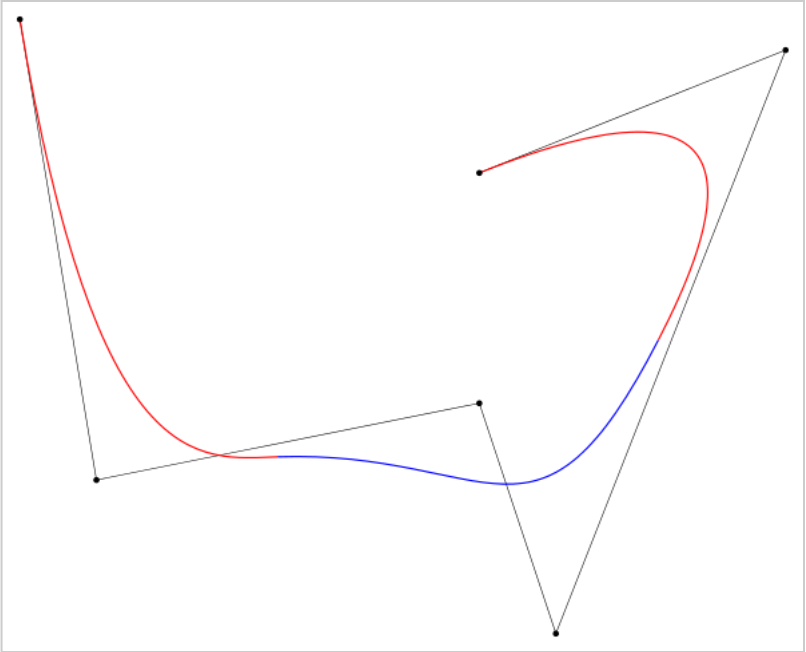
B-spline

From Wikipedia, the free encyclopedia

In the mathematical subfield of numerical analysis, a **B-spline**, or **basis spline**, is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. Any spline function of given degree can be expressed as a linear combination of B-splines of that degree. Cardinal B-splines have knots that are equidistant from each other. B-splines can be used for curve-fitting and numerical differentiation of experimental data.

In the computer-aided design and computer graphics, spline functions are constructed as linear combinations of B-splines with a set of control points.



B-spline with control points/control polygon, and marked component curves

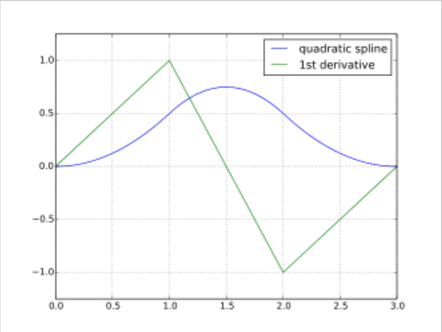
Contents

Introduction

The term "B-spline" was coined by Isaac Jacob Schoenberg[1] and is short for basis spline.[2] A spline function is a piecewise polynomial function of degree $<k$ in a variable x. The places where the pieces meet are known as knots. The number of internal knots must be equal to, or greater than k-1. Thus the spline function has limited support. The key property of spline functions is that they are continuous at the knots. Some derivatives of the spline function may also be continuous, depending on whether the knots are distinct or not. A fundamental theorem states that every spline function of a given degree, smoothness, and domain partition, can be uniquely represented as a linear combination of B-splines of that same degree and smoothness, and over that same partition.

Definition

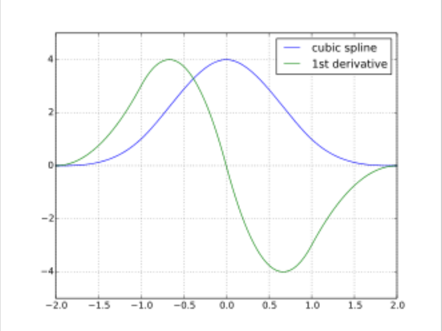
A B-spline of order *n* is a piecewise polynomial function of degree <*n* in a variable *x*. It is defined over a domain $t_0 \le x \le t_m$, m = n. The points where $x = t_j$ are known as knots or break-points. The number of internal knots is equal to the degree of the polynomial if there are no knot multiplicities. The knots must be in ascending order. The number of knots is the minimum for the degree of the B-spline, which has a non-zero value only in the range between the first and last knot. Each piece of the function is a polynomial of degree <*n* between and including adjacent knots. A B-spline is a continuous function at the knots.[note 1] When all internal knots are distinct, its derivatives are also continuous up to the derivative of degree *n*-1. If internal knots are coincident at a given value of *x*, the continuity of derivative order is reduced by 1 for each additional knot.



Cardinal quadratic B-spline with knot vector (0, 0, 0, 1, 2, 3, 3, 3) and control points (0, 0, 1, 0, 0), and its first derivative

For any given set of knots, the B-spline is unique, hence the name, B being short for Basis. The usefulness of B-splines lies in the fact that any spline function of order *n* on a given set of knots can be expressed as a linear combination of B-splines:

$$S_{n,t}(x) = \sum_i \alpha_i B_{i,n}(x).$$



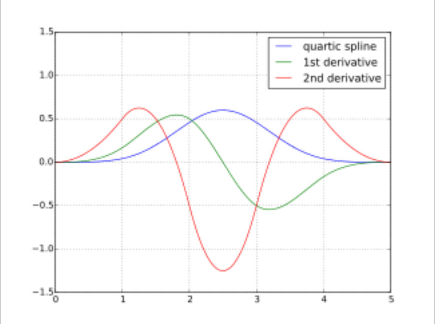
Cardinal cubic B-spline with knot vector (−2, −2, −2, −2, −1, 0, 1, 2, 2, 2, 2) and control points (0, 0, 0, 6, 0, 0, 0), and its first derivative

This follows from the fact that all pieces have the same continuity properties, within their individual range of support, at the knots.[3]

Expressions for the polynomial pieces can be derived by means of the Cox-de Boor recursion formula[4]

$$B_{i,1}(x) := \begin{cases} 1 & \text{if} \quad t_i \le x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,k}(x) := \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x).$$ [5]



Cardinal quartic B-spline with knot vector (0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5) and control points (0, 0, 0, 0, 1, 0, 0, 0, 0), and its first and second derivatives

That is, $B_{j,1}(x)$ is piecewise constant one or zero indicating which knot span *x* is in (zero if knot span *j* is repeated). The recursion equation is in two parts:

$$\frac{x - t_i}{t_{i+k-1} - t_i}$$

ramps from zero to one as *x* goes from t_i to t_{i+k-1} and

$$\frac{t_{i+k} - x}{t_{i+k} - t_{i+1}}$$

ramps from one to zero as *x* goes from t_{i+1} to t_{i+k}. The corresponding *B*s are zero outside those respective ranges. For example, $B_{i,2}(x)$ is a triangular function that is zero below $x = t_i$, ramps to one at $x = t_{i+1}$ and back to zero at and beyond $x = t_{i+2}$. However, because B-spline basis functions have local support, B-splines are typically computed by algorithms that do not need to evaluate basis functions where they are zero, such as de Boor's algorithm.

This relation leads directly to the FORTRAN-coded algorithm BSPLV which generates values of the B-splines of order *n* at *x*.[6] The following scheme illustrates how each piece of degree *n* is a linear combination of the pieces of B-splines of degree *n*-1 to its left.

$$\begin{array}{ccc}
 & & 0 \\
 & 0 & \\
0 & & B_{i-2,3} \\
 & B_{i-1,2} & \\
B_{i,1} & & B_{i-1,3} \\
 & B_{i,2} & \\
0 & & B_{i,3} \\
 & 0 & \\
 & & 0
\end{array}$$

Application of the recursion formula with the knots at 0, 1, 2, and 3 gives the pieces of the uniform B-spline of degree 2

$$\begin{aligned}
&B_1 = x^2/2 \qquad 0 \le x \le 1 \\
&B_2 = (-2x^2 + 6x - 3)/2 \qquad 1 \le x \le 2 \\
&B_3 = (3 - x)^2/2 \qquad 2 \le x \le 3
\end{aligned}$$

These pieces are shown in the diagram. The continuity property of a quadratic spline function and its first derivative at the internal knots are illustrated, as follows

$$\begin{aligned}
&\text{At } x = 1, B_1 = B_2 = 0.5; \frac{dB_1}{dx} = \frac{dB_2}{dx} = 1 \\
&\text{At } x = 2, B_2 = B_3 = 0.5; \frac{dB_2}{dx} = \frac{dB_3}{dx} = -1
\end{aligned}$$

The second derivative of a B-spline of degree 2 is discontinuous at the knots:

$$\frac{d^2B_1}{dx^2} = 1, \frac{d^2B_2}{dx^2} = -2, \frac{d^2B_3}{dx^2} = 1.$$

Faster variants of the de Boor algorithm have been proposed but they suffer from comparatively lower stability.[7][8]

Cardinal B-spline

A cardinal B-spline has a constant separation, *h*, between knots. The cardinal B-splines for a given degree *n* are just shifted copies of each other. They can be obtained from the simpler definition.[9]

$$B_{i,n,t}(x) = \frac{x - t_i}{h} n[0, \ldots, n](. -t_i)_+^{n-1}$$

The "placeholder" notation is used to indicate that the *n*th divided difference of the function $(t - x)_+^{n-1}$ of the two variables *t* and *x* is to be taken by fixing *x* and considering $(t - x)_+^{n-1}$ as a function of *t* alone.

A cardinal B-spline has uniform spaced knots, therefore interpolation between the knots equals convolution with a smoothing kernel.

Example, if we want to interpolate three values inbetween B-spline nodes ($\mathbf{b}$), we can write the signal as:

$$\mathbf{x} = [\mathbf{b}_1, 0, 0, \mathbf{b}_2, 0, 0, \mathbf{b}_3, 0, 0, \ldots., \mathbf{b}_n, 0, 0]$$

Convolution of the signal $\mathbf{x}$ with a rectangle function $\mathbf{h} = [1/3, 1/3, 1/3]$ gives first order interpolated b-spline values. Second-order B-spline interpolation is convolution with a rectangle function twice$\mathbf{y} = \mathbf{x} * \mathbf{h} * \mathbf{h}$, by iterative filtering with a rectangle function higher order interpolation is obtained.

Fast b-spline interpolation on a uniform sample domain can be done by iterative mean-filtering. Alternative, a rectangle function equals Sinc in Fourier domain. Therefore cubic spline interpolation equals multiplying the signal in Fourier domain with Sinc^4.

See Irwin–Hall distribution#Special cases for algebraic expressions for the cardinal B-splines of degree 1-4.

P-spline

The term P-spline stands for "penalized B-spline". It refers to using the B-spline representation where the coefficients are determined partly by the data to be fitted, and partly by an additional penalty function that aims to impose smoothness to avoid overfitting.[10]

Derivative expressions

The derivative of a B-spline of degree *k* is simply a function of B-splines of degree *k*-1.[11]

$$\frac{dB_{i,k}(x)}{dx} = (k-1)\left(\frac{-B_{i+1,k-1}(x)}{t_{i+k} - t_{i+1}} + \frac{B_{i.k-1}(x)}{t_{i+k-1} - t_i}\right)$$

This implies that

$$\frac{d}{dx}\sum_i \alpha_i B_{i,k} = \sum_{i=r-k+2}^{s-1} (k-1)\frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1}\ on[t_r.t_s]$$

which shows that there is a simple relationship between the derivative of a spline function and the B-splines of degree one less.

Relationship to piecewise/composite Bézier

A piecewise/composite Bézier curve is a series of Bézier curves joined with at least C0 continuity (the last point of one curve coincides with the starting point of the next curve). Depending on the application, additional smoothness requirements (such as C1 or C2 continuity) may be added.[12] C1 continuous curves have identical tangents at the breakpoint (where the two curves meet). C2 continuous curves have identical curvature at the breakpoint.[13]

To gain C2 continuity the Bézier Curve loses local control, because to enforce C2 continuity the control points are dependent on each other. If a single control point moves, the whole spline needs to be re-evaluated. B-Splines have both C2 continuity and local control, but they lose the interpolation property of a piecewise Bézier.[14]

Curve fitting

Usually in curve fitting, a set of data points is fitted with a curve defined by some mathematical function. For example common types of curve fitting use a polynomial or a set of exponential functions. When there is no theoretical basis for choosing a fitting function, the curve may be fitted with a spline function composed of a sum of B-splines, using the method of least squares.[15][note 2] Thus, the objective function for least squares minimization is, for a spline function of degree *k*,

$$U = \sum_{all x} \left\{ W(x) \left[y(x) - \sum_i \alpha_i B_{i,k,t}(x) \right] \right\}^2$$

W(x) is a weight and *y(x)* is the datum value at *x*. The coefficients α_i are the parameters to be determined. The knot values may be fixed or they too may be treated as parameters.

The main difficulty in applying this process is in determining the number of knots to use and where they should be placed. de Boor suggests various strategies to address this problem. For instance, the spacing between knots is decreased in proportion to the curvature (2nd. derivative) of the data. A few applications have been published. For instance, the use of B-splines for fitting single Lorentzian and Gaussian curves has been investigated. Optimal spline functions of degrees 3-7 inclusive, based on symmetric arrangements of 5, 6, and 7 knots, have been computed and the method was applied for smoothing and differentiation of spectroscopic curves.[16] In a comparable study, the two-dimensional version of the Savitzky-Golay filtering and the spline method produced better results than moving average or Chebyshev filtering.[17]

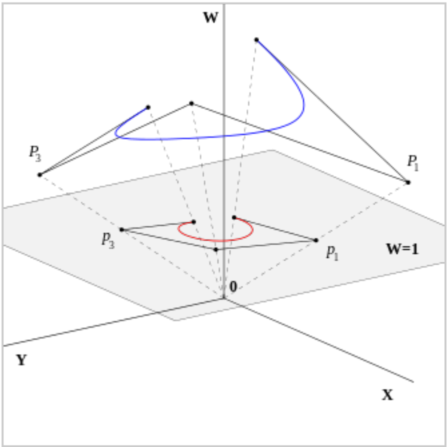
NURBS

In computer aided design, computer aided manufacturing, and computer graphics, a powerful extension of B-splines is non-uniform rational B-splines (NURBS). NURBS are essentially B-splines in homogeneous coordinates. Like B-splines, they are defined by their order, and a knot vector, and a set of control points, but unlike simple B-splines, the control points each have a weight. When the weight is equal to 1, a NURBS is simply a B-spline and as such NURBS generalizes both B-splines and Bézier curves and surfaces, the primary difference being the weighting of the control points which makes NURBS curves "rational".

By evaluating a NURBS at various values of the parameters, the curve can be traced through space; likewise, by evaluating a NURBS surface at various values of the two parameters, the surface can be represented in Cartesian space.

Like B-splines, NURBS control points determine the shape of the curve. Each point of the curve is computed by taking a weighted sum of a number of control points. The weight of each point varies according to the governing parameter. For a curve of degree *d*, the influence of any control point is only nonzero in *d*+1 intervals (knot spans) of the parameter space. Within those intervals, the weight changes according to a polynomial function (basis functions) of degree *d*. At the boundaries of the intervals, the basis functions go smoothly to zero, the smoothness being determined by the degree of the polynomial.

The knot vector is a sequence of parameter values that determines where and how the control points affect the NURBS curve. The number of knots is always equal to the number of control points plus curve degree plus one. Each time the parameter value enters a new knot span, a new control point becomes active, while an old control point is discarded.



NURBS curve – polynomial curve defined in homogeneous coordinates (blue) and its projection on plane – rational curve (red)

A NURBS curve takes the following form:[18]

$$C(u)=\frac{\sum_{i=1}^{k} N_{i,n} w_i P_i}{\sum_{i=1}^{k} N_{i,n} w_i}$$

Here the notation is as follows. *u* is the independent variable (instead of *x*), *k* is the number of control points, *N* is a B-spline (used instead of *B*), *n* is the polynomial degree, *P* is a control point and *w* is a weight. The denominator is a normalizing factor that evaluates to one if all weights are one.

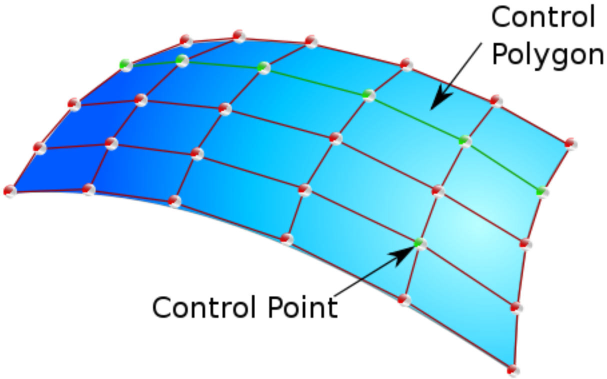
It is customary to write this as

$$C(u)=\sum_{i=1}^{k} R_{i,n}(u) P_i$$

in which the functions

$$R_{i,n}(u)=\frac{N_{i,n}(u) w_i}{\sum_{j=1}^{k} N_{j,n}(u) w_j}$$

are known as the rational basis functions.



A NURBS surface is obtained as the tensor product of two NURBS curves, thus using two independent parameters *u* and *v* (with indices *i* and *j* respectively):[19]

$$S(u,v)=\sum_{i=1}^{k}\sum_{j=1}^{l} R_{i,j}(u,v) P_{i,j}$$

with

$$R_{i,j}(u,v)=\frac{N_{i,n}(u) N_{j,m}(v) w_{i,j}}{\sum_{p=1}^{k}\sum_{q=1}^{l} N_{p,n}(u) N_{q,m}(v) w_{p,q}}$$

as rational basis functions.

See also

- De Boor algorithm
- M-spline
- I-spline
- T-spline
- Bézier curve

- Box spline
- Spline wavelet

Notes

1. Strictly speaking B-splines are usually defined as being left-continuous
2. de Boor gives FORTRAN routines for least-squares fitting of experimental data.

References

1. de Boor, p 114
2. Gary D. Knott (2000), *Interpolating cubic splines (http://books.google.com/books?id=qkGlfJRuRs8C&dq=The+global+cubic+spline+basis+based+on+the+particular+join+order+2+piece-+wise+polynomials+known+as+B-splines+(B+stands+for+%22basis%22)+is+developed+by&source=gbs_navlinks_s)*. Springer. p. 151
3. de Boor, p113.
4. de Boor, p 131.
5. de Boor, p. 131
6. de Boor, p. 134.
7. Lee, E. T. Y. (December 1982). "A Simplified B-Spline Computation Routine". *Computing* (Springer-Verlag) **29** (4): 365–371. doi:10.1007/BF02246763.
8. Lee, E. T. Y. (1986). "Comments on some B-spline algorithms". *Computing* (Springer-Verlag) **36** (3): 229–238. doi:10.1007/BF02240069.
9. de Boor, p 322.
10. Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties (with comments and rejoinder). Statistical Science 11(2): 89-121.
11. de Boor, p. 115
12. Eugene V. Shikin; Alexander I. Plis (14 July 1995). *Handbook on Splines for the User*. CRC Press. pp. 96–. ISBN 978-0-8493-9404-1.
13. Wernecke, Josie (1993). "8". *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN 0201624958.
14. Zorin, Denis (2002), *Bezier Curves and B-splines, Blossoming* (PDF), New York University, retrieved 4 January 2015
15. de Boor, Chapter XIV, p. 235
16. Gans, Peter; Gill, J. Bernard (1984). "Smoothing and Differentiation of Spectroscopic Curves Using Spline Functions". *Applied Spectroscopy* **38** (3): 370–376. doi:10.1366/0003702844555511.
17. Vicsek, Maria; Neal, Sharon L.; Warner, Isiah M. (1986). "Time-Domain Filtering of Two-Dimensional Fluorescence Data". *Applied Spectroscopy* **40** (4): 542–548. doi:10.1366/0003702864508773.
18. Piegl and Tiller, chapter 4, sec. 2
19. Piegl and Tiller, chapter 4, sec. 4

- Carl de Boor (1978). *A Practical Guide to Splines*. Springer-Verlag. ISBN 3-540-90356-9.
- Piegl, Les; Tiller, Wayne (1997). *The NURBS Book* (2nd. ed.). Springer. ISBN 3-540-61545-8.

Further reading

- Richard H. Bartels; John C. Beatty; Brian A. Barsky (1987). *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann. ISBN 978-1-55860-400-1.
- Jean Gallier (1999). *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan Kaufmann. Chapter 6. B-Spline Curves. This book is out of print and freely available from the author.
- Hartmut Prautzsch; Wolfgang Boehm; Marco Paluszny (2002). *Bézier and B-Spline Techniques*. Springer Science & Business Media. ISBN 978-3-540-43761-1.
- David Salomon (2006). *Curves and Surfaces for Computer Graphics*. Springer. Chapter 7. B-Spline Approximation. ISBN 978-0-387-28452-1.

External links

- Weisstein, Eric W., "B-Spline" (http://mathworld.wolfram.com/B-Spline.html), *MathWorld*.
- Mathews, John H. "Module B-Splines".
- Ruf, Johannes. "B-splines of third order on a non-uniform grid" (PDF).
- bivariate B-spline from numpy (http://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.bisplrep.html)
- Interactive B-splines with JSXGraph (http://jsxgraph.uni-bayreuth.de/wiki/index.php/B-splines)
- TinySpline: Opensource C-library with bindings for various languages (https://github.com/retuxx/tinyspline)

Retrieved from "https://en.wikipedia.org/w/index.php?title=B-spline&oldid=727566177"

Categories: Splines (mathematics) | Interpolation