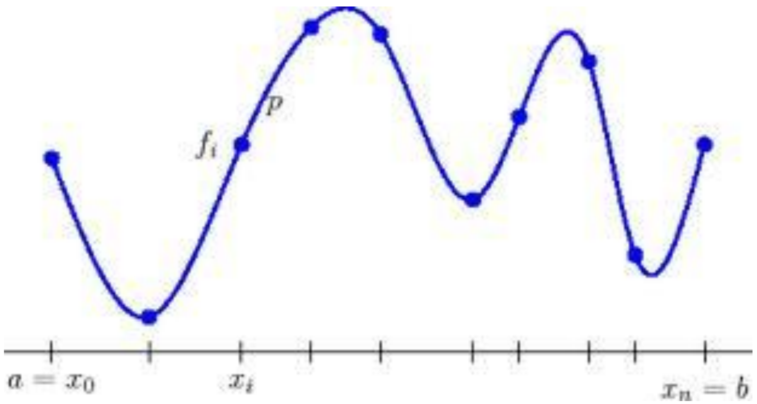
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Programming Task 2: Spline Movement



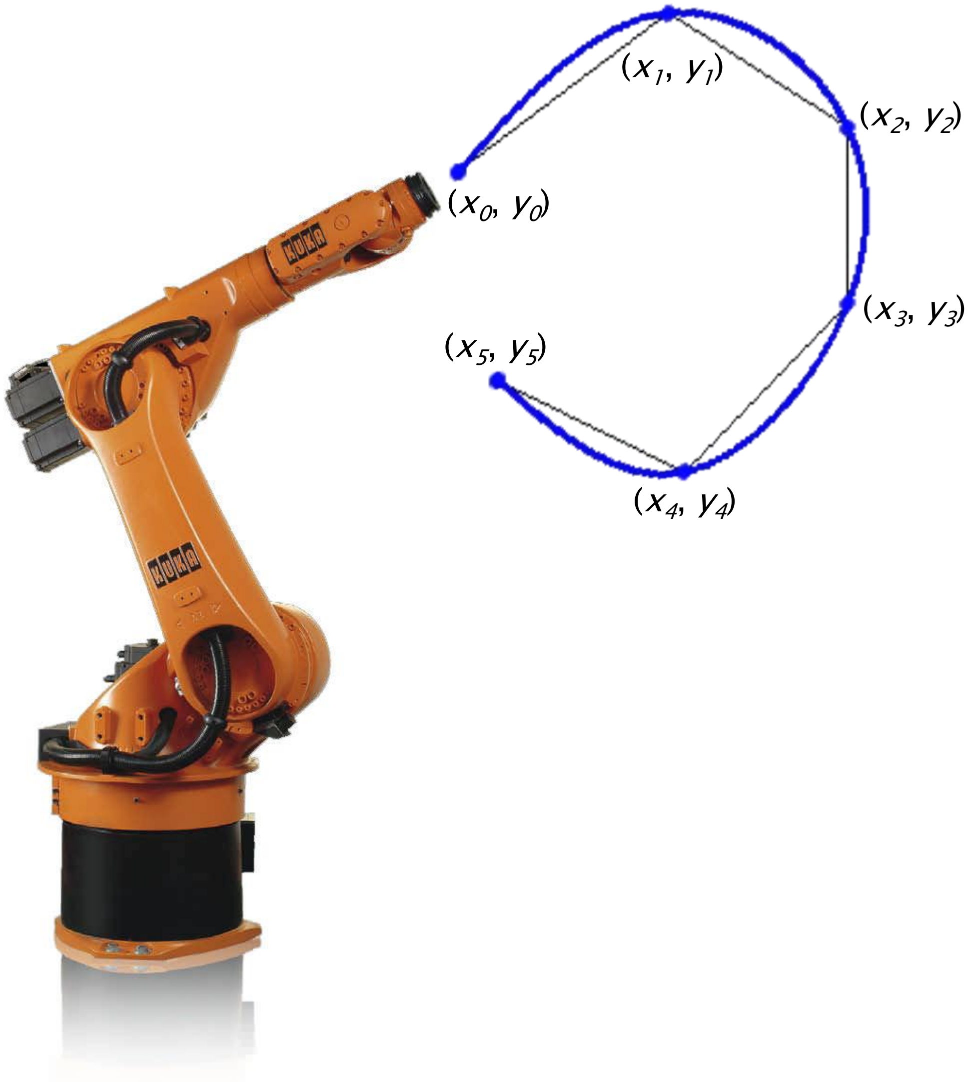
Softwaredevelopment for Industrial Robotics

Marcel Grotzke, Gabriel Kögler, Henri Hamann

30.01.2014

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

Table of content



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

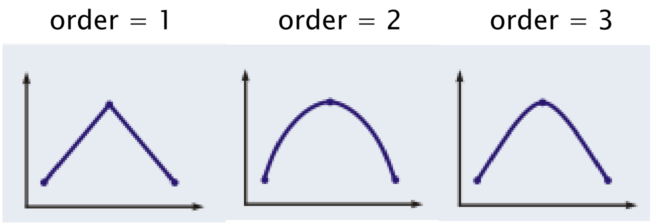
1. Fundamentals – What are splines?

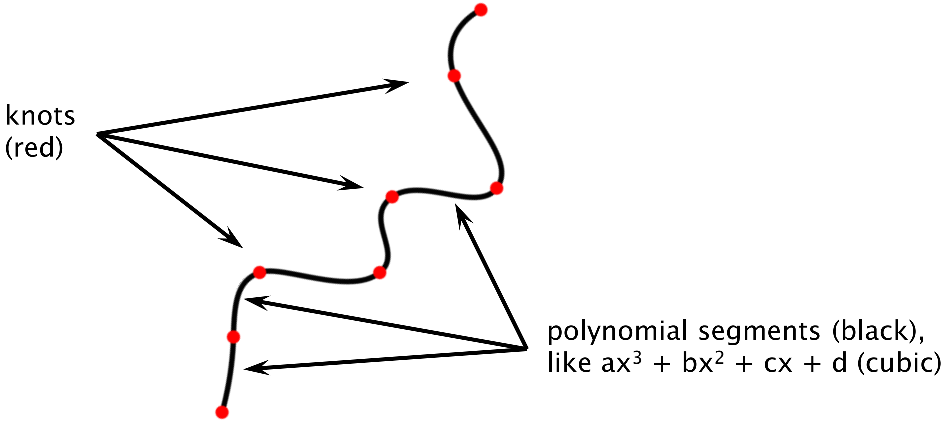
Origin:

- First reference in 1946 [1]
- Suitable to describe curved lines → CAD (curves & freeform surfaces)
- i.e. rollercoaster, automobile and shipbuilding, high-speed railways

Definition:

- *k* knots[1] as base
- *k-1* continuous, smooth[2], polynomial segments of degree[3] *n*
- commonly used splines with degree=3 (cubic) → ax^3+bx^2+cx+d

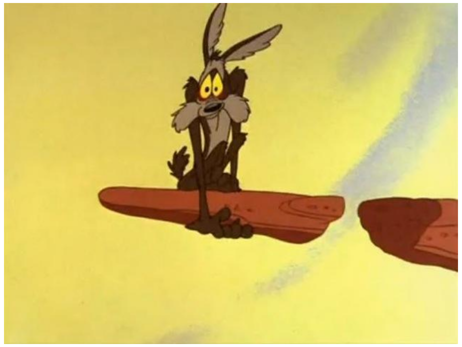




[1] knot = Knotenpunkt, o. Stützpunkt
[2] order = Grad
[3] smooth = stetig

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

1. Fundamentals – Splines have to be…



…continuous:

- Function plot does not jump

(to get an ongoing spline)

…smooth:

- every argument is calculable

(no infinity[1] /overshooting)

…differentiable:

- need for calculating the spline conditions

[1] Unendlichkeit

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

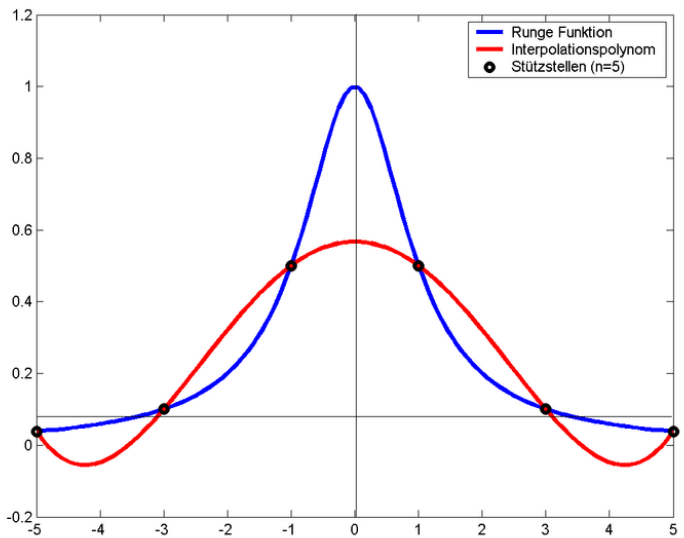
1. Fundamentals – Why Splines?

Typical Problem: Set of data points

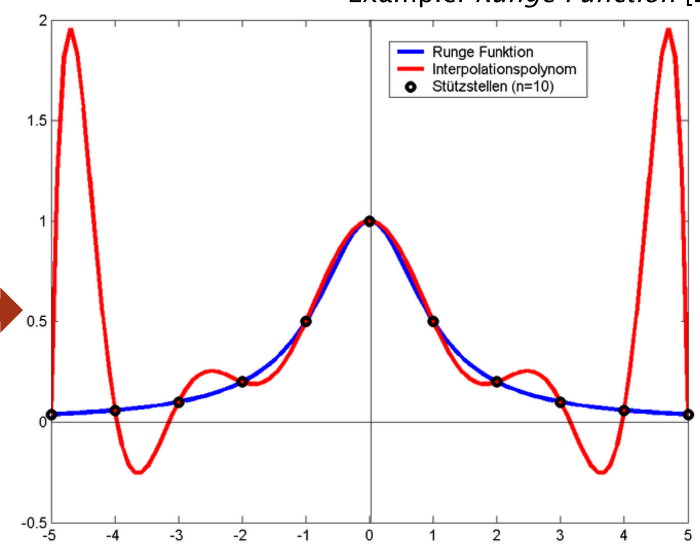
→ ad hoc approach: one (global) polynomial passing every point

But: Increasing number of points means increasing error at interval boundary, too.

Example: *Runge Function* [2]



Increased number of data points

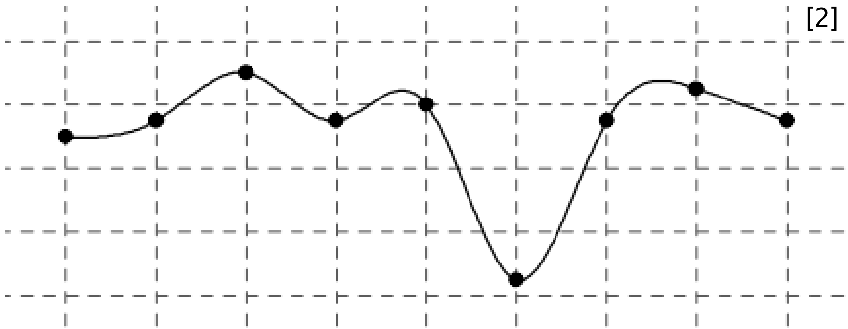


→ better use another interpolation technique, like i.e. spline interpolation

1. Fundamentals – Splines can…

<u>…interpolate the given input points:</u>

- The given input points are part of the spline.

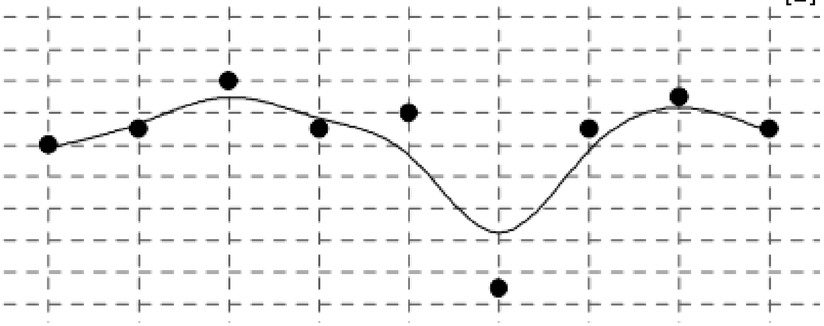


[2]

Inpolation with a cubic spline.

<u>…approximate the given input points:</u>

- The given input points are **not** part of the spline (i.e. *b-spline*).



[2]

Approximation with a cubic b-spline.

[1] unendlich

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

2. Spline Approximation – B-Spline

Approach:

Linear combination of all segments:

$$\boxed{\mathbf{x}(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot \mathbf{d}_i} \tag{1}$$

With:

- Given: n+1 control points $\mathbf{D} = (d_0 \ldots d_n)$ (so called *de Boor points*)
- Basis function $\mathbf{N}_{i,k}(t)$
- Order **k** (degree = k-1)
- Time variable **t** represents the position on the spline
- Knot vector $\mathbf{t}_i = (t_0, \ldots, t_n, \ldots t_{n+k})$ (mathematical construct)

Feature:

- De Boor points influence spline´s shape

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

2. Spline Approximation – B-Spline

Single polynomial:

- Calculated with recursive equation
- Order **k** is a sum from 2 splines with Order **k-1**
- recursive definition:

(2)

$$N_{i,1}(t) = \begin{cases} 1 & \text{für} \quad t_i \le t < t_{i+1} \\ 0 & \text{sonst} \end{cases}$$

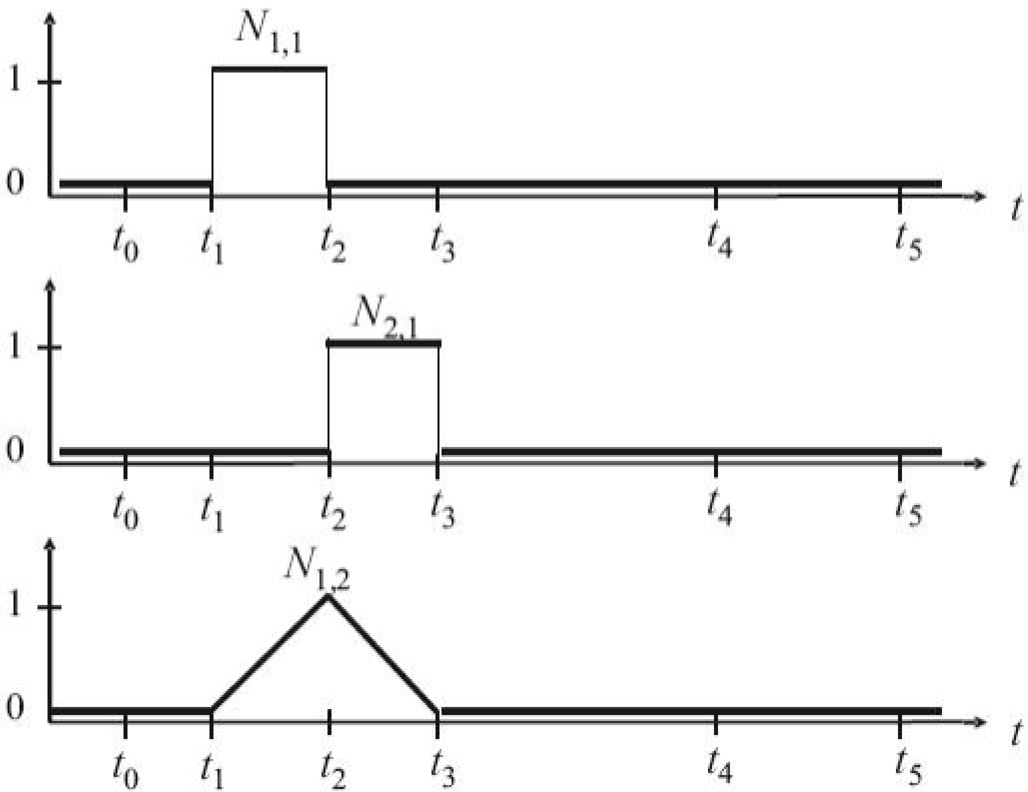
$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

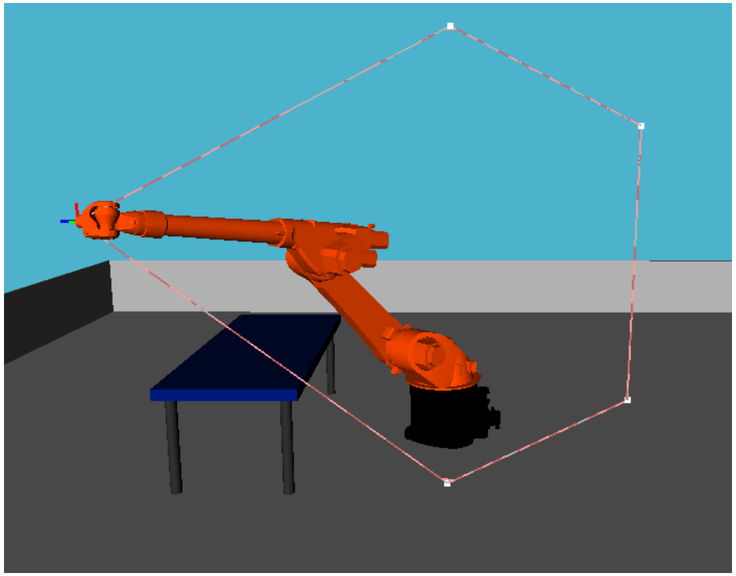
for $k>1$ and $i=0,\ldots,n$

Workaround:

- Calculate the segments considering the knot vector $t_i = (t_0,\ldots,t_n,\ldots t_{n+k})$

2. Spline Approximation – B-Spline



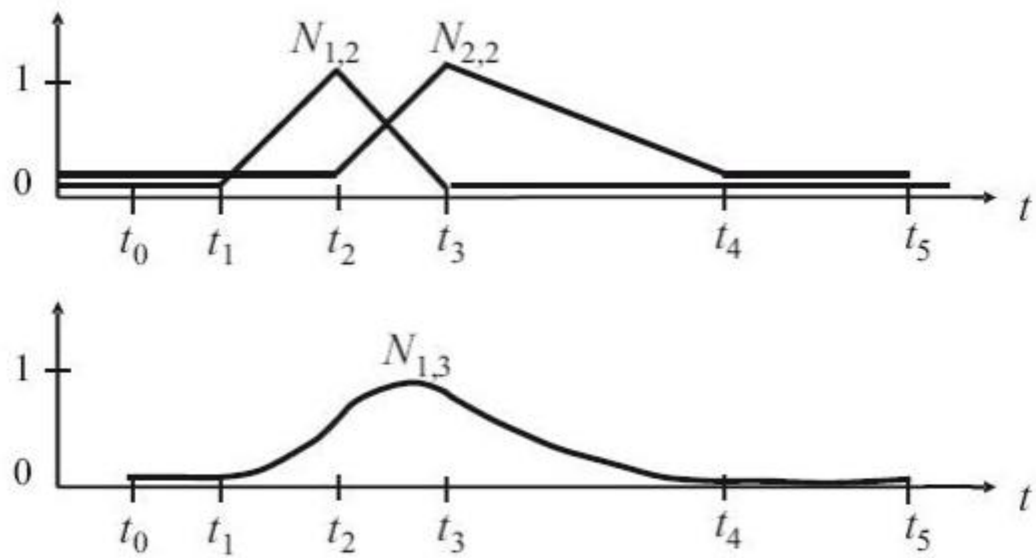


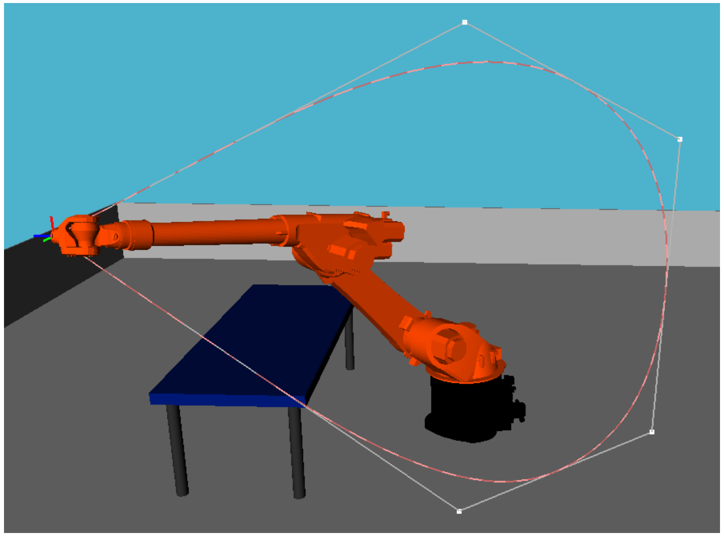
order 1 order 1

order 2

- b-spline of order k = 2
- polynom of degree 1

2. Spline Approximation – B-Spline



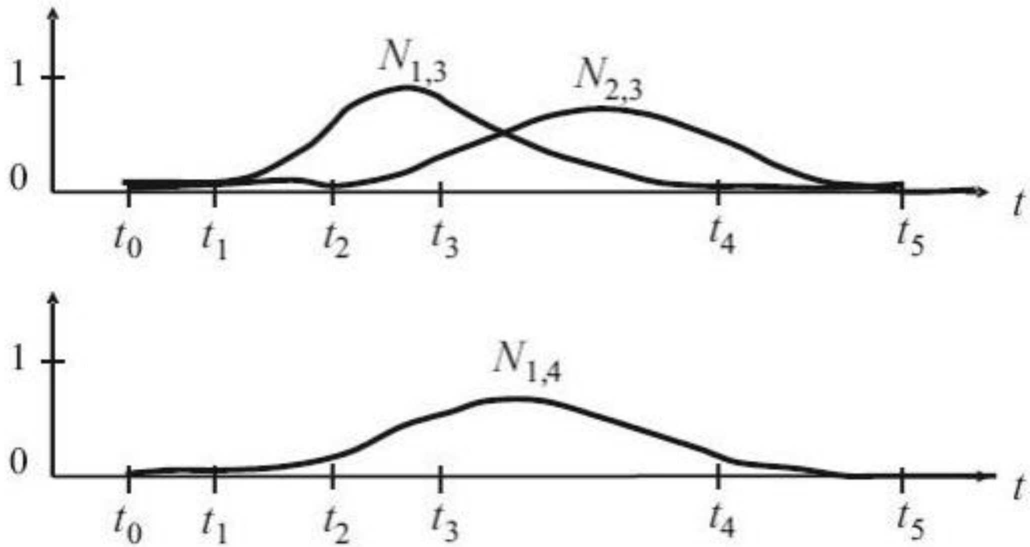


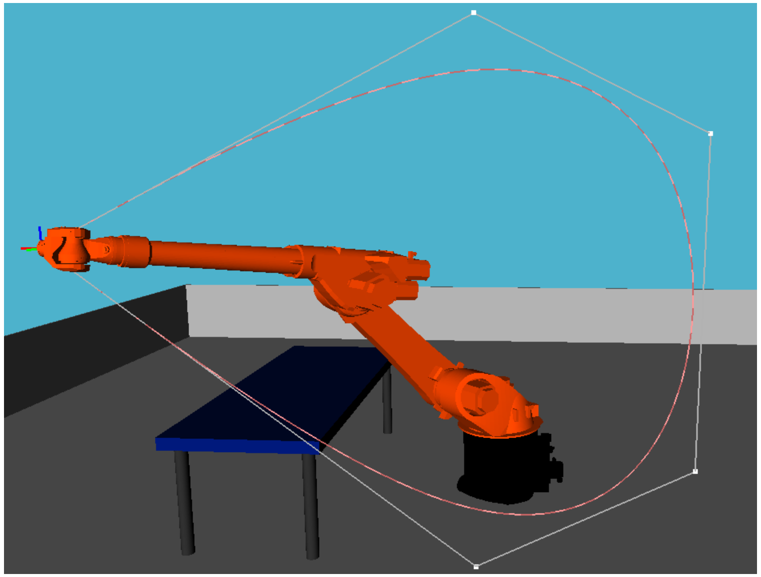
- b-spline of order k = 3
- polynom of degree 2

order 2 order 2

order 3

2. Spline Approximation – B-Spline





- b-spline of order k = 4
- polynom of degree 3

order 3 order 3

order 4

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

2. Spline Approximation – B-Spline

Knot vector $\mathbf{t_i}$:

- $\mathbf{t_i} = (t_0,\ldots,t_n,\ldots t_{n+k})$ condition: $t_i <= t_{i+1}$

- Uniform knot vector: $t_i = t_{i-1} + \text{delta\_t}$ (delta_t = constant)

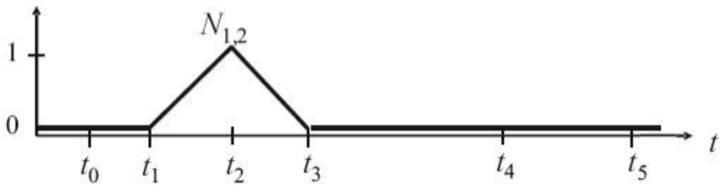
  example: delta_t = 1; $t_i = (t_0, t_1, t_2, t_3) = (0, 1, 2, 3)$

- Non-uniform: delta_t != constant

- knot vector represents a wheigt for the basis functions

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

- Factorises the influence of the *de Boor points*



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

2. Spline Approximation – B-Spline

Demonstration

of B-spline movement

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

3. Spline Interpolation

System of equations: every segment has one equation!

k knots & order=n → ($k-1$) polynomial equations with ($n+1$) variables.

→ total amount of variables = $(n+1) * (k-1)$.

→ with order = 3 → ax^3+bx^2+cx+d → n=4. (our case)

→ amount of variables = 4(k−1) = 4k−4

→ need for conditions to solve system of equations.

Conditions: Are given by spline´s behavior!

- Every knot is part of one segment → $p_i(x_i)=y_i$ → k conditions
- Segment´s contact points → $p_i(x_{i+1})=p_{i+1}(x_{i+1})$ → k−2 conditions
- Same in first derivation → $p_i'(x_{i+1})=p_{i+1}'(x_{i+1})$ → k−2 conditions
- Same in second derivation → $p_i''(x_{i+1})=p_{i+1}''(x_{i+1})$ → k−2 conditions

(4k−6) < (4k−4) → we still need two conditions!

Last two conditions:

- Define the behavior at the spline´s start and end point (i.e. slope)
- Arbitrary decision → $p_0'(x_0)=0$ & $p_k'(x_k)=0$

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

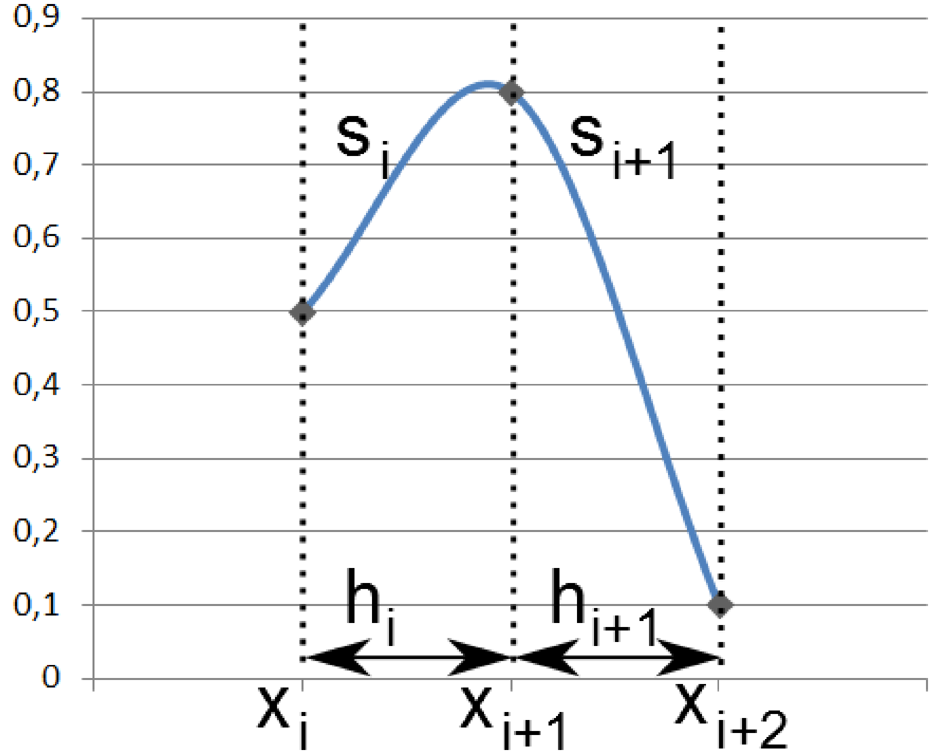
3. Spline Interpolation

- With polynomial s_i between point (x_i, y_i) and point (x_{i+1}, y_{i+1})

$$s_i(x) = a_i\left(x - x_i\right)^3 + b_i\left(x - x_i\right)^2 + c_i\left(x - x_i\right) + d_i$$

→ only one polynomial is active for one input

- Length of the interval $h_i = x_{i+1} - x_i$, where polynomial s_i is "active"



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

3. Spline Interpolation

- Calculation of parameter a_i, b_i, c_i and d_i through curvature k_i
  → curvature is second derivative of a function

$$a_i = \frac{\kappa_{i+1} - \kappa_i}{6h_i} \qquad b_i = \frac{\kappa_i}{2} \qquad c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}\left(2\kappa_i + \kappa_{i+1}\right) \qquad d_i = y_i$$

- Through conditions for spline we can derive this function

$$\frac{2\kappa_i + \kappa_{i-1}}{6} h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} \stackrel{!}{=} -\frac{2\kappa_i + \kappa_{i+1}}{6} h_i + \frac{y_{i+1} - y_i}{h_i}$$

→ this leads to a tridiagonal linear equation system

$$\begin{bmatrix} 2(h_1+h_0) & h_1 & & 0 \\ & \ddots & & \\ h_{i-1} & 2(h_i+h_{i-1}) & h_i & \\ & & \ddots & \\ 0 & & h_{n-1} & 2(h_n+h_{n-1}) \end{bmatrix} \begin{bmatrix} k_0 \\ \vdots \\ k_n \end{bmatrix} = \begin{bmatrix} \frac{6}{h_1}(y_2 - y_1) - \frac{6}{h_0}(y_1 - y_0) \\ \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \\ \frac{6}{h_{n-1}}(y_n - y_{n-1}) - \frac{6}{h_{n-2}}(y_{n-1} - y_{n-2}) \end{bmatrix}$$

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Many thanks for your attention!

www.ovgu.de

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

bibliography

1) Schoenberg: *Contributions to the problem of approximation of equidistant data by analytic functions*, Quart. Appl. Math., vol. 4, pp. 45-99 and 112-141, 1946.
2) Wikipedia: *Polynominterpolation.* Abgerufen am 25.01.14.
3) Wikipedia: *Spline*. Abgerufen am 18.01.14.
4) Moritz Lenz: *Splinefunktionen und ihre Anwendung.* 2003.
5) Schneider: *Splines.* TU Chemniz. Gefunden auf http://www-user.tu-chemnitz.de/~uro/teaching/SS2002-numerik/misc/Splines.pdf. Abgerufen am 15.01.2014.
6) Theisel: *Computer Aided Geometric Design: B-Spline.* Visual Computing, University of Magdeburg. 2013.

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

A. Spline Approximation – B-Spline

Another approach:

- Spline with degree=n is a linear combination from 2 splines with degree=n-1 → recursive defintion.

$$\mathbf{x}(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot \mathbf{d}_i$$

→ input: {Set of points | (x_i,y_i,z_i)}

$$C(u) = \sum_{i=1}^{n-p} P_i \, N_{i,p,\tau}(u) \qquad N_{i,0,\tau}(u) = \begin{cases} 1, & u \in [\tau_i\,, \tau_{i+1}[\\ 0, & \text{sonst} \end{cases}$$

$$N_{i,p,\tau}(u) = \frac{u - \tau_i}{\tau_{i+p} - \tau_i} N_{i,p-1,\tau}(u) \; + \; \frac{\tau_{i+p+1} - u}{\tau_{i+p+1} - \tau_{i+1}} N_{i+1,p-1,\tau}(u)$$

A. Fundamentals – Splines definition

Piecewise polynomial calculation of data set $\{(x_i, y_i), i = 0, 1, \ldots, n\}$

$$S(x) = \begin{cases} S_0(x), & \text{if } x_0 \le x < x_1 \\ S_1(x), & \text{if } x_1 \le x < x_2 \\ \vdots & \\ S_{n-1}(& \end{cases}$$