Interpolation von Punkten durch eine Spline-Funktion

Elkaam Hiba

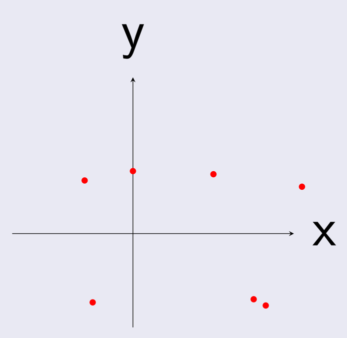
Zusammenfassung

Hier wird eine Familie von Interpolationssplines vorgestellt, die die meisten Spline-Typen aufweisen und daher auch in Anwendungen, die die zukünftigen Punkte nicht kennen, glatte Interpolationskurven erzeugen können, ohne die Notwendigkeit komplexerer Berechnungsmethoden. So ermöglichen die Drei-Punkt-Splines eine größere Einstellfreiheit und können so an die Anwendung in Reichweite angepasst werden.

Motivation

Zweck der Interpolation ist,einen Satz von Werten an vorgegebenen Positionen mit neuen plausiblen Werten im Einklang mit den bereits vorhandenen Werten zu ergänzen.Genauer gesagt besteht das Problem darin, unter Berücksichtigung von Paaren(x_i, y_i) eine Funktion $\Phi = \Phi(x)$ wie $\Phi(x_i) = y_i$ für $i = 0, \dots, n$ zu finden.Es wird dann gesagt , dass y_i interpoliert x_i.

Interpolierenden Anfangspunkten



x	y
0	2
-1	-2.2
2	1.9
3	-2.1
3.3	-2.3
-1.2	1.7
4.2	1.5

Der allgemeine Ausdruck eines Spline-Stücks und seiner ersten Ableitung, die hier ein Polynom des Grades 3 ist:

$$S_i(t) = h_0(t)y_i + h_1(t)y_{i+1} + h_2(t)m_i + h_3(t)m_{i+1}$$

Lineare interpolierende Splines [Nau05]

Lineare Interpolation ist die einfachste Methode, da Linien zwischen zwei benachbarten Punkten verwendet werden.Die Aufgabe besteht darin, gegeben x_i wo $i = 1, \dots, n$,die Aufgabe ist es f(x) zu schätzen. Die lineare Spline $s_L(x)$, die f an diesen Punkten interpoliert, wird definiert durch:[Spä95]

$$S_L(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

wobei $x \in [x_{i-1}, x_i]$, $i = 1, 2, \dots, n$

Sobolev-Räume [Nau05]

Lass p(x) das lineare Polynom sein, das f(x) bei x1 und x2 interpoliert. Dann:

$$E_p = f(x) - p(x) = \frac{f''(\varepsilon)}{2}(x - x1)(x - x2)$$

wobei $x1 \leq \varepsilon \leq x2$ und,

$$|E_p| \leq Ch^2 \qquad , h = x2 - x1$$

Kontinuität

Eine Funktion ist f auf $[a, b]$ **absolut kontinuierlich** ist, wenn ihre Ableitung fast unbegrenzt ist überall in $[a, b]$,ist integrierbar auf $[a, b]$ und erfüllt:

$$\int_x^a v'(s)dx = v(x) - v(a), a \leq x \leq b$$

Interpolationsfehler bei Lineare interpolierende Splines

Wenn $h_i = x_i - x_i - 1$, dann erreicht die Funktion $(x - x_i)(x - x_i - 1)$ ihren maximalen absoluten Wert bei $\frac{x_i + x_i - 1}{2}$, mit einem Maximalwert von $\frac{h_i^2}{4}$. Sei $h = max_{1 \leq i \leq n} h_i$ definieren, dann:

$$\|\mathbf{f} - \mathbf{S_L}\| \leq \frac{1}{8}h^2\|\mathbf{f}''\| \implies \|\mathbf{E_p}\| \leq \frac{1}{8}h^2\|\mathbf{f}''\|$$

Herleitung der Formeln verschiedene kubische Splines

Hermite-Splines: Der allgemeine Hermite-Spline-Stücks , die hier ein Polynom des Grades 3 ist:$hm_{i-1} + 4hm_i + hm_{i+1} = u_i$ wobei $u_i = \frac{6}{x_{i+1} - x_i}(y_{i-1} - 2y_i + y_{i+1})$ für $i \in [0, n-2]$

Spline Kardinal:

$$\begin{aligned} S_i = y_i &+ (\frac{1}{2}y_{i-1} + \frac{1}{2}y_{i+1})t \\ &+ (y_{i-1} - \frac{5}{2} + 2y_{i+1} - \frac{1}{2}y_{i+2})t^2 \\ &+ (-\frac{1}{2}y_{i-1} + \frac{3}{2}y_i - \frac{3}{2}y_{i+1} + \frac{1}{2}y_{i+2})t^3 \end{aligned}$$

Spline Cubic: Sei $f(t)$ eine Funktion,die auf einem Intervall $[a, b]$ definiert wird, und seien $x_0, x, \dots, x_{n+1}$ eindeutige Punkte in $[a, b]$, wobei : $a = x_0 \leq x_1 \leq \dots \leq x_n = b$.

Bedingungen einer Kubische Spline

Eine der folgenden Randbedingungen gewählt werden kann: $s''(a) = s''(b) = 0$, was als natürliche Randbedingungen bezeichnet wird. $s'(a) = f'(a), s'(b) = f'(b)$, genannt die Bedingungen an den festgelegten Grenzen.

Genauigkeit einer Kubische Spline

Ein Spline ist eine flexible Kurvenziehhilfe, die entwickelt wurde, um eine Kurve zu erzeugen $y = v(x) \qquad , x \in [a, b]$ durch vorgeschriebene Punkte in der Weise, dass die Menge

$$\int_a^b \frac{|v''(x)^2|}{(1 + |v'(x)|^2)^3} dx$$

wird über alle Funktionen minimiert, die durch die gleichen Punkte gehen, was der Fall ist, wenn die Krümmung auf [a, b] klein ist.[RT12]

Implementierung

Hermite-Splines:

```
def hermiteSpline(y0, y1, m0, m1, t):
        b0 = 1 - 3 * t2 + 2 * t * t * t
        b1 = 3 * t2 - 2 * t3
        b2 = t - 2 * t2 + t3
        b3 = - t2+ t3
        return b0 * y0 + b1 * y1 + b2 * m0 + b3 * m1
```

Spline Kardinal: def catmullromSpline(y0, y1, y2, y3, t): b0 = -t + 2 * t2 - t2 * t * t b1 = 2 - 5 * t2 + 3 * t3 b2 = t + 4 * t2 - 3 * t3 b3 = -t2 + t3 return (b0 * y0) + (b1 * y1) + (b2 * y2) + (b3 * y3)

Literatur

[Nau05] Joachim Naumann. *Sobolev-Räume*. 2005.

[RT12] Daniel Ruijters und Philippe Thévenaz. „GPU prefilter for accurate cubic B-spline interpolation“. In: *The Computer Journal* 55.1 (2012), S. 15–20.

[Spä95] Helmuth Späth. *One dimensional spline interpolation algorithms*. AK Peters/CRC Press, 1995.

Kontakt: **Elkaam Hiba**, FH Aachen, Campus Jülich; email : elkaam.hiba@fh-aachen.de