Numerische Verfahren

Jens-Peter M. Zemke
zemke@tu-harburg.de

Institut für Numerische Simulation
Technische Universität Hamburg-Harburg

08.04.2008

TUHH
Technische Universität Hamburg-Harburg

Übersicht

Interpolationsproblem

Gegeben sei eine auf einem Intervall I erklärte Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

Interpolationsproblem

Gegeben sei eine auf einem Intervall I erklärte Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

die von n Parametern $a_1, \ldots, a_n$ abhängt.

Interpolationsproblem

Gegeben sei eine auf einem Intervall I erklärte Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

die von n Parametern $a_1, \ldots, a_n$ abhängt.

Zusätzlich seien paarweise verschiedene Knoten (oder Stützstellen) $x_1, \ldots, x_m \in I$, Vielfachheiten $r_1, \ldots, r_m \in \mathbb{N}$ mit $\sum_{i=1}^{m} r_i = n$ und Werte $y_{ij} \in \mathbb{R}$, $i = 1, \ldots, m$, $j = 0, \ldots, r_i - 1$, gegeben.

Interpolationsproblem

Gegeben sei eine auf einem Intervall I erklärte Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

die von n Parametern $a_1, \ldots, a_n$ abhängt.

Zusätzlich seien paarweise verschiedene Knoten (oder Stützstellen) $x_1, \ldots, x_m \in I$, Vielfachheiten $r_1, \ldots, r_m \in \mathbb{N}$ mit $\sum_{i=1}^{m} r_i = n$ und Werte $y_{ij} \in \mathbb{R}$, $i = 1, \ldots, m$, $j = 0, \ldots, r_i - 1$, gegeben.

Das Interpolationsproblem lautet:

Interpolationsproblem

Gegeben sei eine auf einem Intervall I erklärte Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

die von n Parametern $a_1, \ldots, a_n$ abhängt.

Zusätzlich seien paarweise verschiedene Knoten (oder Stützstellen) $x_1, \ldots, x_m \in I$, Vielfachheiten $r_1, \ldots, r_m \in \mathbb{N}$ mit $\sum_{i=1}^{m} r_i = n$ und Werte $y_{ij} \in \mathbb{R}$, $i = 1, \ldots, m$, $j = 0, \ldots, r_i - 1$, gegeben.

Das Interpolationsproblem lautet:

Bestimme die Parameter $a_1, \ldots, a_n$ so, dass die Interpolationsbedingungen

$$\Phi^{(j)}\left(x_i; a_1, \ldots, a_n\right) = y_{ij}, \quad i = 1, \ldots, m, \quad j = 0, \ldots, r_i - 1,$$

erfüllt sind.

Interpolationsproblem

Ist Φ linear in den Parametern a_i, d.h.

$$\Phi(x; a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i \phi_i(x),$$

so heißt das Interpolationsproblem linear.

Interpolationsproblem

Ist Φ linear in den Parametern a_i, d.h.

$$\Phi(x; a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i \phi_i(x),$$

so heißt das Interpolationsproblem linear.

Gilt $r_j = 1$ für alle j, so spricht man von Lagrange-Interpolation, anderenfalls von Hermite-Interpolation.

Interpolationsproblem

Ist Φ linear in den Parametern a_i, d.h.

$$\Phi(x; a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i \phi_i(x),$$

so heißt das Interpolationsproblem linear.

Gilt $r_j = 1$ für alle j, so spricht man von Lagrange-Interpolation, anderenfalls von Hermite-Interpolation.

Bemerkung 2.1: Bei der Lagrange-Interpolation werden nur Funktionswerte, aber keine Ableitungen vorgeschrieben. Man beachte, dass bei der Hermite Interpolation alle Ableitungen der Ordnung $0, \ldots, r_i - 1$ vorgeschrieben werden. Lässt man Lücken bei den vorgeschriebenen Ableitungen zu, so spricht man von einem Hermite-Birkhoff-Interpolationsproblem. □

Interpolationsproblem

Beispiel 2.2: Prominenten Instanzen des Interpolationsproblemes:

1. Polynominterpolation:

$$\Phi(x; a_0, \ldots, a_n) = \sum_{j=0}^{n} a_j x^j$$

2. trigonometrische Interpolation:

$$\Phi(x; a_0, \ldots, a_{2n}) = a_0 + \sum_{j=1}^{n} \left(a_{2j-1} \sin(jx) + a_{2j} \cos(jx)\right)$$

Interpolationsproblem

3. rationale Interpolation:

$$\Phi\left(x; a_0, \ldots, a_n, b_0, \ldots b_p\right) = \frac{\sum_{i=0}^{n} a_i x^i}{\sum_{j=0}^{p} b_j x^j}.$$

4. Spline-Interpolation:

$$\Phi\left(x; a_1, \ldots, a_n\right) = \sum_{i=1}^{n} a_i \phi_i(x),$$

wobei die ϕ_i auf Teilintervallen von I mit Polynomen übereinstimmen.

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Man benötigt die Interpolation zur:

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Man benötigt die Interpolation zur:

1. Bestimmung von Zwischenwerten aus Funktionstafeln,

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Man benötigt die Interpolation zur:

1. Bestimmung von Zwischenwerten aus Funktionstafeln,
2. Herleitung von Formeln zur numerischen Integration,

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Man benötigt die Interpolation zur:

1. Bestimmung von Zwischenwerten aus Funktionstafeln,
2. Herleitung von Formeln zur numerischen Integration,
3. Konvergenzbeschleunigung durch Extrapolation,

Interpolationsproblem

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in den Büchern von Braess, Stoer und Schwarz behandelt.

Man benötigt die Interpolation zur:

1. Bestimmung von Zwischenwerten aus Funktionstafeln,
2. Herleitung von Formeln zur numerischen Integration,
3. Konvergenzbeschleunigung durch Extrapolation,
4. Numerischen Behandlung gewöhnlicher Differentialgleichungen.

Polynominterpolation

Wir betrachten in diesem Abschnitt die Interpolation mit Polynomen. Dabei behandeln wir vor allem das Lagrangesche Interpolationsproblem:

Polynominterpolation

Wir betrachten in diesem Abschnitt die Interpolation mit Polynomen. Dabei behandeln wir vor allem das Lagrangesche Interpolationsproblem:

Gegeben seien $n+1$ *verschiedene Knoten* $x_j \in \mathbb{R}$, $j = 0, \ldots, n$, *und* $n+1$ *nicht notwendig verschiedene Werte* $y_j \in \mathbb{R}$.

Gesucht ist ein Polynom p *vom Höchstgrade* n, *so dass gilt*

$$p(x_j) = y_j \quad \text{für} \quad j = 0, \ldots, n.$$

Polynominterpolation

Wir betrachten in diesem Abschnitt die Interpolation mit Polynomen. Dabei behandeln wir vor allem das Lagrangesche Interpolationsproblem:

Gegeben seien $n+1$ *verschiedene Knoten* $x_j \in \mathbb{R}$, $j = 0, \ldots, n$, *und* $n+1$ *nicht notwendig verschiedene Werte* $y_j \in \mathbb{R}$.

Gesucht ist ein Polynom p *vom Höchstgrade* n, *so dass gilt*

$$p(x_j) = y_j \quad \textit{für} \quad j = 0, \ldots, n.$$

Die Lagrangsche Interpolation hat viel mit einem linearen Gleichungssystem gemein: Es werden ebensoviele Gleichungen wie Parameter vorgegeben und Polynome bilden wie Tupel einen Vektorraum.

Polynominterpolation

Bemerkung 2.3: Die Beweise werden zeigen, dass die Existenz- und Eindeutigkeitsresultate und die Algorithmen zur Berechnung des Interpolationspolynomes ohne Änderungen für *komplexe* Knoten x_j und *komplexe* Daten y_j richtig bleiben. Nur die Fehlerabschätzungen beziehen sich auschließlich auf *reelle* Probleme. □

Polynominterpolation

Bemerkung 2.3: Die Beweise werden zeigen, dass die Existenz- und Eindeutigkeitsresultate und die Algorithmen zur Berechnung des Interpolationspolynomes ohne Änderungen für *komplexe* Knoten x_j und *komplexe* Daten y_j richtig bleiben. Nur die Fehlerabschätzungen beziehen sich auschließlich auf *reelle* Probleme. □

Wir bezeichnen mit Π_n den Vektorraum der Polynome vom Höchstgrad n (entweder mit reellen oder komplexen Koeffizienten).

Lagrangesche Interpolationsformel

Satz 2.4 (Existenz und Eindeutigkeit)

Zu beliebigen $n+1$ Daten $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, mit $x_j \neq x_k$ für $j \neq k$ gibt es genau ein Polynom $p \in \Pi_n$ mit

$$p(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.1}$$

Lagrangesche Interpolationsformel

Satz 2.4 (Existenz und Eindeutigkeit)

Zu beliebigen $n+1$ Daten $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, mit $x_j \neq x_k$ für $j \neq k$ gibt es genau ein Polynom $p \in \Pi_n$ mit

$$p(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.1}$$

Beweis der Eindeutigkeit: Angenommen es gibt zwei Polynome $p_1, p_2 \in \Pi_n$ mit $p_k(x_j) = y_j$, $j = 0, \ldots, n$, $k = 1, 2$.

Lagrangesche Interpolationsformel

Satz 2.4 (Existenz und Eindeutigkeit)

Zu beliebigen $n+1$ Daten $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, mit $x_j \neq x_k$ für $j \neq k$ gibt es genau ein Polynom $p \in \Pi_n$ mit

$$p(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.1}$$

Beweis der Eindeutigkeit: Angenommen es gibt zwei Polynome $p_1, p_2 \in \Pi_n$ mit $p_k(x_j) = y_j$, $j = 0, \ldots, n$, $k = 1, 2$.

Dann besitzt das Polynom $p := p_1 - p_2 \in \Pi_n$ die $n+1$ Nullstellen $x_0, \ldots, x_n$, und daher folgt aus dem Fundamentalsatz der Algebra $p \equiv 0$.

Lagrangesche Interpolationsformel

Existenz: Die Existenz zeigen wir konstruktiv. Es sei

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \ , \quad j = 0, \ldots, n. \tag{2.2}$$

Lagrangesche Interpolationsformel

Existenz: Die Existenz zeigen wir konstruktiv. Es sei

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \Bigg/ \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \ , \quad j = 0, \ldots, n. \tag{2.2}$$

Dann gilt $\ell_j \in \Pi_n$ und $\ell_j(x_k) = \delta_{jk}$ für $j, k = 0, \ldots, n$, und daher erfüllt

$$p(x) := \sum_{j=0}^{n} y_j \ell_j(x) \in \Pi_n \tag{2.3}$$

die Interpolationsbedingungen. ■

Lagrangesche Interpolationsformel

Existenz: Die Existenz zeigen wir konstruktiv. Es sei

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \,, \quad j = 0, \ldots, n. \tag{2.2}$$

Dann gilt $\ell_j \in \Pi_n$ und $\ell_j(x_k) = \delta_{jk}$ für $j, k = 0, \ldots, n$, und daher erfüllt

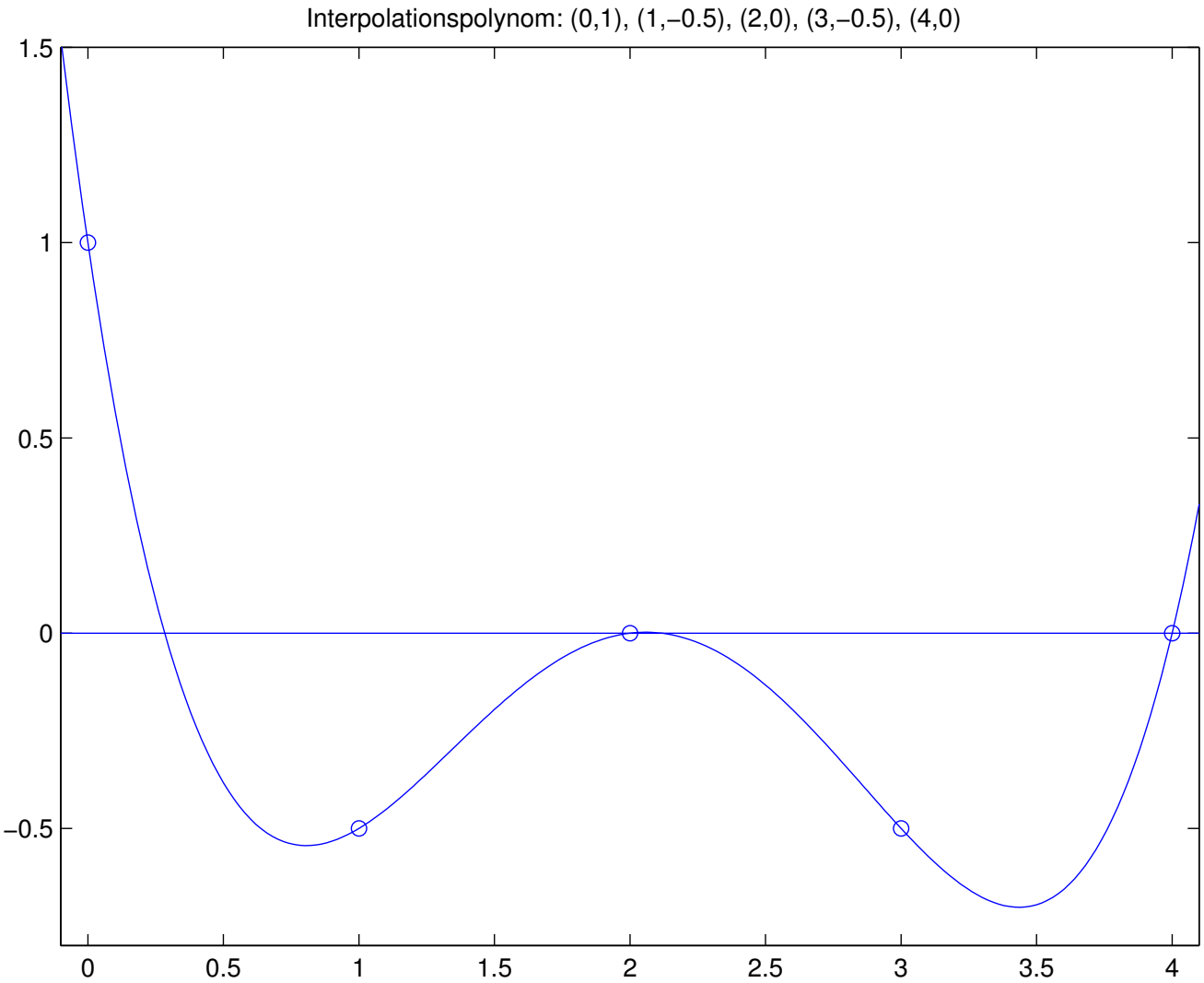
$$p(x) := \sum_{j=0}^{n} y_j \ell_j(x) \in \Pi_n \tag{2.3}$$

die Interpolationsbedingungen. ■

Definition 2.5: Die Darstellung (2.2), (2.3) des Interpolationspolynomes heißt die Lagrangesche Interpolationsformel.

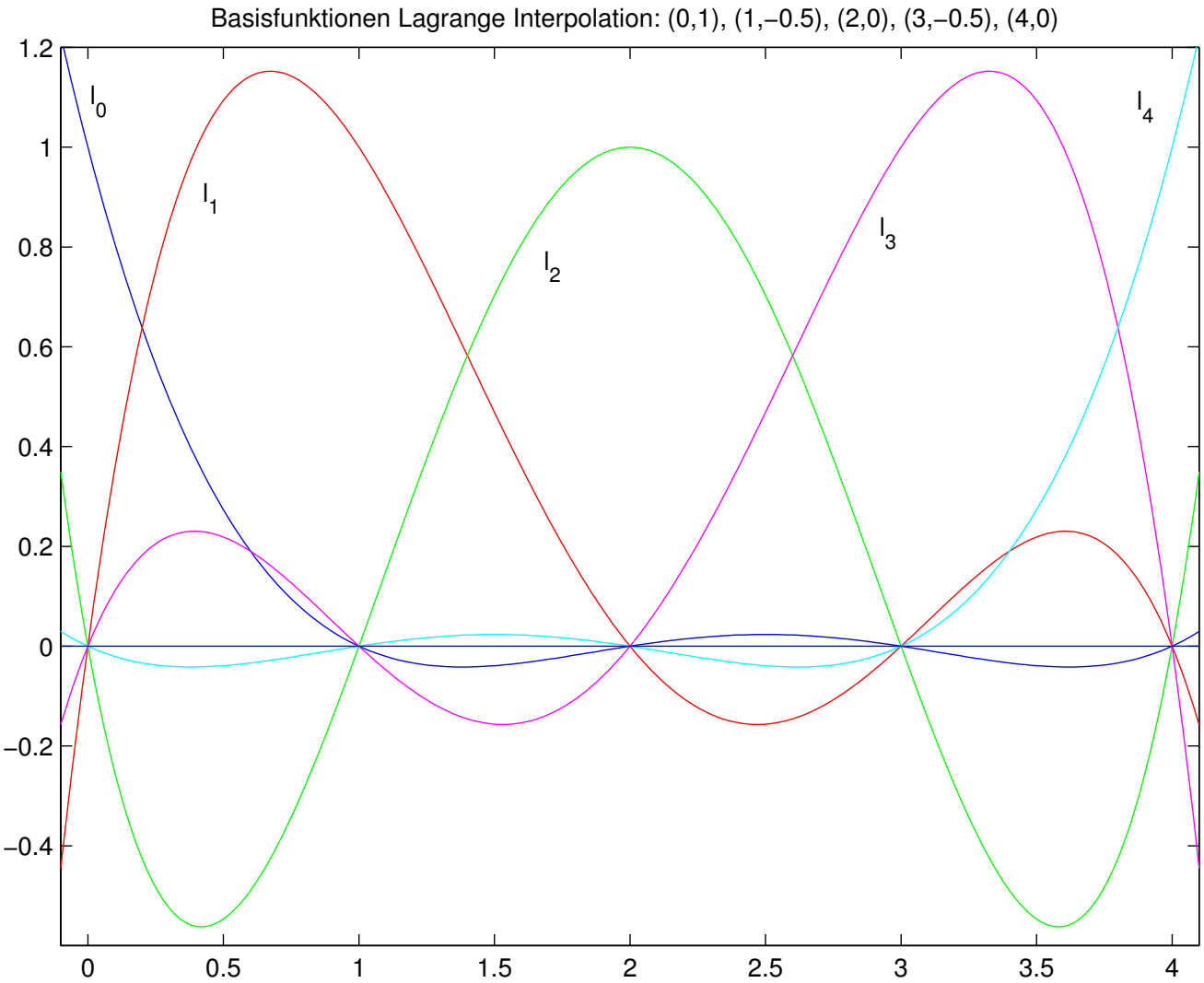
Lagrangesche Interpolationsformel

Beispiel, Teil 1



Lagrangesche Interpolationsformel

Beispiel, Teil 2



Lagrangesche Interpolationsformel

Bemerkung 2.6: Prinzipiell kann man bei dem Ansatz

$$p(x) = \sum_{j=0}^{n} a_j x^j$$

Lagrangesche Interpolationsformel

Bemerkung 2.6: Prinzipiell kann man bei dem Ansatz

$$p(x) = \sum_{j=0}^{n} a_j x^j$$

die Koeffizienten a_j aus dem linearen Gleichungssystem

$$\sum_{j=0}^{n} a_j x_k^j = y_k, \quad k = 0, \ldots, n \tag{2.4}$$

berechnen.

Lagrangesche Interpolationsformel

Für Gleichungssysteme, deren Koeffizientenmatrix

$$(x_k^j)_{j,k=0,\ldots,n}$$

eine Vandermonde-Matrix ist, gibt es spezielle Algorithmen (vgl. Abschnitt 4.5.1 des Skriptes „Grundlagen der Numerischen Mathematik“ von Herrn Voß), die $2.5n^2$ flops (dabei steht flop für Fliesskommazahlenoperation) erfordern. Der Aufwand der folgenden Algorithmen ist geringer. □

Lagrangesche Interpolationsformel

Für Gleichungssysteme, deren Koeffizientenmatrix

$$(x_k^j)_{j,k=0,\ldots,n}$$

eine Vandermonde-Matrix ist, gibt es spezielle Algorithmen (vgl. Abschnitt 4.5.1 des Skriptes „Grundlagen der Numerischen Mathematik“ von Herrn Voß), die $2.5n^2$ flops (dabei steht flop für Fliesskommazahlenoperation) erfordern. Der Aufwand der folgenden Algorithmen ist geringer. □

Bemerkung 2.7: MATLAB stellt die Funktion

```
p=polyfit(x,y,n)
```

zur Verfügung, durch die die Koeffizienten p des Ausgleichspolynomes vom Grad n zu den Daten (x, y) bestimmt werden. Ist $\dim x = n + 1$, so erhält man das Interpolationspolynom. Mit `y=polyval(p,x)` kann man das Polynom dann an der Stelle x (oder den Stellen $x(j)$, falls x ein Vektor ist) auswerten. □

Lagrangesche Interpolationsformel

Bemerkung 2.8: Verwendet man die Lagrangesche Interpolationsformel (2.3) naiv, so benötigt man zur Auswertung von p an einer festen Stelle $\hat{x}$ zur Bereitstellung der $\ell_j(\hat{x})$ aus (2.2) jeweils $4n$ flops

Lagrangesche Interpolationsformel

Bemerkung 2.8: Verwendet man die Lagrangesche Interpolationsformel (2.3) naiv, so benötigt man zur Auswertung von p an einer festen Stelle $\hat{x}$ zur Bereitstellung der $\ell_j(\hat{x})$ aus (2.2) jeweils $4n$ flops

(berechne $(\hat{x} - x_i)/(x_j - x_i)$ für $i = 0, \ldots, n$, $i \neq j$, und das Produkt dieser Quotienten)

Lagrangesche Interpolationsformel

Bemerkung 2.8: Verwendet man die Lagrangesche Interpolationsformel (2.3) naiv, so benötigt man zur Auswertung von p an einer festen Stelle $\hat{x}$ zur Bereitstellung der $\ell_j(\hat{x})$ aus (2.2) jeweils $4n$ flops

(berechne $(\hat{x} - x_i)/(x_j - x_i)$ für $i = 0, \ldots, n$, $i \neq j$, und das Produkt dieser Quotienten)

und zur Auswertung von (2.3) weitere $2n$ flops, zusammen also $4n^2 + O(n)$ flops.

Lagrangesche Interpolationsformel

Bemerkung 2.8: Verwendet man die Lagrangesche Interpolationsformel (2.3) naiv, so benötigt man zur Auswertung von p an einer festen Stelle $\hat{x}$ zur Bereitstellung der $\ell_j(\hat{x})$ aus (2.2) jeweils $4n$ flops

(berechne $(\hat{x} - x_i)/(x_j - x_i)$ für $i = 0, \ldots, n$, $i \neq j$, und das Produkt dieser Quotienten)

und zur Auswertung von (2.3) weitere $2n$ flops, zusammen also $4n^2 + O(n)$ flops.

Dieses kann (vgl. Abschnitt 2.2.1 des Skriptes „Grundlagen der Numerischen Mathematik“von Herrn Voß) reduziert werden auf $1.5n^2 + O(n)$ flops. □

Newtonsche Interpolationsformel

Wir werden nun eine Form des Interpolationspolynomes herleiten, bei der $1.5n(n+1)$ flops zur Vorbereitung benötigt werden und die Auswertung an einer festen Stelle zusätzlich $3n$ flops erfordert. Wir behandeln dazu zunächst die folgende Frage:

Newtonsche Interpolationsformel

Wir werden nun eine Form des Interpolationspolynomes herleiten, bei der $1.5n(n+1)$ flops zur Vorbereitung benötigt werden und die Auswertung an einer festen Stelle zusätzlich $3n$ flops erfordert. Wir behandeln dazu zunächst die folgende Frage:

Das Polynom $p_n(x) \in \Pi_n$ interpoliere die Daten $(x_j, y_j) \in \mathbb{R}^2, j = 0, \ldots, n$.
Wir nehmen ein weiteres Zahlenpaar $(x_{n+1}, y_{n+1}) \in \mathbb{R}^2$, $x_{n+1} \neq x_j$, hinzu.

Newtonsche Interpolationsformel

Wir werden nun eine Form des Interpolationspolynomes herleiten, bei der $1.5n(n+1)$ flops zur Vorbereitung benötigt werden und die Auswertung an einer festen Stelle zusätzlich $3n$ flops erfordert. Wir behandeln dazu zunächst die folgende Frage:

Das Polynom $p_n(x) \in \Pi_n$ interpoliere die Daten $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$.
Wir nehmen ein weiteres Zahlenpaar $(x_{n+1}, y_{n+1}) \in \mathbb{R}^2$, $x_{n+1} \neq x_j$, hinzu.

Kann man dann das Interpolationspolynom $p_{n+1}(x) \in \Pi_{n+1}$ zu den Daten (x_j, y_j), wobei $j = 0, \ldots, n, n+1$, schreiben als

$$p_{n+1}(x) = p_n(x) + f(x),$$

mit einer leicht berechenbaren Funktion $f(x)$?

Newtonsche Interpolationsformel

Wegen $p_n(x) \in \Pi_n$, $p_{n+1}(x) \in \Pi_{n+1}$ gilt $f(x) = p_{n+1}(x) - p_n(x) \in \Pi_{n+1}$,

Newtonsche Interpolationsformel

Wegen $p_n(x) \in \Pi_n$, $p_{n+1}(x) \in \Pi_{n+1}$ gilt $f(x) = p_{n+1}(x) - p_n(x) \in \Pi_{n+1}$, und wegen

$$y_j = p_{n+1}(x_j) = p_n(x_j) + f(x_j) = y_j + f(x_j), \quad j = 0, \ldots, n,$$

gilt $f(x_j) = 0, j = 0, \ldots, n$.

Newtonsche Interpolationsformel

Wegen $p_n(x) \in \Pi_n$, $p_{n+1}(x) \in \Pi_{n+1}$ gilt $f(x) = p_{n+1}(x) - p_n(x) \in \Pi_{n+1}$, und wegen

$$y_j = p_{n+1}(x_j) = p_n(x_j) + f(x_j) = y_j + f(x_j), \quad j = 0, \ldots, n,$$

gilt $f(x_j) = 0, j = 0, \ldots, n$. Daher hat $f(x)$ mit einem $a \in \mathbb{R}$ die Gestalt

$$f(x) = a \prod_{j=0}^{n} (x - x_j).$$

Newtonsche Interpolationsformel

Wegen $p_n(x) \in \Pi_n$, $p_{n+1}(x) \in \Pi_{n+1}$ gilt $f(x) = p_{n+1}(x) - p_n(x) \in \Pi_{n+1}$, und wegen

$$y_j = p_{n+1}(x_j) = p_n(x_j) + f(x_j) = y_j + f(x_j), \quad j = 0, \ldots, n,$$

gilt $f(x_j) = 0, j = 0, \ldots, n$. Daher hat $f(x)$ mit einem $a \in \mathbb{R}$ die Gestalt

$$f(x) = a \prod_{j=0}^{n} (x - x_j).$$

a kann man aus der Interpolationsbedingung

$$y_{n+1} = p_{n+1}(x_{n+1}) = p_n(x_{n+1}) + a \prod_{j=0}^{n} (x_{n+1} - x_j)$$

ermitteln:

$$a = \frac{y_{n+1} - p_n(x_{n+1})}{(x_{n+1} - x_0) \cdot \ldots \cdot (x_{n+1} - x_n)}.$$

Newtonsche Interpolationsformel

Wir wollen nun ein Verfahren zur Berechnung der Zahl a, des führenden Koeffizienten des Interpolationspolynomes, herleiten. Grundlage dafür ist

Newtonsche Interpolationsformel

Wir wollen nun ein Verfahren zur Berechnung der Zahl a, des führenden Koeffizienten des Interpolationspolynomes, herleiten. Grundlage dafür ist

Satz 2.9 (Aitken Lemma)

Es sei zu $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, $x_i \neq x_j$ für $i \neq j$, das Interpolationspolynom gesucht.

Newtonsche Interpolationsformel

Wir wollen nun ein Verfahren zur Berechnung der Zahl a, des führenden Koeffizienten des Interpolationspolynomes, herleiten. Grundlage dafür ist

Satz 2.9 (Aitken Lemma)

Es sei zu $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, $x_i \neq x_j$ für $i \neq j$, das Interpolationspolynom gesucht.

Sei $p_{[0]} \in \Pi_{n-1}$ durch die Interpolationsbedingungen $p_{[0]}(x_j) = y_j$, $j = 0, \ldots, n-1$, festgelegt, und sei $p_{[n]} \in \Pi_{n-1}$ definiert durch $p_{[n]}(x_j) = y_j$, $j = 1, \ldots, n$.

Newtonsche Interpolationsformel

Wir wollen nun ein Verfahren zur Berechnung der Zahl a, des führenden Koeffizienten des Interpolationspolynomes, herleiten. Grundlage dafür ist

Satz 2.9 (Aitken Lemma)

Es sei zu $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, $x_i \neq x_j$ für $i \neq j$, das Interpolationspolynom gesucht.

Sei $p_{[0]} \in \Pi_{n-1}$ durch die Interpolationsbedingungen $p_{[0]}(x_j) = y_j$, $j = 0, \ldots, n-1$, festgelegt, und sei $p_{[n]} \in \Pi_{n-1}$ definiert durch $p_{[n]}(x_j) = y_j$, $j = 1, \ldots, n$.

Dann gilt

$$p(x) = \frac{p_{[0]}(x)(x - x_n) - p_{[n]}(x)(x - x_0)}{x_0 - x_n}.$$

Newtonsche Interpolationsformel

Beweis.

Das angegebene Polynom erfüllt offenbar die Interpolationsbedingungen $p(x_j) = y_j, j = 0, \ldots, n$. □

Newtonsche Interpolationsformel

Beweis.

Das angegebene Polynom erfüllt offenbar die Interpolationsbedingungen $p(x_j) = y_j, j = 0, \ldots, n$. □

Für $0 \leqslant i \leqslant j \leqslant n$ sei nun $p_{ij} \in \Pi_{j-i}$ das Interpolationspolynom, das $p_{ij}(x_k) = y_k$, $k = i, \ldots, j$, erfüllt. Dann folgt aus dem Aitken Lemma, dass die p_{ij} rekursiv berechnet werden können durch

$$p_{ij}(x) = \frac{p_{i,j-1}(x)(x - x_j) - p_{i+1,j}(x)(x - x_i)}{x_i - x_j} \tag{2.5}$$

$$= p_{i+1,j}(x) + (p_{i+1,j}(x) - p_{i,j-1}(x))\frac{x - x_j}{x_j - x_i}$$

Newtonsche Interpolationsformel

Beweis.

Das angegebene Polynom erfüllt offenbar die Interpolationsbedingungen $p(x_j) = y_j, j = 0, \ldots, n$. □

Für $0 \leqslant i \leqslant j \leqslant n$ sei nun $p_{ij} \in \Pi_{j-i}$ das Interpolationspolynom, das $p_{ij}(x_k) = y_k$, $k = i, \ldots, j$, erfüllt. Dann folgt aus dem Aitken Lemma, dass die p_{ij} rekursiv berechnet werden können durch

$$p_{ij}(x) = \frac{p_{i,j-1}(x)(x - x_j) - p_{i+1,j}(x)(x - x_i)}{x_i - x_j} \tag{2.5}$$

$$= p_{i+1,j}(x) + (p_{i+1,j}(x) - p_{i,j-1}(x))\frac{x - x_j}{x_j - x_i}$$

Diese Rekursionsformel kann verwendet werden, um den Wert des Interpolationspolynomes an einer festen Stelle x (aber nicht das Polynom selbst) zu berechnen:

Newtonsche Interpolationsformel

Algorithmus 2.10: (Algorithmus von Neville und Aitken)

```
for j = 0 : n
   t(j) = y(j);
   xj = x - x(j);
   if j > 0
      for i = j-1 : -1 : 0
         t(i) = t(i+1) + (t(i+1) - t(i))*xj /
                                  (x(j) - x(i));
      end
   end
end
p = t(0);
```

Newtonsche Interpolationsformel

Bemerkung 2.11: Mit dem Algorithmus von Neville und Aitken wird das linke Tableau aufgestellt, das die Werte P_{ij} der interpolierenden Polynome p_{ij} an der festen Stelle x enthält. Das rechte Tableau enthält die Reihenfolge, in der die P_{ij} berechnet werden.

$$
\begin{array}{lllll}
x_0 & y_0 = P_{00} & & & \\
x_1 & y_1 = P_{11} & P_{01} & & \\
x_2 & y_2 = P_{22} & P_{12} & P_{02} & \\
x_3 & y_3 = P_{33} & P_{23} & P_{13} & P_{03} \\
x_4 & y_4 = P_{44} & P_{34} & P_{24} & P_{14}
\end{array}
\qquad P_{04} = p(x)
\qquad
\begin{array}{llll}
1 & & & \\
2 & 3 & & \\
4 & 5 & 6 & \\
7 & 8 & 9 & 10
\end{array}
\qquad \square
$$

Newtonsche Interpolationsformel

Bemerkung 2.12: Zur Auswertung des Interpolationspolynomes p an einer festen Stelle benötigt man mit dem Algorithmus von Neville und Aitken offenbar $\frac{5}{2}n^2 + O(n)$ flops. □

Newtonsche Interpolationsformel

Bemerkung 2.12: Zur Auswertung des Interpolationspolynomes p an einer festen Stelle benötigt man mit dem Algorithmus von Neville und Aitken offenbar $\frac{5}{2}n^2 + O(n)$ flops. □

Man erhält aus der Rekursionsformel (2.5) eine weitere Darstellung des Interpolationspolynomes, die Newtonsche Interpolationsformel.

Newtonsche Interpolationsformel

Bemerkung 2.12: Zur Auswertung des Interpolationspolynomes p an einer festen Stelle benötigt man mit dem Algorithmus von Neville und Aitken offenbar $\frac{5}{2}n^2 + O(n)$ flops. □

Man erhält aus der Rekursionsformel (2.5) eine weitere Darstellung des Interpolationspolynomes, die Newtonsche Interpolationsformel.

Da a in

$$p_n(x) = p_{n-1}(x) + a \prod_{j=0}^{n-1} (x - x_j)$$

der Koeffizient bei der höchsten Potenz x^n in $p_n(x)$ ist, liest man aus (2.5) sofort ab:

Newtonsche Interpolationsformel

Satz 2.13 (Newtonsche Interpolationsformel)

Das Interpolationspolynom $p \in \Pi_n$ aus (2.1) hat die Gestalt

$$p(x) = \sum_{j=0}^{n} [x_0, \ldots, x_j] \prod_{k=0}^{j-1} (x - x_k), \tag{2.6}$$

wobei die dividierten Differenzen $[x_0, \ldots, x_j]$ rekursiv definiert sind durch

$$\begin{aligned} [x_j] &:= y_j, \\ [x_k, \ldots, x_j] &:= \frac{[x_{k+1}, \ldots, x_j] - [x_k, \ldots, x_{j-1}]}{x_j - x_k}, \quad j > k \geqslant 0. \end{aligned} \tag{2.7}$$

Newtonsche Interpolationsformel

Die dividierten Differenzen $c_j := [x_0, \ldots, x_j]$ kann man mit folgendem Algorithmus aus den Wertepaaren (x_j, y_j) berechnen:

Algorithmus 2.14: (Dividierte Differenzen)

```
for k = 0 : n
   t(k) = y(k);
   if k > 0
      for i = k-1 : -1 : 0
         t(i) = (t(i+1) - t(i)) / (x(k) - x(i));
      end
   end
   c(k) = t(0);
end
```

Newtonsche Interpolationsformel

Danach kann man das Interpolationspolynom an jeder gewünschten Stelle $x_0 = \texttt{x0}$ mit einem Horner-ähnlichen Schema auswerten:

Algorithmus 2.15:

```
p = c(n);
for i = n-1 : -1 : 0
  p = p * (x0 - x(i)) + c(i);
end
```

Newtonsche Interpolationsformel

Bemerkung 2.16: Die $t(k)$ in dem Algorithmus enthalten die folgenden dividierten Differenzen, die in derselben Reihenfolge wie im Algorithmus von Neville und Aitken berechnet werden:

$$\begin{aligned} t(0) &= y_0 \\ t(1) &= y_1 \\ t(2) &= y_2 \\ t(3) &= y_3 \end{aligned} \qquad \begin{aligned} t(0) &= [x_0, x_1] \\ t(1) &= [x_1, x_2] \\ t(2) &= [x_2, x_3] \end{aligned} \qquad \begin{aligned} t(0) &= [x_0, x_1, x_2] \\ t(1) &= [x_1, x_2, x_3] \end{aligned} \qquad t(0) = [x_0, x_1, x_2, x_3]$$

□

Newtonsche Interpolationsformel

Bemerkung 2.17: Man benötigt $\frac{3}{2}n\left(n+1\right)$ flops zur Berechnung aller c_j, zur Auswertung von p an einer festen Stelle $\hat{x}$ zusätzlich $3n$ flops.

Newtonsche Interpolationsformel

Bemerkung 2.17: Man benötigt $\frac{3}{2}n\,(n+1)$ flops zur Berechnung aller c_j, zur Auswertung von p an einer festen Stelle $\hat{x}$ zusätzlich $3n$ flops.

Die Newtonsche Interpolationsformel liefert also die effizienteste Methode zur Auswertung des Interpolationspolynomes.

Newtonsche Interpolationsformel

Bemerkung 2.17: Man benötigt $\frac{3}{2}n\,(n+1)$ flops zur Berechnung aller c_j, zur Auswertung von p an einer festen Stelle $\hat{x}$ zusätzlich $3n$ flops.

Die Newtonsche Interpolationsformel liefert also die effizienteste Methode zur Auswertung des Interpolationspolynomes.

Selbst wenn man nur an einem Funktionswert interessiert ist, ist der Aufwand geringer als mit dem Algorithmus von Neville und Aitken.

Newtonsche Interpolationsformel

Bemerkung 2.17: Man benötigt $\frac{3}{2}n\,(n+1)$ flops zur Berechnung aller c_j, zur Auswertung von p an einer festen Stelle $\hat{x}$ zusätzlich $3n$ flops.

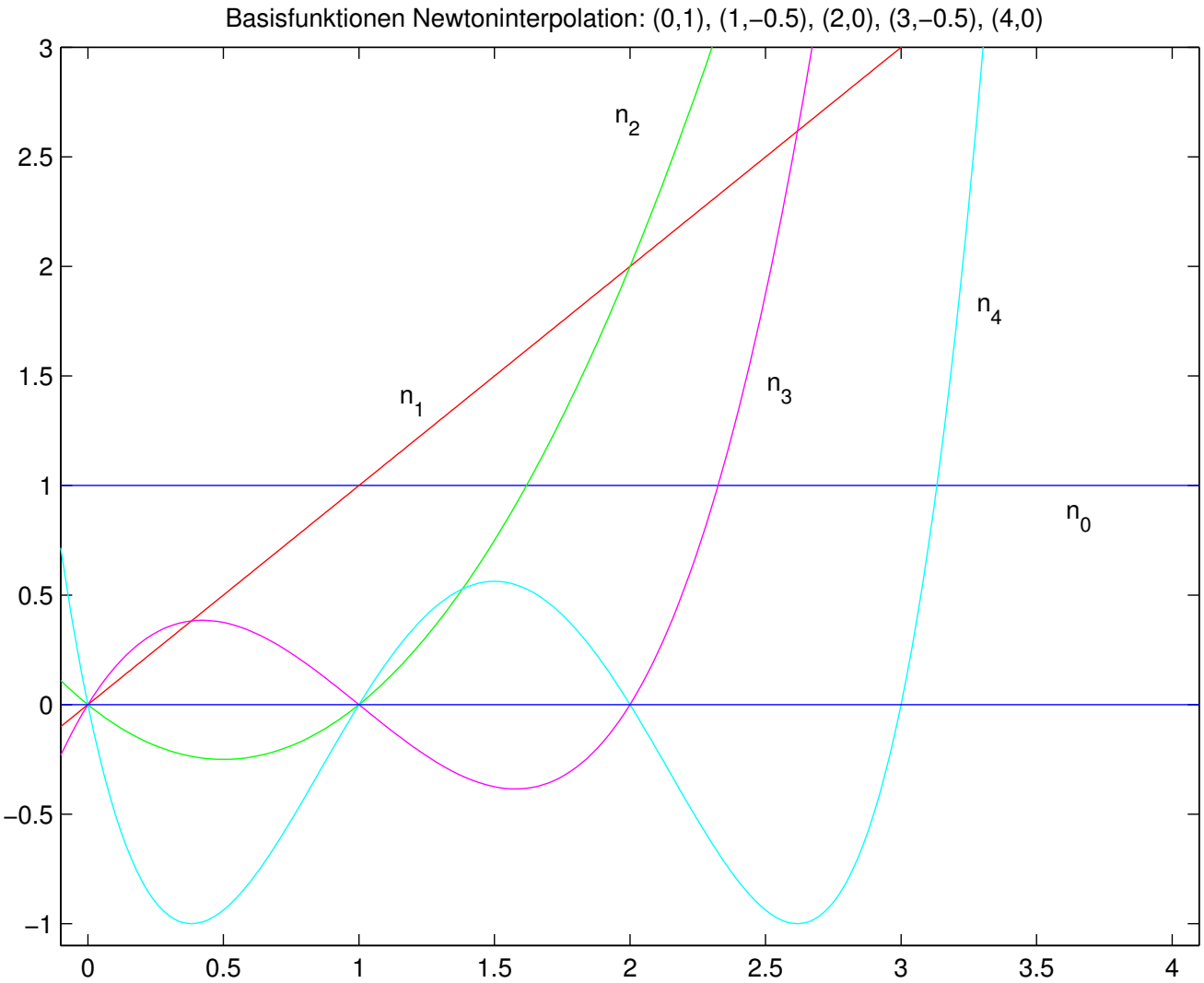
Die Newtonsche Interpolationsformel liefert also die effizienteste Methode zur Auswertung des Interpolationspolynomes.

Selbst wenn man nur an einem Funktionswert interessiert ist, ist der Aufwand geringer als mit dem Algorithmus von Neville und Aitken.

Wegen seiner einfachen Gestalt wird der Algorithmus von Neville und Aitken dennoch in der Praxis verwendet. □

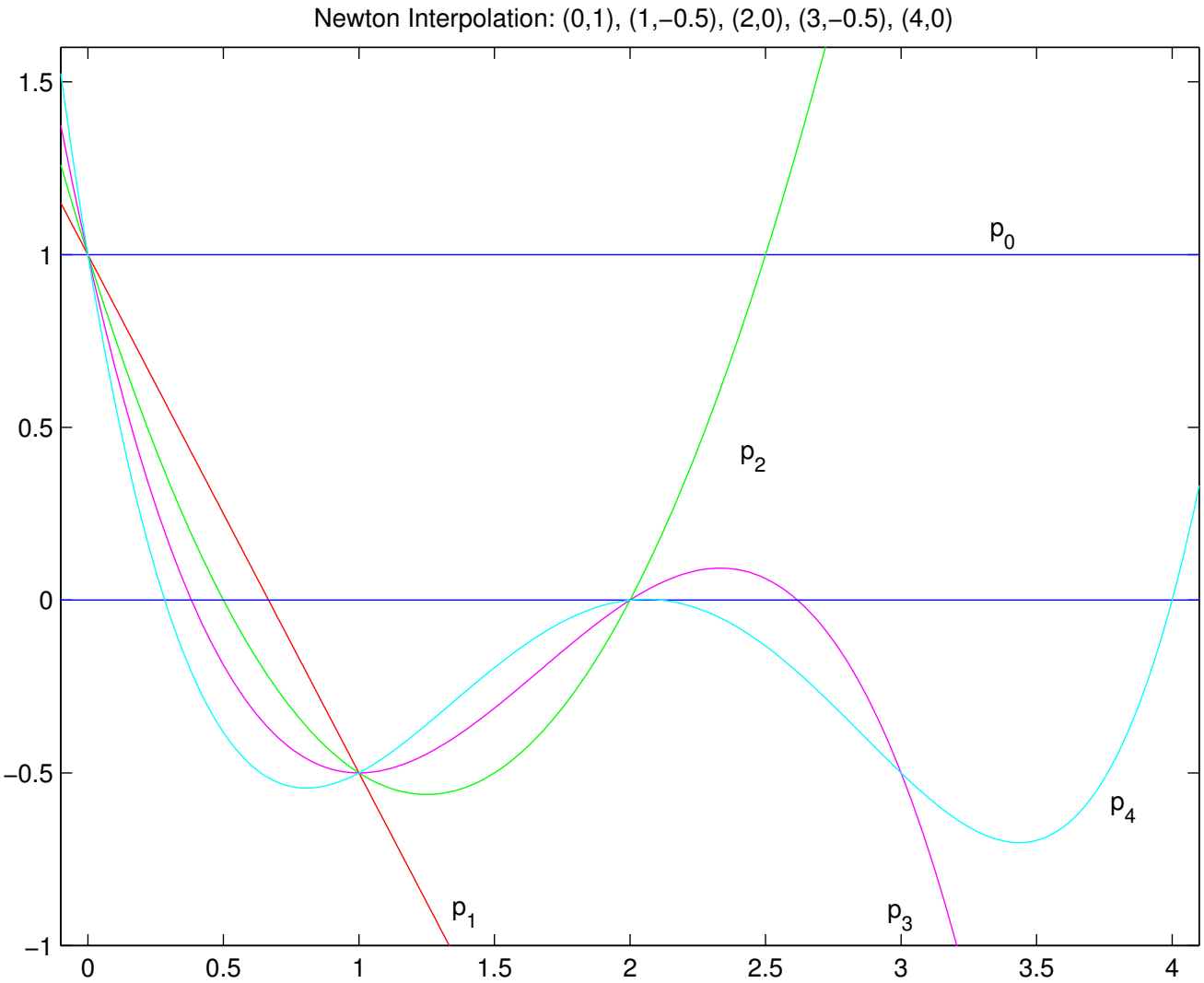
Newtonsche Interpolationsformel

Beispiel, Teil 1



Newtonsche Interpolationsformel

Beispiel, Teil 2



Newtonsche Interpolationsformel

Bemerkung 2.18: Natürlich erhält man für jede Anordnung der Knoten x_i (bei rundungsfehlerfreier Rechnung) dasselbe Interpolationspolynom $p_n(x)$. Will man $p_n(x)$ nur an einer Stelle x (oder in deren Nähe) auswerten, so kann man den Rundungsfehlereinfluss klein halten, indem man die Knoten so numeriert, dass gilt

$$|x - x_i| \leqslant |x - x_{i+1}|, \qquad i = 0, \ldots, n-1.$$ □

Fehlerbetrachtungen

Wir behandeln nun die Frage, wie gut eine gegebene, auf einem reellen Intervall definierte Funktion f durch das Interpolationspolynom zu vorgegebenen Knoten x_i in I approximiert wird (es sei also $y_i = f(x_i)$).

Fehlerbetrachtungen

Wir behandeln nun die Frage, wie gut eine gegebene, auf einem reellen Intervall definierte Funktion f durch das Interpolationspolynom zu vorgegebenen Knoten x_i in I approximiert wird (es sei also $y_i = f(x_i)$).

Beispiel 2.19: Gegeben sei auf dem Intervall $I = [-1, 1]$ die Funktion

$$f(x) = \sin \pi x,$$

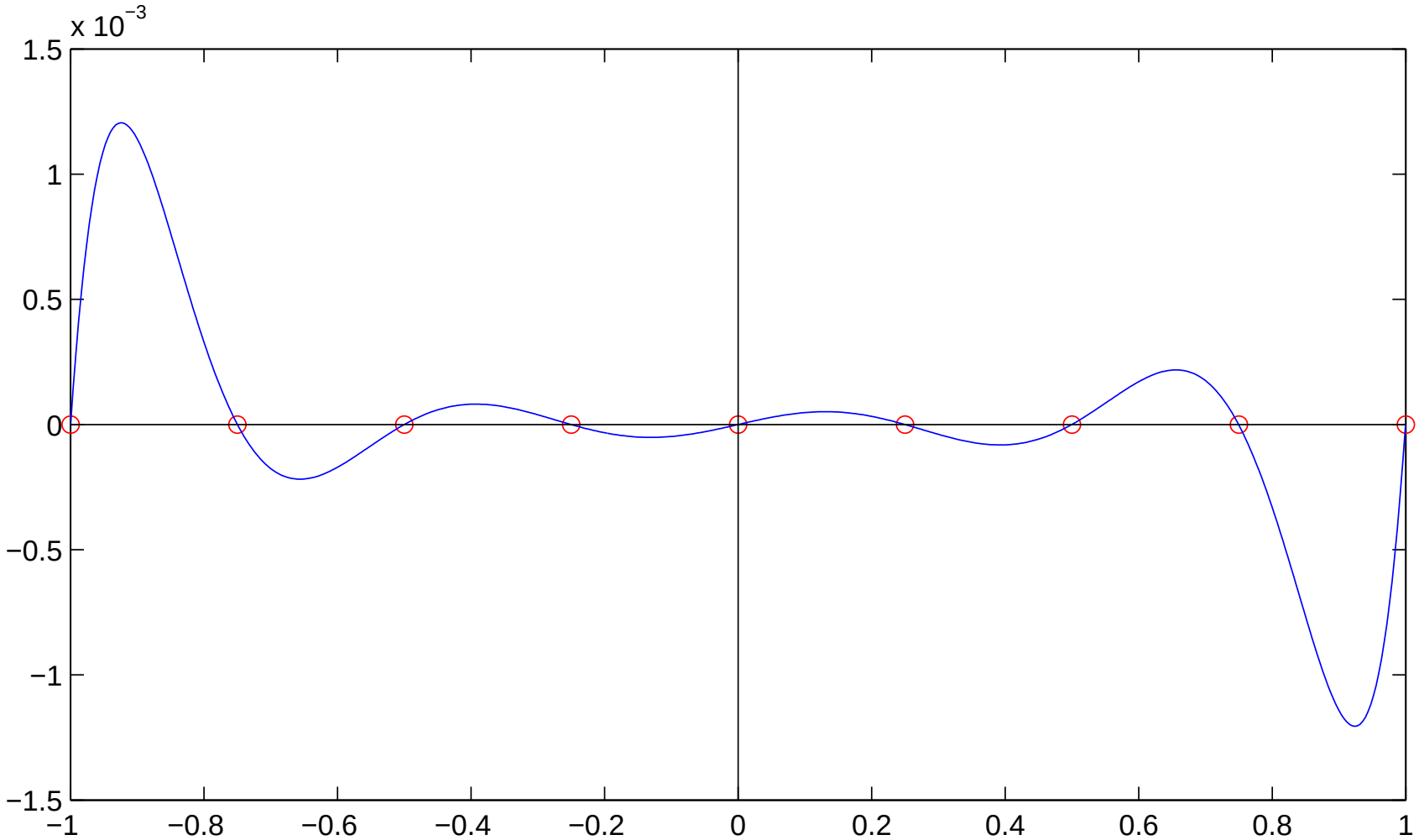
und $p_8(x) \in \Pi_8$ sei das Interpolationspolynom zu den Knoten

$$x_i = -1 + i \cdot 0.25, \quad i = 0, \ldots, 8.$$

Dann erhält man die Fehlerkurve der folgenden Abbildung.

Fehlerbetrachtungen

Fehlerkurve (äquidistante Knoten)



Fehlerbetrachtungen

Das Interpolationspolynom liefert also am Rande des Intervalls eine ziemlich schlechte Approximation für f, obwohl f sehr gutartig ist (beliebig oft differenzierbar). □

Fehlerbetrachtungen

Das Interpolationspolynom liefert also am Rande des Intervalls eine ziemlich schlechte Approximation für f, obwohl f sehr gutartig ist (beliebig oft differenzierbar). □

Das gezeichnete Verhalten ist typisch für die Polynominterpolation mit äquidistanten Knoten bei größerem n. Eine gewisse Erklärung hierfür gibt der folgende Satz.

Fehlerbetrachtungen

Satz 2.20

Sei $f \in C^{n+1}[a,b]$, seien $x_j \in [a,b]$, $j = 0, \ldots, n$, paarweise verschiedene Knoten, und sei $p \in \Pi_n$ das durch $p(x_j) = f(x_j)$, $j = 0, \ldots, n$, bestimmte Interpolationspolynom. Dann gilt:

Fehlerbetrachtungen

Satz 2.20

Sei $f \in C^{n+1}[a,b]$, seien $x_j \in [a,b]$, $j = 0, \ldots, n$, paarweise verschiedene Knoten, und sei $p \in \Pi_n$ das durch $p(x_j) = f(x_j)$, $j = 0, \ldots, n$, bestimmte Interpolationspolynom. Dann gilt:

Zu jedem $x \in [a,b]$ existiert $\xi = \xi(x)$ aus dem kleinsten Intervall $I(x, x_0, \ldots, x_n)$, das alle x_j und x enthält, so dass

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\omega(x) \tag{2.8}$$

gilt mit

$$\omega(x) = \prod_{i=0}^{n} (x - x_i). \tag{2.9}$$

Fehlerbetrachtungen

Satz 2.20

Sei $f \in C^{n+1}[a,b]$, seien $x_j \in [a,b]$, $j = 0, \ldots, n$, paarweise verschiedene Knoten, und sei $p \in \Pi_n$ das durch $p(x_j) = f(x_j)$, $j = 0, \ldots, n$, bestimmte Interpolationspolynom. Dann gilt:

Zu jedem $x \in [a,b]$ existiert $\xi = \xi(x)$ aus dem kleinsten Intervall $I(x, x_0, \ldots, x_n)$, das alle x_j und x enthält, so dass

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\omega(x) \tag{2.8}$$

gilt mit

$$\omega(x) = \prod_{i=0}^{n} (x - x_i). \tag{2.9}$$

Bemerkung 2.21: Das Knotenpolynom ω hat für äquidistante Knoten x_j die Gestalt von $f - p_8$ der Skizze aus der vorherigen Abbildung. □

Fehlerbetrachtungen

Beweis.

Für $x = x_j$, $j = 0, \ldots, n$, ist die Aussage trivial.

Fehlerbetrachtungen

Beweis.

Für $x = x_j$, $j = 0, \ldots, n$, ist die Aussage trivial.

Für festes $x \neq x_j$ betrachten wir die Funktion

$$F(z) := f(z) - p(z) - \alpha\omega(z) \tag{2.10}$$

und bestimmen $\alpha \in \mathbb{R}$ so, dass $F(x) = 0$ gilt.

Fehlerbetrachtungen

Beweis.

Für $x = x_j, j = 0, \ldots, n$, ist die Aussage trivial.

Für festes $x \neq x_j$ betrachten wir die Funktion

$$F(z) := f(z) - p(z) - \alpha\omega(z) \tag{2.10}$$

und bestimmen $\alpha \in \mathbb{R}$ so, dass $F(x) = 0$ gilt.

Dann besitzt F in $I(x, x_0, \ldots, x_n)$ wenigstens $n+2$ Nullstellen. Nach dem Satz von Rolle besitzt F' dort wenigstens $n+1$ Nullstellen, F'' wenigstens n Nullstellen, F''' wenigstens $n-1$ Nullstellen, ..., schließlich hat $F^{(n+1)}$ mindestens eine Nullstelle $\xi \in I(x, x_0, \ldots, x_n)$.

Fehlerbetrachtungen

Beweis.

Für $x = x_j$, $j = 0, \ldots, n$, ist die Aussage trivial.

Für festes $x \neq x_j$ betrachten wir die Funktion

$$F(z) := f(z) - p(z) - \alpha\omega(z) \tag{2.10}$$

und bestimmen $\alpha \in \mathbb{R}$ so, dass $F(x) = 0$ gilt.

Dann besitzt F in $I(x, x_0, \ldots, x_n)$ wenigstens $n+2$ Nullstellen. Nach dem Satz von Rolle besitzt F' dort wenigstens $n+1$ Nullstellen, F'' wenigstens n Nullstellen, F''' wenigstens $n-1$ Nullstellen, ..., schließlich hat $F^{(n+1)}$ mindestens eine Nullstelle $\xi \in I(x, x_0, \ldots, x_n)$. Wegen $p \in \Pi_n$ erhält man aus (2.10)

$$F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - 0 - \alpha \cdot (n+1)! = 0.$$

Hiermit folgt wegen $F(x) = 0$ aus (2.10) die Behauptung. □

Fehlerbetrachtungen

Bemerkung 2.22: Aus Satz 2.20 erhält man eine Abschätzung für die Güte der Approximation einer Funktion durch das Interpolationspolynom:

$$\|f - p\|_\infty := \max_{x\in[a,b]} |f(x) - p(x)| \leqslant \frac{K(f)}{(n+1)!} \|\omega\|_\infty, \tag{2.11}$$

wobei

$$K(f) = \|f^{(n+1)}\|_\infty.$$

□

Fehlerbetrachtungen

Bemerkung 2.23: Man erhält ferner aus Satz 2.20 eine Darstellung der dividierten Differenzen.

Fehlerbetrachtungen

Bemerkung 2.23: Man erhält ferner aus Satz 2.20 eine Darstellung der dividierten Differenzen.

Nimmt man im Newtonschen Interpolationspolynom in Satz 2.13 den Knoten x als weitere Stützstelle mit dem Wert $f(x)$ hinzu, so erhält man für das in Satz 2.13 definierte Polynom p

$$f(x) - p(x) = [x_n, x_{n-1}, \ldots, x_0, x] \prod_{i=0}^{n} (x - x_i). \tag{2.12}$$

Fehlerbetrachtungen

Bemerkung 2.23: Man erhält ferner aus Satz 2.20 eine Darstellung der dividierten Differenzen.

Nimmt man im Newtonschen Interpolationspolynom in Satz 2.13 den Knoten x als weitere Stützstelle mit dem Wert $f(x)$ hinzu, so erhält man für das in Satz 2.13 definierte Polynom p

$$f(x) - p(x) = [x_n, x_{n-1}, \ldots, x_0, x] \prod_{i=0}^{n} (x - x_i). \tag{2.12}$$

Durch Vergleich mit der Abschätzung (2.8) folgt

$$[x_n, \ldots, x_0, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

und damit allgemein für die $n-$te dividierte Differenz

$$[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \quad \text{für ein } \xi \in I(x_0, \ldots x_n). \qquad \square$$

Spline-Interpolation

Die in den vorhergehenden Abschnitten behandelte Polynominterpolation ist zwar leicht durchführbar, hat aber den Nachteil, dass bei Verfeinerung der Zerlegung keine Konvergenz zu erwarten ist.

Spline-Interpolation

Die in den vorhergehenden Abschnitten behandelte Polynominterpolation ist zwar leicht durchführbar, hat aber den Nachteil, dass bei Verfeinerung der Zerlegung keine Konvergenz zu erwarten ist.

Bessere Konvergenzeigenschaften haben die nun zu besprechenden Spline-Funktionen.

Spline-Interpolation

Definition 2.24: Sei

$$\Delta \ : \ a := x_0 < x_1 < \ldots < x_n \ =: \ b \tag{2.13}$$

eine Zerlegung des Intervalls $[a, b]$. Dann bezeichnen wir mit $S(\Delta, p, q)$, $p, q \in \mathbb{N}_0$, $0 \leqslant q < p$, die Menge aller Funktionen $s \in C^q[a, b]$, die auf jedem Teilintervall $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, mit einem Polynom vom Höchstgrad p übereinstimmen.

Spline-Interpolation

Definition 2.24: Sei

$$\Delta \ : \ a := x_0 < x_1 < \ldots < x_n \ =: \ b \tag{2.13}$$

eine Zerlegung des Intervalls $[a, b]$. Dann bezeichnen wir mit $S(\Delta, p, q)$, $p, q \in \mathbb{N}_0$, $0 \leqslant q < p$, die Menge aller Funktionen $s \in C^q[a, b]$, die auf jedem Teilintervall $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, mit einem Polynom vom Höchstgrad p übereinstimmen.

Jedes $s \in S(\Delta, p, q)$ heißt (Polynom-)Spline vom Grade p der Differenzierbarkeitsklasse q zur Zerlegung Δ.

Spline-Interpolation

Am häufigsten treten in den Anwendungen (auf Randwertaufgaben) die Räume $S(\Delta, 3, 2)$ (kubische Splines) und $S(\Delta, 3, 1)$ (kubische Hermite Splines) auf.

Spline-Interpolation

Am häufigsten treten in den Anwendungen (auf Randwertaufgaben) die Räume $S(\Delta, 3, 2)$ (kubische Splines) und $S(\Delta, 3, 1)$ (kubische Hermite Splines) auf.

Daneben untersucht man für spezielle Aufgabenstellungen, bei denen Pole oder verschiedenes Wachstum in verschiedenen Teilen des Intervalls erwartet wird (Grenzschichtprobleme), nichtlineare Splines (rationale oder exponentielle Splines).

Spline-Interpolation

Am häufigsten treten in den Anwendungen (auf Randwertaufgaben) die Räume $S(\Delta, 3, 2)$ (kubische Splines) und $S(\Delta, 3, 1)$ (kubische Hermite Splines) auf.

Daneben untersucht man für spezielle Aufgabenstellungen, bei denen Pole oder verschiedenes Wachstum in verschiedenen Teilen des Intervalls erwartet wird (Grenzschichtprobleme), nichtlineare Splines (rationale oder exponentielle Splines).

Wir behandeln nur $S(\Delta, 3, 2)$, sowie zur Motivation $S(\Delta, 1, 0)$.

Stückweise lineare Funktionen

Es sei $\Delta : a = x_0 < x_1 < \cdots < x_n = b$ eine gegebene Zerlegung des Intervalls $[a, b]$ und

$$S(\Delta, 1, 0) = \left\{ s \in C[a, b] : s\big|_{[x_{i-1}, x_i]} \in \Pi_1,\ i = 1, \ldots, n \right\}$$

die Menge der stückweise linearen Funktionen auf $[a, b]$ zu dieser Zerlegung.

Stückweise lineare Funktionen

Es sei $\Delta : a = x_0 < x_1 < \cdots < x_n = b$ eine gegebene Zerlegung des Intervalls $[a, b]$ und

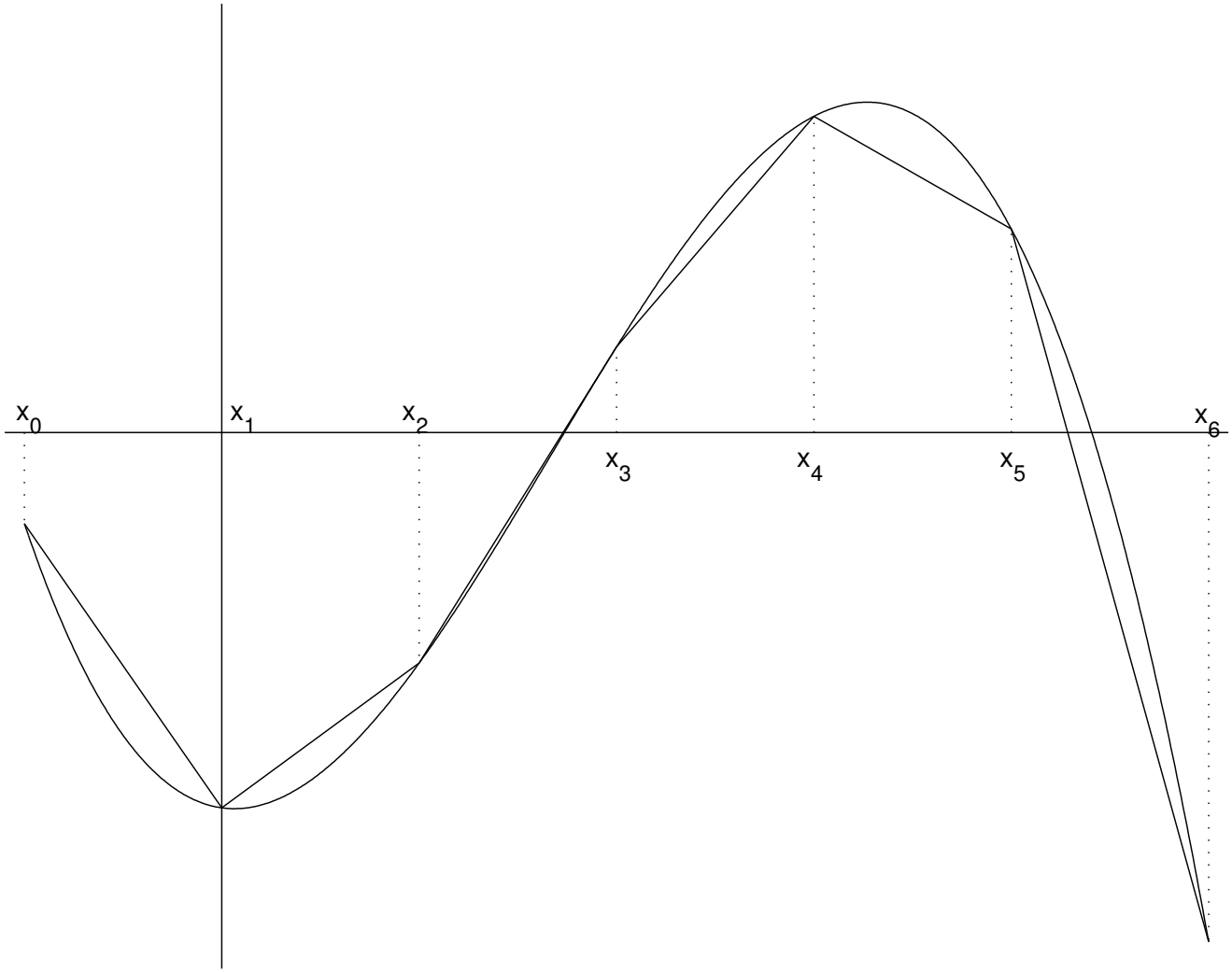
$$S(\Delta, 1, 0) = \left\{ s \in C[a, b] : s\big|_{[x_{i-1}, x_i]} \in \Pi_1,\ i = 1, \ldots, n \right\}$$

die Menge der stückweise linearen Funktionen auf $[a, b]$ zu dieser Zerlegung.

Für gegebene Interpolationsdaten $y_0, \ldots, y_n \in \mathbb{R}$ gibt es offenbar genau einen interpolierenden Spline $s \in S(\Delta, 1, 0)$ mit $s(x_i) = y_i$, $i = 0, \ldots, n$, und dieses s erhält man, indem man in jedem Teilintervall $[x_{i-1}, x_i]$ die Daten (x_{i-1}, y_{i-1}) und (x_i, y_i) durch eine lineare Funktion interpoliert.

Stückweise lineare Funktionen

Stückweise lineare Interpolation



Stückweise lineare Funktionen

Man erhält in den Teilintervallen

$$s(x) = \frac{1}{x_i - x_{i-1}} \big(y_i (x - x_{i-1}) + y_{i-1} (x_i - x) \big), \quad x \in [x_{i-1}, x_i]. \tag{2.14}$$

Stückweise lineare Funktionen

Man erhält in den Teilintervallen

$$s(x) = \frac{1}{x_i - x_{i-1}} \big(y_i (x - x_{i-1}) + y_{i-1} (x_i - x) \big), \quad x \in [x_{i-1}, x_i]. \tag{2.14}$$

Gilt $y_i = f(x_i)$ für eine Funktion $f \in C^2[a,b]$, so erhalten wir aus Satz 2.20 sofort für $x \in [x_{i-1}, x_i]$ den Fehler

$$f(x) - s(x) = \frac{1}{2}(x - x_{i-1})(x - x_i) f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i],$$

Stückweise lineare Funktionen

Man erhält in den Teilintervallen

$$s(x) = \frac{1}{x_i - x_{i-1}} \big(y_i (x - x_{i-1}) + y_{i-1} (x_i - x) \big), \quad x \in [x_{i-1}, x_i]. \tag{2.14}$$

Gilt $y_i = f(x_i)$ für eine Funktion $f \in C^2[a,b]$, so erhalten wir aus Satz 2.20 sofort für $x \in [x_{i-1}, x_i]$ den Fehler

$$f(x) - s(x) = \frac{1}{2}(x - x_{i-1})(x - x_i) f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i],$$

und daher für $x \in [x_{i-1}, x_i]$

$$|f(x) - s(x)| \leqslant \frac{1}{8}(x_i - x_{i-1})^2 |f''(\xi_i)|.$$

Stückweise lineare Funktionen

Man erhält in den Teilintervallen

$$s(x) = \frac{1}{x_i - x_{i-1}} \big(y_i (x - x_{i-1}) + y_{i-1} (x_i - x) \big), \quad x \in [x_{i-1}, x_i]. \tag{2.14}$$

Gilt $y_i = f(x_i)$ für eine Funktion $f \in C^2[a,b]$, so erhalten wir aus Satz 2.20 sofort für $x \in [x_{i-1}, x_i]$ den Fehler

$$f(x) - s(x) = \frac{1}{2}(x - x_{i-1})(x - x_i) f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i],$$

und daher für $x \in [x_{i-1}, x_i]$

$$|f(x) - s(x)| \leqslant \frac{1}{8}(x_i - x_{i-1})^2 |f''(\xi_i)|.$$

Mit $|\Delta| := \max\limits_{i=1,\ldots,n} (x_i - x_{i-1})$ folgt

$$\|f - s\|_\infty \leqslant \frac{1}{8} |\Delta|^2 \, \|f''\|_\infty. \tag{2.15}$$

Stückweise lineare Funktionen

Ist also Δ_n irgendeine Zerlegungsfolge von $[a,b]$ mit

$$\lim_{n\to\infty} |\Delta_n| = 0$$

und $f \in C^2[a,b]$, so konvergiert die zugehörige Folge der interpolierenden Splines $s_n \in S(\Delta_n, 1, 0)$ gleichmäßig gegen f und die Konvergenzgeschwindigkeit wird durch (2.15) beschrieben.

Stückweise lineare Funktionen

Für $f \in C^0[a,b]$ erhält man für $x \in [x_{i-1}, x_i]$ aus (2.14)

$$
\begin{aligned}
|f(x) - s(x)| &= \frac{\left|(x - x_{i-1})(f(x) - f(x_i)) + (x_i - x)(f(x) - f(x_{i-1}))\right|}{x_i - x_{i-1}} \\
&\leqslant \frac{\Big((x - x_{i-1})|f(x) - f(x_i)| + (x_i - x)|f(x) - f(x_{i-1})|\Big)}{x_i - x_{i-1}} \\
&\leqslant \max\left(|f(x) - f(x_i)|, |f(x) - f(x_{i-1})|\right),
\end{aligned}
$$

Stückweise lineare Funktionen

Für $f \in C^0[a,b]$ erhält man für $x \in [x_{i-1}, x_i]$ aus (2.14)

$$\begin{aligned} |f(x) - s(x)| &= \frac{\left|(x - x_{i-1})(f(x) - f(x_i)) + (x_i - x)(f(x) - f(x_{i-1}))\right|}{x_i - x_{i-1}} \\ &\leqslant \frac{\Big((x - x_{i-1})|f(x) - f(x_i)| + (x_i - x)|f(x) - f(x_{i-1})|\Big)}{x_i - x_{i-1}} \\ &\leqslant \max\left(|f(x) - f(x_i)|, |f(x) - f(x_{i-1})|\right), \end{aligned}$$

und daher folgt $\|f - s\|_\infty \leqslant \omega(f, |\Delta|)$, wobei

$$\omega(f,h) := \sup\{|f(x) - f(y)| : x, y \in [a,b],\ |x - y| \leqslant h\}$$

den Stetigkeitsmodul von f zur Schrittweite h bezeichnet.

Stückweise lineare Funktionen

Für $f \in C^0[a,b]$ erhält man für $x \in [x_{i-1}, x_i]$ aus (2.14)

$$\begin{aligned} |f(x) - s(x)| &= \frac{\left|(x - x_{i-1})(f(x) - f(x_i)) + (x_i - x)(f(x) - f(x_{i-1}))\right|}{x_i - x_{i-1}} \\ &\leqslant \frac{\Big((x - x_{i-1})|f(x) - f(x_i)| + (x_i - x)|f(x) - f(x_{i-1})|\Big)}{x_i - x_{i-1}} \\ &\leqslant \max\left(|f(x) - f(x_i)|, |f(x) - f(x_{i-1})|\right), \end{aligned}$$

und daher folgt $\|f - s\|_\infty \leqslant \omega(f, |\Delta|)$, wobei

$$\omega(f,h) := \sup\{|f(x) - f(y)| : x, y \in [a,b],\ |x - y| \leqslant h\}$$

den Stetigkeitsmodul von f zur Schrittweite h bezeichnet.

Ist Δ_n eine Zerlegungsfolge von $[a,b]$ mit $\lim_{n\to\infty} |\Delta_n| = 0$, so konvergieren auch für nur stetiges f die interpolierenden linearen Splines gleichmäßig gegen f.

Stückweise lineare Funktionen

Ähnlich wie bei der Lagrangeschen Interpolation können wir s darstellen als

$$s(x) = \sum_{i=0}^{n} f_i \phi_i(x), \; x \in [a,b]$$

mit den Basisfunktionen ϕ_i, $i = 0, \ldots, n$,

$$\phi_i(x) := \begin{cases} (x - x_{i-1}) \,/\, (x_i - x_{i-1}) & , \quad x \in [x_{i-1}, x_i] \\ (x_{i+1} - x) \,/\, (x_{i+1} - x_i) & , \quad x \in [x_i, x_{i+1}] \\ 0 & , \quad x \notin [x_{i-1}, x_{i+1}] \end{cases}$$

wobei $x_{-1} < a$ und $x_{n+1} > b$ beliebig gewählt sind.

Stückweise lineare Funktionen

Die angegebenen ϕ_i heißen Dachfunktionen (engl.: hat functions).

Stückweise lineare Funktionen

Die angegebenen ϕ_i heißen Dachfunktionen (engl.: hat functions).

Sie besitzen einen lokalen Träger $[x_{i-1}, x_{i+1}]$.

Stückweise lineare Funktionen

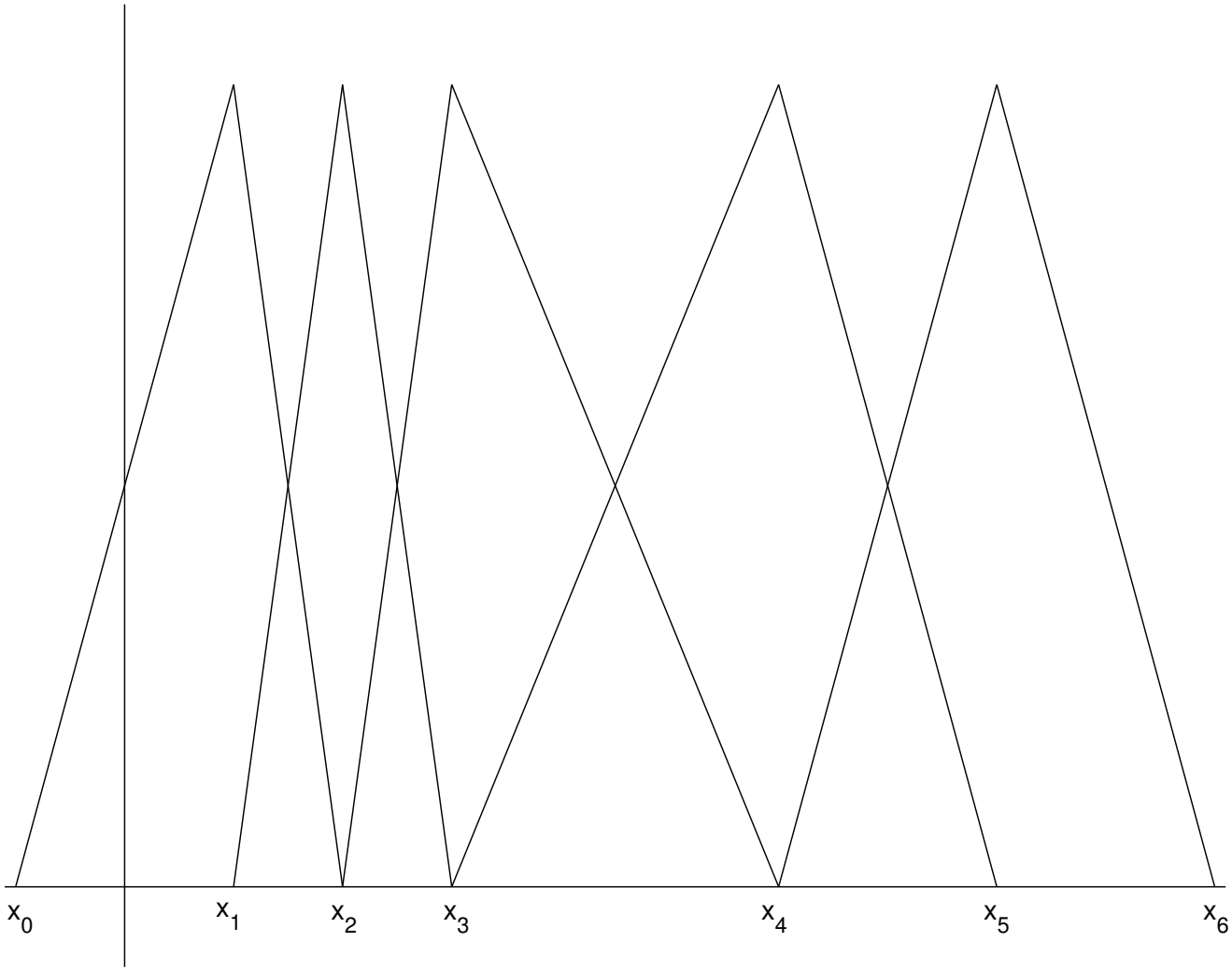
Die angegebenen ϕ_i heißen Dachfunktionen (engl.: hat functions).

Sie besitzen einen lokalen Träger $[x_{i-1}, x_{i+1}]$.

Will man also s an einer Stelle $x \in (x_{i-1}, x_i)$ auswerten, so hat man wegen $\phi_j(x) = 0$ für $j \notin \{i, i-1\}$ nur $\phi_i(x)$ und $\phi_{i-1}(x)$ zu berechnen (anders als bei den $\ell_j(x)$ in Satz 2.4).

Stückweise lineare Funktionen

Dachfunktionen



Kubische Splines

Bei den kubischen Splines $s \in S(\Delta, 3, 2)$ sind die Verhältnisse nicht ganz so einfach wie in den Fällen $S(\Delta, 3, 1)$ und $S(\Delta, 1, 0)$, da man die Interpolationsaufgabe nicht in jedem Teilintervall getrennt ausführen kann.

Kubische Splines

Bei den kubischen Splines $s \in S(\Delta, 3, 2)$ sind die Verhältnisse nicht ganz so einfach wie in den Fällen $S(\Delta, 3, 1)$ und $S(\Delta, 1, 0)$, da man die Interpolationsaufgabe nicht in jedem Teilintervall getrennt ausführen kann.

Ein Spline $s \in S(\Delta, 3, 2)$ ist in jedem Teilintervall $[x_{i-1}, x_i]$ ein Polynom dritten Grades, also ist s durch $4n$ Parameter bestimmt. Durch die Forderung $s \in C^2[a, b]$ werden in jedem inneren Knoten x_i, $i = 1, \ldots, n-1$, drei Bedingungen gegeben:

$$s^{(j)}(x_i - 0) = s^{(j)}(x_i + 0),\ j = 0, 1, 2.$$

Kubische Splines

Daher besitzt ein kubischer Spline noch (wenigstens) $n+3$ Freiheitsgrade, und wir können nicht erwarten, dass s durch die $n+1$ Interpolationsbedingungen

$$s(x_i) = y_i, \ i = 0, \ldots, n,$$

festgelegt ist, sondern es müssen noch zwei „Randbedingungen“ hinzugenommen werden. Dieses kann (je nach Aufgabenstellung) auf verschiedene Weise geschehen.

Kubische Splines

Satz 2.25

Es sei Δ eine Zerlegung von [a,b], und es seien $y_0, \ldots, y_n \in \mathbb{R}$ gegeben.

Kubische Splines

Satz 2.25

Es sei Δ eine Zerlegung von [a,b], und es seien $y_0, \ldots, y_n \in \mathbb{R}$ gegeben.

Dann gibt es für jede der vier folgenden Randbedingungen

1. $s'(x_0) = y'_0, \;\; s'(x_n) = y'_n, \qquad (y'_0, y'_n \in \mathbb{R}$ *gegeben)*
2. $s'(x_0) = s'(x_n), \;\; s''(x_0) = s''(x_n),$
3. $s''(x_0) = s''(x_n) = 0,$
4. $s''(x_0) = y''_0, \;\; s''(x_n) = y''_n, \qquad (y''_0, y''_n \in \mathbb{R}$ *gegeben)*

genau einen interpolierenden kubischen Spline $s \in S(\Delta, 3, 2)$ mit

$$s(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.16}$$

Kubische Splines

Bemerkung 2.26: Wird die Interpolation mit kubischen Splines verwendet, um eine Funktion $f : [a, b] \to \mathbb{R}$ zu approximieren, so wird man die Randbedingungen (i) verwenden, wenn die Ableitung von f in den Randpunkten a und b des Intervalls bekannt sind, die Randbedingung (ii), wenn die Funktion f periodisch mit der Periode $b - a$ ist, und die Randbedingung (iv), wenn zusätzlich zu den Lagrangeschen Interpolationsdaten am Rand des Intervalls die zweiten Ableitungen bekannt sind. □

Kubische Splines

Bemerkung 2.26: Wird die Interpolation mit kubischen Splines verwendet, um eine Funktion $f : [a, b] \to \mathbb{R}$ zu approximieren, so wird man die Randbedingungen (i) verwenden, wenn die Ableitung von f in den Randpunkten a und b des Intervalls bekannt sind, die Randbedingung (ii), wenn die Funktion f periodisch mit der Periode $b - a$ ist, und die Randbedingung (iv), wenn zusätzlich zu den Lagrangeschen Interpolationsdaten am Rand des Intervalls die zweiten Ableitungen bekannt sind. □

Bemerkung 2.27: Erfüllt $s \in S(\Delta, 3, 2)$ die Randbedingung (iii), so kann man s auf $\{x : x < x_0\}$ bzw. $\{x : x > x_n\}$ zweimal stetig differenzierbar als lineare Funktion fortsetzen. Kubische Splines, die man auf diese Weise erhält, heißen natürliche Splines. □

Kubische Splines

Bemerkung 2.28: Im Buch von de Boor werden vier weitere Randbedingungen diskutiert, unter denen die Existenz und Eindeutigkeit des interpolierenden Splines gesichert ist. Unter ihnen besonders wichtig ist die sog. not-a-knot Bedingung. Diese wird verwendet, wenn nichts über das Verhalten der interpolierenden Funktion an den Rändern des Intervalls (außer den Funktionswerten) bekannt ist. Man wählt dann als Knoten für den Spline die Punkte $x_0 < x_2 < \cdots < x_{n-2} < x_n$. Ein Spline zu diesen $n-2$ Intervallen hat $n+1$ Freiheitsgrade und ist durch die $n+1$ Interpolationsbedingungen $s(x_j) = y_j, j = 0, \ldots, n$ eindeutig bestimmt. □

Kubische Splines

Beweis: (von Satz 2.25) Der Beweis ist konstruktiv, er liefert also ein Verfahren zur Berechnung der jeweiligen interpolierenden Splines.

Kubische Splines

Beweis: (von Satz 2.25) Der Beweis ist konstruktiv, er liefert also ein Verfahren zur Berechnung der jeweiligen interpolierenden Splines.

Sei $h_{j+1} := x_{j+1} - x_j$, $j = 0, \ldots, n-1$, und $m_j := s''(x_j)$, $j = 0, \ldots, n$, die zweite Ableitung der gesuchten Splinefunktion an der Stelle x_j.

Kubische Splines

Beweis: (von Satz 2.25) Der Beweis ist konstruktiv, er liefert also ein Verfahren zur Berechnung der jeweiligen interpolierenden Splines.

Sei $h_{j+1} := x_{j+1} - x_j$, $j = 0, \ldots, n-1$, und $m_j := s''(x_j)$, $j = 0, \ldots, n$, die zweite Ableitung der gesuchten Splinefunktion an der Stelle x_j.

Wir wollen zeigen, dass der Vektor $\boldsymbol{m} := (m_0, \ldots, m_n)^T$ für jede der vier Randbedingungen eindeutige Lösung eines linearen Gleichungssystems ist und dass man aus den m_j den Spline $s(x)$ an jeder Stelle $x \in [a, b]$ leicht errechnen kann.

Kubische Splines

s ist in jedem Teilintervall $[x_j, x_{j+1}]$ ein kubisches Polynom. Also ist s'' stückweise linear, und man kann s'' mit Hilfe der m_j so beschreiben (vgl. die Überlegungen in Abschnitt 2.3.1 für $S(\Delta, 1, 0)$).

$$s''(x) = \frac{1}{h_{j+1}} \left(m_j(x_{j+1} - x) + m_{j+1}(x - x_j) \right), \quad x \in [x_j, x_{j+1}].$$

Kubische Splines

s ist in jedem Teilintervall $[x_j, x_{j+1}]$ ein kubisches Polynom. Also ist s'' stückweise linear, und man kann s'' mit Hilfe der m_j so beschreiben (vgl. die Überlegungen in Abschnitt 2.3.1 für $S(\Delta, 1, 0)$).

$$s''(x) = \frac{1}{h_{j+1}} \left(m_j (x_{j+1} - x) + m_{j+1}(x - x_j) \right), \quad x \in [x_j, x_{j+1}].$$

Durch Integration erhält man für $x \in [x_j, x_{j+1}]$, $j = 0, \ldots, n-1$,

$$s'(x) = -m_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + m_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} + \alpha_j \tag{2.17}$$

und

$$s(x) = m_j \frac{(x_{j+1} - x)^3}{6h_{j+1}} + m_{j+1} \frac{(x - x_j)^3}{6h_{j+1}} + \alpha_j (x - x_j) + \beta_j. \tag{2.18}$$

Kubische Splines

Die Integrationskonstanten α_j und β_j erhält man aus den Interpolationsbedingungen

$$\begin{aligned} s(x_j) &= m_j \frac{h_{j+1}^2}{6} + \beta_j = y_j \\ s(x_{j+1}) &= m_{j+1} \frac{h_{j+1}^2}{6} + \alpha_j h_{j+1} + \beta_j = y_{j+1}, \end{aligned}$$

d.h.

$$\begin{aligned} \beta_j &= y_j - m_j \frac{h_{j+1}^2}{6} \\ \alpha_j &= \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1} - m_j). \end{aligned} \tag{2.19}$$

Kubische Splines

Die Integrationskonstanten α_j und β_j erhält man aus den Interpolationsbedingungen

$$\begin{aligned} s(x_j) &= m_j \frac{h_{j+1}^2}{6} + \beta_j = y_j \\ s(x_{j+1}) &= m_{j+1} \frac{h_{j+1}^2}{6} + \alpha_j h_{j+1} + \beta_j = y_{j+1}, \end{aligned}$$

d.h.

$$\begin{aligned} \beta_j &= y_j - m_j \frac{h_{j+1}^2}{6} \\ \alpha_j &= \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1} - m_j). \end{aligned} \tag{2.19}$$

Setzt man α_j und β_j aus (2.19) in (2.18) ein, so erhält man die gewünschte Darstellung von s in Abhängigkeit von den m_j.

Kubische Splines

$n-1$ Bestimmungsgleichungen für die m_j erhält man, indem man die Stetigkeit von s' ausnutzt: Setzt man α_j aus (2.19) in (2.17) ein, so erhält man

$$s'(x) = -m_j \frac{(x_{j+1}-x)^2}{2h_{j+1}} + m_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1}-m_j),\ x \in [x_j, x_{j+1}]. \tag{2.20}$$

Kubische Splines

$n-1$ Bestimmungsgleichungen für die m_j erhält man, indem man die Stetigkeit von s' ausnutzt: Setzt man α_j aus (2.19) in (2.17) ein, so erhält man

$$s'(x) = -m_j \frac{(x_{j+1}-x)^2}{2h_{j+1}} + m_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1}-m_j),\ x \in [x_j, x_{j+1}]. \tag{2.20}$$

Also liefert die Forderung $s'(x_j - 0) = s'(x_j + 0)$

$$\frac{y_j - y_{j-1}}{h_j} + \frac{h_j}{3}m_j + \frac{h_j}{6}m_{j-1} = \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{h_{j+1}}{3}m_j - \frac{h_{j+1}}{6}m_{j+1}\ ,$$

Kubische Splines

$n-1$ Bestimmungsgleichungen für die m_j erhält man, indem man die Stetigkeit von s' ausnutzt: Setzt man α_j aus (2.19) in (2.17) ein, so erhält man

$$s'(x) = -m_j \frac{(x_{j+1}-x)^2}{2h_{j+1}} + m_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1}-m_j),\ x \in [x_j, x_{j+1}]. \tag{2.20}$$

Also liefert die Forderung $s'(x_j - 0) = s'(x_j + 0)$

$$\frac{y_j - y_{j-1}}{h_j} + \frac{h_j}{3}m_j + \frac{h_j}{6}m_{j-1} = \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{h_{j+1}}{3}m_j - \frac{h_{j+1}}{6}m_{j+1}\ ,$$

d.h. für $j = 1, \ldots, n-1$

$$\frac{h_j}{6}m_{j-1} + \frac{h_j + h_{j+1}}{3}m_j + \frac{h_{j+1}}{6}m_{j+1} = \frac{y_{j+1}-y_j}{h_{j+1}} - \frac{y_j - y_{j-1}}{h_j}. \tag{2.21}$$

Kubische Splines

Die restlichen beiden Bedingungen für die m_j erhält man aus den Randbedingungen (i), (ii), (iii) oder (iv).

Kubische Splines

Die restlichen beiden Bedingungen für die m_j erhält man aus den Randbedingungen (i), (ii), (iii) oder (iv).

Für die natürlichen Splines hat man in (2.20)

$$s''(a) = m_0 = 0 = m_n = s''(b)$$

zu setzen, im Fall (iv) die inhomogenen Gleichungen

$$m_0 = y_0'' \quad \text{und} \quad m_n = y_n''.$$

Kubische Splines

Die restlichen beiden Bedingungen für die m_j erhält man aus den Randbedingungen (i), (ii), (iii) oder (iv).

Für die natürlichen Splines hat man in (2.20)

$$s''(a) = m_0 = 0 = m_n = s''(b)$$

zu setzen, im Fall (iv) die inhomogenen Gleichungen

$$m_0 = y_0'' \quad \text{und} \quad m_n = y_n''.$$

Im Fall (i) hat man wegen (2.20) das System (2.21) zu ergänzen durch

$$\begin{aligned} \frac{h_1}{3} m_0 + \frac{h_1}{6} m_1 &= \frac{y_1 - y_0}{h_1} - y_0', \\ \frac{h_n}{6} m_{n-1} + \frac{h_n}{3} m_n &= y_n' - \frac{y_n - y_{n-1}}{h_n}. \end{aligned} \tag{2.22}$$

Kubische Splines

Im Falle periodischer Randbedingungen gilt $m_0 = m_n$, und wegen $s'(a) = s'(b)$ erhält man aus (2.20)

$$\frac{h_1}{6} m_1 + \frac{h_n}{6} m_{n-1} + \frac{h_n + h_1}{3} m_0 = \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}. \tag{2.23}$$

Kubische Splines

Im Falle periodischer Randbedingungen gilt $m_0 = m_n$, und wegen $s'(a) = s'(b)$ erhält man aus (2.20)

$$\frac{h_1}{6} m_1 + \frac{h_n}{6} m_{n-1} + \frac{h_n + h_1}{3} m_0 = \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}. \tag{2.23}$$

In jedem Fall ergeben sich also die m_j als Lösung eines linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.24}$$

wobei für alle Zeilen der Matrix $\boldsymbol{A}$ gilt:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| .$$

Kubische Splines

Im Falle periodischer Randbedingungen gilt $m_0 = m_n$, und wegen $s'(a) = s'(b)$ erhält man aus (2.20)

$$\frac{h_1}{6} m_1 + \frac{h_n}{6} m_{n-1} + \frac{h_n + h_1}{3} m_0 = \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}. \tag{2.23}$$

In jedem Fall ergeben sich also die m_j als Lösung eines linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.24}$$

wobei für alle Zeilen der Matrix $\boldsymbol{A}$ gilt:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| .$$

Die Koeffizientenmatrix ist strikt diagonaldominant. Nach dem Satz von Gerschgorin ist sie regulär, und daher ist das Gleichungssystem (2.24) für jede rechte Seite $\boldsymbol{b}$ eindeutig lösbar. ■

Kubische Splines

Satz 2.29

Sei $f \in C^3[a,b]$ und sei $s \in S(\Delta, 3, 2)$ der interpolierende Spline, der eine der Randbedingungen (i) oder (ii) oder $s''(a) = f''(a)$, $s''(b) = f''(b)$ erfüllt.

Kubische Splines

Satz 2.29

Sei $f \in C^3[a,b]$ und sei $s \in S(\Delta, 3, 2)$ der interpolierende Spline, der eine der Randbedingungen (i) oder (ii) oder $s''(a) = f''(a)$, $s''(b) = f''(b)$ erfüllt.

Dann gilt mit den Konstanten $C_0 = \frac{9}{8}$ und $C_1 = C_2 = \frac{9}{4}$

$$\|s^{(\nu)} - f^{(\nu)}\|_\infty \leqslant C_\nu |\Delta|^{3-\nu}\ \omega(f''', |\Delta|),\ \nu = 0, 1, 2, \tag{2.25}$$

wobei $\omega(g,h)$ den Stetigkeitsmodul von g bezeichnet.

Kubische Splines

Satz 2.29

Sei $f \in C^3[a,b]$ und sei $s \in S(\Delta, 3, 2)$ der interpolierende Spline, der eine der Randbedingungen (i) oder (ii) oder $s''(a) = f''(a)$, $s''(b) = f''(b)$ erfüllt.

Dann gilt mit den Konstanten $C_0 = \frac{9}{8}$ und $C_1 = C_2 = \frac{9}{4}$

$$\|s^{(\nu)} - f^{(\nu)}\|_\infty \leqslant C_\nu |\Delta|^{3-\nu}\, \omega(f''', |\Delta|),\ \nu = 0, 1, 2, \tag{2.25}$$

wobei $\omega(g,h)$ den Stetigkeitsmodul von g bezeichnet.

Genügt f''' einer Lipschitzbedingung

$$|f'''(x) - f'''(y)| \leqslant L|x-y| \quad \text{für alle } x, y \in [a,b],$$

so gilt

$$\|s^{(\nu)} - f^{(\nu)}\|_\infty \leqslant L \cdot C_\nu |\Delta|^{4-\nu}, \quad \nu = 0, 1, 2. \tag{2.26}$$

Kubische Splines

Bemerkung 2.30: Satz 2.29 zeigt, dass Spline-Interpolationen geeignet sind, um Ableitungen von Funktionen, von denen nur diskrete Werte bekannt sind, zu approximieren. □

Kubische Splines

Bemerkung 2.30: Satz 2.29 zeigt, dass Spline-Interpolationen geeignet sind, um Ableitungen von Funktionen, von denen nur diskrete Werte bekannt sind, zu approximieren. □

Auch der Raum der kubischen Splines $S(\Delta, 3, 2)$ besitzt eine lokale Basis.

Kubische Splines

Bemerkung 2.30: Satz 2.29 zeigt, dass Spline-Interpolationen geeignet sind, um Ableitungen von Funktionen, von denen nur diskrete Werte bekannt sind, zu approximieren. □

Auch der Raum der kubischen Splines $S(\Delta, 3, 2)$ besitzt eine lokale Basis.

Seien $x_{-3} < x_{-2} < x_{-1} < a$ und $b < x_{n+1} < x_{n+2} < x_{n+3}$ zusätzliche Knoten. Dann überzeugt man sich leicht (aber mit Hilfe einer längeren Rechnung), dass die Funktionen

Kubische Splines

$$\phi_i(x) = \begin{cases} (x - x_{i-2})_+^3 \Big/ \prod_{j=i-1}^{i+2} (x_j - x_{i-2}) \\ \qquad + (x - x_{i-1})_+^3 \Big/ \prod_{\substack{j=i-2 \\ j \neq i-1}}^{i+2} (x_j - x_{i-1}) \,, & x \leqslant x_i \\ (x_{i+2} - x)_+^3 \Big/ \prod_{j=i-2}^{i+1} (x_{i+2} - x_j) \\ \qquad + (x_{i+1} - x)_+^3 \Big/ \prod_{\substack{j=i-2 \\ j \neq i+1}}^{i+2} (x_{i+1} - x_j) \,, & x \geqslant x_i \end{cases}$$

für $i = -1, \ldots, n+1$, eine Basis von $S(\Delta, 3, 2)$ bilden.

Kubische Splines

Dabei bezeichnet $(x)_+$ die Funktion

$$(x)_+ := \begin{cases} x & \text{für} \quad x \geqslant 0, \\ 0 & \text{für} \quad x \leqslant 0. \end{cases}$$

Kubische Splines

Dabei bezeichnet $(x)_+$ die Funktion

$$(x)_+ := \begin{cases} x & \text{für} \quad x \geqslant 0, \\ 0 & \text{für} \quad x \leqslant 0. \end{cases}$$

Die Basisfunktionen ϕ_j heißen B-Splines. Die folgende Abbildung enthält die Graphen einiger B-Splines.

Kubische Splines

Dabei bezeichnet $(x)_+$ die Funktion

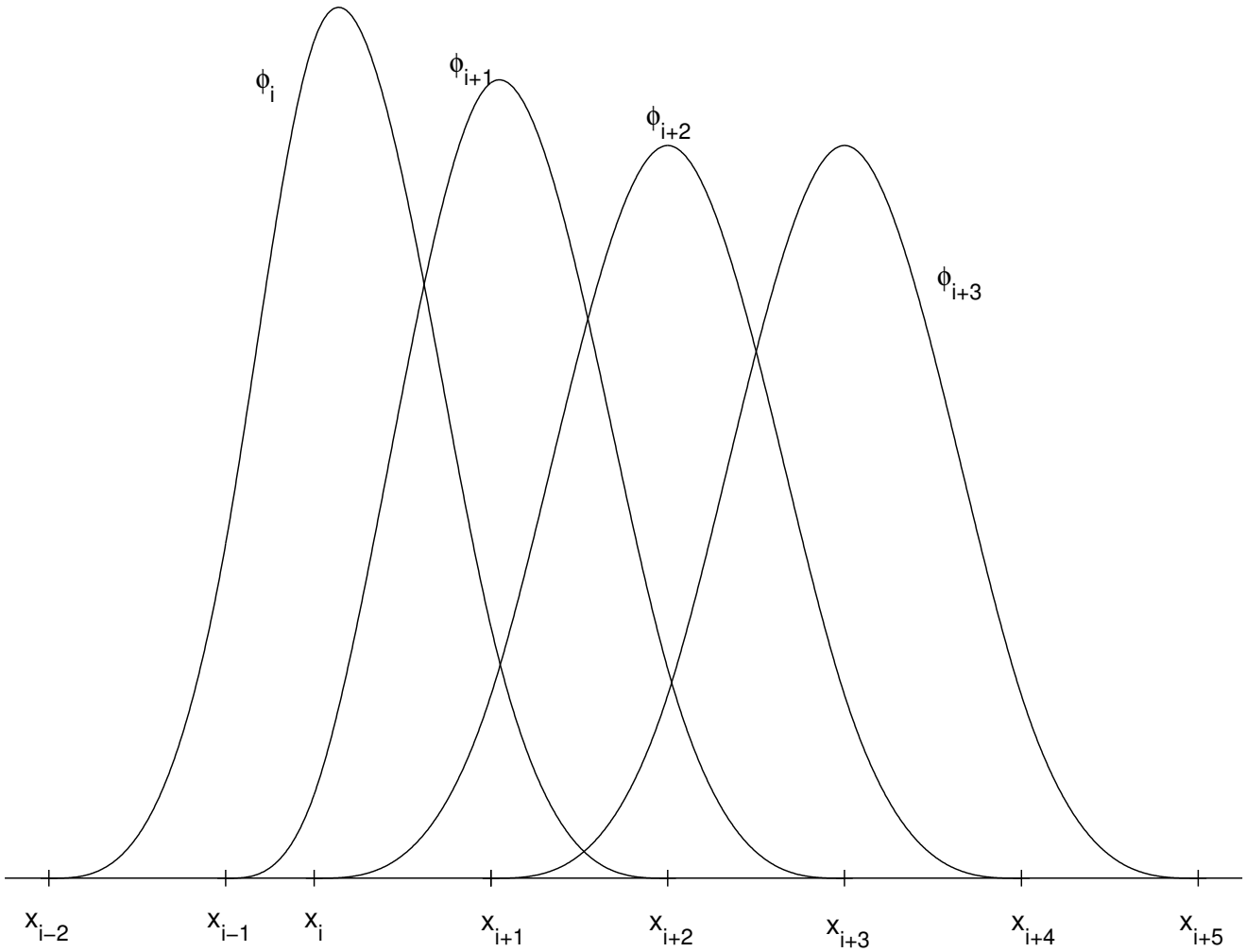
$$(x)_+ := \begin{cases} x & \text{für} \quad x \geqslant 0, \\ 0 & \text{für} \quad x \leqslant 0. \end{cases}$$

Die Basisfunktionen ϕ_j heißen B-Splines. Die folgende Abbildung enthält die Graphen einiger B-Splines.

Jeder B-Spline ist auf 4 Teilintervallen nichttrivial. Man kann zeigen, dass es keine Basis von $S(\Delta, 3, 2)$ mit kleinerem Träger gibt.

Kubische Splines

B-Splines



Kubische Splines

Man kann mit diesen ϕ_i den Ansatz

$$s(x) = \sum_{i=-1}^{n+1} c_i \phi_i(x)$$

machen und zur Lösung des Interpolationsproblems das lineare Gleichungssystem

$$s(x_i) = y_i + \text{Randbedingungen}$$

behandeln.

Kubische Splines

Man kann mit diesen ϕ_i den Ansatz

$$s(x) = \sum_{i=-1}^{n+1} c_i \phi_i(x)$$

machen und zur Lösung des Interpolationsproblems das lineare Gleichungssystem

$$s(x_i) = y_i + \text{Randbedingungen}$$

behandeln.

Dieses besitzt ähnlich wie das für die M_i wieder eine tridiagonale, diagonaldominante Koeffizientenmatrix, ist also problemlos lösbar.

Kubische Splines

Nicht benutzen sollte man jedoch für numerische Zwecke die folgende Basis von $S(\Delta, 3, 2)$, die bisweilen in Büchern angegeben ist:

$$1, x, x^2, x^3, (x - x_i)_+^3, \ i = 1, \ldots, n-1,$$

Kubische Splines

Nicht benutzen sollte man jedoch für numerische Zwecke die folgende Basis von $S(\Delta, 3, 2)$, die bisweilen in Büchern angegeben ist:

$$1, x, x^2, x^3, (x - x_i)_+^3, \ i = 1, \ldots, n-1,$$

Diese ergibt oft schlechte(re) Ergebnisse wegen der schlechten Kondition des entstehenden linearen Gleichungssystems.

Kubische Splines

MATLAB stellt die Funktion `yy=spline(x,y,xx)` zur Verfügung. Besitzen die Vektoren x und y gleiche Länge, so wird der interpolierende kubische Spline mit not-a-knot Randbedingungen bestimmt.

Kubische Splines

MATLAB stellt die Funktion `yy=spline(x,y,xx)` zur Verfügung. Besitzen die Vektoren x und y gleiche Länge, so wird der interpolierende kubische Spline mit not-a-knot Randbedingungen bestimmt.

Ist die Länge von y um 2 größer als die Länge n von x, so werden die erste und letzte Komponente von y als Steigungen von s in $x(1)$ und $x(n)$ gedeutet. Der Vektor yy enthält in der j-ten Komponente $yy(j) = s(xx(j))$.

Kubische Splines

MATLAB stellt die Funktion `yy=spline(x,y,xx)` zur Verfügung. Besitzen die Vektoren x und y gleiche Länge, so wird der interpolierende kubische Spline mit not-a-knot Randbedingungen bestimmt.

Ist die Länge von y um 2 größer als die Länge n von x, so werden die erste und letzte Komponente von y als Steigungen von s in $x(1)$ und $x(n)$ gedeutet. Der Vektor yy enthält in der j-ten Komponente $yy(j) = s(xx(j))$.

Zusätzlich wird von MATLAB die Toolbox SPLINES angeboten. Hierin werden auch andere Randbedingungen für $S(\Delta, p, p-1)$, Glättung von Daten durch Ausgleich und Tensorprodukte von Splines zur Darstellung von Flächen angeboten.