Kapitel 3

Interpolation

Die Interpolation von Funktionen oder Daten ist ein häufig auftretendes Problem sowohl in der Mathematik als auch in vielen Anwendungen.

Das allgemeine Problem, die sogenannte *Dateninterpolation*, entsteht, wenn wir eine Menge von Daten (x_i, f_i) für $i = 0, \ldots, n$ gegeben haben (z.B. Messwerte eines Experiments). Die Problemstellung ist nun wie folgt: Gesucht ist eine Funktion F, für die die Gleichung

$$F(x_i) = f_i \quad \text{für } i = 0, 1, \ldots, n \tag{3.1}$$

gilt.

Ein wichtiger Spezialfall dieses Problems ist die *Funktionsinterpolation*: Nehmen wir an, dass wir eine reellwertige Funktion $f : \mathbb{R} \to \mathbb{R}$ gegeben haben, die aber (z.B. weil keine explizite Formel bekannt ist) sehr kompliziert auszuwerten ist. Ein Beispiel einer solchen Funktion ist die in der Stochastik oft benötigte Gauß-Verteilungsfunktion

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{x} e^{-y^2/2} dy,$$

für die keine geschlossene Formel existiert.

Das Ziel der Interpolation liegt nun darin, eine Funktion $F(x)$ zu bestimmen, die leicht auszuwerten ist, und für die für vorgegebene *Stützstellen* x_0, $x_1, \ldots, x_n$ die Gleichung

$$F(x_i) = f(x_i) \quad \text{für } i = 0, 1, \ldots, n \tag{3.2}$$

gilt. Mit der Schreibweise

$$f_i = f(x_i),$$

erhalten wir hier wieder die Bedingung (3.1), weswegen (3.2) tatsächlich ein Spezialfall von (3.1) ist.

Wir werden in diesem Kapitel zum einen Verfahren zur Lösung von (3.1) entwickeln, die dann selbstverständlich auch auf den Spezialfall (3.2) anwendbar sind. Die Wichtigkeit dieses Spezialfalls liegt in diesem Zusammenhang darin, dass man bei der Interpolation einer Funktion f in natürlicher Weise einen *Interpolationsfehler* über den Abstand zwischen f und F definieren kann, und daher ein Maß für die Güte des Verfahrens erhält. Bei

der Dateninterpolation macht dies keinen rechten Sinn, das es ja keine Funktion f gibt, bezüglich der man einen Fehler messen könnte.

Zum anderen werden wir Verfahren betrachten, die speziell auf die Funktionsapproximation (3.2) zugeschnitten sind, da sich bei diesen die Wahl der Stützstellen x_i aus dem Verfahren ergibt, also nicht beliebig vorgegeben werden kann.

3.1 Polynominterpolation

Eine einfache aber oft sehr effektive Methode zur Interpolation ist die Wahl von F als Polynom, also als Funktion der Form

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m. \tag{3.3}$$

Hierbei werden die Werte a_i, $i = 0, \ldots, m$, die *Koeffizienten* des Polynoms genannt. Die höchste auftretende Potenz (hier also m, falls $a_m \neq 0$) heißt der *Grad* des Polynoms. Um zu betonen, dass wir hier Polynome verwenden, schreiben wir in diesem Abschnitt „P“ statt „F“ für die Interpolationsfunktion. Den Raum der Polynome vom Grad $\leq m$ bezeichnen wir mit $\mathcal{P}_m$. Dieser Funktionenraum ist ein $m+1$-dimensionaler Vektorraum über $\mathbb{R}$ bzw. $\mathbb{C}$ mit Basis $\mathcal{B} = \{1, x, \ldots, x^m\}$, da Addition von Polynomen und Multiplikation mit Skalaren wieder ein Polynom des selben Grads ergeben. Andere Basen dieses Vektorraums werden in den Übungen behandelt.

Das Problem der Polynominterpolation liegt nun darin, ein Polynom P zu bestimmen, das (3.1) erfüllt. Zunächst einmal müssen wir uns dazu überlegen, welchen Grad das gesuchte Polynom haben soll. Hier hilft uns der folgende Satz.

Satz 3.1 Sei $n \in \mathbb{N}$ und seien Daten (x_i, f_i) für $i = 0, \ldots, n$ gegeben, so dass die Stützstellen paarweise verschieden sind, d.h. $x_i \neq x_j$ für alle $i \neq j$. Dann gibt es genau ein Polynom $P \in \mathcal{P}_n$, also vom Grad $\leq n$, das die Bedingung

$$P(x_i) = f_i \quad \text{für } i = 0, 1, \ldots, n$$

erfüllt.

Beweis: Die Koeffizienten a_i des interpolierenden Polynoms erfüllen das lineare Gleichungssystem

$$\begin{pmatrix} 1 & x_0 & \ldots & x_0^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}.$$

Die Determinante dieser Matrix ist

$$\prod_{i=0}^{n} \left(\prod_{j=i+1}^{n} (x_j - x_i) \right)$$

und ist damit ungleich Null, falls die x_i paarweise verschieden sind. Also ist die Matrix invertierbar und das Gleichungssystem besitzt eine eindeutige Lösung. □

Für $n+1$ gegebene Datenpunkte (x_i, f_i) "passt" also gerade ein Polynom vom Grad n.

Nun ist es aus verschiedenen Gründen nicht besonders effizient, dieses lineare Gleichungssystem tatsächlich zu lösen, um die a_i zu bestimmen (wir erinnern daran, dass die direkte Lösung des linearen Gleichungssystems den Aufwand der Ordnung $O(n^3)$ hat). Wir betrachten daher eine andere Technik zur Berechnung des Polynoms P. Beachte, dass diese das gleiche Polynom liefert, auch wenn es auf andere Art dargestellt wird.

3.1.1 Lagrange-Polynome und baryzentrische Koordinaten

Die Idee der Lagrange-Polynome beruht auf einer geschickten Darstellung für Polynome. Für die vorgegebenen Stützstellen x_0, $x_1 \ldots, x_n$ definieren wir für $i = 0, \ldots, n$ die *Lagrange-Polynome* L_i als

$$L_i(x) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Man rechnet leicht nach, dass diese Polynome alle vom Grad n sind, und darüberhinaus die Gleichung

$$L_i(x_k) = \begin{cases} 1 & \text{für } i = k \\ 0 & \text{für } i \neq k \end{cases}$$

erfüllen. Mit Hilfe der L_i kann man das Interpolationspolynom einfach explizit berechnen.

Satz 3.2 Seien Daten (x_i, f_i) für $i = 0, \ldots, n$ mit paarweise verschiedenen Stützstellen x_i gegeben. Dann ist das eindeutige Interpolationspolynom $P(x)$ mit $P(x_i) = f_i$ gegeben durch

$$P(x) = \sum_{i=0}^{n} f_i L_i(x).$$

Beweis: Offensichtlich ist die angegebene Funktion ein Polynom vom Grad $\leq n$. Darüberhinaus gilt

$$P(x_k) = \sum_{i=0}^{n} \underbrace{f_i L_i(x_k)}_{\substack{=0 \text{ falls } i \neq k \\ = f_k \text{ falls } i = k}} = f_k,$$

also gerade die gewünschte Bedingung (3.1). □

Beispiel 3.3 Betrachte die Daten (3, 68), (2, 16), (5, 352). Die zugehörigen Lagrange-Polynome sind gegeben durch

$$L_0(x) = \frac{x-2}{3-2}\,\frac{x-5}{3-5} = -\frac{1}{2}(x-2)(x-5),$$

$$L_1(x) = \frac{x-3}{2-3}\,\frac{x-5}{2-5} = \frac{1}{3}(x-3)(x-5),$$

$$L_2(x) = \frac{x-2}{5-2}\,\frac{x-3}{5-3} = \frac{1}{6}(x-2)(x-3).$$

Damit erhalten wir

$$P(x) = -68\,\frac{1}{2}(x-2)(x-5) + 16\,\frac{1}{3}(x-3)(x-5) + 352\frac{1}{6}(x-2)(x-3).$$

Für $x = 3$ ergibt sich $P(3) = -68\frac{1}{2}(3-2)(3-5) = 68$, für $x = 2$ berechnet man $P(2) = 16\frac{1}{3}(2-3)(2-5) = 16$ und für $x = 5$ erhalten wir $P(5) = 352\frac{1}{6}(5-2)(5-3) = 352$. □

Durch Abzählen der notwendigen Operationen sieht man, dass die direkte Auswertung des Polynoms P in dieser Form den Aufwand $O(n^2)$ besitzt, also deutlich effizienter als die Lösung eines linearen Gleichungssystems ist. Für eine effiziente direkte Auswertung sollte man die Nenner der Lagrange-Polynome vorab berechnen und speichern, damit diese nicht bei jeder Auswertung von P erneut berechnet werden müssen.

Es geht aber noch effizienter, wenn wir die Auswertung der Lagrange-Polynome geschickt umformulieren. Dazu schreiben wir den Zähler von

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

als

$$\frac{\ell(x)}{x - x_i} \quad \text{mit} \quad \ell(x) := \prod_{j=0}^{n} x - x_j.$$

Den Nenner schreiben wir mittels der sogenannten *baryzentrischen Koordinaten*

$$w_i := \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{1}{x_i - x_j}.$$

Dann gilt

$$L_i(x) = \ell(x)\frac{w_i}{x - x_i}$$

und damit

$$P(x) = \sum_{i=0}^{n} L_i(x) f_i = \sum_{i=0}^{n} \ell(x)\frac{w_i}{x - x_i} f_i = \ell(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i} f_i.$$

Beispiel 3.4 Betrachte wiederum die Daten (3; 68), (2; 16), (5; 352). Das zugehörige ℓ ist gegeben durch

$$\ell(x) = (x-2)(x-3)(x-5)$$

und die w_i berechnen sich zu

$$w_0 = \frac{1}{3-2}\,\frac{1}{3-5} = -\frac{1}{2},$$

$$w_1 = \frac{1}{2-3}\,\frac{1}{2-5} = \frac{1}{3},$$

$$w_3 = \frac{1}{5-2}\,\frac{1}{5-3} = \frac{1}{6}.$$

Damit erhalten wir

$$\begin{aligned} P(x) &= \ell(x)\left(\frac{-\frac{1}{2}}{x-3}\,68+\frac{\frac{1}{3}}{x-2}\,16+\frac{\frac{1}{6}}{x-5}\,352\right) \\ &= -\frac{1}{2}(x-2)(x-5)\,68+\frac{1}{3}(x-3)(x-5)\,16+\frac{1}{6}(x-2)(x-3)\,352, \end{aligned}$$

also — wie zu erwarten — das gleiche Polynom wie oben. □

Um dieses Verfahren effizient zu implementieren, teilen wir die Berechnung in zwei Algorithmen auf.

Algorithmus 3.5 (Berechnung der baryzentrischen Koordinaten)
Eingabe: Stützstellen $x_0,\ldots,x_n$

(1) für i von 0 bis n:

(2) setze $w_i := 1$

(3) für j von 0 bis n:

(4) falls $j \neq i$, setze $w_i := w_i/(x_i - x_j)$

(6) Ende der Schleifen

Ausgabe: baryzentrische Koordinaten $w_0,\ldots,w_n$ □

Durch Abzählen der Operationen sieht man leicht, dass die Berechnung der w_i gerade $2(n+1)n = 2n^2 + 2n = O(n^2)$ Operationen benötigt. Dies entspricht der Ordnung des Aufwandes der direkten Auswertung von P. Der Trick liegt nun aber darin, die w_i einmal vorab zu berechnen und die gespeicherten Werte in der Auswertung von P zu verwenden.

Algorithmus 3.6 (Auswertung des Interpolationspolynoms)
Eingabe: Stützstellen $x_0,\ldots,x_n$, Werte $f_0,\ldots,f_n$, baryzentrische Koordinaten $w_0,\ldots,w_n$, Auswertungsstelle x

(0) setze $l := 1$, $s := 0$ (Variablen für ℓ und $\sum_{i=0}^{n} \frac{w_i}{x-x_i} f_i$)

(1) für i von 0 bis n

(2) setze $y := x - x_i$

(3) falls $y = 0$ ist, setze $P := f_i$ und beende den Algorithmus

(4) setze $l := l * y$

(5) setze $s := s + w_i * f_i/y$

(6) Ende der Schleife

(7) Setze $P := l * s$

Ausgabe: Polynomwert $P = P(x)$ □

Durch Abzählen der Operationen sieht man: die Auswertung benötigt gerade $5(n+1)+1 = 5n+6 = O(n)$ Operationen. Sind also die w_i einmal berechnet, so ist die Auswertung für ein gegebenes x deutlich weniger aufwändig als die direkte Auswertung von P. Dies ist z.B. bei der grafischen Darstellung des Polynoms ein wichtiger Vorteil, da das Polynom dabei für viele verschiedene x ausgewertet werden muss.

3.1.2 Fehlerabschätzungen

Wir betrachten in diesem Abschnitt das Problem der Funktionsinterpolation (3.2) für eine gegebene Funktion $f : \mathbb{R} \to \mathbb{R}$. Wir wollen abschätzen, wie groß der Abstand des interpolierenden Polynoms P von der Funktion f ist. Hierbei bezeichnen wir mit $[a, b]$ ein Interpolationsintervall mit der Eigenschaft, dass alle Stützstellen $x_i \in [a, b]$ erfüllen und wir verwenden wieder die Maximumsnorm

$$\|f\|_\infty := \sup_{x\in[a,b]} |f(x)|.$$

Der folgende Satz gibt eine Abschätzung für den Abstand in Abhängigkeit von den Stützstellen an. Hierbei bezeichnet $f^{(k)}$ die k-te Ableitung der Funktion f.

Satz 3.7 Sei f $(n+1)$-mal stetig differenzierbar und sei P das Interpolationspolynom zu den paarweise verschiedenen Stützstellen $x_0, \ldots, x_n$. Dann gelten die folgenden Ausagen.

(i) Für alle $x \in [a, b]$ gibt es ein $\xi \in [a, b]$, so dass die Gleichung

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)$$

gilt.

(ii) Für alle $x \in [a, b]$ gilt die Abschätzung

$$|f(x) - P(x)| \le \|f^{(n+1)}\|_\infty \left|\frac{(x - x_0)(x - x_1)\cdots(x - x_n)}{(n+1)!}\right|.$$

(iii) Es gilt die Abschätzung

$$\|f - P\|_\infty \le \|f^{(n+1)}\|_\infty \frac{(b-a)^{n+1}}{(n+1)!}.$$

Beweis: (i) Wähle ein $x \in [a, b]$ mit $x \neq x_i$, $i = 0, \ldots, n$, und setze

$$c = \frac{f(x) - P(x)}{(x - x_0)\cdots(x - x_n)}.$$

Mit diesem c definieren wir die Funktion

$$\Delta(y) = f(y) - P(y) - c(y - x_0)\cdots(y - x_n).$$

Dann ist $\Delta(y)$ wieder $(n+1)$-mal stetig differenzierbar und hat (mindestens) die $n+2$ Nullstellen $y = x_0, \ldots, x_n$ und $y = x$. Wir nummerieren diese Nullstellen aufsteigend mit der Bezeichnung $y_0^{(0)} < y_1^{(0)} < \ldots < y_{n+1}^{(0)}$. Nach dem Satz von Rolle gilt: Zwischen je zwei Nullstellen der Funktion $\Delta(y)$ liegt (mindestens) eine Nullstelle ihrer Ableitung $\Delta'(y)$. Also liegt für jedes $i = 0, \ldots, n$ zwischen den Werten $y_i^{(0)}$ und $y_{i+1}^{(0)}$ ein Wert $y_i^{(1)}$ mit $\Delta'(y_i^{(1)}) = 0$, und wir erhalten $n+1$ Nullstellen $y_0^{(1)} < y_1^{(1)} < \ldots < y_n^{(1)}$ für die Ableitung $\Delta'(y) = \Delta^{(1)}(y)$. Indem wir induktiv fortfahren, erhalten wir $n+2-k$ Nullstellen $y_0^{(k)} < y_1^{(k)} < \ldots < y_{n+1-k}^{(k)}$ für die Ableitung $\Delta^{(k)}$ und damit für $k = n+1$ eine Nullstelle $\xi = y_0^{(n+1)}$ für die Funktion $\Delta^{(n+1)}$.

Da P ein Polynom vom Grad $\leq n$ ist, folgt $P^{(n+1)}(y) \equiv 0$, außerdem gilt $[(y-x_0)\cdots(y-x_n)]^{(n+1)} = (n+1)!$ für alle $y \in \mathbb{R}$. Damit erhalten wir

$$0 = \Delta^{(n+1)}(\xi) = f^{(n+1)}(\xi) - c(n+1)! = f^{(n+1)}(\xi) - \frac{f(x) - P(x)}{(x-x_0)\cdots(x-x_n)}(n+1)!,$$

also

$$f^{(n+1)}(\xi) = \frac{f(x) - P(x)}{(x-x_0)\cdots(x-x_n)}(n+1)!.$$

Auflösen nach $f(x) - P(x)$ liefert die Gleichung in (i).

(ii) Diese Abschätzung folgt aus (i) wegen

$$\begin{aligned} |f(x) - P(x)| &= \left| \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n) \right| \\ &\leq \max_{y\in[a,b]} \left| \frac{f^{(n+1)}(y)}{(n+1)!}\,(x-x_0)(x-x_1)\cdots(x-x_n) \right|, \end{aligned}$$

da $\xi \in [a,b]$ ist.

(iii) Für alle x und x_i aus $[a,b]$ gilt die Abschätzung

$$|x - x_i| \leq b - a.$$

Damit erhalten wir aus (ii)

$$\|f - P\|_\infty \leq \|f^{(n+1)}\|_\infty \frac{(b-a)^{n+1}}{(n+1)!},$$

also gerade die Behauptung. □

Wir illustrieren diese Abschätzung an einem Beispiel.

Beispiel 3.8 Betrachte die Funktion $f(x) = \sin(x)$ auf dem Intervall $[0, 2\pi]$. Die Ableitungen von f sind

$$f^{(1)}(x) = \cos(x), \quad f^{(2)}(x) = -\sin(x), \quad f^{(3)}(x) = -\cos(x), \quad f^{(4)}(x) = \sin(x), \quad \ldots$$

Für alle diese Funktionen gilt $|f^{(k)}(x)| \leq 1$ für alle $x \in \mathbb{R}$. Mit äquidistanten Stützstellen $x_i = 2\pi i/n$ ergibt sich damit die Abschätzung

$$|f(x) - P(x)| \leq \max_{y \in [a,b]} \left| f^{(n+1)}(y) \right| \frac{(b-a)^{n+1}}{(n+1)!} \leq \frac{(2\pi)^{n+1}}{(n+1)!}.$$

Dieser Term konvergiert für wachsende n sehr schnell gegen 0, weswegen man schon für kleine n eine sehr gute Übereinstimmung der Funktionen erwarten kann, siehe Abbildung 3.1. □

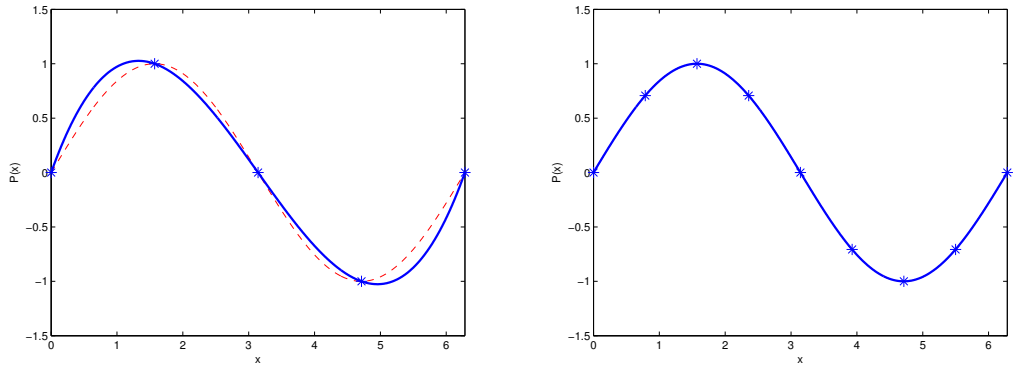

Abbildung 3.1: Links: Interpolationspolynom für 4 (links) und 8 (rechts) äquidistante Stützstellen für die Funktion $f(x) = \sin(x)$ auf $[0, 2\pi]$.

Fügt man jedoch in Beispiel 3.8 weitere Stützstellen ein, beobachtet man Oszillationen am Rand des Intervalls $[0, 2\pi]$ (siehe Abbildung 3.2) — trotz unser Fehlerabschätzung aus Satz 3.7. Dies muss ein numerischer Effekt aus, was eine Untersuchung der Kondition des Interpolationsproblems motiviert.

3.1.3 Kondition

In diesem Abschnitt wollen wir die Kondition der Polynominterpolation betrachten, wobei wir das Polynominterpolationsproblem für fest vorgegebene Stützstellen betrachten. In diesem Fall ist die Abbildung

$$\phi : (f_0, \ldots, f_n) \mapsto \sum_{i=0}^{n} f_i L_i$$

des Datenvektors $(f_0, \ldots, f_n)$ auf das interpolierende Polynom $P \in \mathcal{P}_n$ eine lineare Abbildung $\phi : \mathbb{R}^{n+1} \to \mathcal{P}_n$, weshalb wir die (absolute) Kondition κ_{abs} als induzierte Operatornorm

$$\kappa_{abs} := \|\phi\|_\infty = \sup_{\substack{f \in \mathbb{R}^{n+1} \\ f \neq 0}} \frac{\|\phi(f)\|_\infty}{\|f\|_\infty}$$

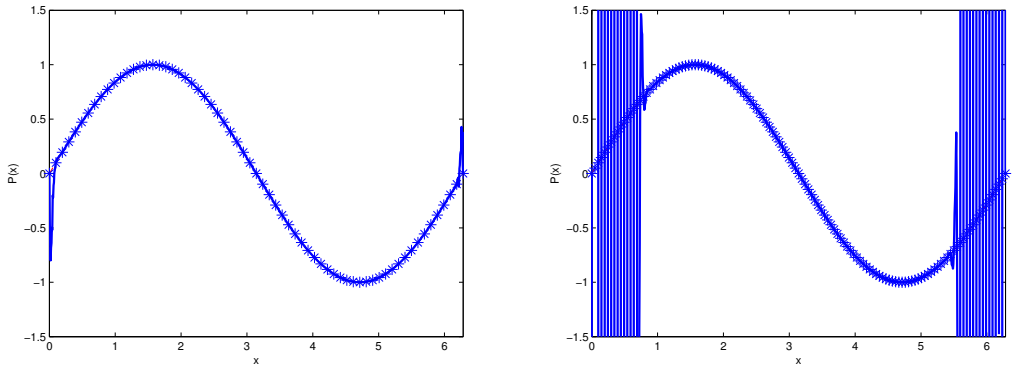

Abbildung 3.2: Links: Interpolationspolynom für 64 (links) und 128 (rechts) äquidistante Stützstellen für die Funktion $f(x) = \sin(x)$ auf $[0, 2\pi]$.

dieser linearen Abbildung berechnen können. Diese induzierte Operatornorm ist die Erweiterung der induzierten Matrixnorm auf lineare Abbildungen, die nicht notwendigerweise durch eine Matrix definiert sind. Im Gegensatz zu den linearen Gleichungssystemen verwenden wir hier die absolute Kondition, weil eine relative Definition hier keine anschauliche Interpretation besitzt. Auf dem Polynomraum $\mathcal{P}_n$ verwenden wir dabei die Maximumsnorm

$$\|P\|_\infty := \max_{x\in[a,b]} |P(x)|,$$

des Raums der stetigen reellwertigen Funktionen $C([a,b],\mathbb{R})$, wobei wir $a = \min_{i=0,\dots,n} x_i$ und $b = \max_{i=0,\dots,n} x_i$ wählen (beachte, dass wir keine Ordnung der Stützstellen x_i vorausgesetzt haben).

Satz 3.9 Seien x_0, $x_1, \dots, x_n$ paarweise verschiedene Stützstellen und L_i die zugehörigen Lagrange-Polynome. Dann ist die absolute Kondition des Interpolationsproblems mit diesen Stützstellen gegeben durch

$$\kappa_{\text{abs}} = \Lambda_n := \left\| \sum_{i=0}^{n} |L_i| \right\|_\infty ,$$

wobei Λ_n als *Lebesgue-Konstante* bezeichnet wird.

Beweis: Es gilt

$$\begin{aligned} |\phi(f)(x)| &= \left| \sum_{i=0}^{n} f_i L_i(x) \right| \le \sum_{i=0}^{n} |f_i|\,|L_i(x)| \\ &\le \|f\|_\infty \max_{x\in[a,b]} \sum_{i=0}^{n} |L_i(x)| = \|f\|_\infty \left\| \sum_{i=0}^{n} |L_i| \right\|_\infty = \|f\|_\infty \Lambda_n, \end{aligned}$$

für alle $x \in [a,b]$, woraus $\|\phi(f)\|_\infty \le \|f\|_\infty \Lambda_n$ für alle $f \in \mathbb{R}^{n+1}$ und damit $\|\phi\| \le \Lambda_n$ folgt. Für die umgekehrte Richtung konstruieren wir ein $g \in \mathbb{R}^{n+1}$ so, dass

$$|\phi(g)(x^*)| = \|g\|_\infty \left\| \sum_{i=0}^{n} |L_i| \right\|_\infty$$

für ein $x^* \in [a,b]$ gilt. Sei dazu $x^* \in [a,b]$ die Stelle, an der die Funktion $x \mapsto \sum_{i=0}^{n} |L_i(x)|$ ihr Maximum annimmt, also

$$\sum_{i=0}^{n} |L_i(x^*)| = \left\| \sum_{i=0}^{n} |L_i| \right\|_\infty = \Lambda_n.$$

Wir wählen $g \in \mathbb{R}^{n+1}$ als $g_i = \operatorname{sgn}(L_i(x^*))$. Dann gilt $\|g\|_\infty = 1$ und $g_i L_i(x^*) = |L_i(x^*)|$, also

$$\|\phi(g)\|_\infty \geq |\phi(g)(x^*)| = \left| \sum_{i=0}^{n} g_i L_i(x^*) \right| = \left| \sum_{i=0}^{n} |L_i(x^*)| \right| = \|g\|_\infty \left| \sum_{i=0}^{n} |L_i(x^*)| \right| = \|g\|_\infty \Lambda_n,$$

weswegen $\|\phi\| \geq \Lambda_n$ ist. Zusammen erhalten wir also die Behauptung $\kappa_{\text{abs}} = \|\phi\| = \Lambda_n$. □

Die Zahl Λ_n hängt natürlich von der Anzahl und Lage der Stützstellen ab. In der folgenden Tabelle 3.1 sind die Konditionen für das Intervall $[-1,1]$ und verschiedene Anzahlen äquidistante Stützstellen $x_i = -1 + 2i/n$ sowie für die sogenannten Tschebyscheff-Stützstellen $x_i = \cos[(2i+1)\pi/(2n+2)]$, die wir in Abschnitt 3.2 näher kennen lernen werden, dargestellt.

n	κ_{abs} für äquidistante Stützstellen	κ_{abs} für Tschebyscheff-Stützstellen
5	3.11	2.10
10	29.89	2.49
15	512.05	2.73
20	10986.53	2.90
60	$2.97 \cdot 10^{15}$	3.58
100	$1.76 \cdot 10^{27}$	3.90

Tabelle 3.1: Kondition κ_{abs} für verschiedene Stützstellen

Man sieht, dass das Problem für äquidistante Stützstellen und große n sehr schlecht konditioniert ist. Dies erklärt die starken Oszillationen bei numerisch erzeugten Interpolationspolynome bei großer Stützstellenanzahl, selbst wenn die zu interpolierende Funktion gutartig ist (vergleiche Abbildung 3.2). Wir werden daher in den nächsten Abschnitten weitere Möglichkeiten zur Interpolation betrachten, die diese Probleme umgehen, indem sie entweder besser positionierte Stützstellen verwenden oder ohne Polynome hohen Grades auskommen.

3.2 Funktionsinterpolation und Stützstellenwahl

In diesem Abschnitt beschäftigen wir uns speziell mit der Frage der Funktionsinterpolation (3.2) durch Polynome. Wie bereits erwähnt, unterscheidet sich diese aus algorithmischer Sicht von der Dateninterpolation (3.1) dadurch, dass man die Stützstellen x_i frei wählen kann. Dies führt auf die Frage, wie man diese Stützstellen für ein gegebenenes Interpolationsintervall $[a,b]$ optimal wählen kann. Wir wollen dieses Problem lösen *ohne* die Kenntnis der zu interpolierenden Funktion f vorauszusetzen.

Dazu betrachten wir die Fehlerabschätzung aus Satz 3.7(ii) für die Funktionsinterpolation. Dort haben wir die Ungleichung

$$|f(x) - P(x)| \leq \|f^{(n+1)}\|_\infty \left| \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} \right|. \tag{3.4}$$

für $x \in [a,b]$ bewiesen. Wir wollen nun untersuchen, wie man die Stützstellen x_i wählen muss, so dass diese Fehlerschranke minimal wird. Da wir hierbei kein spezielles x vorgeben wollen (die Abschätzung soll für alle x optimal sein, also für $\|f-P\|_\infty$), besteht die Aufgabe also darin, Stützstellen $x_0, \ldots, x_n$ zu finden, so dass der Ausdruck

$$\max_{x\in[a,b]} |(x-x_0)(x-x_1)\cdots(x-x_n)| \tag{3.5}$$

minimal wird.

O.B.d.A. betrachten wir nun das Intervall $[a,b] = [-1,1]$, denn wenn wir auf $[-1,1]$ die optimalen Stützstellen x_i gefunden haben, so sind die mittels $\tilde{x}_i = a + (x_i+1)(b-a)/2$ definierten Stützstellen auf $[a,b]$ ebenfalls optimal und es gilt

$$\max_{x\in[a,b]} |(x-\tilde{x}_0)(x-\tilde{x}_1)\cdots(x-\tilde{x}_n)| = \left(\frac{b-a}{2}\right)^{n+1} \max_{x\in[-1,1]} |(x-x_0)(x-x_1)\cdots(x-x_n)|.$$

Definieren wir nun für beliebige Stützstellen $x_0, \ldots, x_n$ das Polynom $R_{n+1}(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$ (führender Koeffizient ist $a_{n+1} = 1$). Dann sind die Stützstellen x_i gerade die Nullstellen von R_{n+1} und der Ausdruck (3.5) ist gerade die Maximumsnorm $\|R_{n+1}\|_\infty$. Die Minimierung von (3.5) ist also äquivalent zur folgenden Problemstellung: Unter allen Polynomen R_{n+1} vom Grad $n+1$ mit führendem Koeffizienten $a_{n+1} = 1$ finde dasjenige mit kleinster Maximumsnorm auf $[-1,1]$.

Der folgende Satz zeigt, dass die bereits in Tabelle 3.1 aufgeführten Tschebyscheff-Knoten (Nullstellen der zugehörigen normierten Tschebyscheff-Polynoms) gerade das Minimum dieses Optimierungsproblems liefern.[1]

Satz 3.10 Die Tschebyscheff-Knoten

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \quad i = 0, \ldots, n,$$

minimieren den Ausdruck (3.5) und damit die Fehlerabschätzung (3.4).

Für die Tschebyscheff-Stützstellen x_i erhält man damit in (3.5)

$$\max_{x\in[-1,1]} |(x-x_0)(x-x_1)\cdots(x-x_n)| = \frac{1}{2^n}.$$

Für allgemeine Intervalle $[a,b]$ ergibt sich daraus für die durch

$$\tilde{x}_i = a + (x_i+1)\,(b-a)/2$$

[1] Ein Beweis findet sich im bereits erwähnten Skript von Prof. Grüne "Einführung in die numerische Mathematik".

gegebenen transformierten Stützstellen

$$\max_{x\in[a,b]} |(x-\tilde{x}_0)(x-\tilde{x}_1)\cdots(x-\tilde{x}_n)| = \frac{(b-a)^{n+1}}{2^{2n+1}} = 2\left(\frac{b-a}{4}\right)^{n+1}.$$

Beachte, dass die Randpunkte -1 und 1 des Interpolationsintervalls keine Tschebyscheff-Knoten und damit keine Stützstellen sind. Wir interpolieren mit dieser Methode also auch außerhalb des durch die Stützstellen definierten Intervalls.

Zusammenfassend kann man feststellen, dass die Tschebyscheff-Stützstellen sowohl dafür sorgen, dass das Interpolationsproblem gut konditioniert ist als auch unsere Abschätzung an den Interpolationsfehler minimieren.

3.3 Splineinterpolation

Wir haben gesehen, dass die Polynominterpolation aus Konditionsgründen problematisch ist, wenn wir viele Stützstellen gegeben haben und diese nicht — wie die Tschebyscheff–Stützstellen — optimal gewählt sind. Dies kann insbesondere bei der Dateninterpolation (3.1) auftreten, wenn die Stützstellen fest vorgegeben sind und nicht frei gewählt werden können. Wir behandeln daher in diesem Abschnitt eine alternative Interpolationstechnik, die auch bei einer großen Anzahl von Stützstellen problemlos funktioniert. Wir betrachten dazu paarweise verschiedene Stützstellen und nehmen an, dass diese aufsteigend angeordnet sind, also $x_0 < x_1 < \ldots < x_n$ gilt.

Die Grundidee der *Splineinterpolation* liegt darin, die interpolierende Funktion (die wir hier mit "S" für "Spline" bezeichnen) nicht global, sondern nur auf jedem Teilintervall $[x_i, x_{i+1}]$ als Polynom zu wählen. Diese Teilpolynome sollten dabei an den Intervallgrenzen nicht beliebig sondern möglichst glatt zusammenlaufen. Eine solche Funktion, die aus glatt zusammengefügten stückweisen Polynomen besteht, nennt man *Spline*.

Formal wird dies durch die folgende Definition präzisiert.

Definition 3.11 Seien $x_0 < x_1 < \ldots < x_n$ Stützstellen und $k \in \mathbb{N}$. Eine stetige und $(k-1)$-mal stetig differenzierbare Funktion $S : [x_0, x_n] \to \mathbb{R}$ heißt *Spline vom Grad k*, falls S auf jedem Intervall $I_i = [x_{i-1}, x_i]$ mit $i = 1, \ldots, n$ durch ein Polynom P_i vom Grad $\leq k$ gegeben ist, d.h. für $x \in I_i$ gilt

$$S(x) = P_i(x) = a_{i\,0} + a_{i\,1}(x - x_{i-1}) + \ldots + a_{i\,k}(x - x_{i-1})^k = \sum_{j=0}^{k} a_{i\,j}(x - x_{i-1})^j. \quad (3.6)$$

Den Raum der Splines vom Grad k zur Stützstellenmenge $\Delta = \{x_0, x_1, \ldots, x_n\}$ bezeichnen wir mit $S_{\Delta,k}$. □

Ein solcher Spline aus Definition 3.11 löst dann das Interpolationsproblem, falls zusätzlich die Bedingung (3.1) erfüllt ist, also $S(x_i) = f_i$ für alle $i = 0, \ldots, n$ gilt.

Bevor wir an die Berechnung von Splines gehen, zeigen wir eine Eigenschaft des Funktionenraums $S_{\Delta,k}$.

Satz 3.12 Sei $\Delta = \{x_0, x_1, \ldots, x_n\}$ mit $x_0 < x_1 < \ldots < x_n$ und $k \in \mathbb{N}$ gegeben. Dann ist der Raum der Splines $S_{\Delta,k}$ ein $k+n$-dimensionaler Vektorraum über $\mathbb{R}$.

Beweis: Sicherlich ist mit S_1 und S_2 auch $aS_1 + bS_2$ für a, $b \in \mathbb{R}$ wieder ein Spline, also ist $S_{\Delta,k}$ ein Vektorraum. Da die Splines linear von den Koeffizienten a_{ij} abhängen, genügt es zur Berechnung der Dimension die Anzahl der freien Parameter a_{ij} zu bestimmen. Auf dem ersten Intervall I_1 haben wir freie Wahl für P_1, also gibt es genau $k+1$ freie Parameter. Auf jedem weiteren Intervall I_i für $i \geq 2$ sind die Werte der j-ten Ableitung $P_i^{(j)}(x_{i-1}) = j!a_{ij}$ für $j = 0, \ldots, k-1$ bereits festgelegt, da die zusammengesetzte Funktion S in x_{i-1} ja stetig und $k-1$-mal stetig differenzierbar ist, es muss also

$$a_{ij} = P_{i-1}^{(j)}(x_{i-1})/j! \quad \text{für } j = 0, \ldots, k-1$$

gelten. Daher ist hier nur noch der Koeffizient a_{ik} frei wählbar, was auf den $n-1$ verbleibenden Intervallen gerade $n-1$ weitere freie Parameter ergibt, also insgesamt $k+n$. □

Tatsächlich kann man beweisen, dass die Spline-Koeffizienten a_{ij} linear voneinander abhängen, statt der im Beweis des Satzes verwendeten Koeffizienten $a_{10}, \ldots, a_{1k}, a_{2k}, \ldots, a_{nk}$ kann man also auch andere $n+k$ Koeffizienten vorgeben und die übrigen daraus berechnen.

Von besonderer praktischer Bedeutung in vielen Anwendungen sind die *kubischen Splines*, also Splines vom Grad $k = 3$; den Grund dafür werden wir etwas später besprechen. Wegen der Darstellung (3.6) von $P_i(x)$ und der Interpolationsbedingung (3.1) sind für die Splineinterpolation die Koeffizienten a_{i0}, $i = 1, 2, \ldots, n$, bereits festgelegt:

$$a_{i0} = f_{i-1} \qquad \text{für} \qquad i = 1, \ldots, n.$$

Entsprechend können wir (3.1) für $x = x_n$ sicherstellen, indem wir

$$a_{n1} = \frac{f_n - a_{n0} - a_{n2}(x_n - x_{n-1})^2 - a_{n3}(x_n - x_{n-1})^3}{x_n - x_{n-1}}.$$

setzen. Dann haben wir insgesamt genau $n+1$ Koeffizienten festgelegt und sichergestellt, dass die Interpolationsbedingung (3.1) für gegebene Daten (x_i, f_i) mit $x_0 < x_1 < \ldots < x_n$ erfüllt ist. Da der Vektorraum $S_{\Delta,3}$ Dimension $n+3$ hat, verbleiben also 2 weitere Koeffizienten, die durch geeignete Bedingungen festgelegt werden müssen, um einen eindeutigen interpolierenden Spline zu erhalten. Diese werden typischerweise in Form von Randbedingungen, also Bedingungen an S oder an Ableitungen von S in den Punkten x_0 und x_n formuliert:

(a) $S''(x_0) = S''(x_n) = 0$ ("natürliche Randbedingungen")

(b) $S'(x_0) = S'(x_n)$ und $S''(x_0) = S''(x_n)$ ("periodische Randbedingungen")

(c) $S'(x_0) = f'(x_0)$ und $S'(x_n) = f'(x_n)$ ("hermite'sche Randbedingungen", nur sinnvoll bei der Funktionsinterpolation)

Dann gibt es genau einen kubischen Spline $S \in S_{\Delta 3}$, der das Interpolationsproblem (3.1) löst und die entsprechende Randbedingung (a), (b) oder (c) erfüllt.

Kubische Splines werden in Anwendungen wie z.B. der Computergrafik bevorzugt verwendet, und wir wollen als nächstes den Grund dafür erläutern. Ein Kriterium zur Wahl der Ordnung eines Splines — speziell bei grafischen Anwendungen, aber auch bei "klassischen" Interpolationsproblemen — ist, dass die Krümmung der interpolierenden Kurve möglichst klein sein soll. Die Krümmung einer Kurve $y(x)$ in einem Punkt x ist gerade gegeben durch die zweite Ableitung $y''(x)$. Die Gesamtkrümmung für alle $x \in [x_0, x_n]$ kann nun auf verschiedene Arten gemessen werden, hier verwenden wir die L_2-Norm $\|\cdot\|_2$ für quadratisch integrierbare Funktionen, die für $g : [x_0, x_n] \to \mathbb{R}$ durch

$$\|g\|_2 := \left(\int_{x_0}^{x_n} g^2(x) dx \right)^{\frac{1}{2}}$$

gegeben ist. Die Krümmung einer zweimal stetig differenzierbaren Funktion $y : [x_0, x_n] \to \mathbb{R}$ über dem gesamten Intervall kann also mittels $\|y''\|_2$ gemessen werden. Hierfür gilt der folgende Satz.

Satz 3.13 Sei $S : [x_0, x_n] \to \mathbb{R}$ ein die Daten (x_i, f_i), $i = 0, \ldots, n$ interpolierender kubischer Spline, der eine der Randbedingungen (a)–(c) erfüllt. Sei $y : [x_0, x_n] \to \mathbb{R}$ eine zweimal stetig differenzierbare Funktion, die ebenfalls das Interpolationsproblem löst und die gleichen Randbedingungen wie S erfüllt. Dann gilt

$$\|S''\|_2 \leq \|y''\|_2.$$

Beweis: Setzen wir die offensichtliche Gleichung $y'' = S'' + (y'' - S'')$ in die quadrierte Norm $\|y''\|_2^2$ ein, so folgt

$$\begin{aligned} \|y''\|_2^2 &= \int_{x_0}^{x_n} (y''(x))^2 dx \\ &= \underbrace{\int_{x_0}^{x_n} (S''(x))^2 dx}_{=\|S''\|_2^2} + \underbrace{2 \int_{x_0}^{x_n} S''(x)(y''(x) - S''(x)) dx}_{=:J} + \underbrace{\int_{x_0}^{x_n} (y''(x) - S''(x))^2 dx}_{\geq 0} \\ &\geq \|S''\|_2^2 + J. \end{aligned}$$

Wir betrachten nun den Term J genauer. Aus jeder der drei Randbedingungen folgt die Gleichung

$$\Big[S''(x)(y'(x) - S'(x)) \Big]_{x=x_0}^{x_n} = S''(x_n)(y'(x_n) - S'(x_n)) - S''(x_0)(y'(x_0) - S'(x_0)) = 0. \quad (3.7)$$

Mit partieller Integration gilt

$$\int_{x_0}^{x_n} S''(x)(y''(x) - S''(x)) dx = \Big[S''(x)(y'(x) - S'(x)) \Big]_{x=x_0}^{x_n} - \int_{x_0}^{x_n} S'''(x)(y'(x) - S'(x)) dx$$

Hierbei ist der erste Summand wegen (3.7) gleich Null. Auf jedem Intervall $I_i = [x_{i-1}, x_i]$ ist $S(x) = P_i(x)$ ein kubisches Polynom, weswegen $S'''(x) \equiv d_i$ konstant für $x \in I_i$ ist. Also

folgt für den zweiten Summanden

$$\begin{aligned}\int_{x_0}^{x_n} S'''(x)(y'(x)-S'(x))dx &= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} d_i(y'(x)-S'(x))dx \\ &= \sum_{i=1}^{n} d_i \int_{x_{i-1}}^{x_i} y'(x)-S'(x)dx \\ &= \sum_{i=1}^{n} d_i \underbrace{[(y(x_i)-y(x_{i-1})-S(x_i)+S(x_{i-1})]}_{=0,\,\text{da } y(x_i)=S(x_i) \text{ und } y(x_{i-1})=S(x_{i-1})} = 0\end{aligned}$$

Damit erhalten wir $J = 0$ und folglich die Behauptung. □

Diese Eigenschaft erklärt auch den Namen Spline: Ein "Spline" ist im Englischen eine dünne Holzlatte. Wenn man diese so verbiegt, dass sie vorgegebenen Punkten folgt (diese also "interpoliert"), so ist auch bei dieser Latte die Krümmung, die hier näherungsweise die notwendige "Biegeenergie" beschreibt, minimal — zumindest für kleine Auslenkungen der Latte.

Wir kommen nun zur praktischen Berechnung der Spline-Koeffizienten für kubische Splines. Hierbei gibt es verschiedene Vorgehensweisen: man kann z.B. direkt ein lineares Gleichungssystem für die $4n$ Koeffizienten a_{ij} für $i = 1, \ldots, n$ aufstellen, was aber wenig effizient ist. Alternativ kann man geschickt gewählte Basis-Funktionen für den Vektorraum $S_{\Delta,3}$ verwenden (sogenannte B-Splines), und S in dieser Basis berechnen; dieser Ansatz wird im Buch von Deuflhard/Hohmann beschrieben. Dies führt auf ein n-dimensionales lineares Gleichungssystem mit tridiagonaler Matrix A. Es werden bei diesem Verfahren allerdings nicht die Koeffizienten a_{ij} berechnet, sondern die Koeffizienten bezüglich der B-Spline-Basis, die Auswertung von S muss demnach ebenfalls über diese Basisfunktionen erfolgen.

Hier stellen wir eine weitere Variante vor, mit der direkt die Koeffizienten a_{ij} berechnet werden, so dass S über die Darstellung in Definition 3.11 ausgewertet werden kann, was z.B. der Vorgehensweise in MATLAB entspricht. Wir kommen hierbei ebenfalls auf ein n-dimensionales lineares Gleichungssystem mit tridiagonaler Matrix A, so dass der Aufwand der Berechnung von der Ordnung $O(n)$ ist. Wir betrachten zuerst die natürlichen Randbedingungen.

Hierzu definieren wir zunächst die Werte

$$f_i'' := S''(x_i) \quad \text{und} \quad h_i := x_i - x_{i-1}$$

für $i = 0, \ldots, n$ bzw. $i = 1, \ldots, n$. Aus der Interpolationsbedingung und der geforderten Stetigkeit der zweiten Ableitung erhält man 4 Gleichungen für die Teilpolynome P_i:

$$P_i(x_{i-1}) = f_{i-1}, \;\; P_i(x_i) = f_i, \;\; P_i''(x_{i-1}) = f_{i-1}'', \;\; P_i''(x_i) = f_i''. \tag{3.8}$$

Löst man diese — unter Ausnutzung der Ableitungsregeln für Polynome — nach den a_{ij}

auf, so erhält man

$$\begin{aligned} a_{i\,0} &= f_{i-1} \\ a_{i\,1} &= \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(f_i'' + 2f_{i-1}'') \\ a_{i\,2} &= \frac{f_{i-1}''}{2} \\ a_{i\,3} &= \frac{f_i'' - f_{i-1}''}{6h_i}. \end{aligned}$$

Da die Werte h_i und f_i ja direkt aus den Daten verfügbar sind, müssen lediglich die Werte f_i'' berechnet werden. Da aus den natürlichen Randbedingungen sofort $f_0'' = 0$ und $f_n'' = 0$ folgt, brauchen nur die Werte $f_1'', \ldots, f_{n-1}''$ berechnet werden.

Beachte, dass wir in (3.8) bereits die Bedingungen an P_i und P_i'' in den Stützstellen verwendet haben. Aus den noch nicht benutzten Gleichungen für die ersten Ableitungen erhält man nun die Gleichungen für die f_i'': Aus $P_i'(x_i) = P_{i+1}'(x_i)$ erhält man

$$a_{i\,1} + 2a_{i\,2}(x_i - x_{i-1}) + 3a_{i\,3}(x_i - x_{i-1})^2 = a_{i+1\,1}$$

für $i = 1, \ldots, n-1$. Indem man hier die Werte f_i'' und h_i gemäß den obigen Gleichungen bzw. Definitionen einsetzt, erhält man

$$h_i f_{i-1}'' + 2(h_i + h_{i+1}) f_i'' + h_{i+1} f_{i+1}'' = 6\frac{f_{i+1} - f_i}{h_{i+1}} - 6\frac{f_i - f_{i-1}}{h_i} =: \delta_i$$

für $i = 1, \ldots, n-1$. Dies liefert genau $n-1$ Gleichungen für die $n-1$ Unbekannten $f_1'', \ldots, f_{n-1}''$. In Matrixform geschrieben erhalten wir so das Gleichungssystem

$$\begin{pmatrix} 2(h_1+h_2) & h_2 & 0 & \cdots & \cdots & 0 \\ h_2 & 2(h_2+h_3) & h_3 & \ddots & & \vdots \\ 0 & h_3 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & h_{n-1} \\ 0 & \cdots & \cdots & 0 & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix} \begin{pmatrix} f_1'' \\ f_2'' \\ \vdots \\ \vdots \\ f_{n-1}'' \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \vdots \\ \delta_{n-1} \end{pmatrix}$$

Zur Berechnung des Interpolationssplines löst man also zunächst dieses Gleichungssystem und berechnet dann gemäß der obigen Formel die Koeffizienten $a_{i\,j}$ aus den f_k''.

Für äquidistante Stützstellen, also $x_k - x_{k-1} = h_k = h$ für alle $k = 1, \ldots, n$, kann man beide Seiten durch h teilen, und erhält so das Gleichungssystem

$$\begin{pmatrix} 4 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 4 & 1 & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ \vdots & & \ddots & 1 & 4 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} f_1'' \\ f_2'' \\ \vdots \\ \vdots \\ f_{n-1}'' \end{pmatrix} = \begin{pmatrix} \tilde{\delta}_1 \\ \tilde{\delta}_2 \\ \vdots \\ \vdots \\ \tilde{\delta}_{n-1} \end{pmatrix}$$

mit $\tilde{\delta_k} = \delta_k/h$, welches ein Beispiel für ein lineares Gleichungssystem mit (offensichtlich) diagonaldominanter Matrix ist.

Für andere Randbedingungen verändert sich dieses Gleichungssystem entsprechend.

Zum Abschluss wollen wir noch kurz auf den Interpolationsfehler bei der Splineinterpolation eingehen. Die Analyse dieses Fehlers ist recht langwierig, das Ergebnis, das wir hier ohne Beweis geben, allerdings sehr leicht darzustellen. Der folgende Satz wurde von C.A. Hall und W.W. Meyer [4] bewiesen.

Satz 3.14 Sei $S \in S_{\Delta,3}$ der interpolierende Spline einer 4 mal stetig differenzierbaren Funktion f mit hermite'schen Randbedingungen und Stützstellen $\Delta = \{x_0, \ldots, x_n\}$. Dann gilt für $h = \max_k(x_k - x_{k-1})$ die Abschätzung

$$\|f - S\|_\infty \leq \frac{5}{384} h^4 \|f^{(4)}\|_\infty$$