De Boor's algorithm

From Wikipedia, the free encyclopedia
(Redirected from De Boor algorithm)

In the mathematical subfield of numerical analysis the **de Boor's algorithm** is a fast and numerically stable algorithm for evaluating spline curves in B-spline form. It is a generalization of the de Casteljau's algorithm for Bézier curves. The algorithm was devised by Carl R. de Boor. Simplified, potentially faster variants of the de Boor algorithm have been created but they suffer from comparatively lower stability.[1][2]

Contents [hide]

Introduction [edit]

The general setting is as follows. We would like to construct a curve whose shape is described by a sequence of *p* points $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{p-1}$, which plays the role of a *control polygon*. The curve can be described as a function $\mathbf{s}(x)$ of one parameter *x*. To pass through the sequence of points, the curve must satisfy $\mathbf{s}(u_0) = \mathbf{d}_0, \ldots, \mathbf{s}(u_{p-1}) = \mathbf{d}_{p-1}$. But this is not quite the case: in general we are satisfied that the curve "approximates" the control polygon. We assume that $u_0, \ldots, u_{p-1}$ are given to us along with $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{p-1}$.

One approach to solve this problem is by splines. A spline is a curve that is a piecewise n^{th} degree polynomial. This means that, on any interval $[u_i, u_{i+1})$, the curve must be equal to a polynomial of degree at most *n*. It may be equal to different polynomials on different intervals. The polynomials must be *synchronized*: when the polynomials from intervals $[u_{i-1}, u_i)$ and $[u_i, u_{i+1})$ meet at the point u_i, they must have the same value at this point and their derivatives must be equal (to ensure that the curve is smooth).

De Boor's algorithm is an algorithm which, given $u_0, \ldots, u_{p-1}$ and $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{p-1}$, finds the value of spline curve $\mathbf{s}(x)$ at a point *x*. It uses O(n^2) + O(n + p) operations where *n* is the degree and *p* the number of control points of *s*.

Outline of the algorithm [edit]

Suppose we want to evaluate the spline curve for a parameter value $x \in [u_\ell, u_{\ell+1}]$. We can express the curve as

$$\mathbf{s}(x) = \sum_{i=0}^{p-1} \mathbf{d}_i N_i^n(x),$$

where[3]

$$N_i^n(x) = \frac{x - u_i}{u_{i+n} - u_i} N_i^{n-1}(x) + \frac{u_{i+n+1} - x}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(x),$$

and

$$N_i^0(x) = \begin{cases} 1, & \text{if } x \in [u_i, u_{i+1}) \\ 0, & \text{otherwise} \end{cases}$$

Due to the spline locality property,

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Português
中文
Edit links

$$\mathbf{s}(x) = \sum_{i=\ell-n}^{\ell} \mathbf{d}_i N_i^n(x)$$

So the value $\mathbf{s}(x)$ is determined by the control points $\mathbf{d}_{\ell-n}, \mathbf{d}_{\ell-n+1}, \dots, \mathbf{d}_\ell$; the other control points $\mathbf{d}_i$ have no influence. De Boor's algorithm, described in the next section, is a procedure which efficiently calculates the expression for $\mathbf{s}(x)$.

The algorithm [edit]

Suppose $x \in [u_\ell, u_{\ell+1})$ and $\mathbf{d}_i^{[0]} = \mathbf{d}_i$ for $i = \ell - n, \dots, \ell$. Now calculate

$$\mathbf{d}_i^{[k]} = (1 - \alpha_{k,i})\mathbf{d}_{i-1}^{[k-1]} + \alpha_{k,i}\mathbf{d}_i^{[k-1]}; \qquad k = 1, \dots, n; \quad i = \ell - n + k, \dots, \ell$$

with

$$\alpha_{k,i} = \frac{x - u_i}{u_{i+n+1-k} - u_i}.$$

Then $\mathbf{s}(x) = \mathbf{d}_\ell^{[n]}$.

See also [edit]

- De Casteljau's algorithm
- Bézier curve
- NURBS

External links [edit]

- De Boor's Algorithm
- The DeBoor-Cox Calculation

Computer code [edit]

TinySpline: Open source C-library for splines which implements De Boor's algorithm

References [edit]

1. ^ Lee, E. T. Y. (December 1982). "A Simplified B-Spline Computation Routine". *Computing* (Springer-Verlag) **29** (4): 365–371. doi:10.1007/BF02246763.
2. ^ Lee, E. T. Y. (1986). "Comments on some B-spline algorithms". *Computing* (Springer-Verlag) **36** (3): 229–238. doi:10.1007/BF02240069.
3. ^ http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-basis.html

Categories: Numerical analysis | Splines | Interpolation