Numerische Verfahren

Interpolation

Prof. Dr.-Ing. K. Warendorf, Prof. Dr.-Ing. P. Wolfsteiner

Hochschule für Angewandte Wissenschaften München
Fakultät 03

WS 13/14

Inhalt

Einleitung

Interpolationsaufgabe

Gegeben seien $n+1$ Stützpunkte (x_i, y_i). Ziel ist es eine Funktion p zu finden, für die gilt: $p(x_i) = y_i$ für alle $i = 1, \dots, n$.

Lösung

Im Normalfall ist die Funktion p ein Polynom.
Durch $n+1$ Stützpunkte (x_i, y_i) ist ein Polynom, das höchstens den Grad n hat, eindeutig bestimmt:

$$p_n(x) = \sum_{k=0}^{n} a_k \cdot x^k$$

Zur Lösung der Interpolationsaufgabe müssen die Koeffizienten a_k berechnet werden. Dafür gibt es verschiedene Ansätze.

Direkter Ansatz

Bestimmung der Koeffizienten durch lineares Gleichungssystem

$$
\begin{array}{ccccccccccc}
a_0 & + & a_1 x_0 & + & a_2 x_0^2 & + & \dots & + & a_n x_0^n & = & y_0 \\
a_0 & + & a_1 x_1 & + & a_2 x_1^2 & + & \dots & + & a_n x_1^n & = & y_1 \\
\vdots & & & & & & \vdots & = & \vdots & & \\
a_0 & + & a_1 x_n & + & a_2 x_n^2 & + & \dots & + & a_n x_n^n & = & y_n
\end{array}
$$

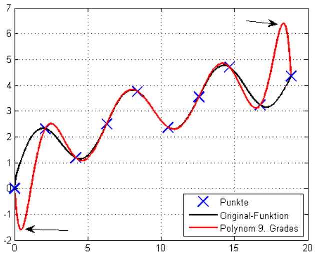
in Matrixschreibweise $X \cdot \vec{a} = \vec{y}$

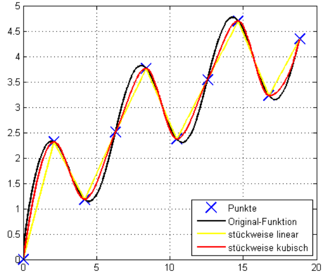
$$
X = \begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ \vdots & & \vdots & \\ 1 & x_n & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}
$$

Die Matrix X ist in der Regel schlecht konditioniert, deswegen sind andere Ansätze zielführender.

Stückweise Interpolation

Ist die Anzahl der Stützpunkte hoch, dann ist es nicht sinnvoll, **ein** Polynom durch alle Punkte zu legen; es entstehen Überschwinger am Intervallrand.





$\Longrightarrow$ Verwendung von Polynomen niedrigerem Grades für Teilintervalle

Definition der kubischen Spline-Interpolation I

Die stückweise Interpolation (zum Beispiel mit Geradenstücken) behebt zwar die Überschwinger, ergibt aber an den Stützpunkten keine **glatte** Funktion.
$\Longrightarrow$ Stückweise Interpolation mit der Forderung nach Differenzierbarkeit in den Stützpunkten.

Definition der kubischen Spline-Interpolation II

Kubische Spline-Interpolation

Gegeben seien $n+1$ Stützpunkte (x_i, y_i) mit $x_0 < x_1 < \cdots < x_n$. Die stückweise

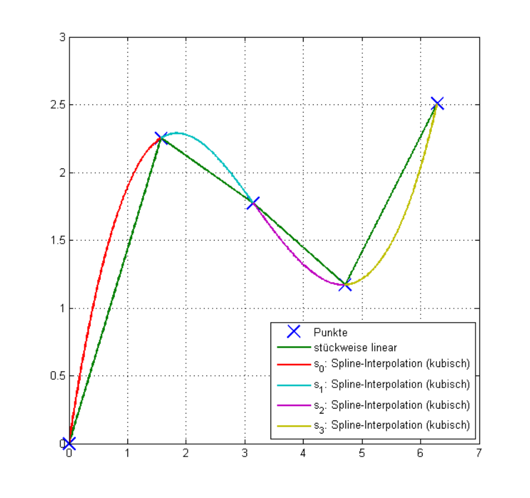
definierte Funktion $S(x) = \begin{cases} S_0(x) & \text{für} \quad x \in [x_0, x_1] \\ & \vdots \\ S_{n-1}(x) & \text{für} \quad x \in [x_{n-1}, x_n] \end{cases}$

heißt **kubische Spline-Funktion**, falls gilt:

1. $S_k = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3,\ k = 0, \ldots, n-1,\ x \in [x_k, x_{k+1}]$
2. Stützpunkte: $S(x_k) = y_k,\ k = 0, \ldots, n$
3. Stetigkeit: $S_k(x_{k+1}) = S_{k+1}(x_{k+1}) = y_{k+1},\ k = 0, \ldots, n-2$
4. Glattheit: $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1}),\ k = 0, \ldots, n-2$
5. Krümmung: $S_k''(x_{k+1}) = S_{k+1}''(x_{k+1}),\ k = 0, \ldots, n-2$

Definition der kubischen Spline-Interpolation III

Vergleich Spline-Interpolation mit stückweise linear



Berechnung der kubischen Spline-Polynome I

Mit $h_k = x_{k+1} - x_k$ ergibt sich durch Einsetzen und Umformen folgende Vorgehensweise[1]:

1. $a_k = y_k,\ k = 0, \dots, n-1$
2. Festlegung von c_0 und c_n durch Randbedingungen
   1. Natürliche kubische Splinefunktion (Krümmung $= 0$):
      $S''(x_0) = 0, S''(x_n) = 0 \Rightarrow c_0 = 0, c_n = 0$
   2. Kubische Splinefunktion mit gegebenen 2. Randableitungen:
      $S''(x_0) = y_0'', S''(x_n) = y_n'' \Rightarrow c_0 = \frac{y_0''}{2}, c_n = \frac{y_n''}{2}$
3. Lösen des Gleichungssystems für $c_k,\ k = 1, \dots, n-1$ (s. nächste Folie)
4. $b_k = \frac{y_{k+1} - y_k}{h_k} - \frac{h_k}{3}(c_{k+1} + 2c_k),\ k = 0, \dots, n-1$
5. $d_k = \frac{1}{3h_k}(c_{k+1} - c_k),\ k = 0, \dots, n-1$

Berechnung der kubischen Spline-Polynome II

Gleichungssystem: $A \cdot c = g$ mit

$$A = \begin{pmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix}$$

$$c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad g = \begin{pmatrix} 3\frac{y_2-y_1}{h_1} - 3\frac{y_1-y_0}{h_0} - h_0 c_0 \\ 3\frac{y_3-y_2}{h_2} - 3\frac{y_2-y_1}{h_1} \\ \vdots \\ 3\frac{y_n-y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1}-y_{n-2}}{h_{n-2}} - h_{n-1} c_n \end{pmatrix}$$

[1] vgl. Engeln-Müllges,Niederdrenk,Wodicka: Numerik Algorithmen, Springer, Berlin 2005