Scientific Visualization, 2022, volume 14, number 1, pages 97 - 108, DOI: 10.26583/sv.14.1.08

One of The Algorithms for Converting Spline Interpolated Curves into B-spline Form

A. S. Minkin[1]

National Research Center «Kurchatov Institute», Moscow, Russia

[1] ORCID: 0000-0001-8789-0779, amink@mail.ru

Abstract

To ensure flexibility and convenience of working with geometric models in CAD systems, algorithms for converting geometric representations are demanded, among which methods of their one-to-one and exact transformation are of great importance. In this paper, we propose a technique for converting a spline curve into a corresponding equivalent B-spline curve based on combining Bezier segments and the removal of multiple knots to obtain a more compact B-spline representation. The justification of the simplified version of the knot-removal algorithm for B-splines is given. This approach makes it possible to construct a B-spline curve based on information about its individual points without using standard fitting tools and complex interpolation schemes.

Keywords: geometric modeling, parametric curves, CAD, cubic splines, B-spline curves, NURBS, Bezier curves.

1. Introduction

An important stage in the design of industrial facilities is the creation of their geometric model, which is usually carried out using specialized software - CAD systems. The underlying part of the CAD system is the geometric core, which allows you to work with various types of curves and surfaces to create two-dimensional and three-dimensional geometric models. The geometric model of an object is usually represented by a hierarchical boundary description (B-rep) [1] considering a geometric object as a set of faces with common boundaries as edges. Various parametric descriptions of curves and surfaces are used to define curved geometry [2-5]. The efficiency of CAD systems depends on the geometric core algorithms for working with various geometric shapes such as splines. A detailed overview and classification of splines are given in [6], however, analysis of all methods and possibilities of transformation between different models is beyond the scope of this article. In this paper, we consider the cubic splines of defect 1 and B-splines and propose a technique for converting spline curves into B-spline form without using special methods of B-splines fitting. We discuss the conditions under which such transformation becomes possible.

One of the best ways to produce a mathematical description of curves that provides a good mechanism of shape control is to use rational B-splines or NURBS [2,4,5]. B-splines are the industry standard for representing composite curves. Like Bezier curves, B-splines are defined by a weighted sum of control points, but, at the same time, they provide local control of curve shape. NURBS (Non-uniform rational B-spline) are a universal mathematical description of various types of curves and surfaces such as straight lines, planes, conic sections [7,8], quadrics, surfaces of rotation [9] and other geometric shapes. Besides, NURBS representation is used in the universal STEP format for data exchange between CAD systems. Thus, NURBS is one of the popular methods of description of parametric curves and surfaces, which is widely used in modern CAD systems, due to its versatility and ease of use. On the one hand, NURBS makes it possible to describe complex curves consisting of many parametric segments. On the other hand, the NURBS defining polygon allows controlling the shape of the curve, but does not explicitly define the points of this curve, determining only the direction of its bend. In this paper, we consider B-splines that are equivalent to NURBS with unit weights. In general, a B-spline curve may not interpolate any of the specified control points of the defining polygon. Low-order spline polynomials such as cubic splines of defect 1 are usually used when it comes to interpolation of given points. Similarly to physical splines, a series of cubic segments is used to construct a mathematical spline, each of which interpolates the endpoints. A cubic spline is defined by points, tangent vectors, and parameter values at the ends of segments, that is, unlike NURBS, explicitly uses additional information about derivatives at the endpoints.

In general, for an unknown original curve, the B-spline fitting problem is solved using more complex algorithms. To interpolate a set of points [10], line segments [11] and to construct transition curves [12] algorithms for solving linear system of equations [13] and different iteration schemes [14] can be used. In [15], an interpolation scheme with fitting of transition curves represented by a pair of Bezier segments is considered.

The goal of the work is to find an effective algorithm for one-to-one and accurate transformation of a curve defined in the form of a cubic spline into the corresponding representation in the form of a B-spline. In this paper, we consider a technique for converting a cubic spline of defect 1 into a B-spline based on combining Bezier segments, followed by obtaining a more compact B-spline description by removing single knots. In this formulation of the problem, a transformation of the curve to B-spline form can be done with linear time complexity, while fitting algorithms based on matrix multiplications [16,17], inversions [18,19], as well as solutions of linear systems [20], have at least quadratic time complexity. The B-spline knot removal algorithm is the reverse of the knot insertion algorithm. There are several variants of the knot insertion algorithms. Insertion of several knots can be implemented using the Oslo algorithm [21,22], insertion of a single knot – using Boehm's algorithm [23]. A modification of the Boehm algorithm for inserting multiple knots is considered in [24]. In this paper, knot removal is used to obtain an equivalent B-spline form with fewer vertices in the defining polygon. As an algorithm for knot removal, in general, any of the options under consideration can be used. In this paper, the rationale for the B-spline knot insertion algorithm is given and a simplified version of the knot removal algorithm is used to obtain the most compact B-spline representation of the original spline curve.

Unlike B-spline fitting based algorithms, which allow approximating a set of vertices with a predetermined accuracy, the proposed algorithm for the transition from splines to B-splines allows equivalent transformations, that is, converting a given curve in the form of a composite cubic spline to a fully equivalent B-spline representation. The same object is known to be represented using different B-spline forms. At the first stage of the algorithm, the identification of a B-spline form with multiple knots is performed, which exactly corresponds to a given spline curve. At the second stage, the resulting B-spline form is optimized by removing multiple knots, which corresponds to a more compact B-spline description with fewer control points.

The article consists of several sections. Section 2 provides a theoretical background on cubic splines with defect 1 and B-splines defined by recursive Cox-deBoor formulas, and also discuss the conditions for equivalence of cubic splines to Bezier segments. Section 3 provides a detailed description of the transform algorithm from a composite cubic spline to a B-spline form, and section 4 discusses knot insertion and a simplified algorithm for a single knot removal to obtain an equivalent B-spline form with fewer knots. Information about the existing software implementation of this technique is given in section 5. An example with the result of the algorithm is discussed in section 6, followed by a conclusion.

2. Properties of cubic spline and B-spline curves

In recent years, various forms of curves and surfaces have been proposed to describe geometric shapes. Among them, cubic splines and B-splines are of particular importance.

According to [2], cubic splines are curves of the smallest degree that admit inflection points and bending in space. Consider the interpolation cubic splines of defect 1 (the difference between the degree and smoothness of the spline), consisting of identical segments, each of which passes through two points. In this case, a cubic spline curve can be described by a polynomial of the 3rd degree with a continuous second derivative at the junction points of segments:

$$P_i(t)=\sum_{j=0}^{3} B_{ij}(t-t_i)^j,\ t_i \le t \le t_{i+1}, \tag{1}$$

where B_{ji} are he control points of the *i*-th spline segment, which are determined by the known coordinates of the points P_i and P_{i+1}, the tangent vectors P_i' and P_{i+1}' and the values of the parameter t_i and t_{i+1} at the ends of the *i*-th segment as follows:

$$B_{i0}=P_i, B_{i1}=P_i', B_{i2}=\frac{3(P_{i+1}-P_i)}{(t_{i+1}-t_i)^2}-\frac{2P_i'}{t_{i+1}-t_i}-\frac{P_{i+1}'}{t_{i+1}-t_i}, B_{i3}=\frac{2(P_i-P_{i+1})}{(t_{i+1}-t_i)^3}+\frac{P_i'}{(t_{i+1}-t_i)^2}+\frac{P_{i+1}'}{(t_{i+1}-t_i)^2}. \tag{2}$$

By simple transformations, a cubic spline segment can be represented in the equivalent to (2) matrix form:

$$P_i(\tau)=[F][G],$$

$$[F]=\left[F_{1i}(\tau)\ \ F_{2i}(\tau)\ \ F_{3i}(\tau)\ \ F_{4i}(\tau)\right],[G]^T=\left[P_i\ \ P_{i+1}\ \ P_i'\ \ P_{i+1}'\right],$$

$$0\le\tau\le 1,\ \tau=\frac{t-t_i}{t_{i+1}-t_i},$$

$$F_{1i}(\tau)=2\tau^3-3\tau^2+1, F_{2i}(\tau)=-2\tau^3+3\tau^2,$$

$$F_{3i}(\tau)=\tau(\tau^2-2\tau+1)(t_{i+1}-t_i), F_{4i}(\tau)=\tau(\tau^2-\tau)(t_{i+1}-t_i).$$

B-spline is an industry standard and is used to describe complex curves and surfaces in most modern CAD systems, due to the following properties:

- B-spline is a composite parametric curve that allows avoiding undesirable oscillations characteristic of high-degree interpolation polynomials when approximating a set of points.
- The B-spline is defined by vertices of the defining polygon which is convenient for controlling the shape of the curve. There is no need to recalculate the entire curve due to a local change of a knot vector.
- A B-spline curve of degree *p* lies inside the union of all convex hulls of *p*+1 consecutive vertices of the defining polygon.
- Any affine and projective transformations can be applied to the curves by applying such transformations to the defining polygon vertices.
- Rational B-splines (NURBS) allow us to describe a fairly wide range of curves such as Bezier curves and conic sections.

The listed features of B-spline curves are determined by the properties of the B-spline basis.

Let $T = (t_0,...,t_r)$ be a non-decreasing sequence of real numbers. Let us call *T* a knot vector and the numbers t_i – knots. Let us denote the $N_{i,p}(t)$ *i*-th normalized basis function of degree *p* and define it as follows (Cox-deBoor formulas):

$$N_{i,0}=\begin{cases}1, \text{ if } t_i \le t \le t_{i+1}\\ 0 \text{ otherwise}\end{cases}$$

$$N_{i,p}(t)=\frac{t-t_i}{t_{i+p}-t_i}N_{i,p-1}(t)+\frac{t_{i+p+1}-t}{t_{i+p+1}-t_{i+1}}N_{i+1,p-1}(t). \tag{3}$$

The values t_i are the elements of the knot vector satisfying the inequality $t_0 \le t_i \le t_{i+1} \le t_r$. Most often, clamped knot vectors are considered for which first and last knot values are repeated with multiplicity equal to basis function degree *p* plus one. Setting *p*+1 multiplicity at the ends of the knot vector avoids reducing the range of the parameter t, characteristic of B-splines with unclamped uniform knot vectors [5], and interpolating the endpoints of the defining polygon.

Note the following properties of the B-spline basis:

1. $N_{i,p}(t) = 0$ for values of t outside the interval $[t_i, t_{i+p+1})$.
2. For $t \in [t_i, t_{i+1})$ no more than $p+1$ basis functions take non-zero values, namely $N_{i-p,p}(t), \ldots, N_{i,p}(t)$.
3. The basis functions are non-negative, i.e. $N_{i,p}(t) \geq 0 \ \forall\ i, p, t$.
4. $\sum_{j=i-p}^{i} N_{j,p}(t) = 1$ for t $\in [t_i, t_{i+1})$.
5. The derivatives of order 1,2,...,p are all continuous for $N_{i,p}(t)$ over the knot interval (t_i, t_{i+1}). The derivatives 1,2,...,$p-q$ are all continuous for $N_{i,p}(t)$ in a knot where q is the multiplicity of the knot to be considered.
6. $N_{i,p}(t)$ has exactly one maximum (except $p = 0$).

Properties 1 and 2 show that the B-spline basis is local, since the functions $N_{i,p}(t)$ differ from zero only in a certain interval, that is, they have a compact support. The multiplicity of knot values allows you to locally control the degree of smoothness of the curve. In general, according to the property 5 of basis functions, C^{p-q} differentiability of the curve is maintained when q multiple knots occur. Accordingly, setting p knots inside the knot vector allows interpolating the selected points of B-spline curve maintaining its continuity. Fig. 1 shows an example of the basis functions of a B-spline curve in case of presence (a) and absence (b) of multiple knot values. The basis functions corresponding to the interpolated points of the curve are marked on the graphs with bold lines.

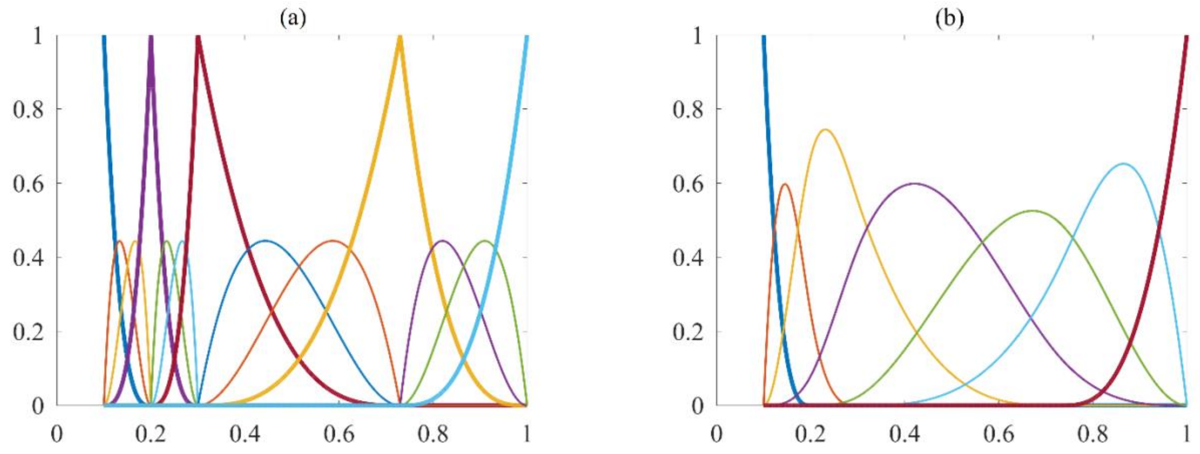

Figure 1. The example of B-spline curve basis functions:
(a) the knot vector (0.1,0.1,0.1,0.1,0.2,0.2,0.2,0.3,0.3,0.3,0.73,0.73,0.73,1,1,1,1),
(b) the knot vector (0.1, 0.1, 0.1, 0.1, 0.2, 0.3, 0.73, 1, 1, 1, 1).

To define a curve of degree p on a set of knots T it is necessary to specify a defining polygon (B_0, ..., B_n). Each vertex B_i of the defining polygon corresponds to a basis function $N_{i,p}(t)$. A point of B-spline curve is calculated from the value of the parameter t as follows:

$$P_i(t) = \sum_{j=0}^{n} B_j N_{j,p}(t), t_i \leq t \leq t_{i+1}, p \leq i < r - p.$$

According to property 2 of the B-spline basis

$$P_i(t) = \sum_{j=i-p}^{i} B_j N_{j,p}(t), t_i \leq t \leq t_{i+1}, p \leq i < r - p. \tag{4}$$

Each segment of the B-spline curve corresponds to a non-degenerate interval $t_i \leq t \leq t_{i+1}$ and is defined by p+1 vertices of the defining polygon. The number of segments of a composite B-spline curve in the absence of multiple knots inside the knot vector can be obtained as follows:

$n+1=ns+p \Rightarrow ns=n-p+1$,

where n+1 is the number of control points of the defining polygon. In the presence of multiple knot values, some of these segments can be interpreted as degenerate.

A clamped knot vector defines the parameters of the segments of a composite B-spline curve, and the number of its elements is

$nk = ns - 1 + 2 \cdot (p + 1) = n + p + 2$, то есть $r = n + p + 1$.

Setting multiple knots inside the knot vector allows you to locally reduce the degree of smoothness of B-spline curve. In case of a B-spline curve of the third degree a pair of multiple knot values makes it possible *to interpolate* individual control points of the defining polygon (Figure 1a) that coincide with the points of the curve. This property is the basis of the transformation technique of a spline curve into a B-spline form, discussed in section 3 of this paper.

Consider a cubic B-spline segment with a clamped knot vector of the form $T_1=(t_i,t_i,t_i,t_i,\ t_{i+1},t_{i+1},t_{i+1},t_{i+1})$. In this case, the B-spline curve is equivalent to the Bezier curve, and the B-spline basis is equivalent to the Bernstein basis [2], which can be, in general, proved by induction on the degree of the polynomial. The cubic Bezier segment corresponding to the knot vector T_1 can be calculated from p+1=4 control points as follows:

$$P_i(t)=\left(\frac{t_{i+1}-t}{t_{i+1}-t_i}\right)^3 B_{i0}+3\left(\frac{t-t_i}{t_{i+1}-t_i}\right)\left(\frac{t_{i+1}-t}{t_{i+1}-t_i}\right)^2 B_{i1}+3\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^2\left(\frac{t_{i+1}-t}{t_{i+1}-t_i}\right)B_{i2}+\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^3 B_{i3}. \tag{5}$$

For $t=t_i\ P_i(t_i)=B_{i0}$, for $t=t_{i+1}\ P_i(t_{i+1})=B_{i3}$.

By differentiating by the parameter t we obtain

$$P_i'(t)=\frac{1}{(t_{i+1}-t_i)^3}\Big\{-3(t_{i+1}-t)^2 B_{i0}+3\Big[(t_{i+1}-t)^2-2(t_{i+1}-t)(t-t_i)\Big]B_{i1}+$$
$$+3\Big[2(t-t_i)(t_{i+1}-t)-(t-t_i)^2\Big]B_{i2}+3(t-t_i)^2 B_{i3}\Big\}.$$

For $t=t_i\ P_i'(t_i)=-\frac{3B_{i0}}{t_{i+1}-t_i}+\frac{3B_{i1}}{t_{i+1}-t_i}$, so $B_{i1}=B_{i0}+(1/3)(t_{i+1}-t_i)P_i'(t_i)$, (6)

for $t=t_{i+1}\ P_i'(t_{i+1})=-\frac{3B_{i2}}{t_{i+1}-t_i}+\frac{3B_{i3}}{t_{i+1}-t_i}$, so $B_{i2}=B_{i3}-(1/3)(t_{i+1}-t_i)P_i'(t_{i+1})$, (7)

that is, the control points B_{i1} and B_{i2} can be calculated based on information about the endpoints B_{i0} and B_{i3} and the derivatives P_i' and P_{i+1}' at the endpoints, which corresponds to similar formulas for derivatives at the endpoints of NURBS curves with a clamped knot vector [5].

Let us show that representations (1), (2) are equivalent to (5). For the sake of simplicity, we consider a special case of cubic segments for the standard range of parameters $0 \le t \le 1$. In this case, according to formulas (1) and (2), the spline curve can be described as follows:

$P_i(t)=B_{i0}^{(1)}+B_{i1}^{(1)}t+B_{i2}^{(1)}t^2+B_{i3}^{(1)}t^3, 0\le t\le 1,$ where

$B_{i0}^{(1)}=P_i, B_{i1}^{(1)}=P_i', B_{i2}^{(1)}=3(P_{i+1}-P_i)-2P_i'-P_{i+1}', B_{i2}^{(1)}=2(P_i-P_{i+1})+P_i'+P_{i+1}'.$

The equation defining the Bezier segment according to (5) for $0 \le t \le 1$ can be rewritten as follows:

$$P_i(t)=(1-t)^3 B_{i0}^{(2)}+3t(1-t)^2 B_{i1}^{(2)}+3t^2(1-t)B_{i2}^{(2)}+t^3B_{i3}^{(2)}=$$
$$=\left(3B_{i1}^{(2)}-3B_{i2}^{(2)}+B_{i3}^{(2)}-B_{i0}^{(2)}\right)t^3+\left(3B_{i0}^{(2)}-6B_{i1}^{(2)}+3B_{i2}^{(2)}\right)t^2+\left(3B_{i1}^{(2)}-3B_{i2}^{(2)}\right)t+B_{i0}^{(2)}.$$

Using expressions (6), (7) for the boundary derivatives of the Bezier segment and equating the coefficients at the same degrees, we obtain

$B_{i0}^{(1)}=B_{i0}^{(2)}=P_i,$

$B_{i1}^{(1)} = 3B_{i1}^{(2)} - 3B_{i0}^{(2)} = 3\left(B_{i0}^{(2)} + (1/3)P_i'\right) - 3B_{i0}^{(2)} = P_i'$,

$B_{i2}^{(1)} = 3B_{i0}^{(2)} - 6B_{i1}^{(2)} + 3B_{i2}^{(2)} = 3B_{i0}^{(2)} - 6\left(B_{i0}^{(2)} + (1/3)P_i'\right) + 3\left(B_{i3}^{(2)} - (1/3)P_{i+1}'\right) = 3\left(P_{i+1} - P_i\right) - 2P_i' - P_{i+1}'$,

$B_{i2}^{(1)} = 3B_{i1}^{(2)} - 3B_{i2}^{(2)} + B_{i3}^{(2)} - B_{i0}^{(2)} = 3\left(B_{i0}^{(2)} + (1/3)P_i'\right) - 3\left(B_{i3}^{(2)} - (1/3)P_{i+1}'\right) + B_{i3}^{(2)} - B_{i0}^{(2)} =$

$= 2\left(P_i - P_{i+1}\right) + P_i' + P_{i+1}'$.

Thus, for this case Bezier segments are equivalent to the cubic splines of defect 1 defined by formulas (1) and (2).

3. Problem statement. The technique of converting a spline curve into a B-spline

Let the original spline curve consist of *ns* segments and (t_0, t_1,..., t_{ns}) is a vector of parameters; P_0, P_1,..., P_{ns} and P_0', P_1',..., P_{ns}' are the set of points and tangent vectors at the ends of segments. Our goal is to transform a spline curve defined as a set of segments into a B-spline form. To achieve this goal we produce a Bezier segment for each segment of the spline curve. The entire set of Bezier segments is transformed into a B-spline curve by means of the interpolation of the endpoints by adding multiples knot values, that is:

1. For each spline segment i, the Bezier segment is calculated using the formulas:
   $B_{i0} = P_i,\ B_{i1} = B_{i0} + (1/3)(t_{i+1} - t_i)P_i',\ B_{i2} = P_{i+1} - (1/3)(t_{i+1} - t_i)P_{i+1}',\ B_{i3} = P_{i+1}$.
2. Let B_{ij} be a control point with the number j corresponding to a segment i. Assuming that $B_{00}=P_0$, $B_{03}=B_{10}=P_1$, $B_{13}=B_{20}=P_2$, ..., $B_{i3}=B_{i+1,0}=P_i$, ..., $B_{ns-2,3}=B_{ns-1,0}=P_{ns-1}$, $B_{ns-1,3}=P_{ns}$; we produce the defining polygon of B-spline in the following form: (B_{00},B_{01},B_{02},B_{03},B_{11},B_{12},B_{13},...$B_{ns-1,1}$,$B_{ns-1,2}$,$B_{ns-1,3}$), $n=p\cdot ns$, where $n+1$ is the number of control points of the defining polygon.
3. To interpolate the endpoints B_{03}, B_{13}, ..., $B_{ns-2,3}$, $B_{ns-1,3}$ we set the clamped knot vector with multiple knot values in the following form:
   (t_0, t_0, t_0, t_0, t_1, t_1, t_1, t_2, t_2, t_2, ... , t_{ns-1}, t_{ns-1}, t_{ns-1}, t_{ns}, t_{ns}, t_{ns}, t_{ns}), $nk=p\cdot(ns+1)+2$.

Thus, the original spline curve can be transformed into B-splines with multiple knot values. The computational complexity of such a transformation is O(*ns*). The resulting B-spline interpolates a subset of the control points corresponding to the set of points P_0, P_1,..., P_{ns} of a multi-segment spline curve. The task set in this way can be considered to be solved, but the resulting representation of the curve in the B-spline form is redundant in terms of the number of segments and can be simplified by converting it into a more compact representation containing fewer control points and fewer knots, respectively. The corresponding transformation can be done using the knot removal algorithm, which is discussed below.

4. Algorithms for single knot insertion and removal

Algorithms for knot insertion and removal are reviewed in [5,12]. A proof of the single knot insertion algorithm alternative to [5] is given below.

Let ($B_0^{(1)}$,..., $B_n^{(1)}$) be the defining polygon of the original B-spline curve and $T^{(1)} = (t_0,\ldots,t_r)$ is its knot vector. Suppose you need to insert a knot $\bar{t} \in \left[t_i, t_{i+1}\right)$, $p \le i < r - p$. Let's state a problem to construct a new B-spline description without changing the shape of the curve. The number of elements of the node vector is a function of the sum of the number of control points and the degree of the curve, therefore, with the degree unchanged, the number of control points after a single knot insertion increases by one. If at the same time ($B_0^{(2)}$,..., $B_{n+1}^{(2)}$) determine the defining polygon of the transformed curve, $N_{j,p}(t)$ and $\bar{N}_{j,p}(t)$ are the basis functions of the original and transformed curve and

$$T^{(2)} = (\bar{t}_0 = t_0, \ldots, \bar{t}_i = t_i, \bar{t}_{i+1} = t, \bar{t}_{i+2} = t_{i+1}, \bar{t}_{r+1} = t_r) \; - \tag{8}$$
is the knot vector of the transformed curve, then

$$\sum_{j=0}^{n} B_j^{(1)} N_{j,p}(t) = \sum_{j=0}^{n+1} B_j^{(2)} \bar{N}_{j,p}(t).$$

As the shape of the curve remains unchanged, then, according to formula (4), for $\forall t \in [t_i, t_{i+1})$

$$B_j^{(1)} = B_j^{(2)}, N_{j,p}(t) = \bar{N}_{j,p}(t) \text{ for } j=0, \ldots, i-p-1, \tag{9}$$

$$B_j^{(1)} = B_{j+1}^{(2)}, N_{j,p}(t) = \bar{N}_{j,p}(t) \text{ for } j=i+1, \ldots, n, \tag{10}$$

$$\sum_{j=i-p}^{i} B_j^{(1)} N_{j,p}(t) = \sum_{j=i-p}^{i+1} B_j^{(2)} \bar{N}_{j,p}(t). \tag{11}$$

It can be shown that the basis functions of the original curve can be expressed in terms of the basis functions of the transformed curve as follows:

$$N_{j,p}(t) = \frac{\bar{t} - \bar{t}_j}{\bar{t}_{j+p+1} - \bar{t}_j} \bar{N}_{j,p}(t) + \frac{\bar{t}_{j+p+2} - \bar{t}}{\bar{t}_{j+p+2} - \bar{t}_{j+1}} \bar{N}_{j+1,p}(t) \text{ for } j=i-p, \ldots, i. \tag{12}$$

The idea of the proof of (12) is given in [5], a proof for non-normalized B-splines is given in [12], divided differences based proof is given in [21]. A simpler proof of formula (12) is given below. It is easy to see the relation between this formula and the Cox-de Boor formula (3):

$$N_{j,p}(t) = \bar{N}_{j,p+1}(t). \tag{13}$$

Formula (12) can be proved based on formulas (3) and (13) by induction on *p* and taking into account (8). For an inductive step, we obtain:

$$N_{j,p+1}(t) = \frac{t - t_j}{t_{j+p+1} - t_j} N_{j,p}(t) + \frac{t_{j+p+2} - t}{t_{j+p+2} - t_{j+1}} N_{j+1,p}(t) = \frac{\bar{t} - \bar{t}_j}{\bar{t}_{j+p+2} - \bar{t}_j} N_{j,p}(t) + \frac{\bar{t}_{j+p+3} - \bar{t}}{\bar{t}_{j+p+3} - \bar{t}_{j+1}} N_{j+1,p}(t) =$$
$$= \frac{\bar{t} - \bar{t}_j}{\bar{t}_{j+p+2} - \bar{t}_j} \bar{N}_{j,p+1}(t) + \frac{\bar{t}_{j+p+2} - \bar{t}}{\bar{t}_{j+p+3} - \bar{t}_{j+1}} \bar{N}_{j+1,p+1}(t).$$

The relation between $B_j^{(1)}$ and $B_j^{(2)}$ for *j=i–p, ..., i* can be obtained by substituting (12) into (11):

$$\left(\frac{\bar{t} - \bar{t}_{i-p}}{\bar{t}_{i+1} - \bar{t}_{i-p}} \bar{N}_{i-p,p} + \frac{\bar{t}_{i+2} - \bar{t}}{\bar{t}_{i+2} - \bar{t}_{i-p+1}} \bar{N}_{i-p+1,p} \right) B_{i-p}^{(1)} + \left(\frac{\bar{t} - \bar{t}_{i-p+1}}{\bar{t}_{i+2} - \bar{t}_{i-p+1}} \bar{N}_{i-p+1,p} + \frac{\bar{t}_{i+3} - \bar{t}}{\bar{t}_{i+3} - \bar{t}_{i-p+2}} \bar{N}_{i-p+2,p} \right) B_{i-p+1}^{(1)} +$$
$$+ \ldots + \left(\frac{\bar{t} - \bar{t}_i}{\bar{t}_{i+p+1} - \bar{t}_i} \bar{N}_{i,p} + \frac{\bar{t}_{i+p+2} - \bar{t}}{\bar{t}_{i+p+2} - \bar{t}_{i+1}} \bar{N}_{i+1,p} \right) B_i^{(1)} = \bar{N}_{i-p,p} B_{i-p}^{(2)} + \bar{N}_{i-p+1,p} B_{i-p+1}^{(2)} + \ldots + \bar{N}_{i+1,p} B_{i+1}^{(2)}.$$

Considering that $\bar{t}_{i+1} = \bar{t}$ and using the knot vector $T^{(1)}$ instead of $T^{(2)}$ we obtain

$$0 = \left(B_{i-p}^{(2)} - B_{i-p}^{(1)} \right) \bar{N}_{i-p,p} + \left(B_{i-p+1}^{(2)} - \frac{t_{i+1} - \bar{t}}{t_{i+1} - t_{i-p+1}} B_{i-p}^{(1)} - \frac{\bar{t} - t_{i-p+1}}{t_{i+1} - t_{i-p+1}} B_{i-p+1}^{(1)} \right) \bar{N}_{i-p+1,p} + \ldots +$$
$$+ \left(B_i^{(2)} - \frac{t_{i+p} - \bar{t}}{t_{i+p} - t_i} B_{i-1}^{(1)} - \frac{\bar{t} - t_i}{t_{i+p} - t_i} B_i^{(1)} \right) \bar{N}_{i-p+1,p} + \left(B_{i+1}^{(2)} - B_i^{(1)} \right) \bar{N}_{i+1,p}.$$

If we set $\alpha_j = (\bar{t} - t_j) / (t_{j+p} - t_j)$ for *j=i–p+1, ..., i*, the coordinates of the new vertices ($B_0^{(2)}, \ldots, B_{n+1}^{(2)}$) can be expressed in terms of the original defining polygon vertices ($B_0^{(1)}, \ldots, B_n^{(1)}$):

$$B_{i-p}^{(2)} = B_{i-p}^{(1)}$$
$$B_j^{(2)} = \frac{t_{j+p} - \bar{t}}{t_{j+p} - t_j} B_{j-1}^{(1)} + \frac{\bar{t} - t_j}{t_{j+p} - t_j} B_j^{(1)} = (1 - \alpha_j) B_{j-1}^{(1)} + \alpha_j B_j^{(1)} \text{ for } j=i-p+1, \ldots, i. \tag{14}$$

$B_{i+1}^{(2)} = B_i^{(1)}$.

By rewriting formulas (14) in reverse order and taking into account (9) and (10), we obtain formulas for a single knot removal:

$B_j^{(1)} = B_j^{(2)}$ for j=0, ..., $i-p$,

$$B_j^{(1)} = \begin{cases} \left(B_j^{(2)} - (1-\alpha_j)B_{j-1}^{(1)}\right)/\alpha_j, \text{ if } \alpha_j > 0 \\ B_{j+1}^{(2)}, \text{ if } \alpha_j = 0 \end{cases} \quad \text{for } j=i-p+1, \ldots, i.$$

$B_j^{(1)} = B_{j+1}^{(2)}$ for j=i+1, ..., n.

In this problem statement, the presented knot removal algorithm can be applied twice for each knot of double multiplicity in order to obtain a more compact B-spline description. Each knot removal operation reduces the number of control points by one without changing the number of B-spline segments making it possible to suggest an algorithm for transformation to an equivalent B-spline representation in linear time, that is, it has computational complexity O(ns) where ns is the number of segments of the original curve in the form of a cubic spline.

This algorithm does not check the possibility of knot removal despite the fact that, in general, according to the property 5 of basis functions, the B-spline curve has a discontinuous first derivative at points of triple multiplicity of the knot value. However, the possibility of double knot removal is justified by the fact that the original cubic spline of defect 1 has the continuous second derivative at the points of internal junction. This makes it possible to abandon the more complex knot removal algorithm discussed in [5]. In general, the technique of converting the cubic spline of defect 1 into a B-spline form can be implemented as an algorithm with computational complexity O(ns).

5. Software implementation

The technique of converting a spline curve into a B-spline form is implemented as a C++ software package. Algorithms for iterative calculation of spline and B-spline curves, algorithms for combining Bezier segments and removing B-spline knots, as well as a general algorithmic implementation of the transformation and optimization procedures for B-spline curves, are implemented as separate functions and software modules. Visualization of curves and basis functions of B-splines is implemented in the form of GNU Octave scientific programming language scripts. The listed software components are freely available at https://github.com/am0606/bsplfit.

6. An example of how algorithms work

Consider the spline curve in the Figure 2a, with the control points and tangent vectors shown in the Table 1.

Table 1. Control points and tangent vectors of the spline curve

Control points	Tangent vectors
(1, 1)	(60, 60)
(3.90873, 2.4881)	(12.2619, -0.357143)
(4.91607, 3.31514)	(8.43441, 10.8827)
(7.24717, 5.63957)	(4.07908, 9.08418)
(10, 6)	(22.2222, -22.2222)

The curve is defined by 5 control points and tangent vectors, that is, it consists of 4 segments. The knot vector is the following: (0.1, 0.2, 0.3, 0.73, 1).

Transformation of a composite spline curve in the Figure 2a into a B-spline form based on the combining of Bezier segments gives the following description of the B-spline curve:

- Knot vector (17 knot values):

(0.1, 0.1, 0.1, 0.1, 0.2, 0.2, 0.2, 0.3, 0.3, 0.3, 0.73, 0.73, 0.73, 1, 1, 1, 1).
- Control points (13 points, n=12): (1, 1), (3, 3), (3.5, 2.5), (3.90873, 2.4881), (4.31746, 2.4762), (4.63492,2.95238), (4.91607,3.31514), (6.125,4.87499), (6.6625,4.3375), (7.24717, 5.63957), (7.61429, 6.45715), (8, 8), (10, 6).

  Removing multiple knots at the endpoints of Bezier segments allows us to obtain the following description of the B-spline curve (Figure 2b):
- Knot vector (11 knot values): (0.1, 0.1, 0.1, 0.1, 0.2, 0.3, 0.73, 1, 1, 1, 1).
- Control points (7 points, n=6): (1,1),(3,3),(4,2),(6,5),(7,4),(8,8),(10,6).

B-spline curve in Figure 2a corresponds to the basis functions set in Figure 1a and B-spline curve in Figure 2b corresponds to the basis functions set in Figure 1b.

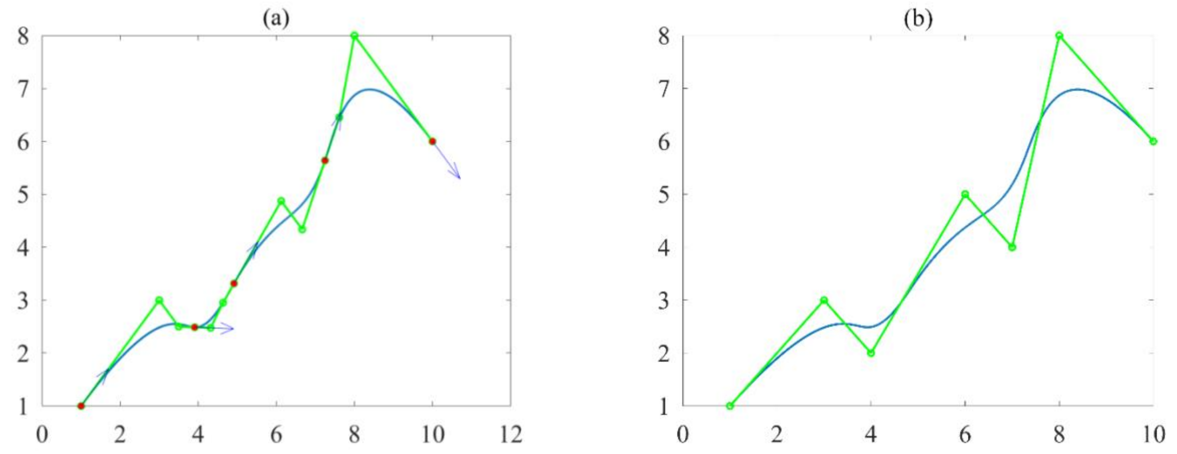

Figure 2. Example of a curve represented by a cubic spline and a B-spline:
(a) A composite spline curve and the corresponding B-spline composed by a set of Bezier segments. Bold dots indicate the endpoints of the spline segments, arrows indicate the directions of tangents along the boundaries of these segments, a polyline is the defining polygon of the B-spline.
(b) Optimal B-spline form. The result obtained by the procedure for removing multiple knots.

Thus, combining Bezier segments equivalent to spline segments makes it possible to obtain a B-spline curve with multiple knots, which can be simplified later using the considered knot removal algorithm. This algorithm is applied 6 times and reduces the number of control points of the B-spline from 13 to 7 by removing degenerate segments.

7. Conclusion

In this paper, we consider the relationship between the representations of curves in the form of cubic splines of defect 1 and B-splines, which can be used to describe the same curve. The representation of the curve as a set of spline segments allows you to interpolate a set of predefined control points, but requires additional information about the derivatives at the segment endpoints. In turn, B-spline curves do not require explicit assignment of derivatives, but, in general, do not interpolate the control points of the defining polygon. At the same time, B-spline is an industry standard convenient for data exchange between different CAD systems, which makes the problem of B-spline fitting to a given set of points as well as converting alternative representations of curves into the B-spline form, important. Such a problem can be efficiently solved for curves defined by cubic splines of defect 1.

Knot insertion allows you to represent the same curve by means of different B-spline descriptions. In this paper, an alternative justification of the algorithms of B-spline knot insertion and removal is given, as well as the conditions for equivalence of cubic splines to third degree Bezier segments. B-spline curves can be composed of individual Bezier segments, and a spline consisting of many segments can be represented as an equivalent B-spline description with multiple knots, which allows you to interpolate the selected vertices.

The B-spline obtained in this way can be transformed into a more compact form using an algorithm for single knot removal. This representation is optimal and fully corresponds to the original description of the curve in the cubic spline form. The knot removal algorithm can also be simplified by taking into account the properties of cubic splines of defect 1.

Thus, in this paper, we propose an original method of B-spline curve construction based on initial representation of this curve in the form of a cubic spline of defect 1 consisting of a set of segments and implement such technique as an algorithm. The described algorithm makes it possible to transform a spline curve into a B-spline form exactly corresponding to it. Such method is an alternative to algorithms for fitting B-spline curves and applying complex interpolation schemes. The algorithm can be implemented as a software module of the geometric core of a CAD system.

References

1. Hiemstra R.R., Shepherd K.M., Johnson M.J., Quan L., Hughes T.J.R., Towards untrimmed NURBS: CAD embedded reparameterization of trimmed B-rep geometry using frame-field guided global parameterization // Comput n Appl Mech Eng. —2020. —Vol.369. —113227.
2. Rogers D.F.,Adams J.A. Mathematical elements for computer graphics. 2nd Edition, McGraw-Hill, New York,1990 (Russ. ed.: *Rogers D.F.,Adams J.A.* Matematicheskie osnovy komp'juternoj grafiki. —M.: Mir, 2001).
3. Faux I.D., Pratt M.J. Computational geometry for design and manufacture, Chichester, West Sussex, John Willey & sons, 1979 (Russ. ed.: Faux I.D., Pratt M.J. Vychislitel'naja geometrija: primenenie v proektirovanii i na proizvodstve. —M.: Mir, 1982).
4. G. Farin. Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide // Academic Press Inc. —1993.
5. Piegl L.A., Tiller W. The NURBS Book. Second edition. New York: Springer–Verlag.—1995–1997.
6. Zadorozhnyi A.G., Kiselev D.S. Postroyenie splaynov s ispol'zovaniem biblioteki OpenGL. — Novosibirsk: NNGU, 2019.
7. Lee E. Rational Bezier Representation for Conies // Geometric Modeling. —1986. — Farin G. (ed.), SIAM. —P. 3–27.
8. Blanc C., Schlick C. Accurate parametrization of conics by NURBS // IEEE Computer Graphics and Applications. —1996. Vol.16, No.6. —P.64–71.
9. Dimas E., Briassoulis D. 3D geometric modelling based on NURBS: a review // Adv Eng Softw. —1999. —Vol.30. — P. 741–751.
10. Wang W., Pottmann H., Liu Y. Fitting B-spline curves to point clouds by curvature-based squared distance minimization // ACM Trans. Graph. —2006. —Vol.25. —P. 214–238.
11. Sun S., Yu D., Wang C., Xie C. A smooth tool path generation and real-time interpolation algorithm based on B-spline curves // Adv Mech Eng. —2018. Vol.10, No.1. —P. 1-14.
12. Golovanov N. Geometric Modeling: The Mathematics of Shapes. CreateSpace Independent Publishing Platform, 2014 (Russ. ed.: Golovanov N. Geometricheskoe modelirovanie. Uchebnoe posobie. —M.: KURS: INFRA-M, 2016).
13. Ueng W.-D., Lai J.-Y., Tsai Y.-C. Unconstrained and constrained curve fitting for reverse engineering // Int J Adv Manuf Technol. —2007. Vol. 33. —P. 1189–1203.
14. Wang G., Shu Q., Wang J. et al. Research on adaptive non-uniform rational B-spline real-time interpolation technology based on acceleration constraints // Int J Adv Manuf Technol. —2017. —Vol.91. —P. 2089–2100.
15. Zhao H., Zhu L., Ding H. A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for CNC machining of short line segments // Int J Mach Tools Manuf. —2013. —Vol. 65. —P. 88–98.

16. Coppersmith D. Winograd S. (1990), Matrix multiplication via arithmetic progressions // J. Symb. Comput. —1990. —Vol. 9 No,3. —P. 251.

17. Cohn H., Kleinberg R., Szegedy B., Umans C. Group-theoretic algorithms for matrix multiplication// 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05). —2005. —P. 379-388.

18. Tveit A. On the complexity of matrix inversion // Mathematical Note. —2003.

19. Krishnamoorthy A., Menon D. Matrix inversion using Cholesky decomposition // Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA). — 2013.— P. 70-72.

20. Pan V. Computer Algorithms for Solving Linear Algebraic Equations // Springer Berlin Heidelberg. —1991. —P.27–56.

21. Cohen E., Lyche T., Riesebfeld R. Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics // Comput Graph Image Process.— 1980. 14. P. 87-111.

22. Prautzsch H. A short proof of the Oslo algorithm // Comput Aided Geom Des. — 1984. —Vol.1,No.1. —P. 95–96.

23. Boehm W. Inserting new knots into B-spline curves // Comput Aided Des. —1980.— Vol.12, No. 4. — P. 199–201.

24. Boehm W., Prautzsch H. The insertion algorithm // Comput Aided Des. —1985.— Vol.17, No.2. —P. 58–59.