On interpolation by spline curves with shape parameters

Miklós Hoffmann[1] and Imre Juhász[2]

[1] Institute of Mathematics and Computer Science, Károly Eszterházy College,
Leányka str. 4., H-3300 Eger, Hungary
hofi@ektf.hu
http://www.ektf.hu/tanszek/matematika/hofi.html
[2] Department of Descriptive Geometry, University of Miskolc,
H-3515 Miskolc-Egyetemváros, Hungary
agtji@uni-miskolc.hu
http://abrpc09.abrg.uni-miskolc.hu/Juhasz/index.html

Abstract. Interpolation of a sequence of points by spline curves generally requires the solution of a large system of equations. In this paper we provide a method which requires only local computation instead of a global system of equations and works for a large class of curves. This is a generalization of a method which previously developed for B-spline, NURBS and trigonometric CB-spline curves. Moreover, instead of numerical shape parameters we provide intuitive, user-friendly, control point based modification of the interpolating curve and the possibility of optimization as well.

Key words: interpolation, spline curve, shape parameter

1 Introduction

Interpolation of an ordered sequence of points is one of the most widely used methods in practical curve modeling, hence there is a vast number of papers and book chapters dealing with this topic (c.f. the books [2], [11]). Designers generally prefer spline curves, where most of the methods work globally. Even in the case of B-spline or NURBS curves, which are standard description methods in curve design and have local control properties, the process of finding control points of an interpolating curve is global and the resulting curve cannot be locally controlled (see e.g. [1], [3], [4], [7]). To overcome this problem some methods have been developed by means of which the shape of the interpolating curve can be adjusted by numerical techniques (see e.g. [8], [9], [12]). Shape parameters and other numerical techniques, however, do not provide intuitive shape control methods such as control point repositioning in approximation. Moreover, in these global methods a large system of equations has to be solved with relatively high computational cost. Especially for large set of data points, local methods have the advantage of solving smaller systems, since the computation of each curve segment is based on only a subset of data. Unfortunately, these local methods typically attain only C^1 continuity [10].

In the last couple of years a new local method has been developed for some types of spline curves that requires only local computation and yields C^2 continuous spline curves. This technique - which is based on linear blending - has been implemented for NURBS in [13], for B-spline curve in [14] and for trigonometric CB-spline curve in [15]. In this method the shape of the interpolating curve can also be adjusted numerically by some shape parameters.

In this paper we will focus on this latter method. In Section 2 we generalize the linear blending interpolation method for a large class of curves. On the other hand, since designers generally prefer geometric entities instead of numerical values, we provide intuitive, control point based modification of the interpolating curve. We let the designer alter the shape of the curve similarly to the approximating curves, meanwhile the interpolation property is continuously preserved. This method will be discussed in Section 3. Finally in Section 4 further generalization in terms of parametrization and the possibility of optimization is provided.

2 Interpolation by linear blending

At first we describe the curve generating method which we refer to as linear blending. Consider the points $\mathbf{p}_i$, $(i = 0, ..., n)$ and the piecewisely defined curve

$$\mathbf{b}(u) = \sum_{i=-1}^{n+1} F_i(u)\mathbf{p}_i, \quad u \in [u_0, u_n], \tag{1}$$

where $\mathbf{p}_i$ are called control points and $F_i(u)$ are basis functions of some space. To ensure that the number of arcs equals the number of control legs, we define two artificial control points $\mathbf{p}_{-1} := \mathbf{p}_0$ and $\mathbf{p}_{n+1} := \mathbf{p}_n$. The only restriction is that each arc of the curve has to be defined by four consecutive control points, that is the arcs of this curve can be written as

$$\mathbf{b}_j(u) = \sum_{l=j-1}^{j+2} F_l(u)\,\mathbf{p}_l, \quad u \in [u_j, u_{j+1}], \quad j = 0, ..., n-1,$$

where the values u_j are called knots.

Now, consider the j^{th} control leg $\mathbf{p}_j, \mathbf{p}_{j+1}, (j = 0, ..., n-1)$ and parameterize this line segment by some function $s(u)$ as

$$\mathbf{l}_j(u) = (1 - s(u))\mathbf{p}_j + s(u)\mathbf{p}_{j+1}, \quad u \in [u_j, u_{j+1}].$$

At this moment the function $s(u)$ can be any function which fulfills two conditions

$$s(u_j) = 0, \quad s(u_{j+1}) = 1. \tag{2}$$

Possible choices of the function will be discussed in detail in Section 4.

Linearly blending the arcs $\mathbf{b}_j(u)$ of the spline curve with the corresponding line segments $\mathbf{l}_j(u)$ we obtain the linear blending curve consisting of the arcs

$$\mathbf{c}_j(u, \alpha) = (1 - \alpha)\mathbf{b}_j(u) + \alpha\mathbf{l}_j(u),$$

where α is a global shape parameter of the curve. To achieve more flexibility in shape modification, the shape parameter α can also be the function of u. A straightforward way is to define it piecewisely by local shape parameters α_i^* associated to each point $\mathbf{p}_i$, and to use the same blending function $s(u)$ as for the line segment, i.e.,

$$\alpha_j(u) = (1 - s(u))\alpha_j^* + s(u)\alpha_{j+1}^*.$$

This way each arc of the linear blending curve will have three parameters, the running parameter u and the two local shape parameters $\alpha_j^*, \alpha_{j+1}^*$:

$$\mathbf{c}_j(u, \alpha_j^*, \alpha_{j+1}^*) = (1 - \alpha_j(u))\mathbf{b}_j(u) + \alpha_j(u)\mathbf{l}_j(u),$$

that is

$$\begin{aligned}\mathbf{c}_j(u, \alpha_j^*, \alpha_{j+1}^*) = (1 - (1 - s(u))\alpha_j^* - s(u)\alpha_{j+1}^*)\mathbf{b}_j(u) + \\ ((1 - s(u))\alpha_j^* + s(u)\alpha_{j+1}^*)\mathbf{l}_j(u).\end{aligned} \tag{3}$$

In order to obtain C^m continuity of the consecutive arcs of the curve, the k^{th} derivatives of the function $s(u)$ have to satisfy the conditions

$$s^{(k)}(u_j) = 0, s^{(k)}(u_{j+1}) = 0 \quad 1 \leq k \leq m, \quad j = 0, ..., n-1. \tag{4}$$

Now, based on the idea described in [13], we modify the linear blending method to interpolate a given sequence of points. As we have seen the curve arc $\mathbf{c}_j(u, \alpha_j^*, \alpha_{j+1}^*)$ is "between" the arc $\mathbf{b}_j(u)$ and the control leg $\mathbf{l}_j(u)$, that is the curve approximates the given points (except the case when all the shape parameters are set to be 1). But there is no specific reason to choose the control leg as the blended line segment, so we can define the linear blending of the spline arc $\mathbf{b}_j(u)$ with some other line segment as well. In this way we can obtain an appropriate line segment by means of which the resulted curve will interpolate the given points.

Let the points $\mathbf{p}_i$, associated parameter values u_i and shape parameters α_i^*, $(i = 0, ..., n)$ be given. If the parameter values u_i are unknown, then choose them by some of the well-known methods, like centripetal or chord length method - these are only initial values which will be allowed to be changed during the interactive modification of the interpolating curve. Now, the problem is to find a linear blending curve having the given shape parameters α_i^*, which interpolates the given points $\mathbf{p}_i$ at the given parameter values u_i.

Consider the approximating spline curve defined by the given points as control points

$$\mathbf{b}_j(u) = \sum_{l=j-1}^{j+2} F_l(u)\,\mathbf{p}_l, \quad u \in [u_j, u_{j+1}], \quad j = 0, ..., n-1.$$

Also consider the parameterized line segments

$$\mathbf{l}_j(u) = (1 - s(u))\mathbf{v}_j + s(u)\mathbf{v}_{j+1}, \quad j = 0, ..., n-1,$$

where the points $\mathbf{v}_j$ are unknown. Analogously to (3), arcs of the linear blending curve can be written as

$$\begin{aligned}\mathbf{c}_j(u,\alpha_j^*,\alpha_{j+1}^*) = (1-(1-s(u))\alpha_j^* - s(u)\alpha_{j+1}^*)\mathbf{b}_j(u) + \\ ((1-s(u))\alpha_j^* + s(u)\alpha_{j+1}^*)((1-s(u))\mathbf{v}_j + s(u)\mathbf{v}_{j+1}).\end{aligned} \tag{5}$$

After some calculations, the interpolation assumptions yield equalities

$$\mathbf{v}_j(u_j,\alpha_j^*) = \mathbf{p}_j + \frac{(1-\alpha_j^*)}{\alpha_j^*}(\mathbf{p}_j - \mathbf{b}(u_j)) \quad j=0,...,n. \tag{6}$$

Using these points, the curve composed of the arcs (5) will interpolate the given points at the given parameter values.

Since we had very little assumptions about the initial base curve $\mathbf{b}(u)$, especially no assumptions about its basis functions $F_i(u)$ and their space, this general method includes the methods described in [13], [14] and [15] as special cases.

As one can observe, there are infinitely many possibilities to choose the points $\mathbf{v}_j$. In the following Section we will discuss the geometric consequences of the alteration of these points, which allows us to modify the interpolating curve in an intuitive way.

3 Control point based modification

Each of the points $\mathbf{v}_i$ depends on two parameters: the corresponding parameter value u_i and the local shape parameter α_i^*. Instead of manipulating these values numerically and calculate the points $\mathbf{v}_i$ and finally the interpolating curve arcs $\mathbf{c}_j(u,\alpha_j^*,\alpha_{j+1}^*)$, our main idea is to start to think in the opposite direction: what happens to these values and what happens to the curve if we change the position of the points $\mathbf{v}_i$?

3.1 Altering the shape parameter α_i^*

The modification of the point $\mathbf{v}_i$ affects only two consecutive arcs of the interpolating curve. If the parameter value u_i is fixed, then due to Equation (6) the point $\mathbf{v}_i$ can move along a straight line defined by the points $\mathbf{p}_i$ and $\mathbf{b}(u_i)$.

What is the geometric effect of this modification to the points of the curve? We describe the movement of a fixed point of the curve arc $\mathbf{c}_j(u,\alpha_j^*,\alpha_{j+1}^*)$ as a function of α_j^*. Fixing $u=\bar{u}$ and $\alpha_{j+1}^* = \bar{\alpha}_{j+1}^*$ and substituting Equation (6) into (5), after some calculation we get

$$\mathbf{c}_j(\bar{u},\alpha_j^*,\bar{\alpha}_{j+1}^*) = \mathbf{A}\alpha_j^* + \frac{\mathbf{B}}{\alpha_j^*} + \mathbf{C}, \quad \alpha_j^* \in \mathbb{R}\setminus\{0\},$$

where $\mathbf{A},\mathbf{B}$ and $\mathbf{C}$ are constants, depending on the knot values u_j, u_{j+1}, the fixed parameters $\bar{u},\bar{\alpha}_{j+1}^*$ and the positions of the points $\mathbf{p}_j,\mathbf{p}_{j+1}$

$$\mathbf{A} = (s(\bar{u})-1)\mathbf{b}(\bar{u}) + (1-s(\bar{u}))((1-s(\bar{u}))\mathbf{b}(u_j) + s(\bar{u})\mathbf{v}_{j+1}),$$

$$\mathbf{B} = s(\bar{u})\,\bar{\alpha}^*_{j+1}(1 - s(\bar{u}))(\mathbf{p}_j - \mathbf{b}(u_j)),$$
$$\mathbf{C} = (1 - s(\bar{u})\,\bar{\alpha}^*_{j+1})\mathbf{b}(\bar{u}) + s(\bar{u})\,\bar{\alpha}^*_{j+1}((1 - s(\bar{u}))\mathbf{b}(u_j) + s(\bar{u})\,\mathbf{v}_{j+1}) + (1 - s(\bar{u}))^2(\mathbf{p}_j - \mathbf{b}(u_j)).$$

This is the equation of a hyperbola with center $\mathbf{C}$ and asymptotic directions $\mathbf{A}$ and $\mathbf{B}$. This means, that modifying the position of the point $\mathbf{v}_j$ along the straight line defined by Equation (6), points of the interpolating curve will move along a well-defined hyperbola (see Fig.1).

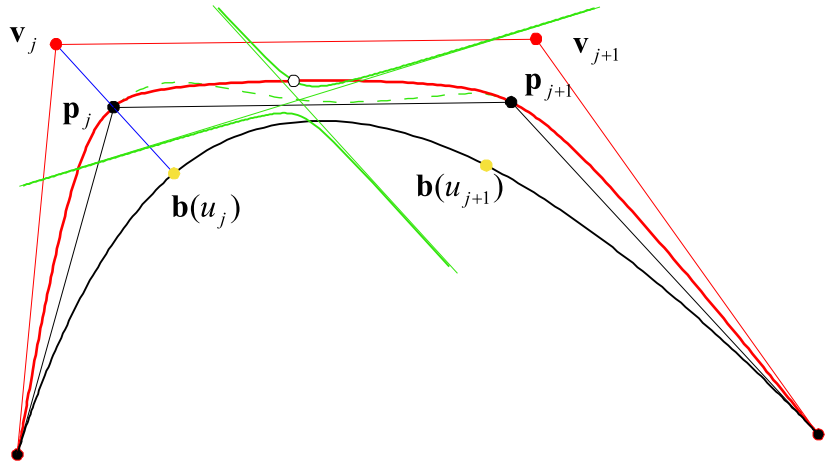

Fig. 1. The interpolating curve (thick red), the base curve (black) and the line segments (red). Modifying the shape parameter α^*_j, the vertex $\mathbf{v}_j$ moves along the line $\mathbf{p}_j\mathbf{b}(u_j)$, while the points of the interpolating curve arc $\mathbf{p}_j\mathbf{p}_{j+1}$ move along hyperbolas. One of the asymptotes of these hyperbolas is always parallel to the direction $\mathbf{p}_j - \mathbf{b}(u_j)$. The green dashed line is the set of the centers of these hyperbolas. Figure shows a cubic B-spline implementation.

Moreover, one of the asymptotes of these hyperbolas are parallel to the direction $\mathbf{p}_j - \mathbf{b}(u_j)$ $\forall u \in [u_j, u_{j+1}]$, and the movement of the points of the interpolating curve flows typically along this asymptote. This hyperbola is degenerated to a straight line when $j = 0$.

3.2 Altering the parameter value u_i

As we have seen, each of the points $\mathbf{v}_i$ depends on two parameters: the corresponding parameter value u_i and the local shape parameter α^*_i. If the shape parameter value α^*_i is fixed, and we let the parameter value u_i vary between u_{i-1} and u_{i+1} then the movement of the point $\mathbf{v}_i$ and finally the change of the shape of the curve will be a little more complicated. This is a consequence of the fact that the value u_i also serves as a knot value of the original base curve

$\mathbf{b}(u)$. Therefore, the alteration of the knot value of a curve changes the shape of this base curve as well. This alteration has been studied for B-spline and NURBS curves in [5], [6]. In general, altering the value $u_i \in [u_{i-1}, u_{i+1}]$ we get a one-parameter family of base curves $\mathbf{b}(u, u_i)$ with family parameter u_i. Let us consider those points of these curves that correspond to $u = u_i$, that is the points $\mathbf{b}(u_i, u_i)$. These points form a curve which will be denoted by

$$\mathbf{e}(u_i) = \mathbf{b}(u_i, u_i), \quad u_i \in [u_{i-1}, u_{i+1}]. \tag{7}$$

It is easy to see from Equation (6), that altering the value u_i, the point $\mathbf{v}_i$ will move along a path curve that can be obtained by a central similarity from the curve (7), where the center of similitude is the given point $\mathbf{p}_i$, and the ratio is $(1 - \alpha_i^*)/\alpha_i^*$

$$\mathbf{v}_i(u_i) = \mathbf{p}_i + \frac{(1 - \alpha_i^*)}{\alpha_i^*} \left(\mathbf{p}_i - \mathbf{e}(u_i)\right). \tag{8}$$

3.3 Repositioning of $\mathbf{v_i}$

As we have learned from the previous subsections, the position of the point $\mathbf{v}_i$ depends on u_i and α_i^* in such a way, that modifying α_i^* the point will move along straight lines passing through $\mathbf{p}_i$, while modifying u_i the point will move along similar path curves (8), where the center of similitude is $\mathbf{p}_i$. Changing the direction of this deduction simply means that repositioning the point $\mathbf{v}_i$ one can calculate the actual values of u_i and α_i^* by finding the corresponding line and a path curve.

The permissible positions of $\mathbf{v}_i$ are on the surface $\mathbf{v}_i(u_i, \alpha_i^*)$ (or a planar region in degenerate cases). The isoparametric lines of this surface are straight lines passing through $\mathbf{p}_i$ and the path curves (8), which means that the permissible positions of $\mathbf{v}_i$ are on the surface of a cone, the apex of which is $\mathbf{p}_i$, and the two limiting generators are the lines connecting $\mathbf{e}(u_{i-1})$, $\mathbf{p}_i$, and $\mathbf{e}(u_{i+1})$, $\mathbf{p}_i$, respectively. If the cone degenerates to a planar region these two lines define the boundary of the permissible region (see Fig.2).

These results enable us to modify the shape of the interpolating curve by an intuitive way, that is similar to control point repositioning in the case of approximating curves. Given a sequence of points $\mathbf{p}_i$, associated parameter values u_i and shape parameters α_i^*, $(i = 0, ..., n)$, we can calculate the initial interpolating curve piecewisely by Equation (5). Then we can change the position of any point $\mathbf{v}_i$ inside its region of permissible positions, without dealing with numerical values and parameters. The effect of this repositioning is described above, the actual values of the parameters can automatically be calculated, thus the new curve will remain interpolating curve. Although, these points are not "real" control points of the curve, the effect of their modifications is intuitive and the designers do not have to bother with numerical parameters.

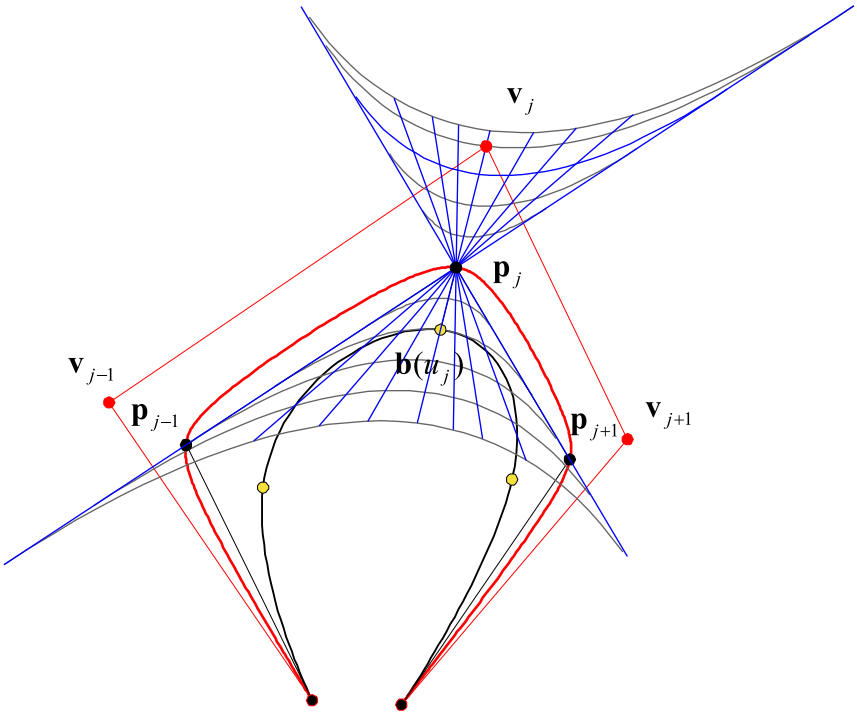

Fig. 2. The interpolating curve (thick red) and its control points $\mathbf{v}_i$. Permissible positions of the point $\mathbf{v}_j$ are on a cone (or planar region). The α_j^* isoparametric curves are straight lines (blue), while the u_i isoparametric curves are path curves (8) (grey). Figure shows a cubic B-spline implementation.

4 Parametrization of the line segment

One of the key steps of constructing the interpolating curve (5) is the parametrization of the line segment $\mathbf{l}_j(u)$ by means of the function $s(u)$. This function is also used for the definition of the shape parameter function $\alpha_j(u)$ from the local shape parameters α_j^* and α_{j+1}^*. Beside the obvious endpoint conditions (2) some continuity conditions (4) also have to be fulfilled by this function. Certainly, the choice of this function depends on the base curve $\mathbf{b}(u)$, more precisely the type of its basis functions $F_i(u)$. If, for example, the base curve is a cubic polynomial curve, the function $s(u)$ should also be cubic polynomial in order to achieve the interpolating curve to be cubic polynomial curve as well. However, in this specific case the function $s(u)$ is bound to be higher degree polynomial by the continuity conditions mentioned above. In [13] the minimum degree polynomial is found to be the quintic Hermite polynomial $6u^5 - 15u^4 + 10u^3$.

Tai and Wang [14] suggested piecewisely defined (i.e. spline) function for $s(u)$ to decrease the degree meanwhile the desired C^2 continuity is preserved.

Instead of the quintic polynomial, they use a function defined piecewisely over the interval $[u_j, u_{j+1}]$ as

$$s(u) = \frac{9}{2}\left(\frac{u-u_j}{u_{j+1}-u_j}\right)^3, \quad u \in \left[u_j, \frac{2}{3}u_j + \frac{1}{3}u_{j+1}\right]$$

$$s(u) = \frac{9}{2}\left[\left(\frac{u-u_j}{u_{j+1}-u_j}\right)^3 - 3\left(\frac{u-u_j}{u_{j+1}-u_j} - \frac{1}{3}\right)^3\right],$$

$$u \in \left[\frac{2}{3}u_j + \frac{1}{3}u_{j+1}, \frac{1}{3}u_j + \frac{2}{3}u_{j+1}\right]$$

$$s(u) = 1 - \frac{9}{2}\left(\frac{u_{j+1}-u}{u_{j+1}-u_j}\right)^3, \quad u \in \left[\frac{1}{3}u_j + \frac{2}{3}u_{j+1}, u_{j+1}\right].$$

A similarly structured trigonometric spline function $s(u)$ has been applied in the case of trigonometric spline base curve in [15].

But these methods can be considered as special cases of the following method defining the appropriate spline function $s(u)$ for parametrization of the line segment. If the base curve is a cubic polynomial curve, then let us consider a general spline function over the interval $[u_j, u_{j+1}]$. We divide the domain of definition into three subintervals with boundaries $u_j < t_1 < t_2 < u_{j+1}$ and we specify cubic polynomials $s_0(u), s_1(u), s_2(u)$ on these smaller ranges. Coefficients of polynomials

$$s_j(u) = a_{j0} + a_{j1}u + a_{j2}u^2 + a_{j3}u^3, \ \ (j = 0, 1, 2)$$

can be computed by means of the continuity constraints. The coefficients of the resulted polynomials are

$$a_{00} = 1,\ a_{01} = 0,\ a_{02} = 0,\ a_{03} = \frac{-1}{t_1 t_2}$$

$$\begin{aligned} a_{10} &= \tfrac{t_2 - t_1 - t_2 t_1}{(t_1-1)(t_1-t_2)}, \\ a_{11} &= \tfrac{3t_1}{(t_1-1)(t_1-t_2)}, \\ a_{12} &= \tfrac{-3}{(t_1-1)(t_1-t_2)}, \\ a_{13} &= \tfrac{1+t_2-t_1}{(t_1-1)(t_1-t_2)t_2} \end{aligned}$$

$$\begin{aligned} a_{20} &= \tfrac{1}{(1-t_1)(1-t_2)}, \\ a_{21} &= \tfrac{-3}{(1-t_1)(1-t_2)}, \\ a_{22} &= \tfrac{3}{(1-t_1)(1-t_2)}, \\ a_{23} &= \tfrac{-1}{(1-t_1)(1-t_2)}. \end{aligned}$$

Thus we have two more free parameters t_1, t_2 that heavily affect the overall parametrization of the interpolating curve, which can be used for optimization,

especially because they have only minor effects on the shape of the curve. Moreover, for other types of base curves this technique can also be applied, even for higher order C^m continuity, when the range has to be divided into $m+1$ parts and the spline function has to be defined over these intervals.

5 Conclusion

An interpolation method has been provided in this paper, that linearly blends the arcs of a base curve and the line segments of a calculated polygon. The method works for a large class of spline curves, actually the only restriction is that each arc has to be determined by four consecutive control points. The method is a generalization of formerly published techniques.

A user-friendly, control point based shape control of the interpolating curve is also proposed, where the vertices of the calculated polygon play the role of control points. The effect of repositioning of these points on the shape is very intuitive, the permissible area of the points are large and easily defined. Therefore, the user does not have to bother with numerical parameters in the fine tuning of the interpolating curve.

Acknowledgements

The authors wish to thank the Hungarian National Scientific Fund (OTKA No. T048523) and the National Office of Research and Technology (Project CHN-37/2005) for their financial support of this research.

References

1. Barsky, B.A., Greenberg, D.P., Determining a set of B-spline control vertices to generate an interpolating surface, Computer Graphics and Image Processing, 14 (1980), 203—226.
2. Hoschek, J., Lasser, D.: Fundamentals of CAGD, AK Peters, Wellesley, MA, 1993.
3. Lavery, J.E., Univariate cubic Lp splines and shape-preserving multiscale interpolation by univariate cubic L1 splines, Computer Aided Geometric Design, 17 (2000), 319—336.
4. Ma, W., Kruth, J.P., NURBS curve and surface fitting and interpolation, in: M.Daehlen, T.Lyche, L.Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Vanderbilt University Press, Nashville & London, 1995.
5. Juhász, I., Hoffmann, M., The effect of knot modifications on the shape of B-spline curves, Journal for Geometry and Graphics, 5 (2001), 111–119.
6. Juhász, I., Hoffmann, M., Modifying a knot of B-spline curves, Computer Aided Geometric Design, 20 (2003), 243–245.
7. Juhász, I., Hoffmann, M., On parametrization of interpolating curves, Journal of Computational and Applied Mathematics, (2007), doi:10.1016/j.cam.2007.05.019
8. Kaklis, P.D., Sapidis, N.S., Convexity-preserving interpolatory parametric splines of non-uniform polynomial degree, Computer Aided Geometric Design, 12 (1995), 1—26.

9. Kong, V.P., Ong, B.H., Shape preserving F3 curve interpolation, Computer Aided Geometric Design, 19 (2002), 239—256.
10. Li, A., Convexity preserving interpolation, Computer Aided Geometric Design, 16 (1999), 127—147.
11. Piegl, L., Tiller, W., The NURBS Book, Springer, Berlin. 1995
12. Sapidis, N., Farin, G., Automatic fairing algorithm for B-spline curves, Computer-Aided Design, 22 (1990), 121—129.
13. Tai, C.-L., Barsky, B.A., Loe, K.-F., An interpolation method with weight and relaxation parameters. In: Cohen, A., Rabut, C., Schumaker, L.L. (eds.): Curve and Surface Fitting: Saint-Malo 1999. Vanderbilt Univ. Press, Nashville, Tenessee (2000) 393–402.
14. Tai, C.-L., Wang, G.-J., Interpolation with slackness and continuity control and convexity-preservation using singular blending. J. Comp. Appl. Math., **172** (2004), 337–361.
15. Pan, Y.-J., Wang, G.-J., Convexity-preserving interpolation of trigonometric polynomial curves with shape parameter. Journal of Zhejiang University Science A, **8** (2007), 1199–1209.