2 INTERPOLATION

Häufig sind in der Praxis z.B. durch Marktanalysen, technische Messungen von einer Funktion nur einzelne Punkte bekannt, aber keine analytische Beschreibung der Funktion, um sie an beliebigen Stellen auswerten zu können. Könnte man die diskreten Daten durch eine (eventuell glatte) Kurve verbinden, so wäre es möglich, die unbekannte Funktion an den dazwischenliegenden Stellen zu schätzen. In anderen Fällen will man eine schwierig berechenbare Funktion näherungsweise durch eine einfachere darstellen. Eine Interpolationsfunktion kann diese Anforderung der Einfachheit erfüllen.

Interpolationsaufgabe: Eine gegebene Funktion $f : I \to \mathbb{R}$ sei geeignet zu approximieren unter der Vorgabe, dass f an diskreten (d.h. endlich vielen) Stützstellen die gegebenen Funktionswerte annehmen soll.

Die Interpolation ist somit eine Art der Approximation. Die Approximationsgüte hängt vom Ansatz ab. Um sie zu schätzen, werden Zusatzinformationen (Co-observations) über die Funktion f benötigt. Diese ergeben sich auch bei Unkenntnis von f häufig in natürlicher Weise: Beschränktheit, Stetigkeit oder Differenzierbarkeit lassen sich häufig voraussetzen.

Bei anderen Approximationsverfahren wie z. B. der Ausgleichungsrechnung wird nicht gefordert, dass die Daten exakt wiedergegeben werden; das unterscheidet diese Verfahren von der Interpolation.

Bemerkung 2.0.1 *i) Ist man am gesamten Verlauf von f interessiert, so sollte man eine Interpoliernde If konstruieren, die sich „möglichst wenig“ von f unterscheidet.*

ii) Diese **Interpolierende** *If sollte eine leicht berechenbare Funktion sein - hierfür eignen sich* **Polynome**, **trigonometrische Funktionen**, **Exponentialfunktionen** *sowie* **rationale Funktionen**.

2.1 DAS ALLGEMEINE INTERPOLATIONSPROBLEM

Manchmal möchte man nicht nur Daten (also Funktionswerte) interpolieren, sondern z.B. auch Steigungen, Krümmungen oder Richtungsableitungen. Dazu ist folgende allgemeine Problemformulierung hilfreich.

Definition 2.1.1 (Lineares Funktional) *Sei $\mathcal{F}$ ein linearer Raum (z.B. ein Funktionenraum $(C[a,b], C^\infty(\Omega), L_2(\Omega), \ldots)$. Eine Abbildung $\mu : \mathcal{F} \to \mathbb{R}$ heißt* ***lineares Funktional****, falls*

$$\mu(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 \mu(f_1) + \alpha_2 \mu(f_2), \quad \alpha_1, \alpha_2 \in \mathbb{R},\ f_1, f_2 \in \mathcal{F}, \tag{2.1}$$

d.h. falls μ reellwertig und linear ist.

Beispiel 2.1.2 Folgende Funktionen sind lineare Funktionale:

(a) $\mathcal{F} = C[a,b],\ x_0 \in [a,b],\ \mu(f) := f(x_0)$, Funktionswerte, Punktauswertungen.

(b) $\mathcal{F} = C^1[a,b],\ x_0 \in [a,b],\ \mu(f) := f'(x_0)$, Steigungen, Ableitungen.

(c) $\mathcal{F} = R[a,b]$,

$$\mu(f) := \int_a^b f(x)\,dx,$$

Integrale.

(d) Mittelwerte, Mediane und andere statistische Größen.

Dann lautet das **allgemeine Interpolationsproblem:** Sei $\mathcal{F}$ ein Funktionenraum und G_n ein $(n+1)$-dimensionaler Teilraum von $\mathcal{F}$. Weiter seien lineare Funktionale $\mu_0, \ldots, \mu_n$ auf G_n gegeben.

$$\begin{cases} \text{Zu } f \in \mathcal{F} & \text{finde } g_n \in G_n \text{ mit} \\ & \mu_j(g_n) = \mu_j(f), \quad j = 0, \ldots, n. \end{cases} \tag{2.2}$$

Man kann nun die Frage der Existenz und Eindeutigkeit beweisen:

Lemma 2.1.3 (Existenz und Eindeutigkeit der Lösung der allgemeinen Interpolationsaufgabe) *Die allgemeine Interpolationsaufgabe (2.2) hat genau dann für jedes $f \in \mathcal{F}$ eine eindeutige Lösung g_n, wenn*

$$\det[(\mu_i(\varphi_j))_{i,j=0}^n] \neq 0$$

für eine (und damit jede) Basis $\{\varphi_0, \ldots, \varphi_n\}$ von G_n gilt.

Beweis. Sei $g_n \in G_n$, dann existieren eindeutig bestimmte Koeffizienten $\{\alpha_0, \ldots, \alpha_n\}$ mit $g_n = \sum_{j=0}^n \alpha_j \varphi_j$, also bedeutet (2.2)

$$\mu_i(g_n) \quad = \quad \sum_{j=0}^n \alpha_j \mu_i(\varphi_j) = (G\alpha)_i \overset{(2.2)}{=} \mu_i(f) = b_i$$

mit $G = (\mu_i(\varphi_j))_{i,j=0}^n$, $\alpha = (\alpha_j)_{j=0}^n$, $b = (b_j)_{j=0}^n$, d.h. $G\alpha = b$. □

Bemerkung 2.1.4 *Je nach Wahl von $g_n \in G_n$ erhält man unterschiedliche Instanzen der Interpolation:*

- *Ist $G_n = \mathbb{P}_n$ redet man von **Polynominterpolation**.*
- *$G_{2n} := \left\{ F(x) := \frac{P(x)}{Q(x)}, P, Q \in \mathbb{P}_n, Q \neq 0 \right\}$ ergibt die **rationale Interpolation**.*
- *Ist $G_n = \mathcal{S}^k(\mathcal{T})$ so spricht man von **Spline-Interpolation** (Vgl. Kapitel 4).*

2.2 POLYNOMINTERPOLATION

Nach dem aus der Analysis bekannten Approximationssatz von Weierstraßist es möglich, jede stetige Funktion beliebig gut durch Polynome zu approximieren. In diesem Abschnitt sei $G_n = \mathbb{P}_n$, d.h. der Vektorraum der Polynome vom Grad kleiner gleich n.

Wenn man nun in der allgemeinen Interpolationsaufgabe (2.2) nur Funktionswerte f_i interpoliert, spricht man von ***Lagrange*-Interpolation**. Werden Funktionswerte und Ableitungen interpoliert, spricht man von ***Hermite*-Interpolation**. Beide Interpolationsaufgaben werden im Folgenden behandelt.

2.2.1 Lagrange-Interpolation

Gegeben seien $(n+1)$ diskrete, paarweise verschiedene **Stützstellen** (auch **Knoten** genannt) $x_0, \ldots, x_n$ und dazugehörige beliebige **Stützwerte** $f_0, \ldots, f_n$.

Gesucht ist nun ein Polynom $P \in \mathbb{P}_n$ vom Grad grad $p \leq n$, d.h.

$$P(x) = a_n x^n + \ldots + a_1 x + a_0, \qquad \text{mit } a_\nu \in \mathbb{R}, \nu = 0, \ldots, n,$$

welches die **Interpolationsbedingungen**

$$P(x_i) = f_i, \qquad i = 0, \ldots, n \tag{2.3}$$

erfüllt.

Sei $\varphi_j \equiv x^j, j = 0, \ldots, n$ die **monomiale Basis** des $\mathbb{P}_n$. Für (2.3) erhalten wir $\mu_i(f) := f(x_i)$ also $\mu_i(\varphi_j) = x_i^j$. Also ist die Matrix $(\mu_i(\varphi_j))_{i,j=0}^n$ die Vandermonde-Matrix, die bekanntermaßen invertierbar ist, falls die x_i paarweise disjunkt sind.

Um eines solches Polynoms $P(x)$ auch explizit zu konstruieren, werden wir zunächst die sogenannte *Lagrange*-Darstellung von Polynomen einführen. Mit dieser Darstellung kann man besonders schön zeigen, dass man immer ein Polynom konstruieren kann, welches die Interpolationsaufgabe erfüllt.

2.2.1.1 *Lagrange*-Darstellung

Zur Konstruktion der *Lagrange*-Darstellung von Polynomen benötigen wir die folgende Definition:

Definition 2.2.1 (*Lagrange*-Polynome) *Die Polynome*

$$L_i^{(n)}(x) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \in \mathbb{P}_n, \quad i = 0, 1, \ldots, n. \tag{2.4}$$

zu den $(n+1)$ diskreten, paarweise verschiedenen Knoten $x_0, \ldots, x_n$ werden **Lagrange-*Basispolynome*** *genannt.*

Bemerkung 2.2.2 (Basis-Darstellung) *Als endlich dimensionalen Vektorraum kann man die Elemente des Vektorraums $\mathbb{P}_n$ der Polynome vom Grad kleiner oder gleich n, d.h., die Polynome, eindeutig als Linearkombination endlicher, linear unabhängiger Teilmengen—einer Basis–darstellen. Wie man sich leicht überlegt, ist die Menge der* Lagrange*-Polynome $\{L_i^{(n)}, i = 0, \ldots, n\}$ eine Basis des $\mathbb{P}_n$ (Übungsaufgabe).*

Bemerkung 2.2.3 (Eigenschaft der *Lagrange*-Polynome) *Per Konstruktion erfüllen die* Lagrange*-Polynome die Knotenbedingung*

$$L_i^{(n)}(x_k) = \delta_{ik} = \begin{cases} 1 & \text{für } i = k, \\ 0 & \text{für } i \neq k. \end{cases} \tag{2.5}$$

Hieraus folgt, das $L_i^{(n)}, i = 0, \ldots, n$ linear unabhängig sind.

Definition 2.2.4 (Lagrange-Darstellung) *Das Polynom*

$$P(x) := \sum_{i=0}^{n} f_i L_i^{(n)}(x) \tag{2.6}$$

wird ***Lagrange-Darstellung*** *des Interpolationspolynoms zu den Stützstellen $(x_0, f_0), \ldots, (x_n, f_n)$ genannt.*

Wie angekündigt, liefert die *Lagrange*-Darstellung einen konstruktiven Beweis für die Existenz und Eindeutigkeit eines Polynoms $P(x)$, das die Interpolationsbedingung (2.3) erfüllt.

Satz 2.2.5 (Existenz und Eindeutigkeit der *Lagrange*-Interpolation) *Zu beliebigen $(n+1)$ Paaren (x_i, f_i), $i = 0, \ldots, n$, mit paarweise verschiedenen Stützstellen $x_0, \ldots, x_n$ existiert genau ein Interpolationspolynom $P \in \mathbb{P}_n$, das (2.3) erfüllt.*

Beweis. **(Existenz:)** Die Existenz des Interpolationspolynoms $P(x)$ zeigen wir auf konstruktive Art. Zu diesem Zweck betrachten wir zu den gegebenen Stützstellen die $(n+1)$ ***Lagrange*-Polynome** $L_i^{(n)}, i = 0, \ldots, n$, aus (2.4). Diese Polynome sind vom echten Grad n und besitzen offensichtlich die Eigenschaft (2.5). Demzufolge besitzt das Polynom

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

wegen (2.5) die geforderten Interpolationseigenschaften:

$$P(x_k) = \sum_{i=0}^{n} f_i L_i^{(n)}(x_k) = \sum_{i=0}^{n} f_i \delta_{ik} = f_k, \quad k = 0, 1, \ldots, n.$$

Ferner ist als Linearkombination von Polynomen vom Grad n der Grad von P kleiner oder gleich n.

(Eindeutigkeit:) Die Eindeutigkeit des Interpolationspolynoms ergibt sich wie folgt: Es seien P und Q zwei Polynome jeweils vom Grad höchstens gleich n, $P, Q \in \mathbb{P}_n$ mit

$$P(x_k) = Q(x_k) = f_k \quad (k = 0, 1, \ldots, n). \tag{2.7}$$

Aus dieser Eigenschaft (2.7) folgt, dass $D(x) := P(x) - Q(x)$ ein Polynom vom Grad kleiner oder gleich n definiert mit den $(n+1)$ paarweise verschiedenen Nullstellen $x_0, \ldots, x_n$. Nach dem Fundamentalsatz der Algebra muss nun aber $D(x) \equiv 0$ gelten, also $P(x) = Q(x)$ sein. □

Bemerkung 2.2.6 (Vor- und Nachteile der *Lagrange*-Darstellung) *Die* Lagrange*-Darstellung (2.6) ist für praktische Zwecke meist zu rechenaufwendig, sie eignet sich jedoch hervorragend für theoretische Fragestellungen. Insbesondere, müssen die Lagrange-Polynome bei Hinzunahme einer weiteren Stützstelle (x_{n+1}, f_{n+1}) komplett neu berechnet werden.*

Im Laufe dieses Abschnitts werden wir eine weitere Basis Darstellung kennenlernen, welche sich besser zur numerischen Berechnung eignet, als die *Lagrange*-Darstellung, dies ist die so genannte *Newton*-Darstelleung. Zunächst aber zur kanonischen Basis Darstellung.

2.2.1.2 Monomiale Basis Darstellung

Eine alternative Basis zur Darstellung des Interpolationspolynoms mit Lagrange-Polynomen bildet die sogenannte **monomiale Basis** $\{1, \ldots, x^n\}$ von $\mathbb{P}_n$, d.h. wir schreiben P in folgender Koeffizientendarstellung

$$P(x) = a_0 + a_1 x + \ldots + a_n x^n.$$

Bemerkung 2.2.7 (*Vandermonde*-Matrix) *Die Interpolationsbedingungen $P(x_i) = f_i$, $i = 0, \ldots, n$ lassen sich wie bereits bemerkt als folgendes lineares Gleichungssystem auffassen:*

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

In Numerik 1 haben wir diese Vandermonde*-Matrix schon kennengelernt und an dortiger Stelle auch eine zum* Gauß*-Algorithmus alternative Möglichkeit angegeben um das dazugehörige lineare Gleichungssystem zu lösen (Aufwand für dieses spezielle Verfahren $5n^2/2+\mathcal{O}(n)$). Das Problem der schlechten Kondition hatten wir auch angesprochen.*

Bemerkung 2.2.8 (Nachteil monomiale Basis Darstellung) *Die Darstellung in monomialer Basis ist numerisch instabil und sollte daher für große n im Allgemeinen nicht verwendet werden.*

2.2.2 Hermite-Interpolation

Um die *Hermite*-Interpolation einzuführen, bei der neben den Funktionswerten $f(x_i)$ auch die Ableitungen gegeben sind, benötigen wir noch einige Notationen. Zunächst sei angemerkt, dass generell Ableitungen verschiedener Ordnung an unterschiedlichen Knoten gegeben sein können.

Notation: Wir definieren die Folge von Stützstellen $\triangle := \{x_j\}_{j=0,\dots,n}$ mit

$$a = x_0 \le x_1 \le \cdots \le x_n = b,$$

wobei Stützstellen auch mehrfach auftreten können. Sind an einer Stelle x_i einerseits der Funktionswert $f(x_i)$ und andererseits die Ableitungen $f'(x_i), \dots, f^{(k)}(x_i)$ gegeben, so soll x_i in obiger Folge $(k+1)$-mal auftreten.

Gleiche Knoten nummerieren wir hierbei mit der Vielfachheit

$$d_i := \max\{j \in \mathbb{N} \mid x_i = x_{i-j}\}$$

von links nach rechts durch; z.B.

x_i	$x_0 = x_1$	$<$	$x_2 = x_3 = x_4$	$<$	x_5	$<$	x_6
d_i	0	1	0	1	2	0	0

Führen wir nun mit diesen Abkürzungen nachfolgende lineare Abbildung

$$\mu_i : C^n[a,b] \to \mathbb{R}, \quad \mu_i(f) := f^{(d_i)}(x_i), \quad i = 0, \dots, n$$

ein, so lautet die **Aufgabe der *Hermite*-Interpolation:** Gegeben μ_i $(i = 0, \dots, n)$. Finde $P \in \mathbb{P}_n$ mit

$$\mu_i(P) = \mu_i(f), \quad i = 0, \dots, n. \tag{2.8}$$

Die Lösung $P = P(f|x_0, \dots, x_n) \in \mathbb{P}_n$ von (2.8) heißt ***Hermite*-Interpolierende**.

Satz 2.2.9 (Existenz und Eindeutigkeit der *Hermite*-Interpolation) *Zu jeder Funktion $f \in C^n[a,b]$ und jeder monotonen Folge*

$$a = x_0 \le x_1 \le \ldots \le x_n = b$$

von (i.Allg. nicht paarweise verschiedenen) Knoten gibt es genau ein Polynom $P \in \mathbb{P}_n$, sodass gilt:

$$\mu_i(P) = \mu_i(f), \quad i = 0, \dots, n.$$

Beweis. Die Abbildung

$$\mu : \mathbb{P}_n \to \mathbb{R}^{n+1}, \; P \mapsto (\mu_0(P), \dots, \mu_n(P))$$

ist offensichtlich eine lineare Abbildung zwischen den $(n+1)$-dimensionalen reellen Vektorräumen $\mathbb{P}_n$ und $\mathbb{R}^{n+1}$, sodass aus der Injektivität der Abbildung bereits die Surjektivität folgen würde

und damit der Satz bewiesen wäre. Somit reicht es die Injektivität der linearen Abbildung zu zeigen.

Da $\mu(P) = 0$ gilt, folgt, dass p mindestens $(n+1)$-Nullstellen inklusive Vielfachheiten besitzt, somit aber das Nullpolynom ist. Da ferner $dim\, \mathbb{P}_n = dim\, \mathbb{R}^{n+1} = n+1$, folgt daraus auch wieder die Existenz. □

Bemerkung 2.2.10 (Eigenschaften der *Hermite*-Interpolation) *Stimmen alle Knoten überein, d.h. $x_0 = x_1 = \cdots = x_n$, dann entspricht das Interpolationspolynom der abgebrochenen Taylor-Reihe um $x = x_0$:*

$$P(f|x_0, \ldots, x_n)(x) := \sum_{j=0}^{n} \frac{(x - x_0)^j}{j!} f^{(j)}(x_0).$$

Die *Hermite*-Interpolation und insbesondere deren effiziente Berechnung sind Thema des nun folgenden Abschnitts. Die dort vorgestellten Schemata zur effizienten numerischen Auswertung sind, wie wir an einem Beispiel sehen werden, allerdings natürlich auch im Fall der *Lagrange*-Interpolation anwendbar.

2.2.3 Auswertung von Polynomen

In diesem Abschnitt werden wir uns unter anderem mit der effizienten Auswertung des so gemannten Interpolationspolynoms beschäftigen:

Definition 2.2.11 (Interpolationspolynom) *Das nach Satz 2.2.5 eindeutig bestimmte Polynom $P \in \mathbb{P}_n$ heißt **Interpolationspolynom** von f zu den paarweise verschiedenen Stützstellen $x_0, \ldots, x_n$ und wird mit*

$$P = P(f|x_0, \ldots, x_n)$$

bezeichnet.

Zur Berechnung kommen je nach Fragestellung zwei unterschiedliche Ansätze zum Einsatz:

- Schema von *Aitken*[1] und *Neville*[2],
- Schema der dividierten Differenzen.

Ist man nur an der Auswertung des Interpolationspolynoms P an einer Stelle x (oder ganz wenigen) interessiert und will das Interpolationspolynom selber nicht explizit ausrechnen so verwendet man das Schema von *Aitken* und *Neville*.

Will man hingegen das Interpolationspolynom an mehreren Stellen auswerten, so wendet man das Schema der dividierten Differenzen an. Hierbei berechnet man die Koeffezienten $a_n = [x_0, \ldots, x_n]f$ des so genannten Newtonschen-Interpolationspolynoms (die dividierten Differenzen) und wendet ein dem Horner-Schema ähnliches Schema an.

Das *Newton*sche-Interpolationspolynom ist hierbei eine Darstellung des Interpolationspolynoms bezüglich der so genannten *Newton*-Basis. Aus dem Blickwinkel der Darstellung des Interpolationspolynoms bzgl. der *Newton*-Basis entspricht das Schema von *Aitken* und *Neville* der Berechnung des Wertes des Interpolationspolynoms an einer gesuchten Stelle ohne die Koeffizienten a_n explizit zu berechnen.

[1] Aitken, Alexander Craig (1895-1967)
[2] Neville, Eric Harold (1889-1961)

2.2.3.1 Schema von *Aitken* und *Neville*

Hier betrachen wir den Fall, dass man nur an der Auswertung des Interpolationspolynoms P an einer Stelle x interessiert ist. Dazu muss man nicht erst P bestimmen, sondern kann $P(x)$ durch rekursive Berechnung effektiver (d.h vor allem effektiver bezüglich des Aufwands) bestimmen. Motiviert wird dies im Folgenden durch das Lemma von *Aitken*.

Lemma 2.2.12 (von Aitken) *Für das Interpolationspolynom $P = P(f|x_0, \ldots, x_n)$ gilt die Rekursionsformel*

$$P(f|x_0, \ldots, x_n)(x) = \frac{(x_0 - x)P(f|x_1, \ldots, x_n)(x) - (x_n - x)P(f|x_0, \ldots, x_{n-1})(x)}{x_0 - x_n}. \quad (2.9)$$

Hierbei gilt also insbesondere $P(f|x_k) = f(x_k)$.

Beweis. Sei $\phi(x)$ definiert als der Term auf der rechten Seite von (2.9). Dann ist $\phi \in \mathbb{P}_n$ und es gilt:

$$\phi(x_i) = \frac{(x_0 - x_i)f(x_i) - (x_n - x_i)f(x_i)}{x_0 - x_n} = f(x_i), \quad i = 1, \ldots, n-1.$$

Ebenso leicht folgt $\phi(x_0) = f(x_0)$ sowie $\phi(x_n) = f(x_n)$ und daher obige Behauptung. □

Herleitung des Algorithmus: Wir definieren $f_i := f(x_i)$ für $i = 0, \ldots, n$. Man beachte hierbei

$$P(f|x_i) = f_i, \quad i = 0, \ldots, n.$$

Für festes x vereinfachen wir die Notation weiterhin durch

$$P_{i,k} := P(f|x_{i-k}, \ldots, x_i)(x), \quad i \geq k.$$

Die Rekursion (2.9) schreibt sich nun

$$P_{n,n} = \frac{(x_0 - x)P_{n,n-1} - (x_n - x)P_{n-1,n-1}}{x_0 - x_n},$$

oder allgemeiner für $P_{i,k}, i \geq k$:

$$\begin{aligned} P_{i,k} &= \frac{\overbrace{(x_{i-k} - x)}^{=(x_{i-k}-x_i+x_i-x)} P_{i,k-1} - (x_i - x)P_{i-1,k-1}}{x_{i-k} - x_i} \\ &= P_{i,k-1} + \frac{x_i - x}{x_{i-k} - x_i}(P_{i,k-1} - P_{i-1,k-1}). \end{aligned}$$

Daraus gewinnen wir den folgenden Algorithmus:

Algorithmus 2.2.1 (*Aitken-Neville*-Verfahren) *Gegeben seien Stützstellen $x_0, x_1, \ldots, x_n$ und Stützwerte $f_0, f_1, \ldots, f_n$.*

1) *Setze $P_{i,0} = f_i$, für i=0,..., n.*

2) *Berechne $P_{i,k} = P_{i,k-1} + \frac{x - x_i}{x_i - x_{i-k}}(P_{i,k-1} - P_{i-1,k-1})$, $j \geq k$.*

Zur praktischen Berechnung eignet sich das **Schema von *Neville***: Nach diesem lässt sich $P_{n,n} = P(f|x_0, \ldots, x_n)(x)$, d.h. das Interpolationspolynom an einer festen Stelle x, ausgehend von den Daten $(f_0 \ldots, f_n)$ wie folgt berechnen:

$$
\begin{array}{ccccccccccc}
f_0 & = & P_{0,0} & & & & & & & & \\
 & & & \searrow & & & & & & & \\
f_1 & = & P_{1,0} & \to & P_{1,1} & & & & & & \\
\vdots & & \vdots & & \vdots & & & & & & \\
\vdots & & \vdots & & & & & & & & \\
 & & & & & \searrow & & & & & \\
\vdots & & \vdots & & \vdots & & \to & P_{n-2,n-2} & & & \\
 & & & & & \searrow & & \searrow & & & \\
f_{n-1} & = & P_{n-1,0} & \to & \cdots & \to & P_{n-1,n-2} & \to & P_{n-1,n-1} & & \\
 & & & \searrow & & \searrow & & \searrow & & \searrow & \\
f_n & = & P_{n,0} & \to & P_{n,1} \cdots & \to & P_{n,n-2} & \to & P_{n,n-1} & \to & P_{n,n}
\end{array}
$$

MATLAB-Funktion: AitkenNeville.m

```
function value = AitkenNeville(x,fx,x0)
% evaluate the Interpolation polynomial given
% by (x,fx) at the point x0
for k = 2:length(fx)
  for j = length(fx):-1:k
    fx(j) = fx(j) + (x0-x(j))/(x(j)-x(j-k+1)) * (fx(j)-fx(j-1));
  end
end
value = fx(end);
```

MATLAB-Beispiel:

Testen wir das *Aitken-Neville*-Verfahren anhand zweier Beispiele. Zum einen werten wir das Interpolationspolynom zu $f(x) = x^2$ und 3 Stützstellen an der Stelle 2 aus.

```
>> x = linspace(0,1,5);
>> AitkenNeville(x,x.^2,2)
ans =
     4
```

Zum anderen werten wir das Interpolationspolynom zu $f(x) = \sin(x)$ und 5 äquidistanten Stützstellen in $[0, 1]$ an der Stelle $\pi/3$ aus. Man beachte, dass $\sin(x) = \sqrt{3}/2$ gilt.

```
>> x = linspace(0,1,5);
>> sqrt(3)/2 - AitkenNeville(x,sin(x),pi/3)
ans =
    4.387286117690792e-005
```

Bemerkung 2.2.13 *Im Gegensatz zur Lagrange-Darstellung lässt sich bei der Berechnung des Interpolationspolynoms an einer festen Stelle mit dem Schema von Aitken-Neville zu $(n+1)$-Paaren ohne Probleme eine weiteres Paar (x_{n+1}, f_{n+1}) hinzu nehmen. Und dies somit zu einer Berechnung des Interpolationspolynoms an einer festen Stelle zu $(n+2)$ Paaren ausweiten.*

2.2.3.2 Newton-Darstellung und dividierte Differenzen

Wir hatten bereits zwei Basis-Darstellungen für $\mathbb{P}_n$ kennengelernt. Allerdings eigenen sich sowohl die *Lagrange*-Darstellung als auch die monomiale Basis Darstellung nicht so gut zur numerischen Berechnung des kompletten Polynoms (also nicht nur der Auswertung an wenigen Stellen). Vor diesem Hintergrund führen wir die *Newton*schen Basispolynome ein, eine Darstellung bzgl. dieser ist für numerische Zwecke besser geeignet.

Das Ziel ist eine Aufdatierungsstrategie bezüglich des Polynomgrades zur Berechnung des kompletten Polynoms. Mit einer Aufdatierungsstrategie kann man dann auch ohne Probleme ein weiteres Paar (x_{n+1}, f_{n+1}) hinzunehmen.

Motivation: Zu gegebenen $(n+1)$ Paaren ist wollen wir das Interpolationspolynom $P(f|x_0,\dots,x_n) \in \mathbb{P}_n$ somit als eine additive Zerlegung

$$\underbrace{P(f|x_0,\dots,x_n)(x)}_{\in\mathbb{P}_n} = \underbrace{P(f|x_0,\dots,x_{n-1})(x)}_{\in\mathbb{P}_{n-1}} + \underbrace{Q_n(x)}_{\in\mathbb{P}_n}$$

darstellen, mit einem Polynom Q_n vom Grad n, welches von den Stützstellen x_i und einem unbekannten Koeffizienten abhängt. Da für

$$Q_n(x_i) = P(f|x_0,\dots,x_n)(x_i) - P(f|x_0,\dots,x_{n-1})(x_i) = 0$$

gilt, muss

$$Q_n(x) = a_n(x-x_0)\cdots(x-x_{n-1})$$

gelten. Für x_n setzt man ein und löst nach a_n auf

$$a_n = \frac{f(x_n) - P(f|x_0,\dots,x_{n-1})(x_n)}{(x_n-x_0)\cdots(x_n-x_{n-1})}$$

Offenbar ist a_n der führende Koeffizient von $P(f|x_0,\dots,x_n)$, d.h. der Koeffizient vor x^n.

Wir definieren nun die bereits erwähnte Newton-Basis:

Definition 2.2.14 (*Newton*-Basis) *Es seien $x_0,\dots x_{n-1} \in \mathbb{R}$ und*

$$\omega_0 := 1, \quad \omega_i(x) := \prod_{j=0}^{i-1}(x-x_j), \quad \omega_i \in \mathbb{P}_i.$$

Wir bezeichnen $\{\omega_0,\dots,\omega_n\}$ als **Newton-*Basis*** *des Polynomraums $\mathbb{P}_n$ mit Basiselementen ω_i.*

Bemerkung 2.2.15 *Man beachte, dass bei der Definition der Newton-Basis weder eine Ordnung der Punkte x_k noch „paarweise verschieden“ vorgeschrieben wurde. Je nach Nummerierung, erhält man somit eine andere Newton-Basis. Dies ist bei der Stabilität der folgenden Verfahren zu berücksichtigen.*

Um nun das Verfahren der dividierten Differenzen herzuleiten, mit dem sich sowohl die *Lagrange*- als auch die *Hermite*-Interpolationsaufgabe effizient lösen lässt, verwenden wir die Darstellung des Interpolationspolynoms in der *Newton*-Basis, d.h. das ***Newton*-Interpolationspolynom**

$$\begin{aligned} P(x) &= c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + \cdots + c_n \prod_{j=0}^{n-1}(x-x_j) \\ &= \sum_{i=0}^{n} a_i \omega_i(x), \end{aligned}$$

wobei sich die unbekannten Koeffizienten $c_0, \ldots, c_n$ prinzipiell aus den Interpolationsbedingungen

$$\begin{array}{rcll} P(x_0) &=& c_0 & = f(x_0) \\ P(x_1) &=& c_0 + c_1(x-x_0) & = f(x_1) \\ P(x_2) &=& c_0 + c_1(x_2-x_0) + c_2(x_2-x_1)(x_2-x_0) & = f(x_2) \\ \vdots & & \vdots & \end{array}$$

sukzessive berechnen lassen (dieses LGS hat Linksdreiecksgestalt!):

$$\begin{aligned} P(x_0) = c_0 &\quad\Longrightarrow\quad c_0 = f(x_0), \\ P(x_1) = c_0 + c_1(x_1 - x_0) &\quad\Longrightarrow\quad c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \end{aligned}$$

Die Koeffizienten $c_k, k = 0, \ldots, n$ nennt man **dividierte Differenzen**.

Definition 2.2.16 (n-te dividierte Differenz) *Der führende Koeffizient a_n des Interpolationspolynoms*

$$P(f|x_0, \ldots, x_n)(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

*von f zu den Knoten $x_0 \le x_1 \le \cdots \le x_n$ heißt **n-te dividierte Differenz** von f an $x_0, \ldots, x_n$ und wird mit*

$$[x_0, \ldots, x_n]f := a_n$$

bezeichnet.

Bemerkung 2.2.17 *Beim Vergleich des* Newton-*Interpolationspolynoms mit dem Interpolationspolynom bzgl. der monomialen Basis*

$$P(x) = \sum_{j=0}^{n} c_i \prod_{i=0}^{j-1}(x - x_i) = \sum_{j=0}^{n} a_j x^j$$

sieht man durch Ausmultiplizieren, dass für die Koeffizienten der höchsten $x-$Potenz $a_n = c_n$ gilt. Damit folgt $a_n = c_n = [x_0, \ldots, x_n]f$

Satz 2.2.18 (*Newton*'sche Interpolationsformel) *Für jede Funktion $f \in C^n(\mathbb{R})$ und Knoten $x_0 \le \cdots \le x_n \in \mathbb{R}$ ist*

$$P(x) = \sum_{i=0}^{n} [x_0, \ldots, x_i]f \cdot \omega_i(x)$$

das Interpolationspolynom $P(f|x_0, \ldots, x_n)$ von f an $x_0, \ldots, x_n$ also $P(f|x_0, \ldots, x_n) = P(x)$. Gilt darüber hinaus $f \in C^{n+1}(\mathbb{R})$, so folgt:

$$f(x) = P(x) + [x_0, \ldots, x_n, x]f \cdot \omega_{n+1}(x)\,. \tag{2.10}$$

Beweis. Wir zeigen die erste Behauptung durch Induktion nach $n \in \mathbb{N}$. Für $n = 0$ ist die Aussage trivialerweise erfüllt. Sei also im Folgenden $n > 0$ und

$$P_{n-1} := P(f|x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} [x_0, \ldots, x_i]f \cdot \omega_i$$

das Interpolationspolynom von f an den Stützstellen $x_0, \ldots, x_{n-1}$. Damit erhalten wir für $P_n = P(f|x_0, \ldots, x_n)$, aufgrund der Definition der dividierten Differenz bzw. von P_n, dass

$$\begin{aligned} P_n(x) &= [x_0, \ldots, x_n]f \cdot x^n + a_{n-1}x^{n-1} + \ldots + a_0 \\ &= [x_0, \ldots, x_n]f \cdot \omega_n(x) + Q_{n-1}(x) \end{aligned}$$

mit einem Polynom $Q_{n-1} \in \mathbb{P}_{n-1}$ gilt. Nun erfüllt aber wegen $\omega_n(x_i) = 0, i = 0, \ldots, n-1$

$$Q_{n-1} = P_n - [x_0, \ldots, x_n]f \cdot \omega_n$$

offensichtlich die Interpolationsaufgabe für $x_0, \ldots, x_{n-1}$, sodass wir erhalten:

$$Q_{n-1} = P_{n-1} = \sum_{i=0}^{n-1} [x_0, \ldots, x_i]f \cdot \omega_i.$$

Dies beweist aber gerade die Aussage des Satzes. Insbesondere folgt nun, dass

$$P_n + [x_0, \ldots, x_n, x]f \cdot \omega_{n+1}$$

die Funktion f an den Knoten $x_0, \ldots, x_n$ und x interpoliert und damit (2.10). □

Aus den Eigenschaften der *Hermite*-Interpolation lassen sich sofort folgende Aussagen über die dividierten Differenzen zu f ableiten:

Lemma 2.2.19 (Eigenschaften der dividierten Differenzen) *i) **(Rekursive Berechnung der dividierten Differenzen)** Für $x_i \neq x_k$ gilt die Rekursionsformel*

$$[x_0, \ldots, x_n]f = \frac{[x_0, \ldots, \widehat{x}_i, \ldots, x_n]f - [x_0, \ldots, \widehat{x}_k, \ldots, x_n]f}{x_k - x_i},$$

wobei $\widehat{\ }$ anzeigt, dass die entsprechende Stützstelle weggelassen wird ('seinen Hut nehmen muss')

*ii) **(Darstellung über abgebrochene Taylor-Reihe)** Für zusammenfallende Knoten $x_0 = \ldots = x_n$ gilt*

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

Bemerkung 2.2.20 *Aussage i) erklärt die Bezeichnung dividierte Differenzen.*

Beweis. Für das *Hermite*-Interpolationspolynom gilt mit $x_i \neq x_k$:

$$P(f|x_0, \ldots, x_n) = \frac{(x_i - x)\overbrace{P(f|x_0, \ldots, \widehat{x}_k, \ldots, x_n)}^{[x_0, \ldots, \widehat{x}_i, \ldots, x_n]f\, x^{n-1} + \mathbb{P}_{n-2}} - (x_k - x)P(f|x_0, \ldots, \widehat{x}_k, \ldots, x_n)}{x_i - x_k}, \tag{2.11}$$

was sich durch Überprüfen der Interpolationseigenschaft zeigen lässt mittels Einsetzen der Definitionen. Aus der Eindeutigkeit des führenden Koeffizienten folgt aus (2.11) unmittelbar Behauptung $i)$. Stimmen dagegen alle Knoten $x_0, \ldots, x_n$ überein, so ist das Interpolationspolynom

$$P(f|x_0, \ldots, x_n)(x) = \sum_{j=0}^{n} \frac{(x - x_0)^j}{j!} f^{(j)}(x_0) =: P(x),$$

wie man durch Einsetzen in μ_i (vgl. (2.8) oben) leicht einsieht. Daraus folgt, dass der führende Koeffizient $f^{(n)}(x_0)/n!$ ist. Somit gilt auch $ii)$. □

Die Auswertung der Rekursionsformel (2.11) erfolgt am zweckmäßigsten im **Schema der dividierten Differenzen** unter Verwendung der Startwerte $[x_i]f = f(x_i)$ für paarweise verschiedene Knoten.

$$
\begin{array}{lllll}
x_0 & [x_0]f & & & \\
x_1 & [x_1]f & [x_0,x_1]f & & \\
x_2 & [x_2]f & [x_1,x_2]f & [x_0,x_1,x_2]f & \\
x_3 & [x_3]f & [x_2,x_3]f & [x_1,x_2,x_3]f & [x_0,x_1,x_2,x_3]f \\
x_4 & [x_4]f & [x_3,x_4]f & [x_2,x_3,x_4]f & [x_1,x_2,x_3,x_4]f \quad [x_0,\ldots,x_4]f
\end{array}
$$

Die gesuchten Koeffizienten a_k des *Newton*-Interpolationspolynoms findet man im obigen Schema der dividierten Differenzen in der oberen Diagonalen.

Nachfolgend werden wir das Schema der dividierten Differenzen anhand zweier Beispiele illustrieren. Zum einen für die klassische *Lagrange*-Interpolation bei der Knoten und Funktionswerte gegeben sind und zum anderen für die *Hermite*-Interpolation.

Beispiel 2.2.21 (*Lagrange*-Interpolation via dem Schema der dividierten Differenzen) Wenn wir das Schema auf gegebene Daten $(x_i, f_i), i = 0,\ldots,3$ anwenden, so erhalten wir

$$
\begin{array}{lllllllll}
x_0 = 0 & : & f_0 = 1 & & & & & & \\
 & & & \searrow & & & & & \\
x_1 = 3/2 & : & f_1 = 2 & \to & 2/3 & & & & \\
 & & & \searrow & & \searrow & & & \\
x_2 = 5/2 & : & f_2 = 2 & \to & 0 & \to & -4/15 & & \\
 & & & \searrow & & \searrow & & \searrow & \\
x_3 = 9/2 & : & f_3 = 1 & \to & -1/2 & \to & -1/6 & \to & 1/45
\end{array}
$$

und das *Newtonsche* Interpolationspolynom lautet demnach

$$P(x) = 1 + \frac{2}{3}(x-0) - \frac{4}{15}(x-0)(x-\frac{3}{2}) + \frac{1}{45}(x-0)(x-\frac{3}{2})(x-\frac{5}{2})$$

MATLAB-Funktionen: NewtonInterpolation.m und EvalNewtonPoly.m

```
function fx = NewtonInterpolation(x,fx)
for k = 2:length(fx)
  for j = length(fx):-1:k
    fx(j) = (fx(j) - fx(j-1))/(x(j)-x(j-k+1));
  end
end
```

```
function value = HornerNewton(a,x0,x)
% evaluate Newton polynom
% p(x0) = a(1) + a(2)*(x0-x(1)) + a(3)*(x0-x(1))*(x0-x(2)) + ...
%       = a(1) + ( (x0-x(1)) * ( a(2) + (x0-x(2)) * ( a(3) ....
value = a(end);
for k=length(a)-1:-1:1
  value = a(k) + value.*(x0-x(k));
end
```

MATLAB-Beispiel:

Testen wir das Differenzen-Verfahren nochmals an den beiden Beispielen aus Bsp. 2.2.3.1. Zum einen werten wir das Interpolationspolynom zu $f(x) = x^2$ und 3 Stützstellen an der Stelle 2 aus.

```
>> x=linspace(0,2,3);
>> a = NewtonInterpolation(x,x.^2)
a =
     0     1     1
>> HornerNewton(a,2,x)
ans =
     4
```

Zum anderen werten das Interpolationspolynom zu $f(x) = \sin(x)$ und 5 äquidistanten Stützstellen in $[0, 1]$ an der Stelle $\pi/3$ aus.

```
>> x=linspace(0,1,5);
>> a = NewtonInterpolation(x,sin(x));
>> sqrt(3)/2-HornerNewton(a,pi/3,x)
ans =
    4.387286117690792e-005
```

Beispiel 2.2.22 (*Hermite*-Interpolation via dem Schema der dividierten Differenzen)
Betrachten wir nun noch abschließend das Schema für die nichttriviale *Hermite*-Interpolationsaufgabe (d.h. auch Ableitungen werden interpoliert). Unter Beachtung von Lemma 2.2.19, insbesondere (2.11) ergibt sich:

$$\begin{array}{lllllll}
x_0: & [x_0]f & & & & & \\
 & & \searrow & & & & \\
x_0: & [x_0]f & \to & [x_0,x_0]f = f'(x_0) & & & \\
 & & \searrow & & \searrow & & \\
x_0: & [x_0]f & \to & [x_0,x_0]f = f'(x_0) & \to & [x_0,x_0,x_0]f = \tfrac{f''}{2}(x_0) & \\
 & & \searrow & & \searrow & & \searrow \\
x_1: & [x_1]f & \to & [x_0,x_1]f & \to & [x_0,x_0,x_1]f & \to \quad [x_0,x_0,x_0,x_1]f
\end{array}$$

Das resultierende Polynom $P(x)$ mit

$$P(x) = f(x_0) + (x-x_0)\Big(f'(x_0) + (x-x_0)\big(f''(x_0) + (x-x_0)[x_0,x_0,x_0x_1]f\big)\Big)$$

erfüllt dann die Interpolationsaufgaben

$$P(x_0) = f(x_0), P'(x_0) = f'(x_0), P''(x_0) = f''(x_0) \text{ und } P(x_1) = f(x_1).$$

Für den speziellen Fall, dass an allen Knoten $x_0, \ldots, x_n$ sowohl f als auch f' interpoliert werden sollen, erhält man die folgende Darstellung des Interpolationspolynoms $P(x)$.

Satz 2.2.23 (Darstellung des Interpolationspolynoms) *Es sei $\omega(x) = (x-x_0)\cdot\cdots\cdot(x-x_n)$ und $L_k^{(n)}(x)$ seien die* Lagrange*-Polynome zu den Knoten $x_0, \ldots, x_n$. Dann hat*

$$P(x) = \sum_{k=1}^{n} f(x_k)\left(1 - \frac{\omega''(x_k)}{\omega'(x_k)}(x-x_k)\right)L_k^{(n)}2(x) + \sum_{k=1}^{n} f'(x_k)(x-x_k)L_k^2(x)$$

die Interpolationseigenschaften

$$P(x_k) = f(x_k) \text{ und } P'(x_k) = f'(x_k), \quad k = 0, \ldots, n.$$

Beweis. Es sei x_ℓ einer der Knoten $x_0, \ldots, x_n$, dann folgt sofort aus der Interpolationseigenschaft der *Lagrange-Polynome*

$$P(x_\ell) = f(x_\ell),$$

da

$$\omega'(x) = \sum_{i=0}^{n} \prod_{\substack{k=0 \\ k\neq i}}^{n} (x - x_k) \text{ und somit } \omega'(x_\ell) = \prod_{\substack{k=0 \\ k\neq \ell}}^{n} (x_\ell - x_k) \neq 0$$

gilt. Für die Ableitung von P ergibt sich

$$\begin{aligned} P'(x) &= \sum_{k=1}^{n} f(x_k)\left[\Big(1 - \frac{\omega''(x_k)}{\omega'(x_k)}(x - x_k)\Big)2(L_k^{(n)}(x))' - \frac{\omega''(x_k)}{\omega'(x_k)} L_k^{(n)}(x)\right] L_k^{(n)}(x) \\ &\quad + \sum_{k=1}^{n} f'(x_k)\Big[(x - x_k)2(L_k^{(n)}(x))' + L_k^{(n)}(x)\Big] L_k^{(n)}(x), \end{aligned}$$

sodass wir nun Folgendes erhalten:

$$P'(x_\ell) = f(x_\ell)\Big(2(L_\ell^{(n)}(x_\ell))' - \frac{\omega''(x_\ell)}{\omega'(x_\ell)}\Big) + f'(x_\ell).$$

Nutzt man aus, dass $L_\ell^{(n)}(x) = \frac{\omega(x)}{(x-x_\ell)\omega'(x_\ell)}$ gilt (siehe Hausübung), so folgt aus

$$\omega(x) = L_\ell^{(n)}(x)(x - x_\ell)\omega'(x_\ell)$$

nach zweimaligem Differenzieren

$$\omega''(x) = (L_\ell^{(n)}(x))''(x - x_\ell)\omega'(x_\ell) + 2L_\ell'(x)\omega'(x_\ell).$$

Damit gilt an der Stelle x_ℓ

$$\frac{\omega''(x_\ell)}{\omega'(x_\ell)} = 2(L_\ell^{(n)}(x_\ell))''$$

Einsetzen in die Ableitung von P schließt den Beweis ab. □

2.2.4 Interpolationsgüte

Geht man davon aus, dass die Stützwerte f_i als Funktionswerte einer Funktion $f : [a, b] \to \mathbb{R}$ an den Stützstellen $x_i \in [a, b]$ gegeben sind, so stellt sich die Frage nach der Güte der Interpolation, d.h. wie gut approximiert das Interpolationspolynom die Funktion auf $[a, b]$.

2.2.4.1 Interpolationsfehler

Bei der nun folgenden Darstellung des Approximationsfehlers

$$\varepsilon(x) := f(x) - P(x)$$

erweisen sich die im vorherigen Abschnitt hergeleiten Eigenschaften der dividierten Differenzen als hilfreich:

Satz 2.2.24 (Interpolationsfehler) *Es sei $f \in C^n[a, b]$ und $f^{(n+1)}(x)$ existiere für alle $x \in (a, b)$; weiterhin gelte $a \leq x_0 \leq x_1 \leq \cdots \leq x_n \leq b$. Dann gilt:*

$$f(x) - P(f|x_0, \ldots, x_n)(x) = \frac{(x - x_0) \cdot \cdots \cdot (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi), \tag{2.12}$$

wobei $\min\{x, x_0, \ldots, x_n\} < \xi < \max\{x, x_0, \ldots, x_n\}$ *ist.*

Beweis. Nach Konstruktion von $P(f|x_0,\dots,x_n)$ gilt:

$$P(f|x_0,\dots,x_n)(x_k) = f(x_k), \quad k = 0,1,\dots,n.$$

Es sei nun x fest und dabei ungleich $x_0, x_1, \dots, x_n$. Ferner sei

$$K(x) := \frac{f(x) - P(f|x_0,\dots,x_n)(x)}{(x-x_0)\cdot\dots\cdot(x-x_n)}. \tag{2.13}$$

Nun betrachten wir die Funktion

$$W(t) := f(t) - P(f|x_0,\dots,x_n)(t) - (t-x_0)\cdot\dots\cdot(t-x_n)K(x). \tag{2.14}$$

Die Funktion $W(t)$ verschwindet also an den Stellen $t = x_0, \dots, t = x_n$ und durch (2.13) auch an der Stelle $t = x$. Nach dem verallgemeinerten Satz von *Rolle*[3] verschwindet die Funktion $W^{(n+1)}(t)$ an einer Stelle ξ mit $\min\{x, x_0, \dots, x_n\} < \xi < \max\{x, x_0, \dots, x_n\}$. Das $(n+1)$-fache Differenzieren von (2.14) nach t liefert

$$W^{(n+1)}(t) = f^{(n+1)}(t) - (n+1)!K(x),$$

sodass gilt:

$$0 = W^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!K(x).$$

Damit erhalten wir aber unmittelbar

$$K(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi). \tag{2.15}$$

Nach Einsetzen von (2.15) in (2.14) erhalten wir die Behauptung für $t = x$, da x Nullstelle von W ist. □

Bemerkung 2.2.25 (Darstellung des Interpolationsfehlers) *Im Beweis zum Approximationsfehler (vgl. Satz, 2.2.24* (2.12)*) haben wir gezeigt*

$$f(x) - P(f|x_0,\dots,x_n)(x) = \frac{\omega_{n+1}(x)}{(n+1)!} f^{(n+1)}(\xi), \ \min\{x_0,\dots,x_n,x\} < \xi < \max\{x_0,\dots,x_n,x\}.$$

Und mit dem Satz über die Newton*-Darstellung gilt:*

$$f(x) - P(f|x_0,\dots,x_n)(x) = [x_0,\dots,x_n,x]f \cdot \omega_{n+1}(x).$$

Somit folgern wir: Für alle Knoten $x_0 \le \dots \le x_n$ existiert ein $\xi \in [x_0, x_n]$, sodass gilt:

$$[x_0,\dots,x_n]f = \frac{f^{(n)}(\xi)}{n!} \tag{2.16}$$

2.2.4.2 *Tschebyscheff*-Interpolation

Wie das folgende Beispiel zeigen wird, hat die Verteilung der Stützstellen $x_0, \dots, x_n$ über das Interpolationsintervall entscheidenden Einfluss auf die Güte der Approximation. Ein klassisches Beispiel hierfür stammt von *Runge*[4]:

Die Interpolationspolynome $P(f|x_1,\dots,x_n)$ zur Funktion $f(x) = \frac{1}{1+x^2}$ im Intervall $I := [-5,5]$ bei äquidistanten Stützstellen $x_k = -5 + \frac{10}{n}k$ zeigen bei wachsendem n einen zunehmenden Interpolationsfehler.

[3] Rolle, Michel (1652-1719)
[4] Runge, Carl (1856-1927)

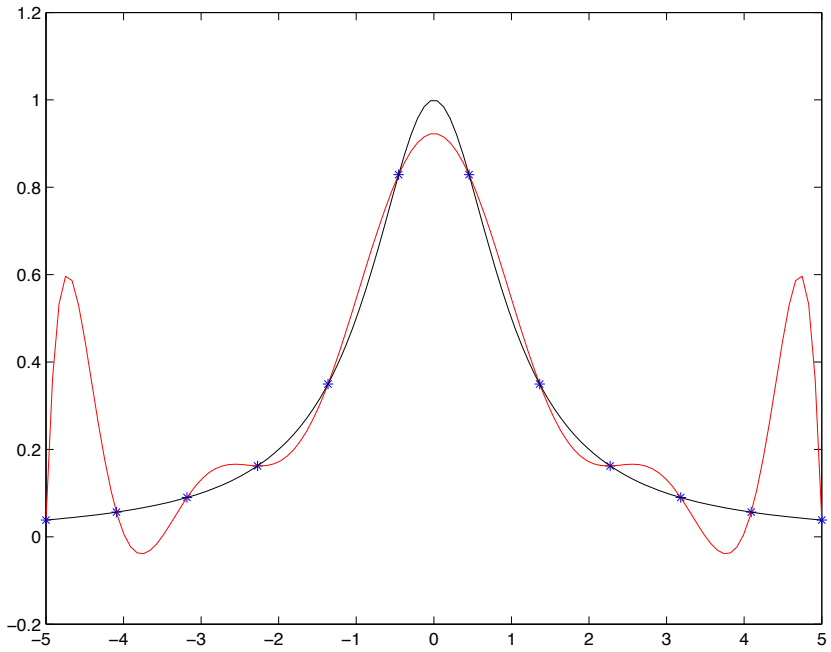
MATLAB-Beispiel:

Das Interpolationspolynom zu 12 äquidistanten Stützstellen und Stützwerten zur Funktion

$$f(x) = \frac{1}{1+x^2}$$

ist in Abb. 2.2.4.2 dargestellt.

```
n = 12;
f = @(x)  1./(x.^2+1);
x = linspace(-5,5,n);
fx = f(x);
s = linspace(x(1),x(end),10*n);
for j=1:length(s)
  ps(j) = AitkenNeville(x,fx,s(j));
end
plot(x,fx,'*',s,ps,'r-',s,f(s),'k')
```



Wie wir im weiteren zeigen werden, kann man bei geschickter, nichtäquidistanter Wahl der Stützstellen dagegen eine Konvergenz erhalten. Genauer gesagt, wählen wir $x_1, \ldots, x_n$ als die Nullstellen der von $[-1,1]$ auf I transformierten ***Tschebyscheff*-Polynome** und erhalten dadurch punktweise Konvergenz für $n \to \infty$. Man beachte das dieses Phänomen nicht von Rundungsfehlern abhängt.

Bei der Berechnung des Approximations- bzw. des Interpolationsfehlers haben wir gesehen, dass

$$f(x) - P(f|x_0, \ldots, x_n)(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x), \quad x \in [a,b]$$

für ein $\xi = \xi(x) \in (a,b)$ gilt. Wir suchen nun Knoten $x_0, \ldots, x_n \in [a,b]$, die das **Minimax-Problem**

$$\max_{x\in[a,b]} |\omega_{n+1}(x)| = \max_{x\in[a,b]} |(x-x_0)\cdot\ldots\cdot(x-x_n)| = \min$$

lösen. Anders formuliert, es gilt das normierte Polynom $\omega_{n+1} \in \mathbb{P}_{n+1}$ mit den reellen Nullstellen $x_0, \ldots, x_n$ zu bestimmen, für das

$$\max_{x\in[a,b]} |\omega_{n+1}(x)| = \min_{x_0,\ldots,x_n}$$

gilt. Im Folgenden werden wir sehen, dass gerade die *Tschebyscheff*-Polynome T_n diese obige *Minimax*-Aufgabe lösen (sie lösen sie bis auf einen skalaren Faktor und eine affine Transformation). Somit sind die Nullstellen der Tschebyscheff-Polynome (bis auf eine affine Transformation) gerade die gesuchten Stützstellen $x_0, \dots x_n$.

Zunächst reduzieren wir das Problem auf das Intervall $[-1, 1]$ mit Hilfe der Umkehrabbildung folgender Abbildung

$$y : [a, b] \to [-1, 1], \;\; x \mapsto y = y(x) = \frac{2x - a - b}{b - a},$$

d.h. die Umkehrabbildung lautet:

$$x : [-1, 1] \to [a, b], \;\; y \mapsto x = x(y) = \frac{b + a}{2} + \frac{b - a}{2} y.$$

Ist jetzt $P \in \mathbb{P}_n$ mit grad $P = n$ und führendem Koeffizienten $a_n = 1$ die Lösung des *Minimax*-problems

$$\max_{y \in [-1,1]} |P(x)| = \min,$$

so stellt $\widehat{P}(x) := P(y(x))$ die Lösung des ursprünglichen Problems mit führendem Koeffizienten $2^n/(b - a)^n$ dar. Für $y \in [-1, 1]$ definieren wir die ***Tschebyscheff*-Polynome** durch (vgl. hierzu auch Kapitel 3):

$$T_n(y) = \cos(n \arccos(y)), \;\; y \in [-1, 1] \tag{2.17}$$

und allgemein für $y \in \mathbb{R}$ durch die **Drei-Term-Rekursion**

$$T_0(y) = 1, \quad T_1(y) = y, \quad T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y), \; k \geq 2. \tag{2.18}$$

Wir benötigen im Folgenden die Eigenschaften der *Tschebyscheff* Polynome, die wir in Kapitel 3 detailliert diskutierten werden und hier der Einfachheit halber bereits schon mal wiedergeben:

Bemerkung 2.2.26 (Eigenschaften der Tschebyscheff-Polynome) (i) *Der führende Koeffizient von T_n ist $a_n = 2^{n-1}$, $n \geq 1$*

(ii) $|T_n(y)| \leq 1$ *für* $y \in [-1, 1]$.

(iii) *Die Nullstellen von $T_n(y)$ sind*

$$y_k := \cos\left(\frac{2k + 1}{2n}\pi\right), \quad k = 0, 1, \dots, n - 1.$$

(iv) $|T_n(y)|$ *nimmt seinen maximalen Wert im Intervall* $[-1, 1]$ *an den* $(n + 1)$ *Extremalstellen* $\overline{y}_k = \cos(\frac{k\pi}{n})$ *für* $k = 0, \dots, n$ *an, d.h.*

$$|T_n(y)| = 1 \quad \Leftrightarrow \quad y = \overline{y}_k = \cos\left(\frac{k\pi}{n}\right) \;\; mit \; k = 0, \dots, n.$$

Satz 2.2.27 (Minimal-Eigenschaft der Tschebyscheff-Polynome) *Jedes Polynom $P \in \mathbb{P}_n$ mit führendem Koeffizienten $a_n \neq 0$ nimmt im Intervall $[-1, 1]$ einen Wert vom Betrag $\geq |a_n|/2^{n-1}$ an. Insbesondere sind die* Tschebyscheff*-Polynome $T_n(y)$ minimal bezüglich der Maximumsnorm $\|f\|_\infty = \max_{y \in [-1,1]} |f(y)|$ unter den Polynomen vom Grad n mit führendem Koeffizienten 2^{n-1}.*

Beweis. **(Annahme:)** Sei $P \in \mathbb{P}_n$ ein Polynom mit führendem Koeffizienten $a_n = 2^{n-1}$ und $|P(y)| < 1$ für $y \in [-1,1]$. Dann ist $T_n - P_n$ ein Polynom vom Grad kleiner oder gleich $(n-1)$ (beide besitzen a_n als führenden Koeffizienten). An den *Tschebyscheff*-Abszissen $\overline{y}_k := \cos(\frac{k\pi}{n})$ gilt:

$$T_n(\overline{y}_{2k}) = 1, \;\; P_n(\overline{y}_{2k}) < 1 \;\; \Rightarrow \quad P_n(\overline{y}_{2k}) - T_n(\overline{y}_{2k}) < 0,$$
$$T_n(\overline{y}_{2k+1}) = -1, \;\; P_n(\overline{y}_{2k+1}) > -1 \;\; \Rightarrow \quad P_n(\overline{y}_{2k+1}) - T_n(\overline{y}_{2k+1}) > 0,$$

d.h. die Differenz $T_n - P_n$ ist an den $(n+1)$-*Tschebyscheff*-Abszissen abwechselnd positiv und negativ, damit besitzt die Differenz mindestens n Nullstellen in $[-1,1]$ im Widerspruch zu $0 \neq T_n - P_n \in \mathbb{P}_{n+1}$. Demnach muss es für jedes Polynom $P \in \mathbb{P}_n$ mit führendem Koeffizienten $a_n = 2^{n-1}$ ein $y \in [-1,1]$ geben derart, dass $|P_n(y)| \geq 1$ erfüllt. Für ein beliebiges $P \in \mathbb{P}_n$ mit $a_n \neq 0$ folgt die Behauptung daraus, dass $\widetilde{P}_n := \frac{2^{n-1}}{a_n} P_n$ ein Polynom mit $\widetilde{a}_n = 2^{n-1}$ ist. □

2.3 Rationale Interpolation

Zur Interpolation einer Funktion, welche einen Pol besitzt oder deren Graph eine Asymptote aufweist sind im Allgemeinen Polynome nicht gut geeignet.

Zu diesem Zweck untersuchen wir im Folgenden die Interpolation mittels einer gebrochen rationalen Funktion

$$R(x) = \frac{p_0 + p_1 x + \ldots + p_\nu x^\nu}{q_0 + q_1 x + \ldots + q_\mu x^\mu} = \frac{P_\nu(x)}{Q_\mu(x)}. \tag{2.19}$$

Die Polynomgrade ν, μ der Polynome $P_\nu \in \mathbb{P}_\nu$ und $Q_\mu \in \mathbb{P}_\mu$ seien hierbei vorgegeben, sodass $R(x)$ an $(n+1)$ paarweise verschiedenen Knoten $x_0, \ldots, x_n$ die vorgegebenen Funktionswerte $f_0 := f(x_0), \ldots, f_n := f(x_n)$ annimmt, somit also gilt:

$$R(x_i) = f_i, \quad i = 0, \ldots, n.$$

Da Zähler und Nenner in (2.19) jeweils mit einer von Null verschiedenen Zahl multipliziert werden dürfen, enthält der Ansatz (2.19) $\nu + \mu + 1$ freie Unbekannte. Damit nun die Anzahl der Interpolationsbedingungen mit der Anzahl der Unbekannten $p_0, p_1, \ldots, p_\nu, q_0, q_1, \ldots, q_\mu$ übereinstimmt, muss zwangsläufig $\nu + \mu = n$ erfüllt sein.

Beispiel 2.3.1 $x_0 = 1, x_1 = 1, x_2 = 2, f_0 = 2, f_1 = 3, f_2 = 3$

Wir setzen $\nu = 0, \mu = 2$, d.h. wir verwenden den Ansatz

$$R(x) = \frac{p_0}{q_0 + q_1 x + q_2 x^2}.$$

Die Interpolationsaufgabe führt uns zum Gleichungssystem

$$P_\nu(x_i) - Q_\mu(x_i) \cdot f(x_i) = 0,$$

d.h. im vorliegenden Fall

$$\begin{aligned} p_0 - 2(q_0 - q_1 + q_2) &= 0 \\ p_0 - 2(q_0 + q_1 + q_2) &= 0 \\ p_0 - 3(q_0 + 2q_1 + 4q_2) &= 0 \end{aligned}$$

Nach der *Cramerschen* Regel besitzt dieses lineare Gleichungssystem die einzige Lösung

$$R(x) = \frac{36}{14 - 3x + x^2}.$$

Beispiel 2.3.2 Wählen wir zu den Daten von Beispiel 2.3.1 den Ansatz $\mu = \nu = 1$, so setzen wir

$$R(x) = \frac{p_0 + p_1 x}{q_0 + q_1 x},$$

bzw.

$$p_0 + p_1 x_i - f_i(q_0 + q_1 x_i) = 0, \quad i = 0, 1, 2$$

Dies führt uns zu folgendem linearen Gleichungssystem:

$$\begin{aligned} p_0 - p_1 - 2(q_0 - q_1) &= 0, \\ p_0 + p_1 - 3(q_0 + q_1) &= 0, \\ p_0 + 2p_1 - 3(q_0 + 2q_1) &= 0, \end{aligned}$$

welches die (bis auf einen gemeinsamen Faktor) eindeutige Lösung

$$p_0 = 3, \;\; p_1 = 3, \;\; q_0 = 1, \;\; q_1 = 1$$

besitzt, sodass wir folgendes Ergebnis erhalten:

$$R(x) = \frac{3 + 3x}{1 + x} = \tilde{R} = 3 \quad \text{[Widerspruch (vgl. } R(x_0) = 2\text{)]}$$

Bemerkung 2.3.3 *Die nichttriviale Lösung des LGS zur Bestimmung von $R(x)$ braucht nicht in jedem Fall die Interpolationsaufgabe zu erfüllen. Da sowohl Zähler als auch Nenner einen gemeinsamen Linearfaktor $(x - x_j)$ besitzen können, wird sich dieser bei der Lösung wegkürzen. Die daraus resultierende gebrochen rationale Funktion $\widetilde{R}(x)$ wird dann im Allgemeinen den Funktionswert f_j an der Stelle x_j nicht annehmen (vgl. hierzu Beispiel 2.3.2).*

Bemerkung 2.3.4 *Auch für die rationale Interpolation existieren* Neville*-artige Algorithmen zur Auswertung des Polynoms, ohne dies vorher bestimmt zu haben bzw. Algorithmen zur Bestimmung der Koeffizienten, ähnlich dem dividierten Differenzen Verfahren. Man schlage z.B. in [?] unter den Stichworten* **Thielscher Kettenbruch**, **reziproke Differenzen** *nach.*

MATLAB-Funktion: RationalAitkenNeville.m

```
1  function value = RationalAitkenNeville(x,fx,x0)
2  % evaluate the rational interpolation polynomial (n,n+1) resp. (n,n
       )
3  % given by (x,fx) at the point x0
4  fxm1 = fx;
5  for j = length(fx):-1:2
6    if x0~=x(j)
7      fx(j) = fx(j) + (x0-x(j))/(x(j)-x(j-1)) * (fx(j)-fx(j-1));
8    else
9      value = fx(j); return
10   end
11 end
12
13 for k = 3:length(fx)
14   temp = fx;
15   for j = length(fx):-1:k
16     d = fx(j)-fx(j-1);
17 %    if fx(j)==fxm1(j-1) % abs(fx(j)-fxm1(j-1))<1e-10*max(1,max(abs
       (fx(j)),abs(fxm1(j-1))))
```

```
18 %       fx(j) = NaN;
19 %     else
20         %((x0-x(j-k+1))/(x0-x(j))*(1-d/(fx(j)-fxm1(j-1)))-1)
21         fx(j) = fx(j)+d/((x0-x(j-k+1))/(x0-x(j))*(1-d/(fx(j)-fxm1(j
               -1)))-1);
22 %     end
23     end
24     fxm1 = temp;
25 end
26 value = fx(end);
```

MATLAB-Beispiel:

Vergleichen wir die rationale Interpolation (Zähler- und Nennerpolynom quadratisch) mit der einfachen Polynominterpolation 4. Ordnung. Die Werte für $\cot(1°), \cot(2°), \ldots, \cot(5°)$ liegen vor und der Wert für $\cot(2°, 30')$ soll durch Interpolation angenähert werden.
Man beachte die unterschiedliche Genauigkeit.

```
>> format long
>> f = @(x) cot(x*pi/180);
>> x = linspace(1,5,5);
>> AitkenNeville(x,f(x),2.5)
ans =
  22.74709740132512
>> RationalAitkenNeville(x,f(x),2.5)
ans =
  22.90376554340344
>> f(2.5)
ans =
  22.90376554843120
```

Das die deutlich bessere Approximation durch das rationale Interpolationspolynom kein einzelner Zufallsbefund ist wird durch die folgende Grafik bestätigt, die den relativen Fehler zwischen Interpolationspolynom und exaktem Wert grafisch wiedergibt.

2.3.1 *Padé*-Approximation

Als *Padé*[5]-Approximation wird eine rationale Funktion bezeichnet, deren Potenzreihenentwicklung mit einer gegebenen Potenzreihe bis zum höchsten Grad übereinstimmt. Falls die rationale Funktion die folgende Form

$$R(x) = \frac{\sum_{k=0}^{m} a_k x^k}{1 + \sum_{k=0}^{n} b_k x^k}$$

hat, dann wird $R(x)$ als *Padé*-Approximation zur Potenzreihe

$$f(x) = \sum_{k=0}^{\infty} c_k x^k$$

bezeichnet, falls

$$R(0) = f(0) \quad \text{und ebenso} \tag{2.20}$$

$$\frac{d}{dx^k} R(x)\Big|_{x=0} = \frac{d}{dx^k} f(x)\Big|_{x=0}, \quad k = 1, 2, \ldots, m+n \tag{2.21}$$

gilt.

[5] Padé, Henri (1863-1953)

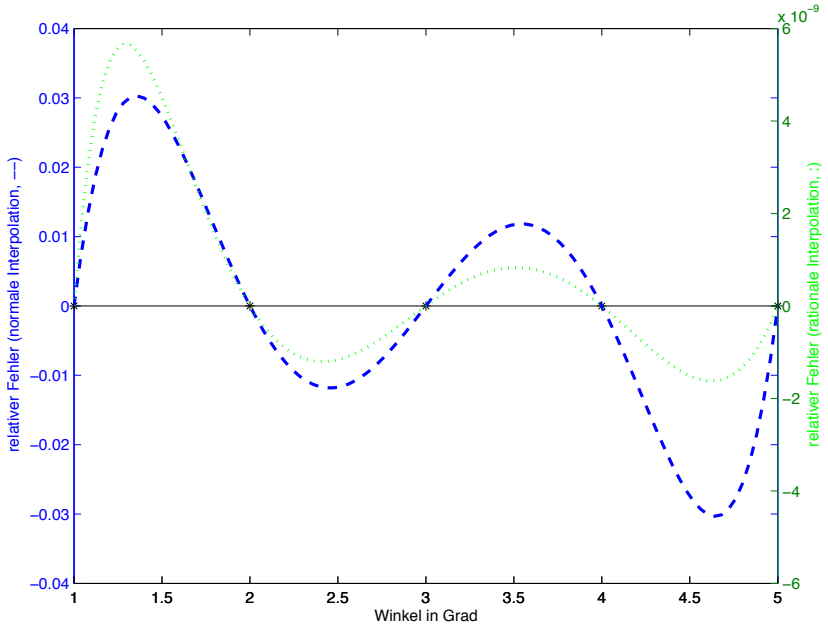


Abb. 2.1: Die Werte für $\cot(1°), \cot(2°), \ldots, \cot(5°)$ liegen vor. Dargestellt ist der relative Fehler zwischen Interpolationspolynom (zu den gegebenen Daten) und der exakten Funktion für das einfache Interpolationspolynom (gestichelte Linie, y-Achse links) mit einem Maximalwert ≈ 0.03 und für ein rationales Interpolationspolynom (gepunktete Linie, y-Achse rechts) mit einen maximalen Absolutwert $\approx 5.8 \cdot 10^{-9}$.

Die Gleichungen (2.20) und (3.43) führen zu $m + n + 1$ Gleichungen mit den Unbekannten $a_0, \ldots, a_m$ und $b_1, \ldots, b_n$. Der einfachste Weg zu diesen Gleichungen zu gelangen, besteht darin, $R(x)$ und $f(x)$ mit dem Nenner von $R(x)$ zu multiplizieren und einen Koeffizientenvergleich in den ersten $m + n + 1$ Monomen $1, x, x^2, \ldots, x^{m+n}$ durchzuführen.

Betrachten wir allerdings nur den Spezialfall $m = n$, so erhalten wir $a_0 = c_0$ und die Verbleibenden Koeffizienten a, b erfüllen die Gleichungen

$$\sum_{l=1}^{n} b_l c_{n-l+k} = -c_{n+k}, \quad k = 1, \ldots, n \tag{2.22}$$

$$\sum_{l=0}^{k} b_l c_{k-l} = a_k, \quad k = 1, \ldots, n \tag{2.23}$$

Man startet also zuerst mit (2.22), welches die Gleichungen zur Bestimmung von $b_1, \ldots, b_n$ darstellen. Obwohl diese Gleichungen eine *Töplitzmatrix* erzeugen, wendet man nicht das spezielle Verfahren für *Töplitzmatrizen* an. Erfahrungen zeigen, dass diese Gleichungen häufig nahezu singulär sind. Die LR-Zerlegung ist hier aus Stabilitätsgründen vorzuziehen. Nachdem die $b's$ nun bekannt sind, liefert (2.23) eine explizite Darstellung für die $a's$.

Die *Padé*-Approximation wird häufig dann angewendet, wenn die zu approximierende Funktion $f(x)$ nicht bekannt ist. Wir gehen davon aus, dass es möglich ist, den Wert von $f(x)$ und einiger Ableitungen an der Stelle $x = 0$, z.B. $f(0), f'(0), f''(0)$ zu bestimmen. Dies sind natürlich bereits die ersten Koeffizienten der Potenzreihenentwicklung, aber es ist keinesfalls klar, ob diese betragsmäßig kleiner werden und ob die Reihe überhaupt konvergiert. Wir wollen die Möglichkeiten der *Padé*-Approximation an einem Beispiel illustrieren, für eine Analyse der Methode verweisen wir auf [**?**].

Beispiel 2.3.5 Man stelle sich vor, man könnte die ersten fünf Terme einer Potenzreihe einer unbekannten Funktion $f(x)$ generieren, und diese sind

$$f(x) \approx 2 + \frac{1}{9}x + \frac{1}{81}x^2 - \frac{49}{8748}x^3 + \frac{175}{78732}x^4 + \dots$$

Wir wählen $m = n = 2$ für unsere *Padé*-Approximation. In der nachfolgenden Abbildung sind die abgebrochene Potenzreihendarstellung, die *Padé*-Approximation und die exakte Funktion

$$f(x) = \left(7 + (1+x)^{4/3}\right)^{1/3}$$

Die dargestellte Funktion f hat einen Verzweigungspunkt bei $x = -1$, daher konvergiert die Potenzreihe lediglich in $-1 < x < 1$. Im größten Teil der nachfolgenden Abbildung ist die Reihe divergent und der Wert der abgebrochenen Entwicklung nahezu sinnlos. Nichtsdestoweniger liefert die *Padé*-Approximation sehr gute Werte zumindest bis $x \sim 10$.

MATLAB-Funktionen: pade.m und horner.m

```
1  function [a,b] = pade(c)
2  c=c(:); n=(length(c)-1)/2;
3  b =[1;-toeplitz(c(n+1:end-1),c(n+1:-1:2))\c(n+2:end)];
4  a = [toeplitz(c(1:n+1),[c(1),zeros(1,n)])*b]
```

```
1  function value = horner(a,x0)
2  value = a(end);
3  for k = length(a)-1:-1:1
4    value = value.*x0 + a(k);
5  end
```

MATLAB-Beispiel:

Vergleich von abgebrochener Reihenentwicklung, *Padé*-Approximation und exakter Darstellung der Funktion

$$f(x) = \left(7 + (1+x)^{4/3}\right)^{1/3}.$$

Das Ergebnis ist in Abb. 2.3.1 dargestellt.

```
>> c = [2,1/9,1/81,-49/8748,175/78732];
>> x = linspace(0,10,400);
>> [a,b] = pade(c)
>> plot(x,(7+(1+x).^(4/3)).^(1/3),'k-', ...
     x,horner(c,x),'b-.', ...
     x,horner(a,x)./ horner(b,x),'r:')
>> axis([0,10,1.5,6])
>> legend('exakt','abgebr. Entwicklung',
   ...
     'Pade',2)
```

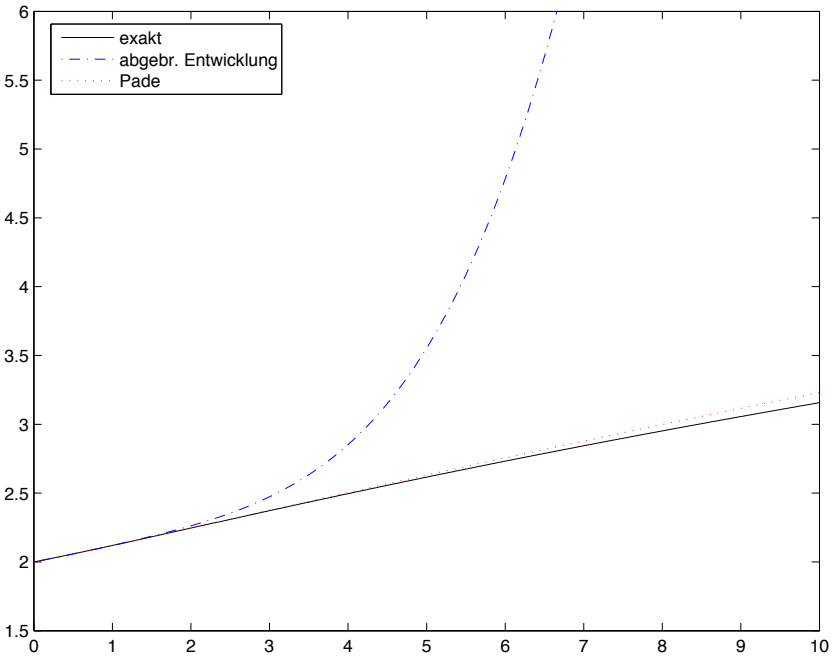


Abb. 2.2: Vergleich von abgebrochener Reihenentwicklung, *Padé*-Approximation und exakter Darstellung der Funktion $f(x) = (7 + (1 + x)^{4/3})^{1/3}$.

3 Numerische Integration und orthogonale Polynome

Die Numerische Integration oder Quadraturtheorie behandelt die Prinzipien und Algorithmen zur numerischen Berechnung von Integralen gegebener Funktionen. Hierbei beschränken wir uns vorerst auf die Berechnung des *Riemann*-Integrals[1]

$$I(f) := \int_a^b f(x)\, dx.$$

Dabei stellt f eine (von Computern ausführbare) Vorschrift dar, die es gestattet zu jedem $x \in [a, b]$ den entsprechenden Funktionswert f zu ermitteln (in den meisten Anwendungen ist f eine stückweise stetige oder stückweise glatte Funktion).

Aufgabe: Gegeben sei $f : [a, b] \to \mathbb{R}$, mit z.B. $f \in C[a, b]$. Approximiere den Ausdruck

$$I(f) := \int_a^b f(x)dx,$$

mit der Vorstellung, dass $I(f)$ nicht, bzw. nicht „leicht“ exakt bestimmt werden kann. Dazu konstruiert man Näherungsformeln („**Quadraturformeln**“)

$$\hat{I}_n : C[a, b] \to \mathbb{R}, \quad f \to \hat{I}_n(f)$$

die das Integral möglichst gut approximieren und dessen Eigenschaften erhalten, so dass der **Quadraturfehler**

$$E_n(f) := I(f) - \hat{I}_n(f)$$

klein wird. Hierzu wird der Integrand durch eine Approximation, welche man z.B. durch Polynominterpolation erhält ersetzt und diese integriert.

Bevor wir diese Idee der Konstruktion von $\hat{I}_n$ vertiefen, Fragen wir uns zunächst, welche **Kondition** das Problem der Integralberechnung besitzt? Mit

$$\|f\|_1 := \int_a^b |f(t)|dt = I(|f|),$$

der so genannten L^1-Norm gilt

Lemma 3.0.1 (Kondition der Integration) *Die absolute und relative Kondition der Integralberechnung bzgl.* $\|\cdot\|_1$ *lauten:*

$$\kappa_{abs} = 1, \quad \kappa_{rel} = \frac{I(|f|)}{|I(f)|}. \tag{3.1}$$

Beweis. Sei $\delta f \in L_1([a, b])$ eine Störung, dann gilt

$$\left| \int_a^b (f + \delta f)dx - \int_a^b f\, dx \right| = \left| \int_a^b \delta f\, dx \right| \leq \int_a^b |\delta f|dx = \|\delta f\|_1$$

und hier gilt Gleichheit genau dann, wenn $\delta f \geq 0$. □

[1] Riemann, Georg Friedrich Bernhard (1826-1866)

D.h. in absoluter Betrachtung ist die Integration ein gutartiges Problem. Relativ betrachtet birgt die Integration z.B. von stark oszillierenden Integranten Schwierigkeiten.

3.1 QUADRATURFORMELN

Die **Grundidee** der Quadratur besteht aus dem Folgenden:

- Unterteile $[a, b]$ in m Teilintervalle $[t_k, t_{k+1}], k = 0, 1, \ldots, m-1$, z.B. äquidistant

$$t_k = a + kH,\ k = 0, \ldots, m,\ H = \frac{b-a}{m}.$$

- Approximiere den Integranden f auf jedem Teilintervall $[t_k, t_{k+1}]$ durch eine „einfach“ zu integrierende Funktion g_k (z.B. ein Polynom vom Grad kleiner gleich n bzgl. $(n+1)$ Stützstellen in dem Intervall $[t_k, t_{k+1}]$) und verwende die Approximation

$$I(f) = \sum_{k=0}^{m-1} \int_{t_k}^{t_{k+1}} f(y)\, dy \approx \sum_{k=0}^{m-1} \int_{t_k}^{t_{k+1}} g_k(y)\, dy.$$

  Für g_k verwenden wir zunächst die Polynominterpolation.

Wir beginnen in diesem Abschnitt mit einfachen Beispielen, bei denen wir jeweils zunächst nur ein Teilintervall betrachten auf welchem wir den Integranden mit einer „einfachen“ Funktion approximieren, dieser Ansatz wird dann jeweils für $m \geq 1$ Teilintervalle „zusammengesetzt“. Dadurch erhält man sog. zusammengesetzte oder summierte Regeln.

Insgesamt betrachten wir drei Beispiele; In den ersten zwei Beispielen verwenden wir auf jedem Teilintervall konstante Funktionen zur Approximation, d.h. konstante Interpolationspolynome nullten Grades bzgl. einer Stützstelle und im dritten Beispiel auf jedem Teilintervall lineare Funktionen, d.h. Interpolationspolynome ersten Grades bzgl. zweier Stützstellen.

Im Anschluss konstruieren dann so genannte interpolatorische Quadraturformeln und motivieren damit die allgemeine Definition von Quadraturformeln. Weiterhin werden wichtige Eigenschaften von Quadraturformeln behandelt.

3.1.1 Einfache Beispiele

Beginnen wir mit den sicherlich einfachsten Quadraturformeln. Allgemein ersetzen wir zunächst den Integranten f auf dem Intervall $[a, b]$ durch eine konstante Funktion.

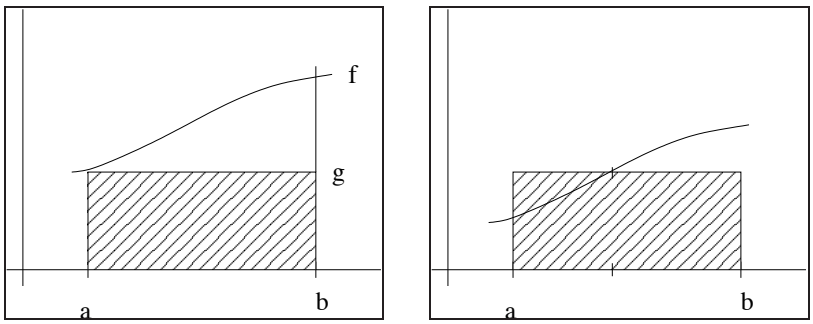


Abb. 3.1: Links: Linke Rechteckregel, rechts: Mittelpunktregel

Beispiel 3.1.1 (Linke Rechteckregel) Zunächst wählen wir als konstante Funktion g den Funktionswert am linken Rand, d.h. $x_0 = a$, $g(y) = f(x_0)$. Zur Veranschaulichung siehe Abbildung 3.1. Die resultierende Quadraturformel

$$R^L(f) := \hat{I}(f) = (b-a)f(a)$$

wird **linke Rechtecksregel** genannt. Natürlich könnte man auch analog den rechten Rand wählen.

Nun zerlegen wir das Integrationsintervall $[a,b]$ in $m \geq 1$ Teilintervalle $I_k = [t_k, t_{k+1}]$, $t_k = a + kH, k = 0, 1, \ldots, m-1$ und Breite $H = (b-a)/m$. Auf jedem Teilintervall wähle als konstante Funktion g_k den Funktionswert am linken Rand des Teilintervalls $g_k(y) := f(t_k)$, $y \in [t_k, t_{k+1}]$. Wir erhalten die Quadraturformel

$$R_m^L(f) := H \sum_{k=0}^{m-1} f(t_k),$$

diese wird **zusammengesetzte linke Rechtecksregel** genannt.

Nun zur Analyse: Sei f hinreichend glatt. Wegen

$$f(y) = f(t_k) + (y - t_k)f'(\xi)$$

mit einem $\xi \in (t_k, t_{k+1})$ (Taylor-Restglied) folgt

$$\begin{aligned}\int_{t_k}^{t_{k+1}} f(y)\,dy &= Hf(t_k) + \int_{t_k}^{t_{k+1}} f'(\xi)(y - t_k)\,dy \\ &= Hf(t_k) + \frac{f'(\xi)}{2}(t_{k+1} - t_k)^2 = \int_{t_k}^{t_{k+1}} g(y)\,dy + \mathcal{O}(H^2).\end{aligned}$$

Damit gilt

$$\begin{aligned}|I(f) - R_m^L(f)| &= \left| \int_a^b f(x)\,dx - H \sum_{k=0}^{m-1} f(t_k) \right| \\ &\leq \sum_{k=0}^{m-1} \underbrace{\left| \int_{t_k}^{t_{k+1}} f(x)\,dx - Hf(t_k) \right|}_{=\mathcal{O}(H^2)} \\ &\overset{m=\frac{b-a}{H}}{=} \mathcal{O}(H).\end{aligned}$$

Beispiel 3.1.2 (Mittelpunktregel) Wir setzen

$$\mathcal{C}_M^{(r)} := \left\{ f : [a,b] \to \mathbb{R} \;\middle|\; f^{(r)} \text{ stetig und } \sup_{a \leq x \leq b} |f^{(r)}(x)| \leq M \right\}$$

für $r \in \mathbb{N}$ und $M > 0$, und betrachten Grafik 3.2: Sei nun $x_0 \in (a,b)$, dann gilt für $f \in C^1[a,b]$:

$$\int_a^b |f - P(f|x_0)|\,dx \;\leq \sup_{a \leq x \leq b} \frac{|f'(x)|}{1!} \int_a^b |x - x_0|\,dx. \tag{3.2}$$

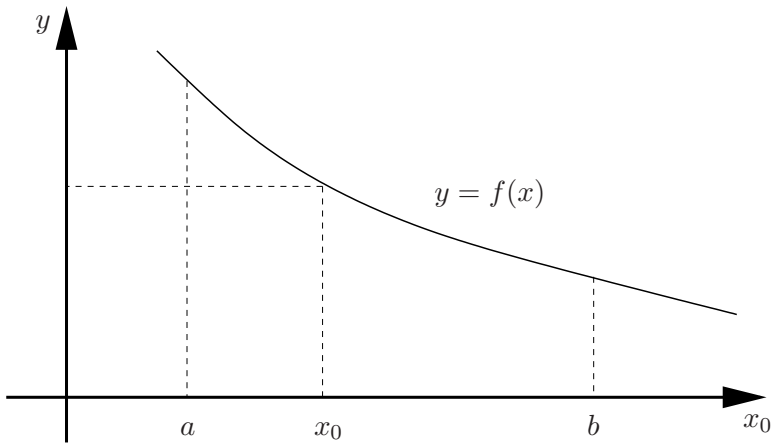


Abb. 3.2: Graph einer Funktion f.

Weiterhin ergibt sich mit $h := b - a$:

$$\begin{aligned} J(x_0) := \int_a^b |x - x_0|\, dx &= -\int_a^{x_0} (x - x_0)\, dx + \int_{x_0}^b (x - x_0)\, dx \\ &= \frac{(x-x_0)^2}{2}\Big|_{x=x_0}^{b} - \frac{(x-x_0)^2}{2}\Big|_{x=a}^{x_0} \\ &= \frac{(b-x_0)^2}{2} + \frac{(a-x_0)^2}{2} \\ &= x_0^2 - x_0(a+b) + \frac{a^2+b^2}{2}. \end{aligned}$$

Wir wollen die Stützstelle x_0 nun so wählen, dass $J(x_0)$ bzw. der rechte Teil der Ungleichung (3.2) minimal wird. Mit

$$J'(x_0) = 2x_0 - (a+b) \stackrel{!}{=} 0$$

ergibt sich als einziger kritischer Punkt $x_0 = \frac{a+b}{2}$, welcher J minimiert, da $J''(x_0) = 2 > 0$. Wir erhalten die **Mittelpunktregel** (zur Veranschaulichung siehe Abbildung 3.1)

$$M(f) := \hat{I}_0(f) := (b-a) \cdot f\Big(\frac{a+b}{2}\Big). \tag{3.3}$$

bei der als konstante Funktion $g(y) = f((a+b)/2)$ genommen wird.
Mit obiger Herleitung haben wir auch gezeigt, dass für $M := \sup_{a \le x \le b} |f'(x)|$ gilt:

$$\begin{aligned} \sup_{f \in \mathcal{C}_M^{(1)}} \left| \int_a^b f(x) - \hat{I}_0(f) \right| &\le \frac{M}{1!} \cdot J\Big(\frac{a+b}{2}\Big) \\ &= M \cdot \Big(\frac{(a+b)^2}{4} - \frac{(a+b)^2}{2} + \frac{a^2+b^2}{2} \Big) \\ &= M \cdot \frac{(b-a)^2}{4}, \end{aligned}$$

also die rechte Seite ein Maß für die von (3.3) in $\mathcal{C}_M^{(1)}$ garantierte Genauigkeit ist.

Nun zerlegen wir das Integrationsintervall $[a, b]$ wieder in $m \ge 1$ Teilintervalle $I_k = [t_k, t_{k+1}]$, $t_k = a + Hk, k = 0, \ldots, m$. Auf jedem Teilintervall setze

$$g_k(y) := f\left(\frac{t_k + t_{k+1}}{2}\right), \quad y \in [t_k, t_{k+1}].$$

Mit $x_k := \frac{1}{2}(t_k + t_{k+1})$ liefert Taylorentwicklung um x_k für $f \in C^3(t_k, t_{k+1})$

$$f(y) = f(x_k) + (y - x_k)f'(x_k) + \frac{1}{2}(y - x_k)^2 f''(x_k) + \mathcal{O}(H^3),$$

und damit

$$\begin{aligned}\int_{t_{k+1}}^{t_k} f(y)\,dy &= \int_{t_k}^{x_k} f(y)\,dy + \int_{x_k}^{t_{k+1}} f(y)\,dy \\ &= \frac{H}{2}f(x_k) + \int_{t_k}^{x_k}(y - x_k)f'(x_k)\,dy + \mathcal{O}(H^3) \\ &\quad + \frac{H}{2}f(x_k) + \int_{x_k}^{t_{k+1}}(y - x_k)f'(x_k)\,dy + \mathcal{O}(H^3) \\ &= Hf(x_k) + \mathcal{O}(H^3),\end{aligned}$$

denn

$$\int_{t_k}^{x_k}(y - x_k)\,dy = \int_{-\frac{H}{2}}^{0} z\,dz = -\int_{0}^{-\frac{H}{2}} z\,dz = -\int_{x_k}^{t_{k+1}}(y - x_k)\,dy.$$

Mit Quadraturknoten $x_k = a + (2k + 1)H/2, k = 0, \ldots, m - 1$ erhalten die **summierte Mittelpunktregel**

$$M_H(f) := \hat{I}_{0,m}(f) := H\sum_{k=0}^{m-1} f\left(\frac{t_k + t_{k+1}}{2}\right) = H\sum_{k=0}^{m-1} f(x_k),$$

mit dem Fehler

$$|I(f) - M_H(f)| = \mathcal{O}(H^2) \quad \text{für } f \in C^3(t_k, t_{k+1}).$$

Nun ersetzten wir den Integranten durch eine lineare Funktion bzw. durch lineare Funktionen auf Teilintervallen.

Beispiel 3.1.3 (Trapezregel) Wählen wir g als lineare Interpolation (Lagrange Interpolationspolynom vom Grad 1) von f bzgl. der Knoten $x_0 = a$ und $x_1 = b$ so erhalten wir die so genannte **Trapezregel**:

$$\hat{I}_1(f) := \frac{b - a}{2}\left(f(a) + f(b)\right).$$

Um die zusammengesetzte Trapezregel zu erhalten, zerlegen wir $[a, b]$ wie im Fall eines Knotens (d.h. $n = 0$) in $m \geq 1$ Teilintervalle $I_k = [t_k, t_{k+1}]$, $t_k = a + Hk, k = 0, \ldots, m$ der Breite $H = b - a/m$ und wähle g_k als lineare Interpolation von f bzgl. der Stützstellen t_k, t_{k+1} mit Quadraturknoten t_k, d.h.

$$g_k(y) = \frac{t_{k+1} - y}{H}f(t_k) + \frac{y - t_k}{H}f(t_{k+1}), \quad y \in [t_k, t_{k+1}]$$

Das führt zu

$$\int_{t_k}^{t_{k+1}} g_k(y)dy = \frac{H}{2}(f(t_{k+1}) + f(t_k)).$$

Dies setzt man zusammen auf $[a, b] = [t_0, t_m]$:

$$\hat{I}_{1,m}(f) := \frac{H}{2}\sum_{k=0}^{m-1}(f(t_{k+1}) + f(t_k))$$

Da jeder Term bis auf den ersten und letzten doppelt gezählt wird erhalten wir die sogenannte **Trapezsumme**

$$T_H(f) := \hat{I}_{1,m}(f) := H\left\{\frac{1}{2}f(a) + f(t_1) + \ldots + f(t_{m-1}) + \frac{1}{2}f(b)\right\}. \tag{3.4}$$

3.1.2 Konstruktion und Definition von Quadraturformeln

In obigen Beispielen haben wir stets den Integranden f durch eine Approximation $\hat{f}$ ersetzt, d.h. $f \approx \hat{f}$ und dann $\hat{f}$ exakt integriert. Wir haben also als Quadratur $\hat{I}_n$

$$\hat{I}_n(f) = I(\hat{f})$$

gewählt. Um höhere Genauigkeit zu erzielen, reichen konstante bzw. lineare Funktionen, wie bei der Mittelpunkts- bzw. der Trapetzregel, offenbar nicht aus.

Die Kernidee der Mittelpunkts- bzw. der Trapezformel liegt eigentlich darin, die Funktion f durch eine Interpolierende $P(f|x_0,\ldots,x_n)$ bzgl. der Stützstellen x_i (die - wie bei der Mittelpunktsregel - von den t_j verschieden seien können) zu ersetzen, sodass sich für diese die Quadratur einfach ausführen läßt. Es liegt somit nahe, als Approximation $\hat{f}$ das Interpolationspolynom an f bezüglich der Knoten zu wählen, d.h.

$$\hat{f}(x) = P(f|x_0,\ldots,x_n)(x).$$

Wir verwenden dann $I\big(P(f|x_0,\ldots,x_n)\big)$ als Approximation zu $I(f) = \int_a^b f(x)dx$, setzen also:

$$\hat{I}_n(f) = I\big(P(f|x_0,\ldots,x_n)\big).$$

Konstruktion interpolatorischer Quadraturformeln: Die Idee bei der Konstruktion von Quadraturformeln ist es $(n+1)$ Stützstellen zu wählen und, wie bereits erwähnt, das Integral über das Interpolationspolynom zu diesen Knoten als Näherung an das Integral von f zu verwenden.

Da ein besimmtes Integral über ein beliebiges Intervall $[a,b]$ - ohne Beschränkung der Allgemeinheit - mittels Variablentransformation auf das Intervall $[0,1]$ transformiert werden kann, betrachten wir zunächst $[a,b] = [0,1]$.
Die auf $[0,1]$ transformierten Stützstellen nennen wir τ_i.
Die $(n+1)$ Stützstellen $\tau_0,\ldots,\tau_n \in [0,1]$ seien der Einfachheit halber äquidistant, d.h. $\tau_i = ih$ für $i = 0,\ldots,n$ mit $h = 1/n$. Mit der Lagrange-Darstellung des Interpolationspolynoms

$$P(y) = \sum_{i=0}^{n} f(\tau_i) L_i^{(n)}(\tau), \quad L_i^{(n)}(y) = \prod_{\substack{j=0 \\ j\neq i}} \frac{(y-\tau_j)}{(\tau_i - \tau_j)}$$

folgt dann

$$\hat{I}_n(f) = \int_0^1 P(\tau)\,d\tau = \int_0^1 \left(\sum_{i=0}^{n} f(\tau_i) L_i^{(n)}(\tau)\right) d\tau = \sum_{i=0}^{n} f(\tau_i) \underbrace{\left(\int_0^1 L_i^{(n)}(\tau)\,d\tau\right)}_{=:\lambda_i} = \sum_{i=0}^{n} f(\tau_i)\lambda_i$$

Dies nennt man eine **Quadraturformel** mit **Gewichten** λ_i und **Knoten** (**Stützstellen**) τ_i.

Um eine allgemeine Quadraturformel für beliebige Intervalle $[a,b]$ zu bekommen, wird das Intervall $[a,b]$ nun auf $[0,1]$ transformiert. Mit $x = a + \tau(b-a)$ und $dx = d\tau(b-a) = (b-a)d\tau$ erhalten wir

$$I(f) = \int_a^b f(x)\,dx = (b-a)\int_0^1 \underbrace{f(a+\tau(b-a))}_{:=\tilde{f}(\tau)}\,d\tau.$$

Damit erhält man die Quadraturformel

$$\hat{I}_n(f) = (b-a)\sum_{i=0}^{n} f(a+\tau_i(b-a))\cdot\lambda_i$$

mit $\lambda_i := \int_0^1 L_i^{(n)}(\tau)\,d\tau$ und $\tau_i = ih$ für $h = 1/n$. Man beachte, die Gewichte λ_i sind unabhängig von den Integrationsgrenzen a, b und vom aktuellen Integranten und müssen für gegebene Stützstellen also nur *einmal* berechnet werden.

Wir fassen dies zusammen und definieren allgemein das lineare Funktional $\hat{I}_n$ als Quadraturformel:

Definition 3.1.4 (Quadraturformel) *Unter einer* Quadraturformel $\hat{I}$ *zur Approximation des bestimmten Integrals*

$$I(f) = \int_a^b f(x)\,dx$$

versteht man

$$\hat{I}_n(f) = (b-a)\sum_{i=0}^{n} \lambda_i f(x_i) \tag{3.5}$$

mit den Knoten $x_0, \ldots, x_n$ *und den* Gewichten $\lambda_0, \ldots, \lambda_n \in \mathbb{R}$, *wobei*

$$\sum_{i=0}^{n} \lambda_i = 1 \tag{3.6}$$

gilt.

Bemerkung 3.1.5

1. *Eine Eigenschaft wie (3.6) wird „Partition der Eins“genannt.*

2. *Sind die Gewichte nicht negativ, dann stellt (3.6) eine Konvexkombination der Funktionswerte $f(x_0), \ldots, f(x_n)$ dar.*

3. *Für die konstante Funktion $f(x) \equiv c = const$, $x \in [a, b]$ folgt aus (3.6)*

$$\hat{I}_n(f) = (b-a)\underbrace{\sum_{i=0}^{n} \lambda_i}_{=1} c = c(b-a) = I(f),$$

d.h., konstante Funktionen werden exakt *integriert. Dies ist der Grund für die Forderung (3.6).*

Definition 3.1.6 (Interpolatorische Quadraturformel) *Sei I ein Integral und I_n eine Quadraturformel dazu. Man nennt die Quaraturformel* **interpolatorisch**, *wenn gilt*

$$\lambda_i = \frac{1}{(b-a)} I(L_i^{(n)}), \quad i = 0, \ldots, n.$$

Bemerkung 3.1.7 *Mit anderen Worten, im Rahmen der Herleitung der Quadraturformeln haben wir interpolatorische Quadraturformeln verwendet. Im Laufe dieses Kapitels werden wir noch andere Quadraturformeln kennen lernen, dies sind die so genannten* **Gauß-*Quadraturformeln*** *(vgl. Abschnitt 3.4).*

3.1.3 Exaktheitsgrad und Quadraturfehler

Aus der Konstruktion der interpolatorischen Quadraturformeln ergeben sich sofort Konsequenzen:

- derartige Quadraturformeln sind für alle $f \in \mathbb{P}_n$ exakt,
- aufgrund der Eindeutigkeit der Polynom-Interpolation ist diese Quadraturformel die *einzige* exakte bzgl. der Knoten $x_0, \ldots, x_n$.

Dies fassen wir in den folgenden Lemma zusammen:

Lemma 3.1.8 (Eindeutige exakte interpolatorische Quadraturformel) *Zu $(n+1)$ paarweise verschiedenen Knoten $x_0, \ldots, x_n$ gibt es genau eine Quadraturformel*

$$\hat{I}_n(f) = (b-a) \sum_{i=0}^{n} \lambda_i f(x_i), \tag{3.7}$$

die für alle $P \in \mathbb{P}_n$ vom Grad kleiner oder gleich n exakt ist.

Beweis. Wie bei der Konstruktion verwenden wird die *Lagrange*-Polynome $L_i^{(n)}$ setzen diese in die Quadraturformel ein und erhalten:

$$I\Big(\sum_{i=0}^{n} f(x_i) L_i^{(n)}(x)\Big) = \sum_{0=1}^{n} f(x_i) I\big(L_i^{(n)}(x)\big) = (b-a) \sum_{i=0}^{n} \lambda_i f(x_i). \tag{3.8}$$

Dadurch erhalten wir die Gewichte

$$\lambda_i = \frac{1}{(b-a)} \int_a^b L_i^{(n)}(x) dx$$

auf eindeutige Weise zurück. □

Nun untersuchen wir allgemein den Fehler von Quadraturformeln. Dazu definieren wir

Definition 3.1.9 (Quadraturfehler, Exaktheitsgrad, Fehlerordnung) *Sei I ein Integral, $\hat{I}_n$ eine Quadraturformel zu $(n+1)$ Knoten.*

1. *Dann heißt $E_n(f) := I(f) - \hat{I}_n(f)$ **Quadraturfehler**.*
2. *Die größte Zahl $r \in \mathbb{N}$ für die gilt*

   $$E_n(f) = I(f) - \hat{I}_n(f) = 0, \quad \forall f \in \mathbb{P}_r$$

   *wird **Exaktheitsgrad** der Quadraturformel genannt.*
3. *Zu gegebenem Exaktheitsgrad r wird $r+1$ **Ordnung** oder **Fehlerordnung der Quadraturformel** $\hat{I}_n$ genannt.*

Bemerkung 3.1.10 (Exaktheitsgrad interpolatorischer Quadraturformeln) *Nach Lemma 3.1.8 hat jede interpolatorische Quadraturformel, die $(n+1)$ Stützstellen verwendet, mindestens den Exaktheitsgrad n.*

Bemerkung 3.1.11 (Test der Ordnung einer Quadraturformel) *Der Quadraturfehler ist offenbar linear. Daher kann die Ordnung einer Quadraturformel $\hat{I}_n$ leicht über die Monome getestet werden: Gilt $E_n(x^j) = 0, j = 0, \ldots, r-1$ und $E_n(x^r) \neq 0$, so hat $\hat{I}_n$ die Ordnung r.*

Hat der Integrand f „schöne“ Eigenschaften, d.h. ist er z.B. $n+1$-mal stetig differenzierbar, so kann man den Interpolationsfehler mit Satz 2.2.24 abschätzen. Integration über den Interpolationsfehler liefert eine Fehlerschranke für den Integrationsfehler.

Lemma 3.1.12 (Fehlerschranken interpolatorischer Quadraturformeln) *Es gilt für*

$$\hat{I}_n(f) := \int_a^b P(f|x_0,\ldots,x_n)(x)dx$$

die Fehlerabschätzung

$$\begin{aligned} |E_n(f)| &\leq (b-a)\left(\max_{y\in[a,b]}\prod_{j=0}^{n}|y-x_j|\right)\frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \\ &\leq \frac{(b-a)^{n+2}}{(n+1)!}\|f^{(n+1)}\|_\infty \end{aligned} \tag{3.9}$$

für $f \in C^{n+1}[a,b]$.

Beweis. Lagrange–Darstellung für $P(f|x_0,\ldots,x_n)$ einsetzen und Fehlerdarstellung der Polynominterpolation anwenden. □

Bemerkung 3.1.13 (Fehlerschranken) *Die Fehlerschranken aus Lemma 3.1.12 sind i.A.* nicht *optimal. D.h., die Schranken sind in der Regel zu groß. Schärfere Schranken können von Hand bewiesen werden (mittels Anwendung der Mittelwertsätze aus der Analysis).*

3.1.4 Konvergenz einer Quadraturformel

Das Ziel ist es, mit möglichst geringem Aufwand ein Integral möglichst genau zu approximieren. Als Maß für die Genauigkeit haben wir im letzten Abschnitt den Quadraturfehler eingeführt. Man kann sich zusätzlich noch fragen, ob, bzw. unter welchen Bedingungen Quadraturformeln, für wachsende Anzahl von Stützstellen gegen das exakte Integral konvergieren.

Wir betrachten dazu in diesem Abschnitt nun Folgen von Quadraturformeln $(\hat{I}_n)_{n\in\mathbb{N}}$ und untersuchen, wann diese gegen das zu approximierende Integral konvergieren. Zunächst definieren wir:

Definition 3.1.14 (Konvergenz einer Quadraturformel) *Sei $(\hat{I}_n)_{n\in\mathbb{N}}$ eine Folge von Quadraturformeln. Gilt*

$$\hat{I}_n(f) \xrightarrow{n\to\infty} \int_a^b f(x)dx \quad \text{für jedes } f\in C[a,b], \tag{3.10}$$

so spricht man von der **Konvergenz der Quadraturformeln** $(\hat{I}_n)_{n\in\mathbb{N}}$.

Die folgenden zwei Sätze liefern nun Bedingungen unter denen Folgen von Quadraturformeln $(\hat{I}_n)_{n\in\mathbb{N}}$ konvergieren. Insbesondere sind dies Bedingungen an die Gewichte.

Satz 3.1.15 (*Szegó*) *Sei $a \leq x_0^{(n)} < \cdots < x_n^{(n)} \leq b$, und seien $\lambda_k^{(n)} \in \mathbb{R}$, $k=0,\ldots,n$. Dann sind die für $n\in\mathbb{N}$ gemäß*

$$\hat{I}_n : \big(C[a,b], \|\cdot\|_\infty\big) \to \big(\mathbb{R}, |\cdot|\big), \quad f \mapsto \hat{I}_n(f) := (b-a)\sum_{i=0}^{n}\lambda_i^{(n)} f\big(x_i^{(n)}\big)$$

definierten Quadraturformeln $\hat{I}_n$ genau dann konvergent, wenn die beiden folgenden Bedingungen erfüllt sind:

(i)

$$\sup_{n\in\mathbb{N}} \sum_{k=0}^{n} \left|\lambda_k^{(n)}\right| < \infty$$

(ii)

$$I(P) = \lim_{n\to\infty} \hat{I}_n(P), \quad \forall\, P \in \mathbb{P} = \mathbb{P}(a,b).$$

wobei $\mathbb{P}(a,b)$ die Menge der Polynome auf dem Intervall (a,b) bezeichnet.

Beweis. Der Beweis dieses Satzes erfordert grundlegende Kenntnisse der Funktionalanalysis und wird deshalb an dieser Stelle ausgespart. Man findet ihn nichtsdestoweniger beispielsweise in [Heuser, Seite 159]. □

Satz 3.1.16 (*Steklov*) *Unter den Voraussetzungen des Satzes von* Szegó *gelte*

$$\lambda_k^{(n)} \geq 0, \quad n \in \mathbb{N}, 0 \leq k \leq n \tag{3.11}$$

für die Gewichte $\lambda_k^{(n)}$ der Quadraturformel $\hat{I}_n$. Dann konvergieren die Quadraturformeln $(\hat{I}_n)_{n\in\mathbb{N}}$ genau dann, wenn sie für alle $P \in \mathbb{P}_n$ konvergieren.

Beweis. Nach dem Satz von *Szegó* ist der Beweis erbracht, wenn dort unter der Voraussetzung von (3.11) (i) aus (ii) folgt. Es gelte also (ii) im Satz von *Szegó*. Für $f \equiv 1$ gilt mit (ii)

$$b - a = I(f) = \lim_{n\to\infty} \hat{I}_n(f) = \lim_{n\to\infty} \sum_{k=0}^{n} \lambda_k^{(n)},$$

was wegen (3.11) die Aussage (i) impliziert. □

3.2 Klassische interpolatorische Quadraturformeln

Wir haben Quadraturformeln über Lagrange-Interpolation zu äquidistanten Knoten konstruiert, d.h. durch Polynominterpolation. Damit haben wir die allgemeine Definition der Quadraturformeln motiviert. Im Fall, dass die Gewichte der Quadraturformel den Integralen über die Lagrange-Polynome entsprechen haben wir von interpolatorischen Quadraturformeln gesprochen.

Im Spezialfall, dass diese Quadraturformeln durch Polynominterpolation an *gleichverteilten* Knoten, d.h. äquidistanter Stützstellen - so wie wir sie auch konstruiert haben - , entstehen, spricht man von ***Newton-Cotes*[2]-Formeln**. In diesem Abschnitt schauen wir uns diesen Spezialfall noch mal etwas genauer an und fassen einige Eigenschaften zusammen. Wir werden sehen, dass die einführenden Beispiele zu diesen klassischen Quadraturformeln gehören.

Außerdem werden wir sehen, dass die Sätze aus dem vorherigen Abschnitt Grenzen für die numerische Stabilität dieses Ansatzes liefern. Dies führt uns auf die **zusammengesetzten *Newton-Cotes*-Formeln**.

[2] Cotes, Roger (1682-1716)

3.2.1 *Newton-Cotes*-Formeln

Definition 3.2.1 (*Newton-Cotes*-Formeln) *Bei äquidistanter Knotenwahl $a \leq x_0 < x_1 \cdots < x_n \leq b$ heißen die resultierenden Integrationsformeln*

$$\hat{I}_n(f) = (b-a)\sum_{i=1}^{n} \lambda_i f(x_i)$$

***Newton-Cotes-Formeln**.*
Die Newton-Cotes-Formeln heißen ***abgeschlossen**, wenn $x_0 = a$ und $x_n = b$, d.h.*

$$x_i = a + ih, \quad h = \frac{b-a}{n}, \quad i = 0, \ldots, n.$$

Die Newton-Cotes-*Formeln heißen* ***offen**, wenn $x_0 < a$ und $x_n < b$, wobei meist*

$$x_i = a + \Big(i + \frac{1}{2}\Big)h \quad \text{mit} \quad h = \frac{b-a}{n+1}, \quad i = 0, \ldots, n \tag{3.12}$$

gewählt wird.

Bemerkung 3.2.2 *Es ist klar, dass die Mittelpunkt- und die Trapezregel, welche wir bereits kennen gelernt haben, abgeschlossene* Newton-Cotes-*Formeln sind. Man erhält sie mit $n = 0, n = 1$.*

Der Ausdruck für die abgeschlossenen *Newton-Cotes*-Gewichte λ_i vereinfacht sich durch die Substitution $s := \frac{(x-a)}{h}$ zu

$$\lambda_i = \frac{1}{b-a}\int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}\, dx = \frac{1}{n}\int_0^n \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(s-j)}{(i-j)}\, ds := \frac{1}{n}\alpha_i$$

und für die offenen *Newton-Cotes*-Formeln erhält man unter der Wahl (3.12) der x_i:

$$\lambda_i = \frac{1}{b-a}\int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}\, dx = \frac{1}{n+1}\int_{-\frac{1}{2}}^{n+\frac{1}{2}} \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(s-j)}{(i-j)}\, ds := \frac{1}{n}\alpha_i$$

Bemerkung 3.2.3 (Praktische Berechnung via a priori Tabellierung der Gewichte) *Die* Newton-Cotes-*Formeln besitzen den Vorteil, dass die Gewichte zwar von n und h abhängen nicht aber von a, b. In der Praxis berechnet man daher* nicht *zunächst $\hat{f} = P(f|\ldots)$ und dann $I_n(\hat{f})$, sondern sucht direkt die Form (3.5) der Quadraturformel, d.h., man berechnet die Gewichte. Diese können* a priori *tabelliert werden.*

Bemerkung 3.2.4 (Offene *Newton-Cotes*-Formeln) *Die offenen* Newton-Cotes-*Formeln kann man vor allem dann verwenden, wenn die Auswertung des Integranden f an den Integrationsgrenzen schwierig ist, z.B. wenn f am Rand eine Singularität hat. Allerdings haben sie eine deutlich schlechtere Ordnung als die ebenfalls offenen Gauß-Quadraturen und finden daher in der Praxis kaum Anwendung.*

Im folgenden Beispiel tabellieren wir daher die Gewichte der wichtigsten geschlossenen *Newton-Cotes*-Formeln. Da die Stützstellen und Gewichte von dem Polynomgrad n abhängen, den wir wählen, schreiben wir hier, wie auch schon zuvor, $x_j^{(n)}$ und $\lambda_j^{(n)}$.

Beispiel 3.2.5 (Geschlossene *Newton-Cotes*-Formeln) Wählen wir $(n+1)$ äquidistante Stützstellen auf $[a, b]$, d.h.

$$x_i^{(n)} := a + \tau_i^{(n)}(b-a), \quad i = 0, \ldots, n,$$

wobei $\tau_i^{(n)} = i/n$, die Stützstellen auf $[0,1]$ sind. Bezeichne $\alpha_i^{(n)} = \lambda_i^{(n)} n$ die Gewichte auf $[0,1]$. Dann gilt mit $h = (b-a)/n$

$$\hat{I}_n(f) := (b-a)\sum_{i=0}^{n} \lambda_i^{(n)} f(a + \tau_i^{(n)}(b-a)) = h\sum_{i=0}^{n} \alpha_i^{(n)} f(a + \tau_i^{(n)}(b-a)). \qquad (3.13)$$

lauten die Eckdaten der wichtigsten geschlossenen *Newton-Cotes*-Formeln

n	$\tau_i^{(n)}$	$\alpha_i^{(n)}$	Restglied	Name
1	$0, 1$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{12}h^3 f''(\xi)$	Trapezregel
2	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$	$\frac{1}{90}2^{-5}h^5 f^{(4)}(\xi)$	Simpson–Regel, Keplersche Fassregel
3	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{3}{80}3^{-5}h^5 f^{(4)}(\xi)$	Newtonsche $3/8$-Regel
4	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{8}{945}4^{-7}h^7 f^{(6)}(\xi)$	Milne–Regel

Als Beispiel betrachte die Simpson-Regel, in obiger Form lautet diese

$$\hat{I}_2(f) = \frac{b-a}{2}\left(\frac{1}{6}f(a) + \frac{2}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{6}f(b)\right).$$

Bemerkung 3.2.6 *In obigem Zugang entspricht die Anzahl der Stützstellen dem Exaktheitsgrad des Interpolationspolynoms und damit der Quadraturformel. Dies wiederum entspricht der Genauigkeit der Quadraturformel. Vielleicht kann man ja mit weniger Stützstelle dieselbe Genauigkeit erreiche, oder — anders ausgedrückt — mit geschickt gewählten Stützstellen eine höhere Genauigkeit erreichen. Mit dieser Frage wollen wir uns nun beschäftigen.*

Bemerkung 3.2.7 (Grenzen der *Newton-Cotes*-Formeln) *Bei den* Newton-Cotes-*Formeln treten durchaus negative Gewichte auf, d.h. der Satz 3.1.16 von* Steklov *kann dann nicht angewandt werden. Jedoch hilft auch der Satz 3.1.15 von* Szegó *nicht weiter, da - wie* G. Polya *zeigte - eine Funktion existiert, für die die* Newton-Cotes-*Formeln nicht konvergieren.*

Für abgeschlossene Newton-Cotes-*Formeln taucht bei $n = 8$, für offene* Newton-Cotes-*Formeln bei $n = 6$ das erste Mal ein negatives Gewicht auf. Man sollte die jeweiligen* Newton-Cotes-*Formeln nicht für n größer als diese jeweiligen Werte verwenden!*

Die Gewichte der Newton-Cotes-*Formeln sind* rational *und können mit* Maple *berechnet werden. Die folgenden Zeilen* Maple-*Code berechnen die Gewichte der abgeschlossenen* Newton-Cotes-*Formeln für $n = 8$ und die der offenen* Newton-Cotes-*Formeln für $n = 6$:*

MAPLE-*Beispiel:*

```
> n:=8:
> zaehler:= (x,i) -> quo(product((x-k),k=0..n),(x-i),x):
> seq(int(zaehler(x,m)/subs(x=m, zaehler(x,m)),x=0..n)/n,m=0..n);

   989    2944     -464    5248     -454     5248     -464     2944     989
 -----, -------, --------,-------, -----, -------, --------, -------, ------
 28350   14175    14175   14175     2835    14175    14175    14175   28350
```

```
> n:=6:
> zaehler:= (x,i) -> quo(product((x-k),k=0..n),(x-i),x):
> seq(int(zaehler(x,m)/subs(x=m, zaehler(x,m)),
      x=-1/2..n+1/2)/(n+1),m=0..n);

  4949     49     6223     -6257     6223     49     4949
 -----, ----, -----, -----, -----, ----, -----
 27648    7680   15360    34560     15360    7680   27648
```

Wie man sieht, enthalten beide Quadratur-Formeln negative Gewichte.

3.2.2 Zusammengesetzte *Newton-Cotes*-Formeln

Wir haben gesehen, dass bei den geschlossenen *Newton-Cotes*-Formeln ab $n = 7$ und für die offenen *Newton-Cotes*-Formeln ab $n = 5$ negative Gewichte auftreten. I.A. können wir keine Konvergenz $\hat{I}_n(f) \to I(f)$ für $n \to \infty$ erwarten.

Einen Ausweg bieten die sogenannten zusammengesetzten oder summierten Quadraturfomeln. Als Spezialfall zusammengesetzter *Newton-Cotes*-Formeln haben wir in den einführenden Beispielen bereits die summierte Mittelpunktsregel und die Trapezsumme kennen gelernt.

Allgemein ist die Idee zusammengesetzter Quadratur, dass man das Integrationsintervall $[a, b]$ in m Teilintervalle zerlegt und die Quadraturformel $\hat{I}_n(f)$ auf die Teilintervalle der Länge $H = (b-a)/m$ anwendet.

$$\hat{I}_{n,m}(f) = \sum_{k=0}^{m-1} \hat{I}_n|_{[t_k,t_{k+1}]}(f) = \sum_{k=0}^{m-1} \sum_{i=1}^{n} \lambda_i^{(n,k)} f(x_i^{(n,k)})$$

Sei nun n fest und man betrachte eine Folge von zusammengesetzte *Newton-Cotes*-Formeln $(\hat{I}_m(f))_{m\in\mathbb{N}}$:

Beispiel 3.2.8 (Anwendung des Satzes von *Szegó* auf die summierte Mittelpunktsregel)
Wir wenden den Satz von *Szegó* auf die summierte Mittelpunktsregel an. Betrachte hierzu eine Folge von Knoten

$$x_k^{(1,m)} := a + \frac{2k+1}{2} \cdot \frac{b-a}{m+1}, \quad k = 0, \ldots, m,$$

mit den Gewichten

$$\lambda_k^{(m)} = \frac{b-a}{m+1}.$$

Es gilt: $\sum_k |\lambda_k^{(m)}| = b - a$, demnach ist (i) in Satz 3.1.15 erfüllt. Weiterhin gilt (ii) aufgrund der Stetigkeit (damit der *Riemann*-Integrierbarkeit) der Polynome auf $[a, b]$ und der Fehlerabschätzung für die summierte Mittelpunktsregel.

Lemma 3.2.9 (Fehlerschätzung der Trapezsumme) *Es gilt*

$$|E_1(f)| = |I(f) - T_H(f)| = \mathcal{O}(H^2) \text{ für } f \in C^2([a, b]) \tag{3.14}$$

Bemerkung 3.2.10 (Ineffizenz bei hoher gewünschter Genauigkeit) *Angenommen, man möchte mit der Trapezsumme (3.4) die Genauigkeit von 10^{-6} erreichen. Die Fehlerschätzung (3.14) besagt dann*

$$|I(f) - T_H(f)| = \mathcal{O}(H^2) \leq 10^{-6},$$

also muss H^2 in der Größenordung von 10^{-6} sein, demnach muss $H \sim 10^{-3}$ gelten und bei äquidistanter Schrittweite folgt dann für die Anzahl der Knoten

$$m = \left[\frac{b-a}{H}\right] \sim 10^3.$$

Dies wäre viel zu ineffizient, wir brauchen Alternativen!

3.3 Extrapolation und Romberg-Integration

Wir wollen nun die Strategie zur nachträglichen Steigerung der Genauigkeit kennenlernen, die auch eine lokal angepasste („adaptive") Wahl der Stützstellen erlaubt. Bisher haben wir Quadraturen bzgl. eines festen Knoten gitters betrachtet. Nun verwenden wir eine Folge von Knotengittern.

Die Idee besteht darin zu einer Folge von absteigenden Schrittweiten verschiedene Approximationen an ein Integral zu berechnen, z.B. die Trapezsummen $T_{H_\ell}, \ell = 1, \ldots, k$. Dann wendet man Polynominterpolation zu den Paaren $(H_1, T_{H_1}), \ldots, (H_k, T_{H_k})$ an, um eine Interpolierende zu bestimmen. Wir wissen, dass

$$\lim_{H\to 0} I_H(f) = \lim_{m\to\infty} \hat{I}_{1,m}(f) = I(f)$$

gilt und werten daher das Polynom bei der optimalen Schrittweite $H = 0$ aus.

Da $H = 0$ außerhalb des gesicherten Bereichs $[H_1, H_k]$ liegt wird dieses Verfahren (welches auch in anderen Bereichen Anwendung findet) **Extrapolation** genannt. Prinzipiell lässt sich dieses Vorgehen auf beliebige Quadraturen anwenden, es eignet sich allerdings besonders gut für die Tapezsumme, von der es eine asymptotische Entwicklung des Fehlers gibt. Wir betrachten in diesem Abschnitt die summierte Trapezregel (3.4), welche wir einerseits bereits im Rahmen der einführenden Beispiele kennen gelernt haben und von der wir andererseits gesehen haben, dass es eine zusammengesetzte Newton-Cotes Formel ist. Zur Erinnerung (3.4) lautete:

$$T_H(f) = \hat{I}_{1,m}(f) = \frac{H}{2}\left\{f(a) + 2\sum_{i=1}^{m-1} f(x_i) + f(b)\right\}, \quad H := \frac{b-a}{m}.$$

Ausgangspunkt ist nun die folgende Aussage:

Lemma 3.3.1 (Asymptotische Entwicklung des Fehlers der Trapezsumme in H^2) *Sei $f \in C^{2\nu}[a,b]$. Der Fehler der Trapezsumme besitzt eine* asymptotische Entwicklung

$$T_H(f) - I(f) = \tau_2 H^2 + \tau_4 H^4 + \cdots + \tau_{2\nu} H^{2\nu} + \mathcal{O}(H^{2\nu+2}) \tag{3.15}$$

für $T_H(f)$ aus (3.4) mit Koeffizienten

$$\tau_{2\ell} = \frac{B_{2\ell}}{(2\ell)!}(f^{(2\ell-1)}(b) - f^{(2\ell-1)}(a)) \tag{3.16}$$

unabhängig von H und $B_{2\ell}$ sind die Bernoulli-Zahlen*, also*

$$B_0 = 1, \quad \sum_{\nu=0}^{n} \binom{n+1}{\nu} B_\nu = 0. \tag{3.17}$$

Beweis. Stoer, Taylor plus Symmetrie. □

Als Ansatz zur Erhöhung der Ordung wählt man eine Kombination aus zwei Trapezsummen zu unterschiedlichen Schrittweiten und hofft dadurch die Ordnung der asymptotischen Entwicklung des Fehlers zu erhöhen. Ersetzt man in (3.15) $T_H(f)$ z.B. durch

$$\tilde{T}_H(f) := \frac{4}{3}T_H(f) - \frac{1}{3}T_{2H}(f), \tag{3.18}$$

dann gilt

$$\begin{aligned}\tilde{T}_H(f) - I(f) &= \frac{4}{3}(T_H(f) - I(f)) - \frac{1}{3}(T_{2H}(f) - I(f)) \\ &= \frac{4}{3}\tau_2 H^2 + \frac{4}{3}\tau_4 H^4 + \cdots + \mathcal{O}(H^{2\nu+2}) \\ &\quad -\frac{1}{3}\tau_2 4H - \frac{1}{3}\tau_4 16H^4 + \cdots + \mathcal{O}(H^{2\nu+2}) \\ &= \mathcal{O}(H^4).\end{aligned}$$

Durch die geschickte Kombination von $T_H(f)$ und $T_{2H}(f)$ fällt also der führende Term vor H^2 in der asymptotischen Entwicklung weg. Diese Idee kann man analog für allgemeine Schrittweiten H_1, H_2 anstelle von $H, 2H$, oder auch für andere Quadraturformeln realisieren.

Die Frage ist nun, wie macht man das praktisch, d.h., wie kommt man an geeignete Kombinationen wie in (3.18)? Eine Möglichkeit wäre, die entsprechenden Gleichungssysteme aufzustellen und zu lösen, dies ist jedoch unbequem, aufwendig und instabil für $H_1 \approx H_2$! Den allgemeinen Zugang nennt man **Extrapolation**. Extrapolation ist hierbei - wie bereits erwähnt - das Bestimmen eines Zusammenhangs über einen gesicherten Zusammenhang hinaus.

3.3.1 Idee der Extrapolation

Die allgemeine **Idee** lautet wie folgt:

- seien H_1 und H_2 zwei Schrittweiten $(k = 2)$
- sei $P \in \mathbb{P}_{k-1} = \mathbb{P}_1$ ein Interpolationspolynom (Gerade) mit

$$T_H(f) = P(H^2) \text{ für } H = H_0, H_1, \tag{3.19}$$

  also

$$P(H^2) = T_{H_1}(f) + \frac{T_{H_2}(f) - T_{H_1}(f)}{H_2^2 - H_1^2}(H^2 - H_1^2)$$

- *extrapoliere* $P(H)$ auf $H = 0$, also

$$\begin{aligned}P(0) &= \frac{1}{H_2^2 - H_1^2}(T_{H_1}(f)(H_2^2 - H_1^2) - H_1^2(T_{H_2}(f) - T_{H_1}(f)) \\ &= (H_2^2 - H_1^2)^{-1}\left(H_2^2 T_{H_1}(f) - H_1^2 T_{H_2}(f)\right) \\ &= \mathcal{O}(H_1^2 H_2^2),\end{aligned}$$

  falls T_H die asymptotische Entwicklung (3.15) besitzt.

Um das entsprechende Resultat allgemein zu beweisen, benötigen wir noch eine Hilfsaussage.