Advanced Numerical Algroirthms with Python FMNN25 2015

Claus Führer

Lund University
claus@maths.lth.se

Lund, Automn 2017

Unit 1: The De Boor Algorithm

The DeBoor Algorithm is the central algorithm for computing splines.

In this course we take this algorithm and splines in general as a first programming task.

1.1: Cubic Spline

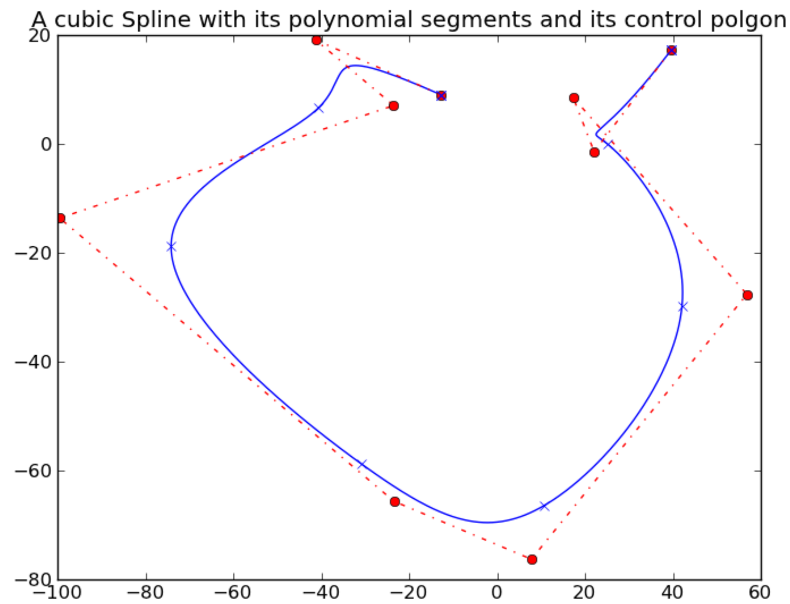
Definition 1. *A function* $s : [u_2, u_{K-2}] \subset \mathbb{R} \to \mathbb{R}^2$ *is called a* cubic spline *if for given node points* $u_0 \leq u_1 \leq \ldots \leq u_K$

- $s \in \mathcal{C}^2([u_2, u_{K-2}])$ *(twice continuously differentiable)*
- $s|_{[u_i,u_{i+1}]} \in \mathcal{P}^3([u_i, u_{i+1}])$ *(cubic polynomial) with* $u_i, u_{i+1} \in [u_2, u_{K-2}]$.

1.2: Basis Representation

A cubic spline has a basis representation

$$s(u) = \sum_{i=0}^{L} d_i N_i^3(u)$$



with $L = K - 2$ being the number of degrees of freedom of the spline (dimension) and $d_i \in \mathbb{R}^2$ its *control points* or *de Boor points*. The control points form the control polygon.

1.3: Basis functions

The basis functions $N_i^3 : [u_0, u_K] \to \mathbb{R}$ are defined recursively:

Definition 2.

$$N_i^0(u) := \begin{cases} 0 & \textit{if } u_{i-1} = u_i \\ 1 & \textit{if } u \in [u_{i-1}, u_i) \\ 0 & \textit{else} \end{cases}$$

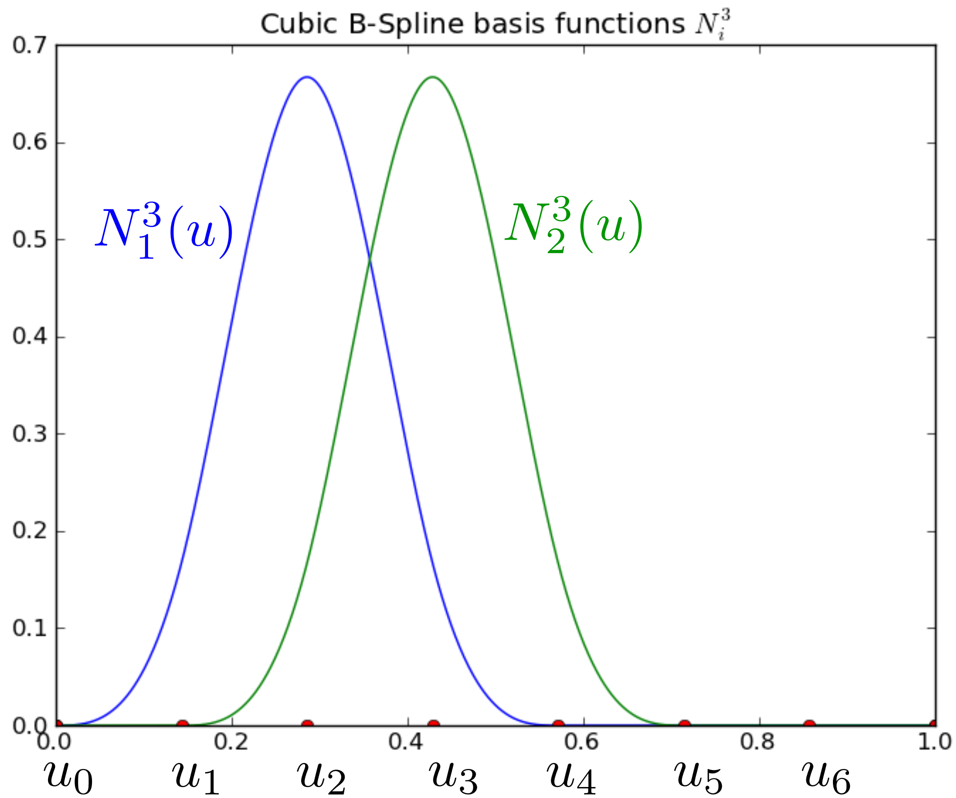
and

$$N_i^k(u) := \frac{u - u_{i-1}}{u_{i+k-1} - u_{i-1}} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_i} N_{i+1}^{k-1}(u)$$

where we use the convention $0/0 = 0$ *if nodes coincide.*

Note: Comparing this definition to other definitions shows an index shift. Here we have the property that $N_i^k(u)$ is nonzero at $u_i \leq \ldots \leq u_{k+i-1}$. The recursion seems to require grid points u_{-1} and u_{K+1} but the location of these points does not affect the final result for $u \in [u_2, u_{K-2}]$ in case of cubic splines as related terms will be multiplied by zero.

1.4 Basis Functions



1.5: Evaluating Splines

Inserting this recursion into $s(u) = \sum_{i=0}^{L} d_i N_i^3(u)$ leads to a recursive evaluation of the spline:

Let

$$\ldots u_{I-2} \le u_{I-1} \le u_I \le u < u_{I+1} \le u_{I+2} \le u_{I+3} \ldots$$

be a subset of the knot sequence. Then we note, that by construction of N_i^3:

$$s(u) = \sum_{i=0}^{L} d_i N_i^3(u) = \sum_{i=I-2}^{I+1} d_i N_i^3(u)$$

Note: N_{I-2}^3 is nonzero at the grid points u_{I-2}, u_{I-1}, u_I and so on.

1.6: Evaluating Splines - Blossoms

Every basis function N_i^3 is nonzero in exactly four intervals and in particular at three grid points. (see picture on Slide 1.4).

We denote the coefficient multiplying a basis function which is not zero at the grid points u_{I-2}, u_{I-1}, u_I by $d[u_{I-2}, u_{I-1}, u_I]$ and define

$$d[u, u_{I-1}, u_I] = \alpha(u) d[u_{I-2}, u_{I-1}, u_I] + (1 - \alpha(u)) d[u_{I-1}, u_I, u_{I+1}]$$

with $\alpha(u)$ being a scalar factor, which we will precise in one of the following slides.

1.7: Evaluating Splines - Blossoms (Cont)

Situation: $u \in [u_I, u_{I+1}]$:

$$
\begin{array}{lclllll}
d_{I-2} & =: & d[u_{I-2}, u_{I-1}, u_I] & & & \\
d_{I-1} & =: & d[u_{I-1}, u_I, u_{I+1}] & d[u, u_{I-1}, u_I] & & \\
d_I & =: & d[u_I, u_{I+1}, u_{I+2}] & d[u, u_I, u_{I+1}] & d[u, u, u_I] & \\
d_{I+1} & =: & d[u_{I+1}, u_{I+2}, u_{I+3}] & d[u, u_{I+1}, u_{I+2}] & d[u, u, u_{I+1}] & \underbrace{d[u, u, u]}_{=s(u)}
\end{array}
$$

The transition from one column to the next is done by linear interpolation:

$$d[u, u_{I-1}, u_I] = \alpha(u) d[u_{I-2}, u_{I-1}, u_I] + (1 - \alpha(u)) d[u_{I-1}, u_I, u_{I+1}]$$

with $\alpha(u)$ ....(see next slide). Note, each blossom contains at least one of the two grid points u_I and u_{I+1} or not a grid point at all.

1.8: Evaluating Splines - Blossoms (Cont)

.... with $\alpha(u)$, which depends on the span of the knot values of the corresponding blossom pair in the following way:

$$\alpha(u) := \frac{u_{\mathrm{rightmostknot}} - u}{u_{\mathrm{rightmostknot}} - u_{\mathrm{leftmostknot}}}$$

where "rightmost" and "leftmost" refers to the knots in the corresponding blossom pair, e.g.

$$d[u, u_{I-1}, u_I] = \alpha(u) d[u_{I-2}, u_{I-1}, u_I] + (1 - \alpha(u)) d[u_{I-1}, u_I, u_{I+1}]$$

with $\alpha(u) = \frac{u_{I+1} - u}{u_{I+1} - u_{I-2}}$.

1.9: De Boor algorithm - summary

The computation of $s(u)$ requires the following steps:

1. Find the "hot" interval $u \in [u_I, u_{I+1}]$
2. Select the corresponding control points $d_{I-3}, \ldots, d_I$
3. Run the blossom recursion to obtain $s(u)$.

1.10: Interpolation

Interpolation task:

Find a cubic spline which passes through given data points $(x_i, y_i), i = 0, \ldots, L$.

Here:

For a grid $u_i, i = 0, \ldots, L+2 = K$ and given data points $(x_i, y_i), i = 0, \ldots, L$ find the control points $d_i, i = 0, \ldots, L$ of the spline s, such that (x_i, y_i) are points of the graph of s.

1.11: Greville abscissae

We consider $s(u) = \begin{pmatrix} s_y(u) \\ s_x(u) \end{pmatrix}$.

A point on the (non parametric) graph of s_x has the form

$$\begin{pmatrix} s_x(u) \\ u \end{pmatrix}$$

The function in the second componend $f(u) = u$ is a special cubic spline with the De Boor points $\xi_i := (u_i + u_{i+1} + u_{i+2})/3$.

The ξ_i are called Greville abscissae.

The interpolation task is: find $d_i = \begin{pmatrix} d_{y_i} \\ d_{x_i} \end{pmatrix}$ such that $s_x(\xi_i) = x_i$ and $s_y(\xi_i) = y_i$.

1.12: Vandermonde like systems

This leads to two linear systems to be solved

$$\begin{pmatrix} N_0^3(\xi_0) & \cdots & N_L^3(\xi_0) \\ \vdots & \cdots & \vdots \\ N_0^3(\xi_L) & \cdots & N_L^3(\xi_L) \end{pmatrix} \begin{pmatrix} d_{x_0} \\ \cdots \\ d_{x_L} \end{pmatrix} = \begin{pmatrix} x_0 \\ \cdots \\ x_L \end{pmatrix}$$

$$\begin{pmatrix} N_0^3(\xi_0) & \cdots & N_L^3(\xi_0) \\ \vdots & \cdots & \vdots \\ N_0^3(\xi_L) & \cdots & N_L^3(\xi_L) \end{pmatrix} \begin{pmatrix} d_{y_0} \\ \cdots \\ d_{y_L} \end{pmatrix} = \begin{pmatrix} y_0 \\ \cdots \\ y_L \end{pmatrix}$$

Note: To be able to evaluate this system, the first and last three grid points need to have multiplicity three: $u_0 = u_1 = u_2$ *and* $u_{K-2} = u_{K-1} = u_K$

1.13: Banded Matrices

As the N_i^3 have compact support which spans at most 4 parameter intervals, the matrices are banded with bandwidth ≤ 4.

To solve the systems in Python use:
`scipy.linalg.solve_banded`.