sizes. For instance, a dot array can be compacted by a factor of ten if described by a curve instead of a polygon [5]. The file size issue has been identified and addressed by the industry in recent years. For instance, mask-writers suppliers proposed a curvilinear format support [6]. Moreover, a curvilinear work group was created in 2019 [7] in order to develop an extended Oasis format supporting curves by generalizing the current P39 standard Oasis format [8]. This new format allows the representation of curvilinear shapes by using curve descriptions, with the following supported type of curves: explicit piecewise Bezier, implicit piecewise Bezier, uniform clamped B-Splines, and uniform periodic B-Splines. Implicit piecewise Bezier and B-Splines are of particular interest since they both offer more compaction properties than the explicit piecewise Bezier [9], from which they are both derived. We analyze the properties, advantages, and disadvantages of implicit Bezier and B-Spline curves. First, we focus our discussion on the definition of such curves as well as the relation with the explicit Bezier curves. Then, we highlight the geometrical properties of both implicit Bezier and B-Spline curves. Finally, we propose a comparative discussion of theses formats properties, as compaction capabilities in terms of number of control points, or predispositions to sizing-like or Boolean operations.

1-The Bezier curve:

Definition (1). A Bezier Curve is a smooth, mathematically defined parametric curve used in computer graphics and design shaped by a set of control points that dictate its path, passing through the start and end points but influenced by the intermediate ones to create precise scalable shapes for fonts. animations, and vector graphics [10].

On a mathematical, as well as on a computational point of view, the simplest type of parametric curve is polynomial curves, that is with a polynomial parameterization as $C_0 + C_1 t + \cdots + C_p t^p$, where $C_0, C_1, \ldots, C_p$ are points. Note that p is the degree of the curve. One of the elegant properties of polynomial curves is *smoothness* of the parameterization, which implies the geometrical smoothness of the curve itself.

An even more convenient class of parametric curves, which is widely spread in industry [11], is Bezier curves. A Bezier curve of degree p is defined by the weighted average1:

$$C(t) = \sum_{i=0}^{p} B_{i,p}(t) P_i\,, \qquad t \in [0,1]$$

,

of the control points $P_0, P_1, \ldots, P_p$. A given weight $B_{i,p}(t) = \binom{p}{i}(1-t)^{p-i}t^i$ is the $\boldsymbol{i}$-th Bernstein polynomial of degree p.

Lemma (1): For the path constructed by two Bezier curve segments $P(t)|_{t\in[t_0,t_1]}$ and $Q(t)|_{t\in[t_1,t_2]}$, if $P(t)$ and $Q(t)$ are at least C^2continuous at t_1 then the path has continuous curvature for every point on it.

Proof. The curvature is expressed in terms of the first and the second derivative of a curve in $k(t) = \frac{|\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)|}{(\dot{x}^2(t)+\dot{y}^2(t))^{\frac{3}{2}}}$. Since the Bezier curves are defined as polynomial functions of t, their k-th derivative for all k = 1,2. are continuous. Hence, they always have continuous curvature for all t. For two different Bezier curves $P(t)$ and $Q(t)$, it is sufficient that $\kappa(t_1)$, the curvature at the junction node is continuous if $\dot{P}(t) = \dot{Q}(t)$ and $\ddot{P}(t) = \ddot{Q}(t)$ are continuous at t_1. Therefore, if $P(t)$ and $Q(t)$ are at least C^2 continuous at t_1 then the path have the curvature continuous for every point on it [12].

2-The B-spline curve:

Definition (2): A B-spline curve is said to be uniform whenever its knots are equally spaced, and non-uniform otherwise. A uniform B-spline curve is said to be open uniform whenever its interior knots are equally spaced, and its exterior knots are same. Similarly, A non-uniform B-spline curve is said to be open non-uniform whenever its exterior knots are same and its interior knots are not equally spaced [13].

In a way which is similar to a Bezier curve, B-Spline parameterization is a weighted average of control points:

$$\boldsymbol{C}(t) = \sum_{i=0}^{n} N_{i,p}(t)\boldsymbol{P}_i,\ t \in [0,1]$$

Where $N_{i,p}$ is the p^{th} degree, i^{th} basis function[3]. $N_{i,p}$ is a p^{th} degree, piecewise polynomial. Contrarily to the Bézier representation, a large number of control points $(n+1)$ is allowed for a low polynomial degree (p). Moreover, the influence of the control points is now local: more precisely, a given point of the curve is only affected by $p+1$ control points, due to the shape of basis functions. This allows a good curve control, and nice numerical properties such as the sparse matrix representation for model fitting.

Remark (1): The coefficients satisfy:

$$0 \leqslant \lambda_{i,k} = \lambda_{k-i,k} \leqslant 1 \text{ for } i = 0, \ldots, \left\lceil \frac{k}{2} \right\rceil$$

and

$$\sum_{i=0}^{\left\lfloor \frac{k}{2} \right\rfloor} \lambda_{2i,k} = \sum_{i=0}^{\left\lfloor \frac{k-1}{2} \right\rfloor} \lambda_{2i+1,k} = \delta_{0,k} + \delta_{1,k} = 1$$

For any polynomial, one degree of freedom can be introduced to its expression by raising its degree by one. Note that B-spline basis functions are piecewise polynomials. Therefore, degree elevation method can theoretically introduce multiple degrees of freedom into the expressions of B-spline curves. However, Theorem 2.1 reveals that, the UBP basis functions are the revised B-spline basis functions, whose expressions are modified by inserting only one parameter by the degree elevation of B-spline. Thus, we will study if there are other more general uniform B-spline basis functions with multiple parameters [14].

Example (1): For $n = 2$ (three control points), the weights are;

$$B_{2,0}(t) = \binom{2}{0} t^0 (1-t)^{2-0} = (1-t)^2$$

$$B_{2,1}(t) = \binom{2}{1} t^1 (1-t)^{2-1} = 2t(1-t)$$

$$B_{2,2}(t) = \binom{2}{2} t^2(1-t)^{2-2} = t^2$$

and the curve is:

$$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

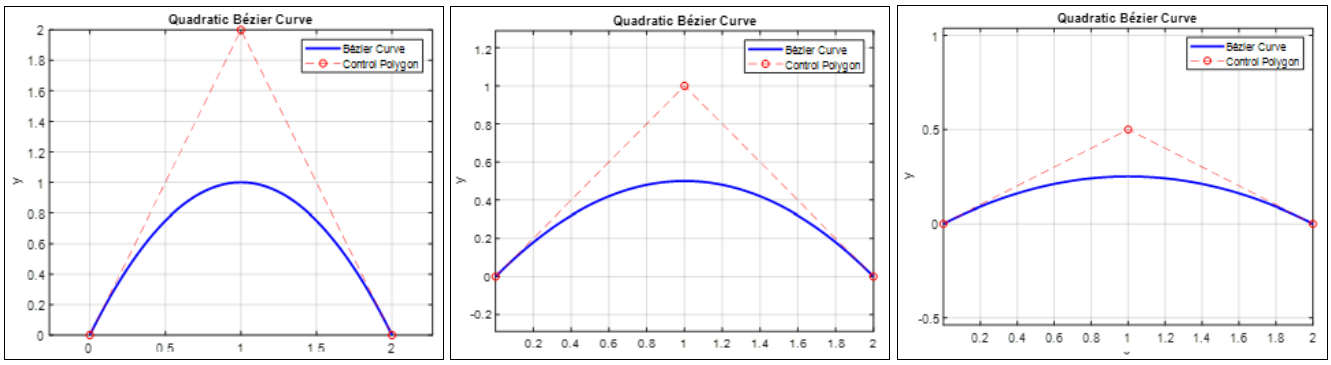
$$= ((1-t)^2, 2t(1-t), t^2)(P_0, P_1, P_2)^T$$

$$= (t^2, t, 1)\begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} P_0 \\ P_1 \\ P_2 \end{pmatrix} \quad (1)$$

This is the quadratic Bezier curve.

MATLAB code to compute the quadratic Bezier curve for n=2 with three control points $P_0, P_1, and\ P_2$.

```
% Define the control points (replace these with actual values)
P0 = [0, 0]; % Example values, replace with your actual coordinates
P1 = [1, 2]; % Example values, replace with your actual coordinates
P2 = [2, 0]; % Example values, replace with your actual coordinates
% Define the parameter t from 0 to 1
t = linspace(0, 1, 100);
% Calculate the Bernstein polynomials
B_2_0 = (1 - t).^2; B_2_1 = 2 * t .* (1 - t); B_2_2 = t.^2;
% Calculate the curve points
P = B_2_0' * P0 + B_2_1' * P1 + B_2_2' * P2;
% Plot the quadratic Bézier curve
figure; plot(P(:,1), P(:,2), 'b', 'LineWidth', 2); hold on;
plot([P0(1) P1(1) P2(1)], [P0(2) P1(2) P2(2)], 'ro--'); % Plot control points and polygon
legend('Bézier Curve', 'Control Polygon'); title('Quadratic Bézier Curve');
xlabel('x'); ylabel('y'); grid on; axis equal;
```



P1(1, 2) **P1(1, 1)** **P1(1, 0.5)**

Fig 1: Bezier Curve for n=2 With Three Control Points P_0 , P_1 , and P_2

In the special case $n = 3$, the four weight functions are:

$$B_{3,0}(t) = \binom{3}{0} t^0(1-t)^{3-0} = (1-t)^3$$

$$B_{3,1}(t) = \binom{3}{1} t^1(1-t)^{3-1} = 3t(1-t)^2$$

$$B_{3,2}(t) = \binom{3}{2} t^2(1-t)^{3-2} = 3t^2(1-t)$$

$$B_{3,3}(t) = \binom{3}{3} t^3(1-t)^{3-3} = t^3$$

and the curve is,

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3 \quad (2)$$

$$= [(1-t)^3, 3t(1-t)^2, 3t^2(1-t), t^3][P_0, P_1, P_2, P_3]^T$$

$$= (t^3, t^2, t, 1)\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

It is clear that $P(t)$ is a cubic polynomial in t. It is the cubic Bezier curve.
In general, the Bezier curve for points $P_0, P_1, \ldots, P_n$ is a polynomial of degree n.
It can be proved by induction that the general, $(n+1)$ -point Bézier curve can be represented by,

$$P(t) = (t^n, t^{n-1}, \ldots, t, 1)N\begin{pmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-1} \\ P_n \end{pmatrix} = T(t) \cdot N \cdot P$$

Where;

$$\mathbf{N} = \begin{pmatrix} \binom{n}{0}\binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \cdots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{0}\binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \cdots & 0 \\ \vdots & \vdots & \cdots & 0 \\ \binom{n}{0}\binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \cdots & 0 \\ \binom{n}{0}\binom{n}{0}(-1)^0 & 0 & \cdots & 0 \end{pmatrix} \quad (5)$$

Matrix **N** is symmetric and its elements below the second diagonal are all zeros. Its determinant therefore equals (up to a sign) the product of the diagonal elements, which are all nonzero. A nonzero determinant implies a nonsingular matrix. Thus, matrix **N** always has an inverse. N can also be written as the product **AB**, Where;

$$\mathbf{A} = \begin{pmatrix} \binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \cdots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \cdots & 0 \\ \vdots & \vdots & \cdots & 0 \\ \binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \cdots & 0 \\ \binom{n}{0}(-1)^0 & 0 & \cdots & 0 \end{pmatrix}$$

and

$$\mathbf{B} = \begin{pmatrix} \binom{n}{0} & 0 & \cdots & 0 \\ 0 & \binom{n}{1} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \binom{n}{n} \end{pmatrix}$$

Figure (2) shows the Bezier **N** matrices for $n = 1,2, \ldots,7$.

$$\mathbf{N}_1 = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\mathbf{N}_2 = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{N}_3 = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{N}_4 = \begin{pmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{N}_5 = \begin{pmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{N}_6 = \begin{pmatrix} 1 & -6 & 15 & -20 & 15 & -6 & 1 \\ -6 & 30 & -60 & 60 & -30 & 6 & 0 \\ 15 & -60 & 90 & -60 & 15 & 0 & 0 \\ -20 & 60 & -60 & 20 & 0 & 0 & 0 \\ 15 & -30 & 15 & 0 & 0 & 0 & 0 \\ -6 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_7 = \begin{pmatrix} -1 & 7 & -21 & 35 & -35 & 21 & -7 & 1 \\ 7 & -42 & 105 & -140 & 105 & -42 & 7 & 0 \\ -21 & 105 & -210 & 210 & -105 & 21 & 0 & 0 \\ 35 & -140 & 210 & -140 & 35 & 0 & 0 & 0 \\ -35 & 105 & -105 & 35 & 0 & 0 & 0 & 0 \\ 21 & -42 & 21 & 0 & 0 & 0 & 0 & 0 \\ -7 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Fig 2: The First Seven Bezier Basis Matrices

Example (2): Five control points p_0 through p_4 are given, implying that n = 4. We select order 3 (i.e., segments that are polynomials of degree 2) to construct the knot sequence (0, 0, 0, 1, 2, 3, 3, 3). The parameter t varies from $t_{k-1} = t_2 = 0$ to $t_{n+1} = t_5 = 3$, so our curve will consist of three segments. Each of the blending functions Ni3(t) (where $0 \le i \le n$) is nonzero over three subintervals of t. The result is,

$$N_{0,3}(t) = (1-t)^2 \quad , \quad 0 \le t < 1$$

$$N_{1,3}(t) = \frac{1}{2}\begin{cases} -3t^2 + 4t & , \quad 0 \le t < 1 \\ (2-t)^2 & , \quad 1 \le t < 2 \end{cases}$$

$$N_{2,3}(t) = \frac{1}{2}\begin{cases} t^2 & , \quad 0 \le t < 1 \\ -2t^2 + 6t - 3 & , \quad 1 \le t < 2 \\ (3-t)^2 & , \quad 2 \le t < 3 \end{cases}$$

$$N_{3,3}(t) = \frac{1}{2}\begin{cases} (t-1)^2 & , \quad 1 \le t < 2 \\ -3t^2 + 14t - 15 & , \quad 2 \le t < 3 \end{cases}$$

$$N_{0,3}(t) = (t-2)^2 \quad , \quad 2 \le t < 3$$

So, the three spline segments are,

$$p_1(t) = (1-t)^2 p_0 + \frac{1}{2} t\,(4-3t) p_1 + \frac{1}{2} t^2 p_2 \quad , \quad 0 \le t < 1$$

$$p_2(t) = \frac{1}{2}(2-t)^2 p_1 + \frac{1}{2}[t\,(2-t) + (t-1)(3-t)]p_2 + \frac{1}{2}(t-1)^2 p_3 \,, \quad 1 \le t < 2$$

$$p_3(t) = \frac{1}{2}(3-t)^2 p_2 + \frac{1}{2}(3-t)(3t-5)p_3 + (t-2)^2 p_4, \quad 2 \le t < 3$$

It is now easy to calculate where each segment starts and ends:

$$p_1(0) = p_0, \qquad p_1(1) = \frac{(p_1 + p_2)}{2}$$

$$p_2(1) = \frac{(p_1 + p_2)}{2}, \quad p_2(2) = \frac{(p_2 + p_3)}{2}$$

$$p_3(2) = \frac{(p_2 + p_3)}{2}, \qquad p_3(3) = p_4$$

Figure below shows a typical example of the three segments (with intentional gaps between them).

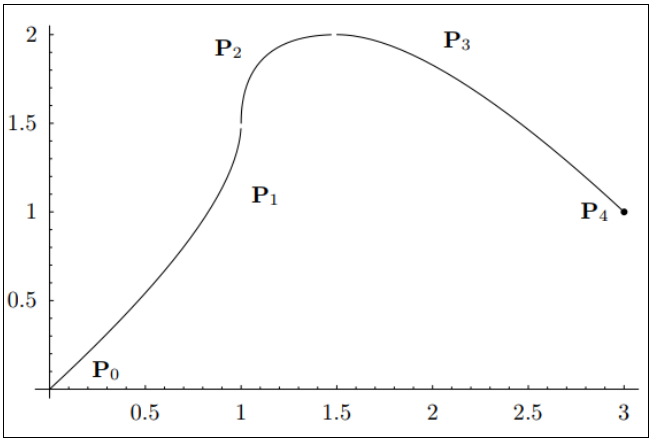


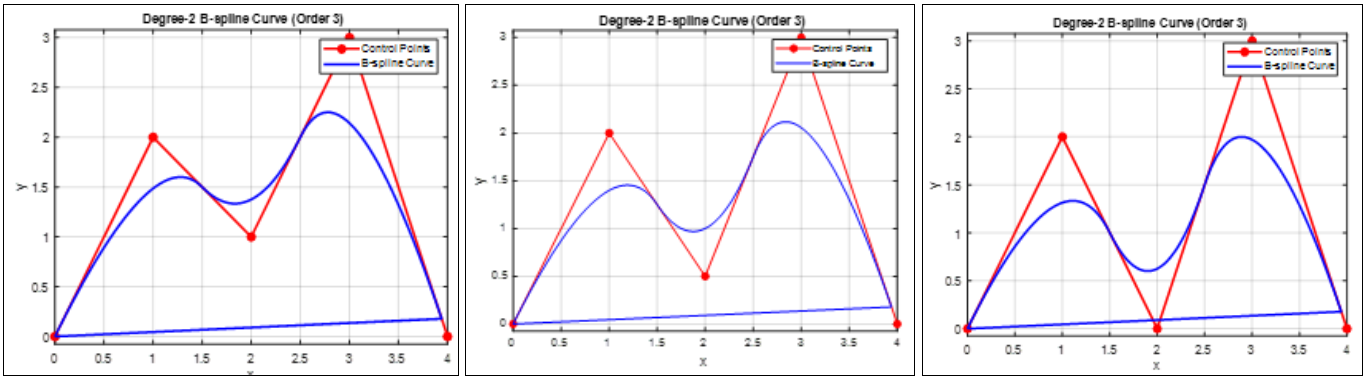
Fig 3: An Open Uniform B-Spline

MATLAB script to generate and plot the B-spline curve

```
% Main script to plot the B-spline curve with order 3 (degree-2 polynomial segments)
% Control points (p_0 to p_4)
P = [0 0; 1 2; 2 1; 3 3; 4 0];
% Knot sequence for degree-2 B-spline (order 3)
knots = [0 0 0 1 2 3 3 3];
% Degree of the B-spline (order 3 means degree 2)
degree = 2
% Parameter range
t_min = knots(degree + 1);nt_max = knots(end - degree);
t_values = linspace(t_min, t_max, 100);
% Calculate points on the B-spline curve
curve_points = zeros(length(t_values), 2);
for j = 1:length(t_values)
    t = t_values(j);
    point = [0 0];
    for i = 1:length(P)
        N = basis_function(i, degree, t, knots);
        point = point + N * P(i, :);
    end
    curve_points(j, :) = point;
end
% Plot control points
plot(P(:, 1), P(:, 2), 'ro-', 'LineWidth', 2, 'MarkerFaceColor', 'r'); hold on;
% Plot B-spline curve
plot(curve_points(:, 1), curve_points(:, 2), 'b-', 'LineWidth', 2);
% Plot settings
xlabel('x'); ylabel('y'); title('Degree-2 B-spline Curve (Order 3)');
```

```
legend('Control Points', 'B-spline Curve'); grid on; axis equal; hold off;
% Basis function definition
function N = basis_function(i, k, t, knots)
    % Adjust indexing for MATLAB (1-based)
    i = i - 1;
    if k == 0
        if i+1 <= length(knots) - 1 && knots(i+1) <= t && t < knots(i+2)
            N = 1;
        else
            N = 0;
        end
    else
        if i+k+1 <= length(knots) - 1 && knots(i+1) ~= knots(i+k+1)
            c1 = (t - knots(i+1)) / (knots(i+k+1) - knots(i+1)) * basis_function(i+1, k-1, t, knots);
        else
            c1 = 0;
        end
        if i+k+2 <= length(knots) - 1 && knots(i+2) ~= knots(i+k+2)
            c2 = (knots(i+k+2) - t) / (knots(i+k+2) - knots(i+2)) * basis_function(i+2, k-1, t, knots);
        else
            c2 = 0;
        end
        N = c1 + c2;
    end
end
```

This will generate and plot the B-spline curve using the specified control points and knot sequence, resulting in a curve with three segments due to the given degree and knot vector.



$P_2(2, 1)$ **$P_2(2, 0.5)$** **$P_2(2, 0)$**

Fig 4: Degree-2 B-spline Curve (Order 3) $P_2(2, 1)$

Five control points are specified. The knot sequence is defined as $[0, 0, 0, 1, 2, 3, 3, 3]$ for a degree-2 B-spline (order 3). The degree of the curve is set to 2 (order 3). The parameter t varies from 0 to 3, which defines the range for the curve. The script calculates points on the curve using the basis functions and control points. The basis functions are evaluated recursively to determine their influence on each segment of the curve. The control points are plotted as red circles. The B-spline curve is plotted in blue.

Results

The most important results i can summarize as follows:

1. The Bezier is defined by Approximating tangents on the curve by using control points but the B-Spline is defined as a linear combination of control points and a B-spline basis.
2. The Bezier approximation provides the global control of curve shape. If a single control point is changed, the shape of curve is disturbed but the B-spline approximation provides the local control of curve shape. If a single control point is changed, only parts of the curve lying under the influence of that control point are disturbed.
3. The Bezier is a special case of a B-spline curve, where the basis function defines the span/influence of each point on the entire curve but the B-Spline is a generalized version of a Bezier curve that consists of Bezier curves as segments.

Recommendations

1- for simple, scalable graphics (logos, icons) use Bezier for simplicity.
2- for complex, high-surfaces (product design, animation) use B-spline for local control and flexibility.
3- for approximating data points B-splines are generally better, avoiding high-degree Bezier curves that become unstable.

References

1. Pang Linyong. Inverse lithography technology: 30 years from concept to practical, full-chip reality. Journal of Micro/Nanopatterning, Materials, and Metrology. 2021; 20(3):030901-030901.
2. Guo Vivian Wei, *et al*. Lithographic benefits and mask manufacturability study of curvilinear masks. Photomask Technology. 2018; 10810. SPIE, 2018.
3. Chen Yung-Yu, *et al*. Curvilinear mask handling in OPC flow. Journal of Micro/Nanopatterning, Materials, and Metrology. 2024; 23(1):011203-011203.
4. Cho Yoonjung, *et al*. Requirements and strategies of High-NA EUV mask using MBMW. Photomask Technology. 2024; 13216. SPIE, 2024.
5. Shendre Abhishek, *et al*. You don't need 1nm contours for curvilinear shapes: Pixel-based computing is the answer. Photomask Technology. 2022; 12293. SPIE, 2022.
6. Choi Jin, *et al*. Status of curvilinear data format working group. Photomask Japan 2022: XXVIII Symposium on Photomask and Next-Generation Lithography Mask Technology. 2022; 12325. SPIE, 2022.
7. Bork Ingo, *et al*. Curvilinear Mask Process Correction-status quo and outlook. 2022 International Workshop on Advanced Patterning Solutions (IWAPS). IEEE, 2022.
8. Venitucci Benjamin, *et al*. B-spline and Bezier curvilinear representations: A comparative discussion. Novel Patterning Technologies. 2024; 12956. SPIE, 2024.
9. Gharat Sayalee, *et al*. A study on various curvilinear data representations and their impact on mask manufacturing flow. Optical Microlithography XXXIV. 2021; 11613. SPIE, 2021.
10. Arsalan hasanvand, Bezier curve, researchgate, 2021.
11. Sederberg Thomas W. Computer aided geometric design, 2012.
12. Ji-wung Choi, Renwick E. Curry, Gabriel Hugh Elkaim, Smooth Path Generation Based on Bezier Curves for Autonomous Vehicles, iaeng.org, 2009.
13. Muhsin Incesu, Sara Yilmaz Evren. The selection of control points for two open non uniform B-spline curves to form Bertrand pairs, tcms.org.ge, 2021.
14. Juan Cao, Guozhao Wang. The structure of uniform B-spline curves with parameters, Sciencedirect, 2008.