HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Mathematisches Institut
Lehrstuhl für Angewandte Mathematik
Prof. Dr. Christiane Helzel

Skript zur Vorlesung

Numerik 1

Vorwort

Die Vorlesung zur Numerik 1 wurde im Sommersemester 2014 von Frau Prof. Dr. Christiane Helzel am Mathematischen Institut der Heinrich-Heine-Universität Düsseldorf gehalten. Dieses Vorlesungsskript entstand aus einer Mitschrift der Studentin Carolin Albrecht. Eine Überarbeitung erfolgte im Sommersemester 2016 von Maximilian Schneiders. Es enthält außerdem einige Ergänzungen zur Vorlesung.

Dieses Skript soll stetig erweitert und verbessert werden. Da Fehler, sowohl inhaltlich als auch sprachlich, nicht ausgeschlossen sind, sind Hinweise auf diese und andere Verbesserungsvorschläge, um die Qualität des Skripts zu verbessern, ausdrücklich erwünscht und können an maximilian.schneiders@uni-duesseldorf.de gesendet werden.

Als weitere Literatur zu dieser Vorlesung dienen die Lehrbücher Stoer/Bulirsch, Numerische Mathematik 1, Springer und P. Deuflhard, A. Hohmann, Numerische Mathematik 1, de Gruyter. Für Beispiele in MATLAB kann das Buch Cleve B. Moler, Numerical Computing with MATLAB, SIAM hinzugezogen werden.

Zuletzt bearbeitet: 29. Juni 2016

Inhaltsverzeichnis

1 Fehleranalyse

Ein Computer kann Zahlen nur mit endlich vielen Ziffern darstellen. Dies führt zwangsläufig zu Fehlern in numerischen Algorithmen. In diesem Kapitel beschäftigen wir uns mit der Darstellung von Zahlen in einem Computer sowie mit den dadurch verbundenen Problemen von numerischen Algorithmen.

1.1 Gleitkommadarstellung

Zahlen werden in einem Computer mit Hilfe von normalisierten Gleitkommazahlen realisiert. Es sei $x \in \mathbb{R}$ eine reelle Zahl. Dann lässt sich diese Zahl in der Form

$$x = \pm m \cdot b^{\pm e}$$

schreiben. Dabei bezeichen wir mit m die *Mantisse*, mit b die *Basis* (im Binärsystem ist $b = 2$ und im Dezimalsystem ist $b = 10$) und mit e den *Exponenten*. Durch den ANSI/IEEE Standard (American National Standard Institute, Institute of Electrical and Electronics Engineers, seit 1985) werden im Computer Darstellungen für binäre Gleitkommazahlen definiert, das heißt wir betrachten Zahlen der Form

$$x = \pm(1 + f)2^{c-1023},$$

wobei die Binärdarstellung von f maximal 52 Bits benutzt, das heißt es ist

$$0 \leq 2^{52} \cdot f < 2^{52} \quad \text{mit} \quad 0 \leq f < 1$$

und

$$f = f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + \cdots + f_{52} \cdot 2^{-52}$$

mit $f_i \in \{0, 1\}$. Die Zahl c wird wie folgt charakterisiert: Es werden 11 Bits für den Exponenten vergeben, das heißt wir erhalten

$$c = c_0 \cdot 2^0 + c_1 \cdot 2^1 + \cdots + c_{10} \cdot 2^{10} \quad \text{mit} \quad c_i \in \{0, 1\},$$

also $0 \leq c \leq 2^0 + 2^1 + \cdots + 2^{10} = 2047$ und damit

$$-1023 \leq e \leq 1024.$$

Die Werte $e = -1023$ und $e = 1024$ werden zur Speicherung besonderer „Zahlen“ verwendet:

- $e = 1024$ und $f = 0$: Inf (infinity).
- $e = 1024$ und $f \neq 0$: NaN (not a number).
- $e = -1023$ und $f = 0, c = 0$: Darstellung der Null.

```
>> 1/Inf
ans =
     0

>> Inf+Inf
ans =
   Inf

>> 0/0
ans =
   NaN

>> Inf-Inf
ans =
   NaN
```

Beispiele in MATLAB

Für alle anderen Zahlen gilt $-1022 \leq e \leq 1023$. Gleitkommazahlen (mit doppelter Genauigkeit) werden mit insgesamt 64 Bits gespeichert:

- 52 Bits für die Mantisse,
- 11 Bits für den Exponenten,
- 1 Bit für das Vorzeichen.

Bemerkung. Die Verwendung der Gleitkommadarstellung im numerischen Rechnen ist wesentlich, um Zahlen sehr unterschiedlicher Größe bearbeiten zu können. Des Weiteren liefert die Darstellung von m eine Beschränkung an die Genauigkeit und die Darstellung von e eine Beschränkung der Größenordnung.

Es gelte nun $-1022 \leq e \leq 1023$. Dann gilt für die größte/kleinste Gleitkommazahl

$$x_{\max,\min} = \pm[1 + 2^{-1} + 2^{-2} + \cdots + 2^{-52}] \cdot 2^{1023} \approx \pm 1{,}8 \cdot 10^{308}.$$

Für die kleinste positive/größte negative Gleitkommazahl erhalten wir

$$x_{\text{posmin}} = [1 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 0 \cdot 2^{-52}] \cdot 2^{-1022} \approx 2{,}2 \cdot 10^{-308},$$
$$x_{\text{negmax}} = -2^{-1022} \approx -2{,}2 \cdot 10^{-308}.$$

Mit $A = A(b, f, c)$ bezeichnen wir das *numerische Gleitkommagitter*. Die Menge A ist die Menge der in der Form $x = \pm(1 + f) \cdot b^{c-1023}$ darstellbaren Maschinenzahlen. Dabei beschreibt f die Darstellung der Mantisse und c die Darstellung des Exponenten. Der zulässige Bereich wird definiert als

$$D := [x_{\min}, x_{\text{negmax}}] \cup \{0\} \cup [x_{\text{posmin}}, x_{\max}].$$

```
>> realmin % kleinste positive Zahl
ans =
  2.2251e-308

>> realmax % größte positive Zahl
ans =
  1.7977e+308

>> 2*2^1023 % Overflow
ans =
   Inf

>> 2^-1022
ans =
  2.2251e-308

>> eps*2^-1022
ans =
  4.9407e-324

>> eps*2^-1023
ans =
     0
```

Beispiele in MATLAB

Wir definieren auf D die Rundungsoperation $\mathrm{rd} : D \to A$ mit

$$|x - \mathrm{rd}(x)| = \min_{y \in A}(|x - y|) \quad \text{für alle} \quad x \in D.$$

Beispiel:

$$\mathrm{rd}(x) = \mathrm{sign}(x) \cdot \begin{cases} (1 + f_1 \cdot 2^{-1} + \cdots + f_{52} \cdot 2^{-52}) \cdot 2^{c-1023}, & f_{53} = 0, \\ (1 + f_1 \cdot 2^{-1} + \cdots + f_{52} \cdot 2^{-52} + 2^{-52}) \cdot 2^{c-1023}, & f_{53} = 1. \end{cases}$$

(Man nimmt dabei an, dass der Rechner intern mit mehr Stellen arbeitet.) Dann gilt:

- absoluter Rundungsfehler: $|x - \mathrm{rd}(x)| \leq \frac{1}{2} \cdot 2^{-52} \cdot 2^e$,
- relativer Rundungsfehler: $\left|\frac{x-\mathrm{rd}(x)}{x}\right| \leq \frac{1}{2} \cdot \frac{2^{-52}}{(1+f)\cdot 2^e} < 2^{-53}$,
- maximaler Rundungsfehler $\frac{\mathrm{eps}}{2} = \max\limits_{x\in D, x\neq 0} \left|\frac{\mathrm{rd}(x)-x}{x}\right|$.

Dabei bezeichnen wir mit eps die *Maschinengenauigkeit*. Im IEEE-Format gilt

$$\frac{\mathrm{eps}}{2} = \frac{1}{2} \cdot 2^{-52} = 2^{-53} \approx 1{,}11 \cdot 10^{-16} \quad \text{bzw.} \quad \mathrm{eps} \approx 2{,}2204 \cdot 10^{-16}.$$

Die Maschinengenauigkeit gibt den Abstand von 1 zur nächstgelegenen Gleitkommazahl an bzw. den relativen Abstand zwischen den Gleitkommazahlen.

```
format long;
myeps = 1.d0;

while (1.d0+myeps)>1.d0
    myeps = myeps/2.d0;
end
2*myeps
```

MATLAB-Code 1: Berechnung der Maschinengenauigkeit in MATLAB

Für $x \in D$ gilt $\mathrm{rd}(x) = x(1+\varepsilon)$ mit $|\varepsilon| \leq \mathrm{eps}$. Die Grundoperationen $* \in \{+, -, \cdot, /\}$ werden im Computer durch Maschinenoperationen $\circledast \in \{\oplus, \ominus, \odot, \oslash\}$ ersetzt. Diese sind so definiert: Für $x, y \in A$ und $x * y \in D$ gilt $x \circledast y = (x * y)(1+\varepsilon)$ mit $|\varepsilon| \leq \mathrm{eps}$.

Bemerkung. Für $x, y \in A$ gilt:

(i) $(x \oplus y) \oplus z \neq x \circledast (y \oplus z)$ (kein Assoziativgesetz),

(ii) $(x \oplus y) \odot z \neq (x \odot z) \oplus (y \odot z)$ (kein Distributivgesetz),

(iii) $x \oplus y = x$ für $|y| \leq \frac{|x|}{2}\mathrm{eps}$.

Aus (iii) lässt sich die Größe der Maschinengenauigkeit eps experimentell bestimmen (vergleiche MATLAB-Code 1).

1.2 Konditionierung numerischer Aufgaben

In diesem Abschnitt beschäftigen wir uns mit der Frage, wie sich Störungen in der Eingabegröße auf das Resultat auswirken, unabhängig vom gewählten Algorithmus. Eine numerische Aufgabe wird als *gut konditioniert* bezeichnet, wenn eine kleine Störung der

Eingabedaten auch nur eine kleine Änderung im Ergebnis zur Folge hat. Ansonsten wird die numerische Aufgabe als *schlecht konditioniert* bezeichnet. Unter einer numerischen Aufgabe versteht man die Berechnung endlich vieler Größen y_i $(i = 1, \ldots, n)$ aus Größen x_j $(j = 1, \ldots, m)$ mittels einer funktionalen Vorschrift $y_i = f_i(x_1, \ldots, x_m)$. Schreibweise: $y = f(x)$, $x = (x_1, \ldots, x_m)^T$, $y = (y_1, \ldots, y_m)^T$, $f = (f_1, \ldots, f_m)^T$.

```
clear all;

A = [1.2969 0.8648; 0.2161 0.1441];
b1 = [0.8642; 0.1440];

A\b1
%%
b2 = [0.86419999; 0.14400001];
A\b2
```

MATLAB-Code 2: Beispiel für eine schlecht konditionierte Aufgabe in MATLAB

Der obige MATLAB-Code berechnet die Lösung eines linearen Gleichungssystems. Im ersten Fall ist die Lösung gegeben durch $(x, y) = (2, -2)^T$. Durch geringe Störungen der rechten Seite ergibt sich die Lösung $(x, y) = (0{,}9911, -0{,}4870)^T$, also ein völlig anderes Ergebnis. Diese Aufgabe ist somit schlecht konditioniert.

Definition 1.1. Bei Verwendung fehlerhafter Eingabedaten $x_j + \Delta x_j$ (z.B. aufgrund des Rundungsfehlers) ergeben sich fehlerhafte Resultate $y_i + \Delta y_i$. Wir bezeichnen $|\Delta y_i|$ als den *absoluten Fehler* und $\left|\frac{\Delta y_i}{y_i}\right|$ für $y_i \neq 0$ als den *relativen Fehler*.

Definition 1.2 (Landau-Symbole)**.** Für Funktionen $g(t)$ und $h(t)$ der Variablen $t \in \mathbb{R}_+$ bedeutet die Schreibweise

$$g(t) = \mathcal{O}(h(t)) \quad \text{für} \quad t \to 0,$$

dass für kleine $t \in]0, t_0]$ mit einer Konstanten $c \geq 0$ gilt:

$$|g(t)| \leq c \cdot |h(t)|.$$

Entsprechend bedeutet

$$g(t) = o(h(t)) \quad \text{für} \quad t \to 0,$$

dass für kleine $t \in]0, t_0]$ mit einer Funktion $c(t) \to 0$ für $t \to 0$ gilt:

$$|g(t)| \leq c(t) \cdot |h(t)|.$$

Differentielle Fehleranalyse

Es sei $|\Delta x_j| \ll |x_j|$ (relativ kleiner Datenfehler) und $f_i = f_i(x_1, \ldots, x_m)$ zweimal stetig partiell differentierbar nach den Argumenten x_j. Dann gilt mit Taylorentwicklung:

$$\Delta y_i = f_i(x + \Delta x) - f_i(x) = \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}(x)\Delta x_j + R_i^f(x, \Delta x), \quad i = 1, \ldots, m, \qquad (*)$$

wobei $R_i^f(x, \Delta x) = \mathcal{O}(|\Delta x|^2)$ das Restglied ist. Aus $(*)$ folgt in erster Näherung

$$\Delta y_i \approx \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}(x)\Delta x_j,$$

das heißt Gleichheit gilt in erster Näherung bis auf einen Term der Ordnung $\mathcal{O}\left(|\Delta x|^2\right)$. Für den relativen Fehler (komponentenweise) folgt mit $y_i = f_i(x)$ also

$$\frac{\Delta y_i}{y_i} \approx \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}(x)\frac{\Delta x_j}{y_i} = \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}(x)\frac{x_j}{f_i(x)}\frac{\Delta x_j}{x_j}.$$

Definition 1.3. Die Größen

$$K_{i,j}(x) := \frac{\partial f_i}{\partial x_j}(x)\frac{x_j}{f_i(x)}$$

heißen *relative Konditionszahlen* der Funktion f im Punkt x. Sie sind ein Maß dafür, wie sich kleine relative Fehler in den Ausgangsdaten im Ergebnis auswirken. Man nennt die Aufgabe, $y = f(x)$ aus x zu berechnen, s*chlecht konditioniert*, wenn $|K_{i,j}(x)| \gg 1$ ist, andernfalls *gut konditioniert*.

Grundoperationen

1) Wir betrachten für $x_1, x_2 \in \mathbb{R}$, $x_1 \cdot x_2 \neq 0$ die Addition $y = f(x_1, x_2) = x_1 + x_2$. Dann gilt

$$K_1 = \frac{\partial f}{\partial x_1} \cdot \frac{x_1}{f} = 1 \cdot \frac{x_1}{x_1 + x_2} = \frac{1}{1 + \frac{x_2}{x_1}},$$
$$K_2 = \frac{\partial f}{\partial x_2} \cdot \frac{x_2}{f} = 1 \cdot \frac{x_2}{x_1 + x_2} = \frac{1}{1 + \frac{x_1}{x_2}}.$$

Für $\frac{x_1}{x_2} \approx -1$ (also für die Subtraktion zweier fast gleich großer Zahlen) ist die Addition schlecht konditioniert.

Definition 1.4 (Auslöschung)**.** Unter *Auslöschung* versteht man den Verlust an wesentlichen Dezimalstellen bei der Subtraktion von Zahlen gleichen Vorzeichens. Dies ist gefährlich im Fall, dass eine oder beide Zahlen keine Maschinenzahlen sind und diese vor Ausführung der Operation gerundet werden. Bei der Subtraktion von Maschinenzahlen ist Auslöschung natürlich unschädlich.

Beispiel (Dezimale Gleitpunktrechnung mit vier signifikanten Ziffern). Gegeben seien

$$x_1 = 0{,}11258762 \cdot 10^2, \quad x_2 = 0{,}11244891 \cdot 10^2.$$

Da wir nur vier signifikante Ziffern betrachten, erhalten wir

$$\mathrm{rd}(x_1) = 0{,}1125 \cdot 10^2, \quad \mathrm{rd}(x_2) = 0{,}1124 \cdot 10^2$$

sowie jeweils einen absoluten Fehler von

$$|\Delta x_1| = 0{,}001238, \quad |\Delta x_2| = 0{,}004891.$$

Der relative Fehler in der Eingabe beträgt somit

$$\left|\frac{\Delta x_1}{x_1}\right| = 1{,}099 \cdot 10^{-4}, \quad \left|\frac{\Delta x_2}{x_2}\right| = 4{,}3495 \cdot 10^{-4}.$$

Nun gilt

$$x_1 + x_2 = 0{,}22503653 \cdot 10^2, \quad \mathrm{rd}(x_1) + \mathrm{rd}(x_2) = 0{,}2250 \cdot 10^2$$

und

$$x_1 - x_2 = 0{,}13871 \cdot 10^{-1}, \quad \mathrm{rd}(x_1) - \mathrm{rd}(x_2) = 0{,}2000 \cdot 10^{-1}.$$

Wir berechnen für den relativen Fehler der Subtraktion

$$\left|\frac{(x_1 - x_2) - (\mathrm{rd}(x_1) - \mathrm{rd}(x_2))}{x_1 - x_2}\right| = 0{,}44185.$$

Mit $f(x_1, x_2) = x_1 - x_2$ erhalten wir für die relativen Konditionszahlen die Werte

$$K_1 = \left|\frac{\partial f}{x_1} \cdot \frac{x_1}{f}\right| = \left|\frac{x_1}{x_1 - x_2}\right| = 811{,}67, \quad K_2 = \left|\frac{\partial f}{x_2} \cdot \frac{x_2}{f}\right| = \left|-\frac{x_2}{x_1 - x_2}\right| = 810{,}67,$$

also

$$K_1 \left|\frac{\Delta x_1}{x_1}\right| + K_2 \left|\frac{\Delta x_2}{x_2}\right| = 0{,}44139.$$

Die Subtraktion fast gleich großer Zahlen ist also sehr schlecht konditioniert. Im Gegensatz dazu gilt für die Addition

$$K_1 = 0{,}5003, \quad K_2 = 0{,}4997.$$

2) Betrachten wir nun die Multiplikation $y = f(x_1, x_2) = x_1 \cdot x_2$. Dann gilt

$$K_1 = \frac{\partial f}{\partial x_1} \cdot \frac{x_1}{f} = \frac{x_1 x_2}{x_1 x_2} = 1,$$
$$K_2 = \frac{\partial f}{\partial x_2} \cdot \frac{x_2}{f} = \frac{x_2 x_1}{x_2 x_1} = 1.$$

Das heißt die Multiplikation ist gut konditioniert. Damit ist auch die Division gut konditioniert.

Lösung quadratischer Aufgaben

Betrachte für $p, q \in \mathbb{R}$, $q \neq 0$ und $\frac{p^2}{4} - q > 0$ die quadratische Gleichung

$$y^2 + py + q = 0.$$

Die Lösungsformel für diese Gleichung liefert die Nullstellen

$$y_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

Außerdem gilt $y_1 + y_2 = -p$ und $y_1 y_2 = q$ (Satz von Vieta). Man kann nun zeigen (Übung), dass die Berechnung von y_1 und y_2 für $y_1 \approx y_2$ schlecht konditioniert ist.

1.3 Stabilität numerischer Algorithmen

Betrachte eine numerische Aufgabe $y = f(x)$ mit $f : \mathbb{R}^m \to \mathbb{R}^n$. Ein *Verfahren* oder *Algorithmus* zur Berechnung von y aus x ist eine endliche (oder abzählbar unendliche) Folge von „elementaren" Abbildungen $\varphi^{(k)}$ (z. B. arithmetische Grundoperationen), die durch sukzessive Anwendung einen Näherungswert $\tilde{y}$ zu y liefert.

Definition 1.5. Bei der Durchführung des Algorithmus (auf einer Rechenanlage) treten in jedem Schritt Fehler auf (z. B. Rundungsfehler), die sich bis zum Ende der Rechnung akkumulieren können. Der Algorithmus wird *stabil* (oder gutartig) genannt, wenn die im Verlauf der Ausführung akkumulierten Fehler den durch die Konditionierung der Aufgabe bedingten unvermeidbaren Problemfehler nicht übersteigen.

Beispielsweise sei das Polynom $p(x) = (x-1)^7$ gegeben. Abbildung 1.1 zeigt den Plot von p in einer kleinen Umgebung der 7-fachen Nullstelle $x = 1$. Während die Auswertung der Faktorisierung $p(x) = (x-1)^7$ einen stabilen Algorithmus liefert, ist der

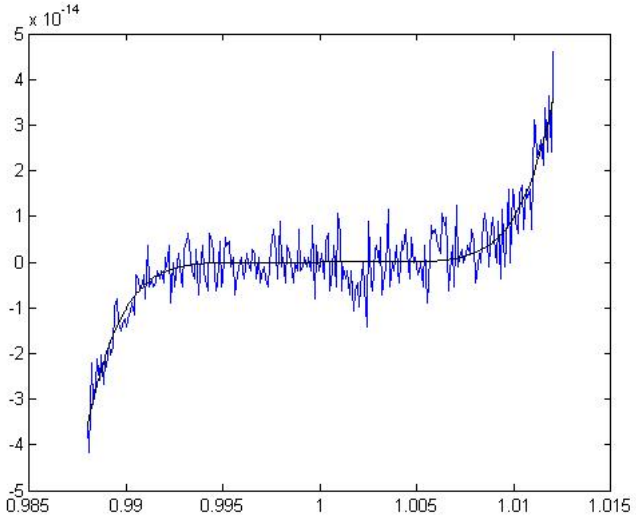

Abbildung 1.1: Plot eines Polynoms siebten Grades, das auf zwei verschiedene Weisen berechnet wurde.

Algorithmus, der das Polynom in der Form

$$p(x) = \sum_{k=0}^{7} a_k x^k = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

auswertet, sehr instabil.

```
x = 0.988:0.0001:1.012;
y1 = x.^7 - 7*x.^6 + 21*x.^5 -35*x.^4 + 35*x.^3 -21*x.^2 + 7*x -1;
plot(x,y1)
%%

y2 = (x-1).^7;
hold on;
plot(x,y2,'k')
```

MATLAB-Code 3: Berechnung eines Polynoms siebten Grades auf zwei verschiedene Weisen.

Eine der Hauptaufgaben der numerischen Mathematik ist es, für die in den Anwendungen auftretenden Aufgaben, stabile Lösungsalgorithmen zu finden. Kommen wir nun nochmal zurück zur Lösung quadratischer Gleichungen.

Für $q \neq 0$, $p > 0$ und $q < \frac{p^2}{4}$ sind die Lösungen von $y^2 + py + q = 0$ gegeben durch

$$y_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

Betrachte nun den Fall $|\frac{y_2}{y_1}| \gg 1$, das heißt $q \ll \frac{p^2}{4}$. Damit ist die Berechnung von y_2 gut konditioniert. Ein Algorithmus zur Berechnung von y_1 und y_2 ist dann wie folgt:

1. $u = \left(\frac{p}{2}\right)^2$,
2. $v = u - q$,
3. $w = \sqrt{v} \quad (\geq 0)$.

Da $p > 0$ ist, wird zur Vermeidung von Auslöschung zunächst

$$y_2 = -\frac{p}{2} - w = f(p, w)$$

berechnet.

Fehlerfortpflanzung

Betrachten wir als erstes den relativen Fehler von y_2. Dann gilt mit

$$\frac{\partial f}{\partial p} \cdot \frac{p}{f} = -\frac{1}{2} \cdot \frac{p}{-\frac{p}{2} - w} = \frac{1}{1 + \frac{2w}{p}} \quad \text{und} \quad \frac{\partial f}{\partial w} \cdot \frac{w}{f} = -1 \cdot \frac{w}{-\frac{p}{2} - w} = \frac{1}{\frac{p}{2w} + 1}$$

die Abschätzung

$$\begin{aligned} \left|\frac{\Delta y_2}{y_2}\right| &\leq \left|\frac{\partial f}{\partial p} \cdot \frac{p}{f}\right| \left|\frac{\Delta p}{p}\right| + \left|\frac{\partial f}{\partial w} \cdot \frac{w}{f}\right| \left|\frac{\Delta w}{w}\right| \\ &= \left|\frac{1}{1 + \frac{2w}{p}}\right| \left|\frac{\Delta p}{p}\right| + \left|\frac{1}{1 + \frac{p}{2w}}\right| \left|\frac{\Delta w}{w}\right| \\ &< 1 \cdot \left|\frac{\Delta p}{p}\right| + 1 \cdot \left|\frac{\Delta w}{w}\right|. \end{aligned}$$

Das heißt die Fehlerforpflanzung in diesem Schritt ist akzeptabel.

Für die Berechnung von y_1 betrachten wir jetzt zwei Varianten:

- Variante A: Berechne $y_1 = -\frac{p}{2} + w$ nach der Lösungsformel,
- Variante B: Berechne $y_1 = \frac{q}{y_2}$ mit Hilfe des Satzes von Vieta.

Für die erste Variante gilt $y_1 = f(p, w) = -\frac{p}{2} + w$. Mit

$$\frac{\partial f}{\partial p} \cdot \frac{p}{f} = -\frac{1}{2} \cdot \frac{p}{-\frac{p}{2} + w} = \frac{1}{1 - \frac{2w}{p}} \quad \text{und} \quad \frac{\partial f}{\partial w} \cdot \frac{w}{f} = 1 \cdot \frac{w}{-\frac{p}{2} + w} = \frac{1}{1 - \frac{p}{2w}}$$

ergibt sich für den relativen Fehler von y_1 dann

$$\left|\frac{\Delta y_1}{y_1}\right| \leq \left|\frac{1}{1-\frac{2w}{p}}\right| \left|\frac{\Delta p}{p}\right| + \left|\frac{1}{1-\frac{p}{2w}}\right| \left|\frac{\Delta w}{w}\right|.$$

Da $\frac{1}{1-\frac{p}{2w}} \gg 1$ und $\frac{1}{1-\frac{2w}{p}} \gg 1$, ist dieser Algorithmus für $q \ll \frac{p^2}{4}$ sehr instabil.

Für die zweite Variante hingegen setzen wir $y_1 = f(q, y_2) = \frac{q}{y_2}$. Diesmal berechnen wir

$$\frac{\partial f}{\partial q} \cdot \frac{q}{f} = \frac{1}{y_2} \cdot \frac{q}{\frac{q}{y_2}} = 1 \quad \text{und} \quad \frac{\partial f}{\partial y_2} \cdot \frac{y_2}{f} = \frac{q}{y_2^2} \cdot \frac{y_2}{\frac{q}{y_2}} = 1.$$

Dann lässt sich der relative Fehler von y_1 abschätzen durch

$$\left|\frac{\Delta y_1}{y_1}\right| \leq \left|\frac{\Delta q}{q}\right| + \left|\frac{\Delta y_2}{y_2}\right|.$$

Da der relative Fehler von y_2 klein ist, ist auch der relative Fehler von y_1 klein. Dieser Algorithmus ist also stabil.

Bemerkung. Bei der Lösung quadratischer Gleichungen sollten nicht beide Wurzeln aus der Lösungsformel berechnet werden.

Integral-Rekursion

Aufgabe: Berechne

$$I_n = \frac{1}{e} \int_0^1 x^n e^x \, \mathrm{d}x \quad \text{für} \quad n = 0, 1, \ldots, 30.$$

Partielle Integration liefert uns

$$\begin{aligned} I_n &= \frac{1}{e} \int_0^1 x^n e^x \, \mathrm{d}x \\ &= \frac{1}{e} \left[x^n e^x\right]_0^1 - \int_0^1 n x^{n-1} e^x \, \mathrm{d}x \\ &= \frac{1}{e} \left(e - n \int_0^1 x^{n-1} e^x \, \mathrm{d}x \right) \\ &= 1 - n I_{n-1}. \end{aligned}$$

Damit erhalten wir die Zwei-Term-Rekursion

$$I_n = 1 - nI_{n-1}.$$

Als Startwert berechnen wir

$$I_0 = \frac{1}{e} \int_0^1 e^x \, \mathrm{d}x = \frac{1}{e}(e-1) \approx 0{,}63212\ldots$$

Ausgehend von I_0 lassen sich aus dieser Formel beliebige Integrale I_n für $n > 0$ berechnen.

```
I(0)  =  6.3212e-001
I(5)  =  1.4553e-001
I(10) =  8.3877e-002
I(15) =  5.9034e-002
I(16) =  5.5459e-002
I(17) =  5.7192e-002
I(18) = -2.9454e-002
I(19) =  1.5596e+000
I(20) = -3.0192e+001
I(21) =  6.3504e+002
I(22) = -1.3970e+004
```

Einige Integrale der Integral-Rekursion. Ab I_{18} bekommt man völlig falsche Werte.

Wir erhalten

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \,\mathrm{d}x \geq 0, \quad \text{da} \quad x^n e^x > 0 \quad \text{für alle} \quad x \in (0,1).$$

Wegen $e^x < e$ für alle $x \in (0,1)$ können wir weiterhin

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \,\mathrm{d}x \leq \frac{1}{e} e \int_0^1 x^n \,\mathrm{d}x = \left[\frac{x^{n+1}}{n+1}\right]_0^1 = \frac{1}{n+1}$$

abschätzen und bekommen somit $0 \leq I_n \leq \frac{1}{n+1}$ für alle $n > 0$. Damit sehen wir, dass die Vorwärtsrekursion ab I_{18} deutlich falsche Ergebnisse liefert.

Es sei nun I_n der exakte Wert und $\tilde{I}_n$ ein mit Rundungsfehler behafteter Wert. Weiterhin sei $\Delta I_0 = \tilde{I}_0 - I_0$ der Fehler im Startwert, etwa $|\Delta I_0| \approx 10^{-16}$. Dann gilt $I_n = 1 - nI_{n-1}$ und $\tilde{I}_n = 1 - n\tilde{I}_{n-1}$. Daraus folgt $\tilde{I}_n - I_n = -n\left(\tilde{I}_{n-1} - I_{n-1}\right)$, also $\Delta I_n = -n\Delta I_{n-1}$. Sukzessives Einsetzen liefert

$$\Delta I_n = -n\Delta I_{n-1} = -n(-(-n-1))\Delta I_{n-2} = \cdots = (-1)^n n!\Delta I_0$$

und wir erhalten die Fehlerformel $\Delta I_n = (-1)^n n!\Delta I_0$. Dies erklärt das alternierende Vorzeichen für großes n. Für $n = 18$ haben wir $18!10^{-16} = 0{,}64023$ und für $n = 19$ haben wir bereits $19!10^{-16} = 12{,}16$. Der Algorithmus ist also instabil.

Eine Alternative zur Berechnung der Integrale ist die Rückwärtsrekursion $I_{n-1} = \frac{1-I_n}{n}$. Das Problem hier: Der Startwert ist unbekannt. Wir können aber $I_n = \frac{1}{n+1}$ benutzen. Die Fehlerformel liefert $\Delta I_0 = \frac{1}{(-1)^n \cdot n!}\Delta I_n$. Auf diese Weise liefert MATLAB den Wert $I_0 \approx 0{,}632120558828555$, wohingegen der „exakte“ Wert $\frac{e-1}{e} = 0{,}632120558828558$ ist. Der Algorithmus ist also stabil.

```
%% Integral-Rekursion:
%% Vorwärtsrekursion
I = zeros(30,1);
I(1) = (exp(1)-1.)/exp(1);
for i=2:30
    I(i) = 1 - (i-1)*I(i-1);
end

%% Rückwärtsrekursion
n0=15;
I = zeros(n0,1);
I(n0) = 1./(n0+1);
for i=n0-1:-1:1
    I(i) = (1.-I(i+1))/(i);
end
```

MATLAB-Code 4: Vorwärts- und Rückwärtsrekursion für die Integral-Rekursion in MATLAB.

2 Interpolation und Approximation

Häufig tritt in der numerischen Mathematik die Situation auf, dass statt einer Funktion $f : \mathbb{R} \to \mathbb{R}$ nur einige diskrete Funktionswerte $f(x_i)$ und eventuell noch Ableitungen $f^{(j)}(x_i)$ an endlich vielen Punkten x_i gegeben sind. Zum Beispiel, wenn f nur in Form von experimentellen Daten $f(x_0), \ldots, f(x_n)$ vorliegt und man an weiteren Funktionswerten zwischen den tabellierten Werten interessiert ist. Das Ziel ist es also, aus den gegebenen Daten eine Funktion φ zu konstruieren, die

$$\varphi^{(j)}(x_i) = f^{(j)}(x_i)$$

für alle i und j erfüllt und sich möglichst wenig von f unterscheidet, also $\|\varphi - f\|$ „klein“ und φ leicht auswertbar. Leicht auswertbar sind beispielsweise Funktionen aus P, wobei P eine Klasse von einfach strukturierten Funktionen ist, wie etwa

- Polynome $p(x) = a_0 + a_1 x + \cdots + a_n x^n$,
- rationale Funktionen $r(x) = \frac{a_0 + \cdots + a_n x^n}{b_0 + \cdots + b_n x^n}$,
- oder trigonometrische Polynome $t(x) = \frac{1}{2}a_0 + \sum_{k=1}^{b} \{a_k cos(kx) + b_k sin(kx)\}$.

Definition 2.1. Geschieht die Zuordnung eines Elements $\varphi \in P$ zur Funktion f durch Fixierung von Funktionswerten

$$\varphi(x_i) = f(x_i), \quad i = 0, \ldots, n,$$

so spricht man von *Interpolation*. Ist $\varphi \in P$ als in einem gewissen Sinne „beste“ Darstellung von f zu bestimmen, zum Beispiel

$$\max_{a \le x \le b} |f(x) - \varphi(x)| \quad \text{minimal für} \quad \varphi \in P$$

oder

$$\left(\int_a^b |f(x) - \varphi(x)|^2 \, \mathrm{d}x \right)^{1/2} \quad \text{minimal für} \quad \varphi \in P,$$

so spricht man allgemein von *Approximation*. Die jeweilige Wahl der Konstruktion von $\varphi \in P$ hängt von der zu erfüllenden Aufgabe ab. Offenbar ist die Interpolation eine spezielle Art der Approximation mit

$$\max_{i=0,\ldots,n} |f(x) - \varphi(x)| \quad \text{minimal für} \quad \varphi \in P.$$

2.1 Polynominterpolation

In diesem Abschnitt betrachten wir $P = \mathbb{P}_n$, den Vektorraum der Polynome vom Grad kleiner oder gleich n, also

$$\mathbb{P}_n = \{p(x) = a_0 + a_1 x + \cdots + a_n x^n \mid a_i \in \mathbb{R}, i = 0, \ldots, n\}.$$

Definition 2.2. Die Polynom-Interpolationsaufgabe besteht darin, zu $n+1$ paarweise verschiedenen Stützstellen (auch Knoten genannt) $x_0, \ldots, x_n \in \mathbb{R}$ und gegebenen Knotenwerten $y_0, \ldots, y_n \in \mathbb{R}$, ein Polynom $p \in \mathbb{P}_n$ mit der Eigenschaft

$$p(x_i) = y_i, \quad i = 0, \ldots, n \qquad \text{(Interpolationsbedingung)}$$

zu bestimmen.

Satz 2.3. *Die Polynom-Interpolationsaufgabe ist eindeutig lösbar.*

Beweis. Eindeutigkeit: Seien $p, q \in \mathbb{P}_n$ zwei interpolierende Polynome mit

$$p(x_i) = q(x_i), \quad i = 0, \ldots, n.$$

Dann ist $p - q \in \mathbb{P}_n$ ein Polynom mit den $n+1$ Nullstellen $x_0, \ldots, x_n$, da alle Knoten paarweise verschieden sind. Aus dem Satz von Rolle folgt damit sofort, dass $p - q$ das Nullpolynom ist, also $p = q$.

Existenz: Wir wollen ein Polynom $p(x) = a_n x^n + \cdots + a_1 x + a_0$ konstruieren, das die Interpolationsaufgabe erfüllt. Die Koeffizienten erhalten wir als Lösung des linearen Gleichungssystems

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}}_{=:V_n} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Dieses lineare Gleichungssystem hat genau dann eine eindeutige Lösung, wenn $\det(V_n) \neq 0$. Man kann zeigen (siehe Übung), dass

$$\det(V_n) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_j - x_i),$$

also $\det(V_n) \neq 0$ genau dann, wenn die Knoten paarweise verschieden sind. □

Bemerkung. Für größere Gleichungssysteme ist die Vandermonde-Matrix V_n im Allgemeinen schlecht konditioniert. Daher ist dieser Ansatz ungeeignet zur Berechnung des Interpolationspolynoms.

2.1.1 Darstellung des Interpolationspolynoms

Es seien $x_0 < x_1 < \cdots < x_n$ paarweise verschiedene Knoten.

Satz 2.4 (Lagrange-Darstellung)**.** *Das (eindeutige) Interpolationspolynom*

$$p_n(x) := p(f|x_0, \ldots, x_n) \in \mathbb{P}_n \quad \textit{mit} \quad p_n(x_j) = f(x_j), \quad j = 0, \ldots, n$$

lässt sich in der Form

$$p_n(x) = \sum_{j=0}^{n} f(x_j) l_{jn}(x) \quad \textit{mit} \quad l_{jn}(x) = \prod_{k \neq j}^{n} \frac{x - x_k}{x_j - x_k}, \quad j = 0, \ldots, n$$

schreiben. Die $l_{jn}(x)$ heißen Lagrange-Fundamentalpolynome*.*

Beweis. Es gilt $l_{jn} \in \mathbb{P}_n$ für alle j und somit $p_n \in \mathbb{P}_n$. Weiterhin ist $l_{jn}(x_i = \delta_{ij}$, wobei δ_{ij} das Kronecker-Symbol ist. Und schließlich folgt damit

$$p_n(x_i) = \sum_{j=0}^{n} f(x_j) l_{jn}(x_i) = f(x_i).$$

Also löst p_n die Interpolationsaufgabe. Die Eindeutigkeit folgt mit Satz 2.3. □

Der Vorteil dieser Darstellung liegt in der Nützlichkeit für theoretische Betrachtungen. Der Nachteil dieser Darstellung hingegen ist, dass bei Hinzunahme einer neuen Stützstelle die gesamte Darstellung neu berechnet werden muss. Außerdem ist für Stützstellen $x_i \approx x_j$ diese Darstellung sehr ungenau. Dies führt uns zu einer alternativen Darstellung, der Newton-Darstellung.

Lemma 2.5. *Für die Lagrange-Interpolationspolynome*

$$p_{n-1} = p(f|x_0, \ldots, x_{n-1}) \in \mathbb{P}_{n-1} \quad \textit{und} \quad p_n = p(f|x_0, \ldots, x_n) \in \mathbb{P}_n$$

gilt

$$p_n(x) = p_{n-1}(x) + \delta_n w_n(x)$$

mit

$$w_n(x) = \prod_{i=0}^{n-1} (x - x_i) \in \mathbb{P}_n \quad \textit{und} \quad \delta_n = \frac{f(x_n) - P_{n-1}(x_n)}{w_n(x_n)}.$$

Hierbei nennt man $\{w_i \mid i = 0, \ldots, n\}$ die Newton-Basis *von $\mathbb{P}_n$ mit*

$$w_0 = 1, w_1 = (x - x_0), w_2 = (x - x_0)(x - x_1), \ldots$$

Beweis. Per Definition gilt $p_{n-1} \in \mathbb{P}_{n-1}$ und $w_n \in \mathbb{P}_n$. Damit folgt

$$q_n(x) = p_{n-1}(x) + \delta_n w_n(x) \in \mathbb{P}_n.$$

Es bleibt zu zeigen, dass q die Interpolationsbedingung an den Stützstellen $x_0, \ldots, x_n$ erfüllt. Für $i = 0, \ldots, n-1$ gilt

$$q_n(x_i) = p_{n-1}(x_i) + \delta_n w_n(x_i) = p_{n-1}(x_i) = f(x_i),$$

da $w_n(x_i) = 0$ nach Definition. Es sei nun $i = n$. Dann ist

$$\begin{aligned} q_n(x_n) &= p_{n-1}(x_n) + \delta_n w_n(x_n) \\ &= p_{n-1}(x_n) + \left(\frac{f(x_n) - p_{n-1}(x_n)}{w_n(x_n)} \right) w_n(x_n) \\ &= f(x_n). \end{aligned}$$

Falls $\delta_n = 0$ ist, so folgt bereits $p_{n-1}(x_n) = f(x_n)$ und Satz 2.3 liefert schließlich $q_n = p_n$ und damit die Eindeutigkeit des Interpolationspolynoms von f an den Stützstellen $x_0, \ldots, x_n$. □

Der Koeffizient δ_n hängt von f und den Stützstellen x_i ab. Daher hat sich auch die Schreibweise $\delta_n = [x_0, \ldots, x_n]f$ oder $f[x_0, \ldots, x_n]$ eingebürgert. Eine wiederholte Anwendung der Argumentation in Lemma 2.5 liefert uns den folgenden Satz.

Satz 2.6 (Newtonsche Darstellung der Interpolationspolynoms)**.** *Das Interpolationspolynom zu f und Knoten $x_0, \ldots, x_n$ ist gegeben durch*

$$p(f|x_0, \ldots, x_n) = \sum_{i=0}^{n} ([x_0, \ldots, x_i]f) w_i(x) \quad \textit{mit} \quad w_i(x) = \prod_{j=0}^{i-1} (x - x_j),$$

also

$$p(f|x_0, \ldots, x_n) = [x_0]f + [x_0, x_1]f w_1(x) + \cdots + [x_0, \ldots, x_n]f w_n(x).$$

Lemma 2.7 (Aitken)**.** *Es gilt*

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

Die Interpolierende an $x_0, \ldots, x_n$ ist also eine lineare Interpolation zwischen zwei Interpolationspolynomen auf $x_1, \ldots, x_n$ und $x_0, \ldots, x_{n-1}$.

Beweis. Wir setzen x_i in die rechte Seite ein und schauen uns die Terme für verschiedene Werte von i getrennt an. Es sei also zunächst $0 < i < n$. Dann gilt:

$$\begin{aligned} &\frac{x_i - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x_i) + \frac{x_n - x_i}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x_i) \\ &= \frac{x_i - x_0}{x_n - x_0} f(x_i) + \frac{x_n - x_i}{x_n - x_0} f(x_i) = f(x_i) = p(f|x_0, \ldots, x_n)(x_i). \end{aligned}$$

Für den Fall $i = 0$ fällt der erste Summand weg und wir erhalten

$$\frac{x_n - x_0}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x_0) = p(f|x_0, \ldots, x_{n-1})(x_0) = f(x_0).$$

Analog fällt für $i = n$ der zweite Summand weg und es folgt

$$\frac{x_n - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x_0) = p(f|x_1, \ldots, x_n)(x_0) = f(x_0).$$

Also stimmen beide Seiten für alle $x_0, \ldots, x_n$ überein und sind eindeutiges Interpolationspolynom vom Grad kleiner oder gleich n. □

Bemerkung 2.8. i) Für paarweise verschiedene x_i gilt

$$[x_0, \ldots, x_n]f = \frac{[x_1, \ldots, x_n]f - [x_0, \ldots, x_{n-1}]f}{x_n - x_0}.$$

Diesen Ausdruck nennt man *dividierte Differenz* der Ordnung n von f.

ii) Die dividierte Differenz $[x_0, \ldots, x_n]f$ ist der führende Koeffizient a_n des Interpolationspolynoms

$$p(f|x_0, \ldots, x_n)(x) = a_n x^n + \cdots + a_1 x + a_0.$$

Beweis. Zu i): Nach Lemma 2.7 gilt

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

Setze die Newton-Darstellung für das Interpolationspolynom ein und führe Koeffizientenvergleich für den führenden Koeffizienten durch. Die Newton-Darstellung ist

$$p(f|x_0, \ldots, x_n)(x) = \sum_{i=0}^{n} ([x_0, \ldots, x_i]f) w_i(x) \quad \text{mit} \quad w_i(x) = \prod_{j=0}^{i-1} (x - x_j).$$

Koeffizientenvergleich liefert uns jetzt

$$[x_0, \ldots, x_n]f = \frac{1}{x_n - x_0}[x_1, \ldots, x_n]f - \frac{1}{x_n - x_0}[x_0, \ldots, x_{n-1}]f.$$

Zu ii): Folgt aus Lemma 2.5. □

Schema zur Berechnung der dividierten Differenzen für gegebene Startwerte $[x_i]f = f(x_i)$ für $i = 0, \ldots, n$:

$$
\begin{array}{llll}
x_0 & [x_0]f & & \\
 & & [x_0, x_1]f & \\
x_1 & [x_1]f & & [x_0, x_1, x_2]f \\
 & & [x_1, x_2]f & & [x_0, x_1, x_2, x_3]f \\
x_2 & [x_2]f & & [x_1, x_2, x_3]f \\
 & & [x_2, x_3]f & \\
x_3 & [x_3]f & &
\end{array}
$$

Algorithmus 2.9 (Berechnung dividierter Differenzen).

```
function d = DivDiff(x,f)
    n = size(x,1);

    %Matrix in der die div. Differenzen gespeichert werden.
    d = zeros(n);

    %Schritt 1: Setze erste Spalte.
    d(:,1)=f(x);

    %Schritt 2: Berechne restliche Werte.
    for k = 2:n
        for i = 1:n-k+1
            d(i,k) = (d(i+1,k-1) - d(i,k-1))/(x(i+k-1)-x(i));
        end
    end
end
```

MATLAB-Code 5: Berechnung dividierter Differenzen

Beachte: Zur effizienten algorithmischen Beschreibung wird ein doppelter Index benutzt.

Algorithmus 2.10 (Auswertung der Newton-Interpolationsformel (Horner-Schema)).

```
        %Berechne div. Differenzen.
        D=DivDiff(x,f);

        %Vektor fuer die Loesung.
        Y = zeros(size(X,1),1);

        %Berechnung des Interpolationspolynoms (Alg. 2.10)
        Y(:) = D(1,end);
        for k = size(D,1)-1:-1:1
            Y(:) = D(1,k) + (X(:)-x(k)).*Y(:);
        end
```

MATLAB-Code 6: Auswertung der Newton-Interpolationsformel basierend auf Algorithmus 2.9

Aufwand: Es werden $n^2/2$ Divisionen zur Berechnung von $\Delta_{0,i}$ benötigt (wird einmal durchgeführt) und n Multiplikationen zur Berechnung von y^* (wird für jeden Auswertungspunkt x^* durchgeführt) benötigt.

Bemerkung 2.11. Für die dividierten Differenzen gilt:

(i) $[x_0, \ldots, x_n]f$ ist unabhängig von der Reihenfolge der x_i.

(ii) Ist $f \in C^n(I, \mathbb{R})$, $I = [a, b]$, $a \leq x_0 < x_1 < \cdots < x_n \leq b$, so gibt es ein $\eta \in I$ mit $[x_0, \ldots, x_n]f = \frac{1}{n!} f^{(n)}(\eta)$.

(iii) Ist $p \in \mathbb{P}_{n-1}$, so gilt $[x_0, \ldots, x_n]p = 0$

Beweis. Zu i): Folgt aus der Newton-Darstellung des Interpolationspolynoms.

Zu ii): Sei $g := f - p(f|x_0, \ldots, x_n) \in C^n(I, \mathbb{R})$. Dann hat g die $n+1$ Nullstellen $x_0, \ldots, x_n$. Aus dem Satz von Rolle folgt die Existenz eines $\eta \in I$ mit

$$0 = g^{(n)}(\eta) = f^{(n)}(\eta) - p(f|x_0, \ldots x_n)^{(n)}(\eta) = f^{(n)}(\eta) - n![x_0, \ldots, x_n]f.$$

Zu iii): Ist klar, denn der n-te Koeffizient eines Polynoms vom Grad kleiner oder gleich $n - 1$ verschwindet. □

2.1.2 Verfahrensfehler der Polynominterpolation

Wir stellen uns die Frage, wie stark das Interpolationspolynom von der zu interpolierenden Funktion abweicht. Das Interpolationspolynom erkennt nicht, von welcher Funktion die Daten kommen. Für festes $\overline{x} \in [x_0, x_n]$ gilt aber immer

$$f(\overline{x}) = P(f|x_0, \ldots, x_n, \bar{x})(\overline{x}) \quad \text{mit} \quad P(f|x_0, \ldots, x_n, \overline{x}) \in \mathbb{P}_{n+1}.$$

Daraus folgt mit Lemma 2.5

$$\begin{aligned} f(x) - P(f|x_0, \ldots, x_n)(x) &= P(f|x_0, \ldots, x_n, x)(x) - P(f|x_0, \ldots, x_n)(x) \\ &= [x_0, \ldots, x_n, x]f \cdot w_{n+1}(x) \end{aligned}$$

mit

$$w_{n+1}(x) = \prod_{i=0}^{n} (x - x_i).$$

Wir wollen die linke Seite abschätzen. Nun gibt es nach Bemerkung 2.11 (ii) ein $\xi \in I = [a, b]$, $a \leq x_0$, $b \geq x_n$ mit

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

falls $f \in C^{n+1}(I, \mathbb{R})$. Damit erhalten wir die folgende Abschätzung für den Interpolationsfehler:

Satz 2.12. *Sei $f \in C^{n+1}([x_0, x_n], \mathbb{R})$. Dann existiert ein $\xi(x) \in [x_0, x_n]$ mit*

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

Insbesondere gilt:

$$\|f - P(f|x_0, \ldots x_n)\|_\infty \leq \|w_{n+1}\|_\infty \frac{\|f^{(n+1)}\|_\infty}{(n+1)!},$$

wobei $\|f\|_\infty = \max\limits_{x\in[x_0,x_n]} |f(x)|$ ist.

Bemerkung. (i) Der Faktor $\|f^{(n+1)}\|_\infty$ hängt von f, aber nicht von den x_i ab,

(ii) und $w_{n+1} = \prod\limits_{j=0}^{n}(x - x_j)$ hängt von den Stützstellen ab, nicht jedoch von f.

Eine grobe Abschätzung erhält man durch $|w_{n+1}(x)| \leq (b-a)^{n+1}$ für alle $x \in [a, b]$. Hierbei geht jedoch der Einfluss der Wahl der Sütztstellen verloren.

Bemerkung. Wählt man als Stützstellen für die Interpolationsaufgabe auf $[-1, 1]$ die Nullstellen der Tschebyscheff-Polynome, so wird $\|w_{n+1}\|_{\infty,[-1,1]}$ unter allen (normierten) Polynomen mit reellen Nullstellen minimal (Übung).

Die *Tschebyscheff-Polynome* sind eine spezielle Folge von Polynomen T_k, die bezüglich eines speziellen gewichteten Skalarprodukts auf $[-1, 1]$ orthogonal sind, das heißt

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} \cdot T_n(x) \cdot T_m(x)\, \mathrm{d}x = \begin{cases} 0, & m \leq n, \\ \pi, & n = m = 0, \\ \frac{\pi}{2} & n = m \neq 0. \end{cases}$$

Explizit lautet das k-te Polynom

$$T_k(x) = \cos(k \cdot \arccos(x)), \quad x \in [-1, 1].$$

Die Tschebyscheff-Polynome lassen sich auch durch eine Drei-Term-Rekursion berechnen und es gilt

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2, \quad T_0(x) = 1, \quad T_1(x) = x.$$

Des Weiteren sind die Nullstellen von $T_k(x)$ gegeben durch

$$x_j = \cos\left(\frac{2j+1}{2k}\pi\right), \quad j = 0, \ldots, k-1.$$

Grenzen der Polynominterpolation

Durch Erhöhung der Anzahl der Stützstellen n kann man nicht automatisch eine beliebig genaue Approximation an f bezüglich der $\|\cdot\|_\infty$-Norm erreichen. Runge hat hierfür ein Gegenbeispiel angegeben. Betrachte

$$f(x) = \frac{1}{1+x^2} \in C^\infty, \quad x \in [-c, c], \quad c > 0$$

mit äquidistanten Stützstellen

$$x_i = -c + \frac{2c}{n} i, \quad i = 0, \ldots, n.$$

Es gilt:

- $\|f - P_n\|_\infty \to \infty$ für $n \to \infty$, falls $c > \frac{e}{2}$,
- $\|f - P_n\|_\infty \to \infty$ für $n \to \infty$, falls $c \leq \frac{e}{2}$.

Im divergenten Fall treten insbesondere an den Intervallrändern immer stärkere Oszillationen auf. Die bezeichnet man auch als das Runge-Phänomen.

Bemerkung. Der Weierstraßsche Approximationssatz besagt, dass jede Funktion $f \in C([a, b])$ beliebig gut gleichmäßig auf $[a, b]$ durch Polynome approximiert werden kann. Die Vermutung, dass dies mit Interpolationspolynomen geschehen kann, ist jedoch im Allgemeinen falsch.

```
% Interpolation
% Runge Testproblem

c_vec = [exp(1)/2+3 exp(1)/2-.5];    % c.
n_vec = [3 10 20 40];                  % n.

f = @(x) 1./(1+x.^2);                %Funktion.

for n = n_vec
p=1;                                  %Subplot-Nummer.
figure;
for c = c_vec
    a = -c;
    b = c;
    X=[-c:2*c/100:c]';                % X fuer den Plot.

    for j = 1:2
        if j==1
            % Aequidistant
            TITLE = [ 'Aequidistant', ' mit c = ',num2str(c), ', n = '...
```

```
                ,num2str(n)];
            x=[-c:2*c/n:c]';
        else
            % Tschebyscheff
            TITLE = ['Tschebyscheff', ' mit c = ',num2str(c), ', n = '...
                ,num2str(n)];
            x=zeros(n,1);
            for i=1:n
                xi = cos(pi*(2*i-1)/(2*n));
                x(i) = a + (b - a)/2*(xi+1);
            end
        end

        %Berechne div. Differenzen.
        D=DivDiff(x,f);

        %Vektor fuer die Loesung.
        Y = zeros(size(X,1),1);

        %Berechnung des Interpolationspolynoms (Alg. 2.10)
        Y(:) = D(1,end);
        for k = size(D,1)-1:-1:1
            Y(:) = D(1,k) + (X(:)-x(k)).*Y(:);
        end

        %Plot
        subplot(2,2,p)
        plot(X,f(X),'-k',X,Y,'-.')
        title(TITLE)
        legend('f(f)','P(x)');
        p=p+1;
    end
end
end
```

MATLAB-Code 7: Runge-Phänomen

2.1.3 Die baryzentrische Darstellung des Interpolationspolynoms

Um das Problem des Runge-Phänomens zu umgehen, kann man noch eine andere Darstellung des Interpolationspolynoms betrachten, die baryzentrische Darstellung. Wir haben zuvor die Lagrange-Darstellung

$$p(x) = \sum_{j=0}^{n} l_{jn}(x) f_j \quad \text{mit} \quad l_{jn}(x) = \prod_{k \neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)}$$

betrachtet, wobei $l_{jn}(x_k) = \delta_{jk}$ gilt.

Vor- und Nachteile:

+ Ideal, um etwas zu beweisen.
- Auswertung von $p(x)$ benötigt $\mathcal{O}(n^2)$ Rechenoperationen.
- Hinzufügen von (x_{n+1}, f_{n+1}) ändert alles.
- Numerisch instabil.

Weiterhin haben wir auch die Newton-Darstellung

$$p(x) = \sum_{j=0}^{n} [x_0, \ldots, x_j] f \prod_{i=0}^{j-1} (x - x_i)$$

mit

$$[x_j, \ldots, x_k]f = \frac{[x_{j+1}, \ldots, x_k]f - [x_j, \ldots, x_{k-1}]f}{x_k - x_j} \quad \text{und} \quad [x_j]f = f_j, \quad j = 0, \ldots, n$$

kennengelernt.

Vor- und Nachteile:

+ Auswertung von $p(x)$ benötigt $\mathcal{O}(n)$ Rechenoperationen.
+ Hinzufügen von (x_{n+1}, f_{n+1}) ist leicht umsetzbar.
- Rekursive Formel für $[x_0, \ldots, x_j]f$ benötigt $\mathcal{O}(n^2)$ Operationen.
- $[x_0, \ldots, x_j]f$ ist abhängig von der Funktion f.

Modifizierte Lagrange-Darstellung

Es sei nun

$$l(x) := (x - x_0) \cdot \cdots \cdot (x - x_n) \quad \text{mit Stützkoeffizienten} \quad \lambda_j = \prod_{k \neq j}^{n} \frac{1}{(x_j - x_k)} = \frac{1}{l'(x_j)}.$$

Jetzt können wir die Lagrange-Basispolynome darstellen durch

$$l_{jn}(x) = \prod_{k \neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)} = l(x) \frac{\lambda_j}{x - x_j}$$

und erhalten für das Interpolatinspolynom

$$p(x) = l(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j} f_j.$$

Diese Darstellung hat die folgenden Eigenschaften:

- Einmalig $\mathcal{O}(n^2)$ Rechenoperationen zur Berechnung von λ_j (unabhängig von f_j).
- $\mathcal{O}(n)$ Rechenoperationen zur Berechnung von $p(x)$.
- Hinzufügen von (x_{n+1}, f_{n+1}):
  - Dividiere λ_j durch $x_j - x_{n+1} \to n+1$ Operationen.
  - Berechne $\lambda_{n+1} \to n+1$ Operationen.

Mit Hilfe der λ_j und $f(x) \equiv 1$ lässt sich dann $l(x)$ berechnen durch

$$1 = \sum_{j=0}^{n} l_{jn}(x) = l(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j},$$

also

$$l(x) = \frac{1}{\sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}}.$$

Wir haben also gezeigt:

Satz 2.13 (Baryzentrische Interpolationsformel)**.** *Das Interpolationspolynom $p \in \mathbb{P}_n$ zu den $(n+1)$ Knoten $x_0, \ldots, x_n$ und den $(n+1$ Daten $f_0, \ldots, f_n$ hat die Form*

$$p(x) = \begin{cases} \sum_{j=0}^{n} \frac{\lambda_j f_j}{x - x_j} \Big/ \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}, & x \neq x_j, \\ f_j, & x = x_j. \end{cases}$$

Die Stützkoeffizienten haben die Form

$$\lambda_j = \prod_{k \neq j}^{n} \frac{1}{x_j - x_k}.$$

Baryzentrische Interpolation in Tschebyscheff-Knoten

Die Tschebyscheff-Knoten 1. Art sind definiert durch die Nullstellen der Tschebyscheff-Polynome

$$x_j = \cos\left(\frac{(2j+1)\pi}{2n+2}\right), \quad 0 \leq j \leq n$$

Dann erhalten wir für die Gewichte unter Vernachlässigung aller von j unabhängigen Bestandteile

$$\lambda_j = (-1)^j \sin\left(\frac{(2j+1)\pi}{2n+2}\right).$$

In praktischen Rechnungen benutzt man meist die Tschebyscheff-Knoten 2. Art, die gegeben sind durch

$$x_j = \cos(j\pi/n), \quad 0 \le j \le n.$$

In diesem Fall erhält man die Gewichte

$$\lambda_j = (-1)^j \delta_j, \quad \text{mit} \quad \delta_j = \begin{cases} \frac{1}{2}, & j = 0, n, \\ 1, & \text{sonst.} \end{cases}$$

Das Interpolationspolynom hat dann die Form

$$p(x) = \sum_{j=0}^{n}{}' \frac{(-1)^j f_j}{x - x_j} \Big/ \sum_{j=0}^{n}{}' \frac{(-1)^j}{x - x_j}, \quad \text{und} \quad p(x_j) = f_j, \quad j = 0, \ldots, n.$$

Dabei bedeutet $\sum'$, dass der erste und letzte Summand jeweils mit $1/2$ multipliziert werden. Im Gegensatz zu den vorherigen Darstellungen des Interpolationspolynoms, liefert die baryzentrische Darstellung einen stabilen Algorithmus zur Berechnung des Interpolationspolynoms.

```
clear all;
close all;
n = 10000;
%fun = @(x) abs(x)+0.5*x-x.^2;
fun = @(x) 1./(1+x.^2);
cc = exp(1)/2;
x = cos(pi*(0:n)'/n)*cc;
f = fun(x);
xref = linspace(-1,1,5000)*cc;
fref = fun(xref);

c = [1/2; ones(n-1,1); 1/2].*(-1).^((0:n)');
xx = cc*linspace(-1,1,5*n)';
numer = zeros(size(xx));
denom = zeros(size(xx));
for j = 1:n+1
   xdiff = xx-x(j);
   temp = c(j)./xdiff;
   numer = numer + temp*f(j);
   denom = denom + temp;
end
ff = numer./denom;
plot(xref,fref,'k-',xx,ff,'ro',x,fun(x),'b+');
```

MATLAB-Code 8: Das Beispiel von Runge mit baryzentrischer Darstellung

2.2 Hermite-Interpolation

Definition 2.14. Die Hermite-Interpolationsaufgabe lautet:

gegeben:	Knoten	$x_0 < x_1 < \cdots < x_m,$	$m \geq 0,$
	Werte	$y_i^{(k)} \in \mathbb{R},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i,$
			$\mu_i \geq 0$ für $0 \leq i \leq m,$
gesucht:	$p \in \mathbb{P}_n, n = m + \sum_{i=0}^{m} \mu_i,$	mit $p(x_i)^k = y_i^{(k)},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i.$

Satz 2.15. *Die Hermite-Interpolationsaufgabe besitzt eine eindeutige Lösung.*

Beweis. Der Beweis wird analog zu dem von Satz 2.3 geführt (Übung). □

Die Hermite-Interpolation ist eine Verallgemeinerung der klassischen Polynom-Interpolation, bei der man die Interpolation von Ableitungen formal als zusammenfallende Stützstellen interpretiert, das heißt $a \leq x_0 \leq x_1 \leq \cdots \leq x_n \leq b$. Sind in einem Knoten x_i die Funktionswerte $f(x_i)$ und die Ableitungen $f'(x_i), \ldots, f^{(\mu_i)}(x_i)$ bis zum Grad μ_i gegeben, so soll x_i genau $(\mu_i + 1)$-mal auftreten. Sind alle Knoten paarweise verschieden, so erhalten wir die klassische Polynom-Interpolation. Stimmen alle Knoten überein, das heißt $x_0 = x_1 = \cdots = x_n$, so ist das Interpolationspolynom die abgebrochene Taylor-Reihe

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

um x_0. Wir können das Interpolationspolynom wieder in der Newton-Darstellung

$$p = \sum_{i=0}^{n} [x_0, \ldots, x_i] f \cdot w_i, \quad w_i = \prod_{j=0}^{i-1} (x - x_j)$$

schreiben. Beachte: Die bisherige Definition der dividierten Differenzen können wir nicht ohne Weiteres verwenden, da Stützstellen doppelt auftreten können. Die Kombination von klassischer Interpolation und Taylor-Reihe ergibt

$$[x_i, \ldots, x_{i+k}] f = \frac{f^{(k)}(x_i)}{k!}, \qquad \text{falls} \quad x_i = x_{i+k},$$
$$[x_i, \ldots, x_{i+k}] f = \frac{[x_{i+1}, \ldots, x_{i+k}] f - [x_i, \ldots, x_{i+k-1}] f}{x_{i+k} - x_i} \qquad \text{sonst.}$$

Beispiel. Finde das Hermite-Interpolationspolynom $p \in \mathbb{P}_4$ mit

$$\begin{aligned} &p(0) = f(0) = -1, && p'(0) = f'(0) = -2, \\ &p(1) = f(1) = 0, && p'(1) = f'(1) = 10, && p''(1) = f''(1) = 40. \end{aligned}$$

Wir verdoppeln den ersten und verdreifachen den zweiten Knoten und erhalten die dividierten Differenzen

$$\begin{array}{llllll} x_0 = 0 & [x_0]f = -1 & & & & \\ & & [x_0,x_1]f = -2 & & & \\ x_1 = 0 & [x_1]f = -1 & & [x_0,x_1,x_2]f = 3 & & \\ & & [x_1,x_2]f = 1 & & [x_0,\dots,x_3]f = 6 & \\ x_2 = 1 & [x_2]f = 0 & & [x_1,x_2,x_3]f = 9 & & [x_0,\dots,x_4]f = 5 \\ & & [x_2,x_3]f = 10 & & [x_1,\dots,x_4]f = 11 & \\ x_3 = 1 & [x_3]f = 0 & & [x_2,x_3,x_4]f = 20 & & \\ & & [x_3,x_4]f = 10 & & & \\ x_4 = 1 & [x_4]f = 0 & & & & \end{array}$$

Daraus ergibt sich das Hermite-Interpolationspolynom als

$$\begin{aligned} P(f|x_0,\dots,x_4) &= -1 - 2(x-x_0) + 3(x-x_0)(x-x_1) \\ &\quad + 6(x-x_0)(x-x_1)(x-x_2) + 5(x-x_0)(x-x_1)(x-x_2)(x-x_3) \\ &= -1 - 2x + 3x^2 + 6x^3(x-1) + 5x^4(x-1)^2. \end{aligned}$$

Satz 2.16 (Fehler der Hermite-Interpolation)**.** *Ist $f \in C^{n+1}([x_0, x_m])$, so gibt es zu jedem $x \in [x_0, x_m]$ ein $\xi(x) \in [x_0, x_m]$, so dass für die Lösung $p \in \mathbb{P}_n$ der Hermite-Interpolationsaufgabe*

$$\begin{aligned} f(x) - p(x) &= [x_0,\dots,x_0,\dots,x_m,\dots x_m,x] f \prod_{i=0}^{m} (x-x_i)^{\mu_i+1} \\ &= \frac{1}{(n+1)!} f^{(n+1)}(\xi(x)) \prod_{i=0}^{m} (x-x_i)^{\mu_i+1} \end{aligned}$$

gilt.

Beweis. Der Beweis funktioniert wie in Satz 2.12. □

2.3 Spline-Interpolation

Im Folgenden sei $I = [a, b]$ ein Intervall mit $a < b$. Weiter sei X eine Zerlegung von I, das heißt $X : a = x_0 < x_1 < \cdots < x_n = b$, $n \in \mathbb{N}$. Es sei $f_j \in \mathbb{R}$ für $j = 0, \dots, n$.

Definition 2.17. Es sei $k \in \mathbb{N}$. Dann heißt

$$S_k(X) = \left\{ S \in C^{k-1}(I) \mid 1 \leq j \leq n, s|_{[x_{j-1},x_j]} \in \mathbb{P}_k \right\}$$

Raum der polynomialen Spline-Funktionen oder *Splines* vom Grad k auf der Zerlegung X.

Beispiel. (i) Für $k = 1$ erhält man stetige Polygonzüge. Die Interpolationsaufgabe lautet: Finde $s \in S_1(X)$ mit $s(x_j) = f_j$ für $j = 0, \ldots, n$.

(ii) Für $k = 2$ erhält man quadratische Splines (wenig genutzt), also stückweise Polynome 2. Grades, die global einmal stetig differenzierbar sind. Beachte: Die Interpolationsaufgabe $s(x_j) = f_j$ für $j = 0, \ldots, n$ enthält eine Interpolationsbedingung zu wenig. Man kann zum Beispiel $s'(x_0) = f'(x_0)$ oder $s'(x_n) = f'(x_n)$ zusätzlich angeben.

(iii) Für $k = 3$ erhält man kubische Splines (häufig genutzt), also stückweise Polynome 3. Grades, die global zweimal stetig differenzierbar sind. Beachte: Hier enthält die Interpolationsaufgabe zwei Bedingungen zu wenig. Wähle zum Beispiel $s'(x_0) = f'(x_0)$ und $s'(x_n) = f'(x_n)$ als zusätzliche Interpolationsbedingungen.

Dimensionsbetrachtungen

Gesucht ist $s \in S_3(X)$ mit $s(x_j) = f_j$ für $j = 0, \ldots n$. Da wir Polynome 3. Grades betrachten, haben wir in jedem Teilintervall $[x_i, x_{i+1}]$ von I genau 4 Parameter. Dies liefert uns insgesamt $4n$ Parameter für das gesamte Intervall I. Durch die Stetigkeitsbedingungen $s^{(i)}(x_j - 0) = s^{(i)}(x_j + 0)$ für $i = 0, 1, 2$ und $j = 1, \ldots, n-1$ haben wir bereits $3(n-1)$ Bedingungen gegeben. Durch die Interpolationsbedingungen $s(x_j) = f_j$ für $j = 0, \ldots, n$ erhalten wir weitere $n + 1$ Bedingungen. Wir können also noch

$$4n - 3(n-1) - (n+1) = 2$$

weitere Bedingungen stellen.

Woher kommt das Interesse an diesen Funktionen? Ingenieure benutzten früher Splines, das heißt „dünne Holzlatten“ bei der Konstruktion (zum Beispiel im Schiffsbau). Man zwang Splines durch bestimmte Knotenpunkte. Dadurch stellte sich für Rumpflinien von Schiffen eine günstige Kurve ein. Die Holzlatte nimmt eine Lage mit minimaler potentieller Energie an, das heißt

$$\int_a^b \frac{y''(x)^2}{(1 + y'(x)^2)^{3/2}} \, \mathrm{d}x$$

wird minimal. Falls $|y'(x)|$ klein ist, erhalten wir näherungsweise

$$\int_a^b y''(x)^2 \, \mathrm{d}x$$

minimal. Außerhalb von $[a, b]$ verläuft die Latte linear, das heißt wegen $y \in C^2(\mathbb{R})$ gilt $y''(x) = 0$ in $(-\infty, a]$ und $[b, \infty)$. Insbesondere gilt $y''(x_0) = y''(x_n)$ (natürliche Randbedingung). Daraus ergibt sich die

Aufgabenstellung 2.18. *Es seien die Zerlegung X und Daten $f_0, \ldots, f_n$ gegeben und es sei*

$$A = \{f \in C^2(I) \mid f(x_j) = f_j,\ j = 0, \ldots, n\}.$$

Gesucht ist $\tilde{f} \in A$ mit $E(\tilde{f}) \leq E(f)$ für alle $f \in A$. Dabei sei

$$E(f) := \int_a^b (f''(x))^2 \, \mathrm{d}x$$

(dies ist nur eine Halbnorm, denn $|f| = 0 \Leftrightarrow f = 0$ ist nicht erfüllt).

2.3.1 Berechnung kubischer Splines

Satz 2.19. *Es existiert eine Lösung $\tilde{f} \in A$ von Problem 2.18. Für dieses $\tilde{f}$ gilt $\tilde{f} \in S_3(X)$ und $f''(x_0) = f''(x_n) = 0$. Diese Lösung ist eindeutig.*

Beweis. Zur Minimaltiät: Es seien $f, s \in A$. Dabei sei $s \in S_3(X)$ ein natürlicher kubischer Spline (das heißt ein kubischer Spline mit $s''(x_0) = s''(x_n) = 0$). Wir müssen zeigen, dass $E(s) \leq E(f)$ gilt. Es gilt

$$\begin{aligned}
0 \leq E(f - s) &= \int_a^b (f''(x) - s''(x))^2 \, \mathrm{d}x \\
&= \int_a^b (f''(x))^2 \, \mathrm{d}x - 2 \int_a^b (f''(x) s''(x)) \, \mathrm{d}x + \int_a^b (s''(x))^2 \, \mathrm{d}x \\
&= \int_a^b (f''(x))^2 \, \mathrm{d}x - 2 \int_a^b (f''(x) - s''(x)) s''(x) \, \mathrm{d}x - \int_a^b (s''(x))^2 \, \mathrm{d}x.
\end{aligned}$$

Wir zeigen nun, dass der zweite Term verschwindet, womit $E(s) \leq E(f)$ gezeigt wäre. Definiere $I_j = (x_{j-1}, x_j)$. Dann gilt

$$\int_a^b (f''(x) - s''(x)) s''(x) \, \mathrm{d}x = \sum_{j=1}^n \int_{I_j} (f''(x) - s''(x)) s''(x) \, \mathrm{d}x.$$

Wegen $s \in C^\infty(I_j)$ erhalten wir mit partieller Integration und der Tatsache, dass s''' konstant ist

$$\begin{aligned}\int_{I_j}(f''-s'')s''\,\mathrm{d}x &= \big[(f'-s')s''\big]_{x_{j-1}}^{x_j} - \int_{I_j}(f'-s')s'''\,\mathrm{d}x \\ &= \big[(f'-s')s''\big]_{x_{j-1}}^{x_j} - s'''\big((f-s)(x_j)-(f-s)(x_{j-1})\big) \\ &= \big[(f'-s')s''\big]_{x_{j-1}}^{x_j},\end{aligned}$$

da der zweite Term aufgrund der Interpolationsbedingungen verschwindet. Folglich erhalten wir

$$\int_a^b (f''-s'')s''\,\mathrm{d}x = \sum_{j=1}^{n}\big[(f'-s')s''\big]_{x_{j-1}}^{x_j} = \big[(f'-s')s''\big]_{x_0}^{x_n} = 0,$$

da s stetig auf I ist und wegen $s''(x_0) = s''(x_n) = 0$. Also gilt

$$0 \le \int_a^b (f''-s'')^2\,\mathrm{d}x = \int_a^b (f'')^2\,\mathrm{d}x - \int_a^b (s'')^2\,\mathrm{d}x$$

für alle $f \in A$. Das heißt aber gerade, dass s die Lösung von Problem 2.18 ist, denn $E(s) \le E(f)$. Wir haben also bisher gezeigt, dass wenn es eine Lösung von Problem 2.18 gibt, diese ein natürlicher kubischer Spline ist.

Zur Eindeutigkeit: Es seien $s, \tilde{s} \in S_3(X) \cap A$ Lösungen von Problem 2.18. Dann gilt: sowohl $E(s) \le E(\tilde{s})$ (nach obiger Rechnung), als auch $E(\tilde{s}) \le E(s)$, also $E(\tilde{s}) = E(s)$. Dann folgt mit obiger Rechnung sowohl $E(s) \le E(\tilde{s})$ als auch $E(\tilde{s}) \le E(s)$ und damit $E(\tilde{s}) = E(s)$. Außerdem liefert die obige Rechnung

$$E(\tilde{s}-s) = \int_a^b (\tilde{s}''-s'')^2\,\mathrm{d}x = \int_a^b (\tilde{s}'')^2\,\mathrm{d}x - \int_a^b (s'')^2\,\mathrm{d}x = E(\tilde{s}) - E(s) = 0.$$

Also gilt $\tilde{s}''(x) - s''(x) = 0$ für alle $x \in [a,b]$. Mit Integration folgt nun $\tilde{s}(x) = s(x) + cx + d$ mit $c, d \in \mathbb{R}$. Wegen der Interpolationsbedingung sind s und $\tilde{s}$ mindestens an x_0 und x_n gleich. Daraus folgt wegen

$$\begin{aligned}\tilde{s}(x_0) &= s(x_0) + cx_0 + d, \\ \tilde{s}(x_n) &= s(x_n) + cx_n + d,\end{aligned}$$

dass $cx_0 + d = 0 = cx_n + d$, woraus wiederum $cx_0 = cx_n$ folgt und wegen $x_0 \neq x_n$ schließlich $c = 0$ und damit auch $d = 0$, also $s = \tilde{s}$.

Zur Existenz (konstruktiv): Wir wollen schließlich noch zeigen, dass es einen natürlichen kubischen Spline gibt. Im folgenden werden wir aber die Konstruktion eines

solchen Splines etwas allgemeiner darstellen und auch andere Interpolationsprobleme lösen.

Es sei $s \in S_3(X)$ und s'' ist in jedem Teilintervall ein Polynom 1. Grades und global ein stetiger Polygonzug, also

$$s''(x)\Big|_{I_j} = a_j x + b_j, \quad a_j, b_j \in \mathbb{R}, \quad j = 1, \ldots, n.$$

Setze $M_j := s''(x_j)$ für $j = 0, \ldots, n$ und $h_j = x_j - x_{j-1}$ für $j = 1, \ldots, n$. Dann folgt

$$s''(x)\Big|_{I_j} = \frac{1}{h_j}\big(M_{j-1}(x_j - x) + M_j(x - x_{j-1})\big),$$

wegen der Stetigkeit von s''. Es gilt also

$$\begin{aligned} s'(x) &= \frac{1}{h_j}\left(M_j\frac{(x-x_{j-1})^2}{2} - M_{j-1}\frac{(x_j-x)^2}{2}\right) + c_j && \text{in } I_j, \\ s(x) &= \frac{1}{h_j}\left(M_j\frac{(x-x_{j-1})^3}{6} + M_{j-1}\frac{(x_j-x)^3}{6}\right) + c_j\,(x-x_{j-1}) + d_j && \text{in } I_j. \end{aligned}$$

Die Stetigkeit der ersten Ableitung in x_j für $j = 1, \ldots, n-1$ liefert $s'(x_j - 0) = s'(x_j + 0)$, woraus

$$\frac{h_j}{2}M_j + c_j = -M_j\frac{h_{j+1}}{2} + c_{j+1} \tag{$*$}$$

folgt und die Interpolationsbedingungen $s(x_j) = f(x_j)$ für $j = 0, \ldots, n$ ergeben

$$\begin{aligned} f_{j-1} &= \frac{h_j^2}{6}M_{j-1} + d_j && \text{in } I_j, \\ f_j &= \frac{h_j^2}{6}M_j + c_j h_j + d_j && \text{in } I_j \end{aligned}$$

und damit

$$\frac{f_j - f_{j-1}}{h_j} = \frac{h_j}{6}(M_j - M_{j-1}) + c_j. \tag{$**$}$$

Folglich gilt

$$c_j = [x_{j-1}, x_j]\,f - \frac{h_j}{6}(M_j - M_{j-1}), \quad d_j = f_{j-1} - \frac{h_j^2}{6}M_{j-1}, \quad j = 1, \ldots, n.$$

Setze nun c_j in $(*)$ ein und erhalte

$$h_j M_{j-1} + 2(h_j + h_{j+1})M_j + h_{j+1}M_{j+1} = 6\left([x_j, x_{j+1}]\,f - [x_{j-1}, x_j]\,f\right), \quad j = 1, \ldots, n-1.$$

Dies sind $n-1$ lineare Gleichungen für $n+1$ Unbekannte $M_0, \ldots, M_n$. Dieses Gleichungssystem wird in praktischen Rechnungen benutzt. Für theoretische Betrachtungen dividieren wir noch durch $h_j + h_{j+1} = x_{j+1} - x_{j-1}$ und erhalten

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = 6\,[x_{j-1}, x_j, x_{j+1}]\,f \tag{GS}$$

mit

$$\mu_j = \frac{h_j}{h_j + h_{j+1}}, \quad \lambda_j = \frac{h_{j+1}}{h_j + h_{j+1}}, \quad j = 1, \ldots, n-1,$$

wobei $\mu_j, \lambda_j > 0$ und $\mu_j + \lambda_j = 1$ gilt. □

Möglichkeiten zur Bestimmung der M_i

1. Fall: $M_0 = M_n = 0$ (natürliche Splines)

Das $(n-1)\times(n-1)$-Gleichungssystem mit den Unbekannten $M_1, \ldots, M_{n-1}$ ist gegeben durch

$$\begin{pmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ \\ \vdots \\ \\ M_{n-1} \end{pmatrix} = 6 \begin{pmatrix} [x_0, x_1, x_2]\, f \\ \\ \vdots \\ \\ [x_{n-2}, x_{n-1}, x_n]\, f \end{pmatrix}$$

2. Fall: Zusätzliche Interpolationsbedingungen (vollständige Randbedingungen)

Wir fordern zusätzlich $s'(x_0) = f'(x_0)$ und $s'(x_n) = f'(x_n)$. Für $j = 1$ folgt aus der Interpolationsbedingung $(**)$

$$[x_0, x_1]\, f = \frac{f_1 - f_0}{h_1} = \frac{h_1}{6}(M_1 - M_0) + c_1$$

und aus

$$s'(x) = \frac{1}{h_j}\left[M_j \frac{(x - x_j)^2}{2} - M_{j-1}\frac{(x_j - x)^2}{2}\right] + c_j \quad \text{in} \quad I_j$$

folgt für $x = x_0$ und $j = 1$ die Gleichung

$$s'(x_0) = f'(x_0) = [x_0, x_0]\, f = \frac{1}{h_1}\left[-M_0 \frac{h_1^2}{2}\right] + c_1 = -M_0 \frac{h_1}{2} + c_1.$$

Damit erhalten wir

$$[x_0, x_0, x_1]\, f = \frac{[x_0, x_1]\, f - f'(x_0)}{x_1 - x_0} = \frac{1}{6}(M_1 - M_0) + \frac{c_1}{h_1} + \frac{M_0}{2} - \frac{c_1}{h_1} = \frac{1}{6}(M_1 + 2M_0).$$

Eine analoge Gleichung wird aus der Bedingung $s'(x_n) = f'(x_n)$ hergeleitet. Zusätzlich zu (GS) erhalten wir somit

$$2M_0 + M_1 = 6\,[x_0, x_0, x_1]\, f,$$
$$M_{n-1} + 2M_n = 6\,[x_{n-1}, x_n, x_n]\, f,$$

also das $(n+1) \times (n+1)$-Gleichungssystem

$$\begin{pmatrix} 2 & 1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-2} \\ & & & 1 & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ \\ \vdots \\ \\ M_{n-1} \end{pmatrix} = 6 \begin{pmatrix} \delta_0 \\ \\ \vdots \\ \\ \delta_n \end{pmatrix}$$

mit

$$\delta_0 = [x_0, x_0, x_1]\, f, \quad \delta_n = [x_{n-1}, x_n, x_n]\, f, \quad \delta_j = [x_{j-1}, x_j, x_{j+1}]\, f, \quad j = 1, \ldots, n-1.$$

3. Fall: Periodische Daten $f_n = f_0$

In diesem Fall setzen wir das Gitter mit der Periode $(b-a)$ periodisch auf $\mathbb{R}$ fort. Wir erhalten dadurch aus der Periodizität von $s \in C^2$ zwei zusätzliche Gleichungen $s'(x_0) = s'(x_n)$ und $s''(x_0) = s''(x_n)$. Es folgt $M_0 = M_n$ und damit verbleiben n Unbekannte $M_1, \ldots, M_{n-1}, M_n = M_0$. Folglich erhalten wir eine zusätzliche Bestimmungsgleichung

$$\mu_n M_{n-1} + 2M_n + \lambda_n M_{n+1} = 6\tilde{\delta}_n,$$

wobei $\tilde{\delta}_n = [x_{n-1}, x_n, x_{n+1}]\, f$ und $x_{n+1} = x_1 + (b-a)$. Das Gleichungssystem lautet dann:

$$\begin{pmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-2} \\ \lambda_n & & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ \\ \vdots \\ \\ M_n \end{pmatrix} = 6 \begin{pmatrix} \delta_1 \\ \\ \vdots \\ \\ \tilde{\delta}_n \end{pmatrix}$$

mit δ_i wie in Fall 2.

4. Fall: not-a-knot-Spline

Als zusätzliche Bedingungen setzen wir $s'''(x_1 - 0) = s'''(x_1 + 0)$ und $s'''(x_{n-1} + 0) = s'''(x_{n+1} + 0)$, das heißt s ist auf $[x_0, x_2]$ und $[x_{n-2}, x_n]$ ein Polynom 3. Grades, x_1 und x_{n-1} sind also keine Knoten im eigentlichen Sinne. Es gilt

$$s'''(x) = \frac{1}{h_j} = \frac{1}{h_j}(M_j - M_{j-1})$$

und damit

$$s'''(x) = \frac{1}{h_1}(M_1 - M_0) \quad \text{in } I_1,$$
$$s'''(x) = \frac{1}{h_2}(M_2 - M_1) \quad \text{in } I_2.$$

Die erste zusätzliche Bedingung liefert uns

$$\frac{1}{h_1}(M_1 - M_0) = \frac{1}{h_2}(M_2 - M_1),$$

woraus wir

$$M_0\lambda_1 - M_1 + M_2\mu_1 = 0$$

erhalten. Analog ergibt sich für I_{n-1} und I_n mit der zweiten zusätzlichen Bedingung

$$M_{n-2}\lambda_{n-1} - M_{n-1} + M_n\mu_{n-1} = 0.$$

Insgesamt ist das Gleichungssystem dann durch

$$\begin{pmatrix} \lambda_1 & -1 & \mu_1 & & & & \\ \mu_1 & 2 & \lambda_1 & & & & \\ & \mu_2 & 2 & \lambda_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\ & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & \lambda_{n-1} & -1 & \mu_{n-1} \end{pmatrix} \begin{pmatrix} M_1 \\ \\ \\ \vdots \\ \\ \\ M_n \end{pmatrix} = 6 \begin{pmatrix} 0 \\ \delta_1 \\ \\ \vdots \\ \\ \delta_n \\ 0 \end{pmatrix}$$

gegeben, wobei auch hier δ_i wie in den Fällen zuvor definiert ist.

Nun ergibt sich die Frage, ob diese linearen Gleichungssysteme lösbar sind.

Definition 2.20. Eine Matrix $A \in \mathbb{R}^{nxn}$ (oder $A \in \mathbb{C}^{nxn}$) heißt *(strikt) diagonaldominant*, falls

$$|a_{ii}| > \sum_{k=1, k\neq i}^{n} |a_{ik}|, \quad i = 1, \ldots, n.$$

Sie heißt *schwach diagonaldominant*, wenn stattdessen

$$|a_{ii}| \geq \sum_{k=1, k\neq i}^{n} |a_{ik}|, \quad i = 1, \ldots, n$$

und für mindestens einen Index die strikte Ungleichung gilt.

Lemma 2.21. *Es sei $A \in \mathbb{R}^{nxn}$ (oder $A \in \mathbb{C}^{nxn}$) eine strikt diagonaldominante Matrix. Dann ist A invertierbar und es gilt:*

$$\|Az\|_\infty \geq c\|z\|_\infty \quad \textit{für alle} \quad z \in \mathbb{R}^n \quad (\textit{oder } z \in \mathbb{C}^n)$$

mit

$$c := \min_{1\leq i\leq n} \left(|a_{ii}| - \sum_{j\neq i}^{n} |a_{ij}| \right).$$

Beweis. Zum Beweis benutzen wir die Aussage A ist invertierbar genau dann, wenn $Ax = 0$ nur die triviale Lösung besitzt. Es seien $x \in \mathbb{R}^n \setminus \{0\}$ und $b = Ax$. Weiter sei $k \in \{1, \ldots, n\}$ so gewählt, dass $\|x\|_\infty = |x_k|$. Dann gilt

$$\begin{aligned}
\|b\|_\infty = \|Ax\|_\infty &= \max_{1\leq i\leq n} \left| \sum_{j=1}^{n} a_{ij}x_j \right| \\
&\geq \sum_{j=1}^{n} a_{kj}x_j = \left| a_{kk}\frac{x_k}{|x_k|} + \sum_{j\neq k}^{n} a_{kj}\frac{x_j}{|x_k|} \right| \|x\|_\infty \\
&\geq \left(|a_{kk}| - \sum_{j\neq k}^{n} |a_{kj}| \right) \|x\|_\infty \\
&\geq \min_{1\leq i\leq n} \left(|a_{ii}| - \sum_{j\neq i}^{n} |a_{ij}| \right) \|x\|_\infty = c\|x\|_\infty > 0.
\end{aligned}$$

Also folgt aus $x \neq 0$, dass $Ax = b \neq 0$ und damit ist A invertierbar. Außerdem gilt $\|Ax\|_\infty \geq c\|x\|_\infty$ für alle $x \in \mathbb{R}^n$. Der Beweis funktioniert für $x \in \mathbb{C}^n$ analog. □

Bemerkung. Mit Lemma 2.21 folgt die Lösbarkeit für die Spline-Interpolation mit natürlichen, vollständigen und periodischen Randbedingungen.

Lemma 2.22. *Das im 4. Fall auftretende („not-a-knot") lineare Gleichungssystem ist eindeutig lösbar.*

Beweis. Wir wollen zeigen, dass $Ax = 0$ nur die triviale Lösung besitzt. Dazu formen wir die Matrix A aus dem 4. Fall zunächst so um, dass wir die Matrix

$$\tilde{A} = \begin{pmatrix}
1 & 1 & 1 & & & & \\
0 & \mu_1 - 2 & \mu_1 - \lambda_1 & & & & \\
 & \mu_2 & 2 & \lambda_2 & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\
 & & & & \lambda_{n-1} - \mu_{n-1} & \lambda_{n-1} - 2 & 0 \\
 & & & & 1 & 1 & 1
\end{pmatrix}$$

erhalten. Bezeichne A_i die i-Zeile von A. Dann gilt nämlich $\tilde{A}_1 = A_1 + A_2$ und unter Verwendung von $\mu_1 + \lambda_1 = 1$ gilt $\tilde{A}_2 = \mu_1 A_1 - \lambda_1 A_2$. Analog erhält man die letzten beiden Zeilen von $\tilde{A}$. Wir betrachten nun $\tilde{A}x = 0$. Dann folgt aus

$$\left(\begin{array}{c|ccccc|c} 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ \hline 0 & & & & & & 0 \\ \vdots & & & A_{22} & & & \vdots \\ 0 & & & & & & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 & 1 & 1 \end{array}\right) \begin{pmatrix} x_1 \\ \vec{x_2} \\ x_3 \end{pmatrix} = 0$$

mit Lemma 2.21 zunächst $\vec{x}_2 = 0$, da A_{22} strikt diagonaldominant ist.. Weiterhin sehen wir dann aber anhand der ersten und letzten Zeile, dass gerade $x_1 + 0 = 0$ und $x_3 + 0 = 0$ ist. Also hat $Ax = 0$ nur die triviale Lösung und A ist invertierbar. □

Im folgenden Satz fassen wir nochmal alle betrachteten Interpolationsprobleme zusammen.

Satz 2.23. *Die Interpolationsprobleme mit kubischen*

1) natürlichen Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &M_0 = M_n = 0 \quad (\textit{oder } M_0, M_n \textit{ beliebig}), \end{aligned}$$

2) vollständigen Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'(x_0) = f'(x_0), \\ &s'(x_n) = f'(x_n), \end{aligned}$$

3) periodischen Splines

$$\begin{aligned} &f(x_0) = f(x_n), \\ &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'(x_0) = s'(x_n), \\ &s''(x_0) = s''(x_n), \end{aligned}$$

4) not-a-knot-Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'''(x_1 - 0) = s'''(x_1 + 0), \\ &s'''(x_{n-1} - 0) = s'''(x_{n-1} + 0) \end{aligned}$$

sind stets eindeutig lösbar.

Beweis. Die entsprechenden lineare Gleichungssysteme sind nach Lemma 2.21 und Lemma 2.22 eindeutig lösbar. Folglich existiert s'' eindeutig. Also existiert ein s, dass die Interpolationsaufgabe löst. Ist $\tilde{s} \in S_3(X)$ eine weitere Lösung, so folgt $s - \tilde{s} = cx + d$, da $s'' = \tilde{s}''$. Aus der Interpolationsbedingung für $j = 0$ und $j = n$ folgt dann aber $s = \tilde{s}$. □

2.3.2 Konvergenz kubischer Splines

Es seien $X : a = x_0 < \cdots < x_n = b$ eine Zerlegung von $I = [a, b]$ sowie

$$h := \max_{j=1,\ldots,n} (x_j - x_{j-1}) \qquad \text{der maximale Gitterabstand und}$$

$$h_{min} := \min_{j=1,\ldots,n} (x_j - x_{j-1}) \qquad \text{der minimale Gitterabstand.}$$

Wir sind nun am Verhalten des Fehlers $f - s$ für $n \to \infty$ interessiert.

Satz 2.24. *Es sei $f \in C^4([a, b])$ mit $f''(a) = f''(b) = 0$. Es sei weiter s der kubische natürliche Spline, der f auf der Zerlegung X interpoliert. Dann gilt die Fehlerabschätzung*

$$\|f - s\|_{\infty,[a,b]} \leq h^4 \|f^{(4)}\|_{\infty,[a,b]}.$$

Beweis. Sei $0 \leq k \leq n - 1$ und $p(f - s|x_k, x_{k+1})(x) \in \mathbb{P}_1$ das lineare Polynom, das $f - s$ an den Knoten x_k und x_{k+1} interpoliert. Wegen

$$0 = p(f - s|x_k, x_{k+1})(x_k) = p(f - s|x_k, x_{k+1})(x_{k+1})$$

gilt dann aber $p(f - s|x_k, x_{k+1})(x) \equiv 0$. Somit liefert Satz 2.12 auf $[x_k, x_{k+1}]$ den Fehler

$$\begin{aligned} \|(f - s) - p(f - s|x_k, x_{k+1})\|_\infty &= \|f - s\|_\infty \\ &\leq \|(x - x_k)(x - x_{k+1})\|_\infty \frac{1}{2!} \|f'' - s''\|_\infty \\ &\leq \frac{1}{2} h_{k+1}^2 \|f'' - s''\|_\infty. \end{aligned} \tag{1}$$

Wir wollen $\|f'' - s''\|_{\infty,[x_k,x_{k+1}]}$ weiter abschätzen. Dazu sei $p_k = p(f''|x_k, x_{k+1})(x) \in \mathbb{P}_1$ das lineare Polynom, das f'' an den Knoten x_k und x_{k+1} interpoliert. Dann gilt unter erneuter Anwendung von Satz 2.12 und der Tatsache, dass p_k und s'' linear sind, die Abschätzung

$$\begin{aligned} \|f'' - s''\|_{\infty,[x_k,x_{k+1}]} &\leq \|f'' - p_k\|_{\infty,[x_k,x_{k+1}]} + \|p_k - s''\|_{\infty,[x_k,x_{k+1}]} \\ &\leq \frac{1}{2} h_{k+1}^2 \|f^{(4)}\|_{\infty,[x_k,x_{k+1}]} + \max_{l=k,k+1} |f''(x_l) - s''(x_l)| \end{aligned} \tag{2}$$

Wir setzen nun $M_l = s''(x_l)$. Dann bleibt noch $\max\limits_{l=k,k+1} |f''(x_l) - M_l|$ abzuschätzen. Es sei

$$A = \begin{pmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-2} \\ \lambda_n & & & \mu_n & 2 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, \quad z \in \mathbb{R}^n.$$

Dann gilt nach Lemma 2.21 (mit $c := 2 - \mu - \lambda = 1$) gerade

$$\|z\|_\infty \leq \|Az\|_\infty.$$

Dies wenden wir nun auf $f''(x_l) - M_l$ an. Mit $f''(a) = f''(b) = 0$ gilt

$$\begin{aligned} \max_{0\leq l\leq n} |f''(x_l) - M_l| &= \max_{1\leq l\leq n-1} |f''(x_l) - M_l| \\ &\leq \max_{1\leq l\leq n-1} |\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - \mu_l M_{l-1} - 2M_l - \lambda_l M_{l+1}| \\ &= \max_{1\leq l\leq n-1} |\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - 6[x_{l-1}, x_l, x_{l+1}]f|. \end{aligned}$$

Bei der letzten Gleichung wurde die Darstellung der rechten Seite des linearen Gleichungssystems zur Berechnung von M benutzt. Wir berechnen die vorkommenen dividierten Differenzen und betrachten im Folgenden den Ausdruck

$$\left|\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - \frac{6}{h_l + h_{l+1}} \left(\frac{f(x_{l+1} - f(x_l}{h_{l+1}} - \frac{f(x_l) - f(x_{l-1})}{h_l} \right) \right|. \tag{$*$}$$

Wir führen nun Taylor-Entwicklung um x_l durch und erhalten die Gleichungen

$$\begin{aligned} f''(x_{l-1}) &= f''(x_l) + f'''(x_l)(-h_l) + \frac{1}{2} f^{(4)}(x_l - \Theta_1 h_l) h_l^2, \\ f''(x_{l+1}) &= f''(x_l) + f'''(x_l) h_{l+1} + \frac{1}{2} f^{(4)}(x_l + \Theta_2 h_{l+1}) h_{l+1}^2, \\ f(x_{l-1}) &= f(x_l) + f'(x_l)(-h_l) + \frac{1}{2} f''(x_l) h_l^2 - \frac{1}{6} f'''(x_l) h_l^3 + \frac{1}{24} f^{(4)}(x_l - \Theta_3 h_l) h_l^4, \\ f(x_{l+1}) &= f(x_l) + f'(x_l) h_{l+1} + \frac{1}{2} f''(x_l) h_{l+1}^2 + \frac{1}{6} f'''(x_l) h_{l+1}^3 + \frac{1}{24} f^{(4)}(x_l + \Theta_4 h_{l+1}) h_{l+1}^4. \end{aligned}$$

Weiterhin erhalten wir dann mit $\mu_l = \frac{h_l}{h_l + h_{l+1}}$ und $\lambda_l = \frac{h_{l+1}}{h_l + h_{l+1}}$ die beiden Gleichungen

$$\begin{aligned} &\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) \\ &= 3f''(x_l) + (-\mu_l h_l + \lambda_l h_{l+1}) f'''(x_l) + \frac{\mu_l h_l^2}{2} f^{(4)}(x_l - \Theta_1 h_l) + \frac{\lambda_l h_{l+1}^2}{2} f^{(4)}(x_l + \Theta_2 h_{l+1}) \end{aligned}$$

sowie

$$
\begin{aligned}
&- \frac{6}{h_l + h_{l+1}} \left(\frac{f(x_{l+1} - f(x_l}{h_{l+1}} - \frac{f(x_l) - f(x_{l-1})}{h_l} \right) \\
&= -3f''(x_l) - (-\mu_l h_l + \lambda_l h_{l+1}) f'''(x_l) \\
&\quad - \frac{1}{h_l + h_{l+1}} \left(\frac{h_{l+1}^3}{4} f^{(4)}(x_l + \Theta_4 h_{l+1}) + \frac{h_l^3}{4} f^{(4)}(x_l - \Theta_3 h_l) \right).
\end{aligned}
$$

Addieren wir die letzten beiden Gleichungen und setzen diese in $(*)$ ein, so erhalten wir die Abschätzung

$$
\begin{aligned}
(*) &= \left| \frac{1}{h_l + h_{l+1}} \left(\frac{h_l^3}{2} f^{(4)}(x_l - \Theta_1 h_l) + \frac{h_{l+1}^3}{2} f^{(4)}(x_l + \Theta_2 h_{l+1}) \right. \right. \\
&\qquad \left. \left. - \frac{h_{l+1}^3}{4} f^{(4)}(x_l + \Theta_4 h_{l+1}) + \frac{h_l^3}{4} f^{(4)}(x_l - \Theta_3 h_l) \right) \right| \\
&\leq \frac{3}{2} h^2 \| f^{(4)} \|_{\infty, [x_{l-1}, x_{l+1}]}
\end{aligned}
$$

und damit insgesamt

$$
\max_{l=k,k+1} |f''(x_l) - s''(x_l)| \leq \max_{1 \leq l \leq n-1} |f''(x_l) - s''(x_l)| \leq \frac{3}{2} h^2 \| f^{(4)} \|_{\infty, [a,b]}. \tag{3}
$$

Aus den Abschätzungen (1), (2) und (3) folgt dann die Behauptung. □

3 Numerische Integration

Sei $f : I \to \mathbb{R}$ eine stetige Funktion auf $I = [a, b]$ und f habe eine Stammfunktion F. Dann gilt nach dem Hauptsatz der Differential- und Integralrechnung

$$I(f) = \int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$$

Diese Formel lässt sich aber nicht ohne Weiteres anwenden, denn

1) Die Stammfunktion F ist im Allgemeinen nicht bekannt. Daraus folgt die Notwendigkeit, das Integral näherungsweise zu bestimmen.

2) Häufig kenn man vom Integranden f und einige seiner Ableitungen nur an einzelnen Punkten des Intervalls I.

Eine Approximation des Integrals ist zum Beispiel über riemannsche Summen möglich. Für Punkte $x_i = a + ih$, $i = 0, \ldots, n$ mit $h = \frac{b-a}{n}$ gilt

$$I(f) \approx S_n(f) = h \sum_{i=1}^{n} f(t_i).$$

Es gilt $S_n(f) \to I(f)$ für $n \to \infty$, aber im Allgemeinen ist die Konvergenz langsam.

3.1 Einfache Integrationsformeln (Quadraturformeln)

Im folgenden Abschnitt werden wir zwei einfache Integrationsformeln kennenlernen.

1. Trapezregel

Es sei $p \in \mathbb{P}_1$ mit $p(a) = f(a)$ und $p(b) = f(b)$, also

$$p(x) = \frac{(b-x)f(a) + (x-a)f(b)}{b-a}.$$

Dann gilt

$$T(f) = \int_a^b p(x)\,\mathrm{d}x = \left[\frac{f(b)}{b-a}\frac{1}{2}(x-a)^2 - \frac{f(a)}{b-a}\frac{1}{2}(b-x)^2\right]_a^b$$
$$= \frac{1}{2}(b-a)f(b) + \frac{1}{2}(b-a)f(a)$$
$$= h\frac{f(a)+f(b)}{2}$$

mit $h = b - a$. Damit erhalten wir für das Integral

$$I(f) = T(f) + R(f)$$

für einen Fehler $R(f)$. Dieser Fehler berechnet sich mit der Formel aus Satz 2.12 und dem Mittelwertsatz der Integralrechnung zu

$$R(f) = I(f) - T(f) = \int_a^b f(x) - p(x)\,\mathrm{d}x$$
$$= \int_a^b \frac{(x-a)(x-b)}{2!} f''(\xi(x))\,\mathrm{d}x$$
$$= f''(\xi_0)\int_a^b \frac{(x-a)(x-b)}{2!}\,\mathrm{d}x$$
$$= -\frac{1}{12}(b-a)^3 f''(\xi_0),$$

wobei $\xi_0 \in [a,b]$.

2. Simpson-Formel

Es sei $p \in \mathbb{P}_2$ und p interpoliere f in den Punkten a, $\frac{a+b}{2}$ und b. Wir erhalten

$$S(f) = \int_a^b p(x)dx = \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right), \quad h = \frac{b-a}{2}.$$

Man kann zeigen (siehe Übung), dass die Simpson-Formel nicht nur Polynome 2. Grades, sondern auch Polynome 3. Grades exakt integriert. Daraus folgt, ähnlich wie bei der Trapezregel, die Fehlerabschätzung

$$R(f) = I(f) - S(f) = -\frac{1}{90}h^5 f^{(4)}(\xi), \quad \xi \in [a,b],$$

falls $f \in C^4([a,b])$.

Weitere Quadraturformeln

Das Vorgehen für die Trapez- und Simpson-Regel lässt sich für beliebige Interpolationspolynome von f verallgemeinern. Dabei ersetzt man den Integranden durch ein Interpolationspolynom $p \in \mathbb{P}_n$ an den Stellen $a = x_0 < x_1, \ldots, < x_n = b$ und setzt

$$Q(f) = \int_a^b p(x)\,\mathrm{d}x = \sum_{i=0}^{n} c_i f(x_i). \qquad \text{(Interpolatorische Integrationsformel)}$$

Diese Formeln erben allerdings die schlechten Eigenschaften der Interpolation mit Polynomen hohen Grades. Eine Ausnahme bilden hier die sogenannten Gauß-Quadraturformeln.

Eine weitere Möglichkeit besteht darin, das Intervall $I = [a, b]$ in Teilintervalle $I_1, \ldots, I_n$ zu zerlegen und in jedem Teilintervall I_i eine einfache Integrationsformel (zum Beispiel die Trapezregel) anzuwenden. Dieser Vorgang ähnelt dem Übergang von der Interpolation mit Polynomen zur Spline-Interpolation.

3.2 Iterierte Trapezregel (Romberg-Verfahren)

Als Beispiel für die oben erwähnte zweite Möglichkeit wollen wir hier die iterierte Trapezregel erläutern. Wir betrachten Teilintervalle der Länge $h = \frac{b-a}{n}$, $n \in \mathbb{N}$. Dann gilt

$$\begin{aligned} Q(f) &= \sum_{i=0}^{n-1} h \frac{f(a+ih) + f(a+(i-1)h)}{2} \\ &= h\left(\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right) \\ &= T_h(f) \end{aligned}$$

und für das Integral somit

$$I(f) = T_h(f) + R_h(f)$$

für einen von h abhängigen Fehler $R_h(f)$.

Satz 3.1. *Es gilt*

1) *Ist $f \in C([a,b])$, so gilt $R_h(f) \to 0$, das heißt $T_h(f) \to I(f)$ für $h \to 0$.*

2) *Gilt sogar $f \in C^2([a,b])$, so folgt die Fehlerabschätzung $R_h(f) = \mathcal{O}(h^2)$.*

Beweis. Wir können $T_h(f)$ als riemannsche Summe auffassen und schreiben

$$T_h(f) = \sum_{i=1}^{n} f(\xi_i)(x_i - x_{i-1}), \quad \xi_i \in [x_{i-1}, x_i],$$

wobei hier $x_i = a + ih$ für $i = 0, \ldots, n$ ist und ξ_i so gewählt ist, dass mit dem Zwischenwertsastz gerade

$$f(\xi_i) = \frac{f(x_{i-1}) + f(x_i)}{2}$$

gilt. Nach Definition des riemannschen Integrals gilt:

Sei $f : [a, b] \to \mathbb{R}$ eine Riemann-integrierbare Funktion. Dann existiert zu jedem $\varepsilon > 0$ ein $\delta > 0$, so dass für jede Unterteilung $a = x_0 < x_1 \cdots < x_n = b$ des Intervalls $[a, b]$ der Feinheit $\leq \delta$, das heißt $|x_i - x_{i-1}| \leq \delta$ für alle $i = 0, \ldots, n$, und jede Wahl von Stützstellen $\xi_i \in [x_{i-1}, x_i]$ die Abschätzung

$$\left| \int_a^b f(x)\, \mathrm{d}x - \sum_{i=1}^{n} f(\xi_i(x_i - x_{i-1}) \right| \leq \varepsilon$$

gilt.

Hieraus folgt die erste Behauptung. Die zweite Aussage folgt direkt aus der Fehlerabschätzung der Trapezregel, denn

$$\begin{aligned} |R_h(f)| &= \left| \sum_{i=1}^{n} \left(-\frac{1}{12} h^3 f''(\xi_i) \right) \right|, \quad \xi_i \in [x_{i-1}, x_i] \\ &\leq \frac{1}{12} h^3 n \|f''\|_{\infty,[a,b]} \\ &= \frac{b-a}{12} h^2 \|f''\|_{\infty,[a,b]} \\ &= \mathcal{O}(h^2). \end{aligned}$$

□

Die folgenden beiden Abschnitte stellen Hilfsmittel bereit, die später im Abschnitt Extrapolation, Romberg-Verfahren ausgenutzt werden.

3.2.1 Bernoulli-Polynome

Definition 3.2. Das durch die folgende Rekursionsvorschrift festgelegte System von Polynomen B_r heißt das System der *Bernoulli-Polynome*.

(i) $B_0(x) = 1$,

(ii) $B'_{r+1}(x) = B_r(x)$, $r = 0, 1, 2, \ldots$,

(iii) $\int_0^1 B_r(x)\,\mathrm{d}x = 0$

Die ersten Bernoulli-Polynome sind

- $B_0(x) = 1$,
- $B_1(x) = x - \frac{1}{2}$,
- $B_2(x) = \frac{1}{2}x^2 - \frac{1}{2}x + \frac{1}{12}$,
- $B_3(x) = \frac{1}{6}x^3 - \frac{1}{4}x^2 + \frac{1}{12}x$.

Offensichtlich gilt $B_r \in \mathbb{P}_r$.

Im Folgenden wollen wir drei Eigenschaften der Bernoulli-Polynome beweisen.

Satz 3.3. *Es gilt* $B_r(x) = (-1)^{r+1}B_r(1-x)$.

Beweis. Wir beweisen dies mittels vollständiger Induktion über r. Für $r = 0$ ist die Gleichung klar. Angenommen, die Gleichung gelte für ein r. Dann folgt die Bildung der Stammfunktion

$$B_{r+1}(x) = (-1)^{r+1}B_{r+1}(1-x) + c \tag{1}$$

mit einer Konstanten c. Hieraus erhält man

$$\int_0^1 B_{r+1}(x)\,\mathrm{d}x = \int_0^1 (-1)^{r+1}B_{r+1}(1-x)\,\mathrm{d}x + c. \tag{2}$$

Substituieren wir nun $1 - x = u$, so folgt $\mathrm{d}u = -\mathrm{d}x$ und damit

$$\int_0^1 B_{r+1}(1-x)\,\mathrm{d}x = -\int_1^0 B_{r+1}(u)\,\mathrm{d}u = \int_0^1 B_{r+1}(u)\,\mathrm{d}u = 0$$

nach (iii). Aus (2) ergibt sich schließlich, dass $c = 0$ sein muss und es folgt aus (1) sofort die Behauptung für $r + 1$. □

Satz 3.4. *Für* $r \geq 2$ *gilt* $B_r(1) = B_r(0)$.

Beweis. Nach (ii) und (iii) gilt

$$0 = \int_0^1 B_{r+1}\,\mathrm{d}x = B_r(1) - B_r(0).$$ □

Satz 3.5. *Für aller* ≥ 1 *gilt* $B_{2r+1}(0) = B_{2r+1}(\frac{1}{2}) = B_{2r+1}(1) = 0$

Beweis. Nach Satz 3.3 ist

$$B_{2r+1}(x) = -B_{2r+1}(1-x). \tag{3}$$

Damit folgt für $x = \frac{1}{2}$ aber sofort $B_{2r+1}(\frac{1}{2}) = 0$. Ein Vergleich mit Satz 3.4 liefert für $x = 0$ die Beziehung

$$B_{2r+1}(0) = B_{2r+1}(1) = -B_{2r+1}(0) = -B_{2r+1}(1),$$

also $B_{2r+1}(0) = B_{2r+1}(1) = 0$. □

3.2.2 Die Eulersche Summenformel

In diesem Abschnitt wollen wir eine genauere Darstellung der Struktur des Fehlers der iterierten Trapezregel betrachten, das heißt eine Entwicklung nach Potenzen von h^2. Dazu sei $f : [a, b] \to \mathbb{R}$ genügend oft differenzierbar, $h := \frac{b-a}{n}$, $s = a + th$ für $t \in [0, n]$. Weiter sei $g(t) = f(a + th)$ für $t \in [0, n]$ und es gelte $\mathrm{d}s = h\mathrm{d}t$. Für genügend oft differenzierbares f gilt folglich

$$g^{(i)}(t) = f^{(i)}(a + th)h^i$$

und

$$\begin{aligned}
\int_a^b f(s)\,\mathrm{d}s &= h\int_0^n f(a+th)\,\mathrm{d}t \\
&= h\int_0^n g(t)\,\mathrm{d}t \\
&= h\sum_{r=1}^{n}\int_{r-1}^{r} g(t)\,\mathrm{d}t \\
&= h(T_h(g) + R_h(g))
\end{aligned}$$

Um $R_h(g)$ zu berechnen, betrachten wir für $r = 1, \ldots, n$ die Integrale

$$\int_{r-1}^{r} g(t)\,\mathrm{d}t = \int_{r-1}^{r} g(t)\cdot 1\,\mathrm{d}t = g(t)\left[t - \left(r - \frac{1}{2}\right)\right]_{r-1}^{r} - \int_{r-1}^{r}\left(t - \left(r - \frac{1}{2}\right)\right) g'(t)\,\mathrm{d}t,$$

wobei wir bei der partiellen Integration die Stammfunktion von $F' = 1$ so gewählt haben, dass

$$\int_{r-1}^{r} F(t)\,\mathrm{d}t = 0$$

gilt. Weiterhin können wir

$$F(t) = t - \left(r - \frac{1}{2}\right) = t - (r-1) - \frac{1}{2} = t - \lfloor t \rfloor - \frac{1}{2}, \quad t \in [r-1, r]$$

schreiben, um uns vom Index r zu lösen. Hier bezeichnet $\lfloor t \rfloor$ die größte ganze Zahl $m \in \mathbb{Z}$ mit $m \leq t$. insgesamt erhalten wir dann

$$\begin{aligned}
\int_0^n g(t)\,\mathrm{d}t &= \sum_{r=1}^{n} \int_{r-1}^{r} g(t)\,\mathrm{d}t \\
&= \sum_{r=1}^{n} \left[g(t) \cdot F(t)\right]_{r-1}^{r} - \sum_{r=1}^{n} \int_{r-1}^{r} F(t) g'(t)\,\mathrm{d}t \\
&= \underbrace{\frac{1}{2}g(0) + g(1) + \cdots + g(n-1) + \frac{1}{2}g(n)}_{=T_h(g)} \underbrace{- \int_0^n F(t) g'(t) dt}_{=R_h(t)}
\end{aligned}$$

Wenden wir nun partielle Integration auf $R_h(g)$ an und benutzen die Darstellung der Bernoulli-Polynome, so erhalten wir

$$\begin{aligned}
R_h(g) &= -\int_0^n F(t) g'(t)\,\mathrm{d}t \\
&= -\sum_{r=1}^{n} \left(\int_{r-1}^{r} \left(t - \lfloor t \rfloor - \frac{1}{2}\right) g'(t) \mathrm{d}t \right) \\
&= -\sum_{r=1}^{n} \left(\int_{r-1}^{r} B_1\left(t - \lfloor t \rfloor\right) g'(t)\,\mathrm{d}t \right) \\
&= -\sum_{r=1}^{n} \left[B_2(t-(r-1)) g'(t)\right]_{r-1}^{r} - \int_{r-1}^{r} B_2\left(t - \lfloor t \rfloor\right) g''(t)\,\mathrm{d}t \\
&= \dots \\
&= -\sum_{r=1}^{n} \sum_{j=2}^{m} (-1)^j \left(B_j(1) g^{(j-1)}(r) - B_j(0) g^{(j-1)}(r-1) \right) + (-1)^{m+1} \int_0^n B_m\left(t - \lfloor t \rfloor\right) g^{(m)}(t)\,\mathrm{d}t \\
&= -\sum_{j=2}^{m} (-1)^j \left(B_j(1) g^{(j-1)}(n) - B_j(0) g^{(j-1)}(0) \right) + (-1)^{m+1} \int_0^n B_m\left(t - \lfloor t \rfloor\right) g^{(m)}(t)\,\mathrm{d}t
\end{aligned}$$

Dabei folgt die letzte Gleichung aufgrund der Tatsache, dass nach Satz 3.4 $B_j(1) = B_j(0)$ gilt und somit die inneren Terme der Summe über r verschwinden. Wenden wir jetzt Satz 3.5, also $B_{2r+1}(1) = B_{2r+1}(0)$, an, so bekommen wir

$$-\sum_{i=1}^{k}\Big(B_{2i}(1)g^{(2i-1)}(n) - B_{2i}(0)g^{(2i-1)}(0)\Big) + \begin{cases} -\int_0^n B_{2k}\left(t-\lfloor t\rfloor\right) g^{(2k)}(t)\,\mathrm{d}t, & m=2k,\\ +\int_0^n B_{2k+1}\left(t-\lfloor t\rfloor\right) g^{(2k+1)}(t)\,\mathrm{d}t, & m=2k+1. \end{cases}$$

Den unteren Integralterm können wir noch einmal wie oben partiellen intergrieren und wir erhalten dann

$$\begin{aligned}\int_0^n B_{2k+1}(t-\lfloor t\rfloor)g^{(2k+1)}(t)\,\mathrm{d}t &= B_{2k+2}(1)g^{(2k+2)}(n) - B_{2k+2}(0)g^{(2k+2)}(0)\\ &\quad - \int_0^n B_{2k+2}(t-\lfloor t\rfloor)g^{(2k+2)}(t)\,\mathrm{d}t,\end{aligned}$$

so dass wir erneut mit Satz 3.4

$$R_h(g) = -\sum_{i=1}^{k+1} B_{2i}(0)\big(g^{(2i-1)}(n) - g^{(2i-1)}(0)\big) + \int_0^n B_{2k+2}(t-\lfloor t\rfloor)g^{(2k+2)}(t)\,\mathrm{d}t$$

erhalten. Insgesamt haben wir dann

$$\begin{aligned}\int_a^b f(s)\,\mathrm{d}s &= h\int_0^n f(a+th)\,\mathrm{d}t = h\int_0^n g(t)\mathrm{d}t\\ &= h\left[\frac{1}{2}g(0) + \sum_{r=1}^{n-1} g(r) + \frac{1}{2}g(n) - \sum_{i=1}^{k+1} B_{2i}(0)\big(g^{(2i-1)}(n) - g^{(2i-1)}(0)\big)\right.\\ &\quad \left. + \int_0^n B_{2k+2}(t-\lfloor t\rfloor)g^{(2k+2)}(t)\,\mathrm{d}t\right]\\ &= h\left[\frac{1}{2}f(a) + \sum_{r=1}^{n-1} f(a+rh) + \frac{1}{2}f(b) - \sum_{i=1}^{k+1} h^{2i-1}B_{2i}(0)\big(f^{(2i-1)}(b) - f^{(2i-1)}(a)\big)\right.\\ &\quad \left. + h^{2k+1}\int_a^b B_{2k+2}\left(\frac{s-a}{h} - \left\lfloor\frac{s-a}{h}\right\rfloor\right) f^{(2k+2)}(s)\,\mathrm{d}s\right]\end{aligned}$$

Wir haben also den folgenden Satz bewiesen.

Satz 3.6. *Für $f \in C^{2k+2}([a,b])$ gilt*

$$T_h(f) = \int_a^b f(s)\,\mathrm{d}s + \sum_{i=1}^{k+1} c_i h^{2i} - h^{2k+2} I_{2k+2}(f)$$

mit $c_i = B_{2i}(0)\big(f^{(2i-1)}(b) - f^{(2i-1)}(a)\big)$ *und*

$$I_{2k+2}(f) = \int_a^b B_{2k+2}\left(\frac{s-a}{h} - \left\lfloor \frac{s-a}{h} \right\rfloor\right) f^{(2k+2)}(s)\,\mathrm{d}s \in \mathbb{R}.$$

Folglich besitzt der Fehler

$$R_h(f) = \int_a^b f(s)\,\mathrm{d}s - T_h(f)$$

eine Entwicklung nach Potenzen von h^2

Bemerkung. Insbesondere gilt für periodische Funktionen $f \in C^{2m}(\mathbb{R})$ mit Periode $b-a$ die Gleichung

$$R_h(f) = h^{2m} I_{2m}(f), \quad I_{2m}(f) \in \mathbb{R}.$$

3.2.3 Extrapolation, Romberg-Verfahren

Wir werden nun eine andere Art von Integrationsverfahren kennenlernen, die auf der Trapezsumme basieren. Die bisher besprochenen Quadraturformeln bezogen sich alle auf ein festes Gitter $t_0, \ldots, t_n$ von Knoten, an denen die Funktion ausgewertet wurde. Im Gegensatz dazu benutzen wir bei der Romberg-Quadratur eine Folge von Gittern und versuchen aus den dazu gehörenden Trapezsummen eine bessere Approximation des Integrals zu konstruieren.

Es sei $h = \frac{b-a}{n}$ und $0 < q < 1$ mit $\frac{b-a}{qh} \in \mathbb{N}$. Es gilt $T_h(f) = I(f) - R_h(f)$. Seien weiter $T_h(f)$ und $T_{qh}(f)$ berechnet. Dann folgt

$$\begin{aligned} T_h(f) &= I(f) + c_1 h^2 + c_2 h^4 + \ldots \\ T_{qh}(f) &= I(f) + c_1 (qh)^2 + c_2 (qh)^4 + \ldots \end{aligned}$$

Wir multiplizieren die erste Gleichung mit q^2 und subtrahieren die zweite Gleichung von der ersten Gleichung. Somit erhalten wir

$$(q^2-1)I(f) = q^2 T_h(f) - T_{qh}(f) - c_2(q^2-q^4)h^4 - c_3(q^2-q^6)h^6 + \ldots$$

Folglich ist

$$I(f) = \frac{q^2 T_h(f) - T_{qh}(f)}{q^2-1} + c_2 q^2 h^4 + c_3 q^2(1+q^2)h^6 + \ldots$$

und wir sehen, dass die Fehlerentwicklung bei h^4 beginnt. Dieses Vorgehen kann man weiter wiederholen.

Extrapolation

Es seien $h_0 > h_1 > \cdots > h_n > 0$ Schrittweiten mit $\frac{b-a}{h_i} \in \mathbb{N}$ für $i = 0, \ldots, n$. Dann lautet das Vorgehen wie folgt:

1) Berechne $T_{h_i}(f)$ für alle $i = 0, \ldots, n$.

2) Interpoliere $(h_i^2, T_{h_i}(f))$ für $i = 0, \ldots, n$ mit einem Polynom $p \in \mathbb{P}_n$ in h^2, das heißt
$$p(h^2) = a_0 + a_1 h^2 + a_2 h^4 + \cdots + a_n h^{2n}$$
mit
$$p(h_i^2) = T_{h_i}(f), \quad i = 0, \ldots, n.$$

3) Betrachte $p(0)$ als neue Näherung für $I(f)$ (Extrapolation auf $h = 0$).

Es gibt zwei Möglichkeiten, die Folge der Schrittweiten zu wählen:

- Romberg-Folge: $h_0 = b - a$, $\frac{h_0}{2}, \frac{h_0}{4}, \ldots,$
- Bulirsch-Folge: $h_0 = b - a$, $\frac{h_0}{2}, \frac{h_0}{3}, \ldots$

Dabei wächst die Anzahl der Funktionsauswertungen bei der Bulirsch-Folge weniger schnell, aber man erhält eine langsamere Konvergenz.

Allgemein gilt: Die Berechnung der Größen $T_{00}, \ldots, T_{n0}$ kann vergleichsweise viel Zeit in Anspruch nehmen, während die Berechnung der anderen Größen T_{ij} für $j \geq i$ nicht ins Gewicht fällt, aber im Allgemeinen einen großen Gewinn bringt.

Bemerkung 3.7. (1) Für $h_0 = (b - a)$, und $h_1 = \frac{h_0}{2}$ folgt
$$T_{1,1} = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$
Dies ist gerade die Simpson-Formel, was sich leicht durch Nachrechnen überprüfen lässt (Übung).

(2) Abbruchkriterium: Setze $U_{j,k} := 2T_{j+1,k} - T_{j,k}$. Dann lässt sich zeigen (Bulirsch/-Stoer), dass asymptotisch
$$U_{j,k} - I(f) \approx I(f) - T_{j,k}$$
gilt. Daraus folgt
$$U_{j,k} - T_{j,k} \approx 2(I(f) - T_{j,k})$$
und
$$2(T_{j+1,k} - T_{j,k}) = U_{j,k} - T_{j,k} \approx 2(I(f) - T_{j,k}).$$

Damit lässt sich dann

$$|T_{j+1,k} - T_{j,k}| \leq \varepsilon$$

als geeignetes Abbruchkriterium festlegen, denn dann gilt

$$|I(f) - T_{j,k}| \approx \varepsilon,$$

das heißt der Fehler ist klein. In der Praxis bedeutet das: Falls $T_{j,k}$ dieses Kriterium erfüllt, berechne $T_{j+1,k+1}$ und verwende diesen Wert als endgültige Näherung.

(3) Für die Ordnung des Verfahrens gilt

$$|T_{j,k} - I(f)| = \mathcal{O}\left(\prod_{i=j-k}^{j} h_i^2\right),$$

das heißt die j-te Spalte des Romberg-Dreiecks (iterierte Trapezregel) konvergiert mit h^{2j+2} gegen $I(f)$.

Beispiel. Betrachte

$$\int_0^1 e^t \,\mathrm{d}t \approx 1{,}718\,828\,459.$$

Weiter sei $h_0 = b - a$ und $h_i = \frac{h_{i-1}}{2}$ (Romberg-Folge). Dann gilt

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\left(\frac{h_{j-k}}{h_j}\right)^2 - 1} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{4^k - 1} = \frac{4^k T_{j,k-1} - T_{j-1,k-1}}{4^k - 1}.$$

Das Romberg-Dreieck sieht dann wie folgt aus:

	$k=0$	1	2	3
$j=0$	1,8591			
1	1,7539	1,718 861		
2	1,7272	1,718 318	1,718 282 687	
3	1,7205	1,718 264	1,718 281 842	1,718 281 829
$\vdots$	$\vdots$			
12	1,718 281 839			

3.3 Newton-Cotes-Formeln und daraus abgeleitete Verfahren

Definition 3.8. Es seien $a \leq x_0 < \cdots < x_n \leq b$. Ein lineares Funktional der Form

$$Q_n(f) = \sum_{j=0}^{n} a_j f(x_j), \quad a_j \in \mathbb{R}$$

heißt *Integrationsformel (Quadraturformel)*. Sie heißt *abgeschlossen*, wenn $a = x_0$ und $x_n = b$. Im Falle $a < x_0$ und $b > x_n$ heißt sie *offen*. Integrationsformeln heißen *interpolatorische Integrationsformeln*, wenn sie auf der Integration des die Funktion f interpolierenden Polynoms $p_f(x) = p(f|x_0, \ldots, x_n)(x) \in \mathbb{P}_n$ beruht. *Newton-Cotes-Formeln* sind interpolatorische Integrationsformeln mit äquidistanten Stützstellen.

Es sei $p_f \in \mathbb{P}_n$ mit $p_f(x_j) = f_j$ für $j = 0, \ldots, n$. Die Lagrange-Darstellung von p_f ist gegeben durch

$$p_f(x) = \sum_{j=0}^{n} f_j l_{jn}(x), \quad l_{jn}(x) = \prod_{i \neq j}^{n} \frac{x - x_i}{x_j - x_i}.$$

Dann gilt

$$Q_n(f) := \int_a^b p_f(x)\,\mathrm{d}x = \sum_{j=0}^{n} f(x_j) \int_a^b l_{jn}(x)\,\mathrm{d}x = \sum_{j=0}^{n} a_j f(x_j)$$

mit

$$a_j = \int_a^b l_{jn}(x)\,\mathrm{d}x.$$

Bemerkung 3.9. Zu $n+1$ paarweise verschiedenen Knoten $x_0, \ldots, x_n$ gibt es genau eine Quadraturformel $Q_n : C([a,b]) \to \mathbb{R}$ der Gestalt

$$Q_n(f) = \sum_{j=0}^{n} a_j f(x_j), \quad a_j \in \mathbb{R},$$

die auf $\mathbb{P}_n$ exakt ist, für die also

$$Q_n(p) = \int_a^b p(x)\,\mathrm{d}x$$

für alle $p \in \mathbb{P}_n$ gilt.

Beweis. Setze die zu den Knoten x_j gehörenden Lagrange-Polynome $l_{jn} \in \mathbb{P}_n$, die nach Voraussetzung exakt integriert werden, in die Quadraturformel ein und erhalte

$$Q_n(l_{jn}) = \sum_{i=0}^{n} a_i l_{jn}(x_i) = \sum_{i=0}^{n} a_i \delta_{ji} = a_i, \quad j = 0, \ldots, n$$

und dadurch die Gewichte $a_0, \ldots, a_n$ auf eindeutige Weise zurück. □

Definition 3.10 (Ordnung einer Quadraturformel)**.** Eine Quadraturformel Q_n hat die Ordnung m, wenn durch sie mindestens alle Polynome aus $\mathbb{P}_{m-1}$ exakt integriert werden. Das heißt die interpolatorischen Quadraturformeln zu $n+1$ Stützstellen sind mindestens von der Ordnung $n+1$.

Definition 3.11. Ein Quadraturverfahren ist eine Folge von Quadraturformeln wachsender Ordnung zum gleichen Grundintervall.

Klassische Quadraturformeln auf $[0,1]$:

	Stützstellen	Gewichte a_j
Rechteckregel	0	1
Mittelpunktregel	$\frac{1}{2}$	1
Trapezregel	$0,1$	$\frac{1}{2},\frac{1}{2}$
Simpsonregel	$0,\frac{1}{2},1$	$\frac{1}{6},\frac{2}{3},\frac{1}{6}$
$3/8-$ Regel	$0,\frac{1}{3},\frac{2}{3},1$	$\frac{1}{8},\frac{3}{8},\frac{3}{8},\frac{1}{8}$
Milne $-$ Regel	$0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1$	$\frac{7}{90},\frac{32}{90},\frac{12}{90},\frac{32}{90},\frac{7}{90}$

Für $n > 6$ treten negative Gewichte auf, so dass die Formeln numerisch unbrauchbar werden. Stattdessen benutzt man zusammengesetzte Quadraturformeln (zum Beispiel die iterierte Trapezregel).

Transformationsformel: Es sei eine Quadraturformel zum Grundintervall $[a,b]$ gegeben, also

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{r=0}^{n} a_r f(x_r) + R[f],$$

aber die zu integrierende Funktion g sei aus $[\tilde{a},\tilde{b}]$. Dann gilt

$$\int_{\tilde{a}}^{\tilde{b}} g(x)\,\mathrm{d}x \approx \sum_{r=0}^{n} \tilde{a}_r g(\tilde{x}_r)$$

mit $\tilde{a}_r = \frac{\tilde{b}-\tilde{a}}{b-a} a_r$ und $\tilde{x}_r = \tilde{a} + (x_r - a)\left(\frac{\tilde{b}-\tilde{a}}{b-a}\right)$ (siehe Übung).

Definition 3.12. Es sei Q_n eine Quadraturformel auf dem Grundintervall $[0,1]$ und es sei $H = \frac{b-a}{m}$. Ihre auf $[a+(r-1)H, a+rH]$ Transformierte werde mit $Q_n^{(r)}$ bezeichnet. Dann heißt:

$$Q_{(m)} := Q_n^{(1)} + Q_n^{(2)} + \ldots Q_n^{(m)}$$

die durch m-fache Anwendung der Elementarformel Q_n entstandene zusammengesetzte Quadraturformel. Ein zusammengesetztes Quadraturverfahren erhalten wir durch $m = 1,2,\ldots$-fache Anwendnung einer festen Elementarformel, zum Beispiel die iterierte Trapezregel oder das iterierte Simpson-Verfahren.

Satz 3.13. *Ein zusammengesetztes Quadraturverfahren ist genau dann konvergent, wenn seine Elementarformeln Konstanten exakt integrieren.*

Beweis. Es sei

$$Q_n^{\mathrm{el}}(f) = \sum_{r=0}^{n} a_r f(x_r)$$

eine Elementarformel auf $[0,1]$ und

$$Q_n^{(r)}(f) = \sum_{\mu=0}^{n} a_\mu H f(a + (r-1)H + x_\mu H)$$

sei eine Elementarformel auf $[(r-1)H, rH]$. Dann gilt für das zusammengesetzte Quadraturverfahren

$$\begin{aligned} Q_{(m)}(f) &= \sum_{r=1}^{m} \sum_{\mu=0}^{n} a_\mu H f\big(a + (r-1+x_\mu)H\big), \quad m = 1, 2, \dots \\ &= \sum_{\mu=0}^{n} a_\mu \underbrace{\left[H \sum_{r=1}^{m} f\big(a + (r-1+x_\mu)H\big) \right]}_{\text{Riemannsche Summe für das Integral}}. \end{aligned}$$

Also gilt

$$\lim_{m\to\infty} Q_{(m)}(f) = \sum_{\mu=0}^{n} a_\mu \int_a^b f(x)\,\mathrm{d}x$$

genau dann, wenn

$$\sum_{\mu=0}^{n} a_\mu = 1. \qquad \square$$

3.4 Gaußsche Quadraturformeln

Die interpolatorischen Quadraturformeln

$$Q_n(f) = \sum_{i=0}^{n} a_i f(x_i)$$

zu den Stützstellen $x_0, \dots, x_n \in [a,b]$ sind nach Konstruktion mindestens von der Ordnung $n+1$, das heißt für ihr Restglied gilt

$$R_n(p) = I(p) - Q_n(p) = 0, \quad \text{für alle} \quad p \in \mathbb{P}_n.$$

Unser Ziel in diesem Abschnitt ist es, die Stützstellen $x_0, \dots, x_n$ und Gewichte $a_0, \dots, a_n$ so zu wählen, dass die Ordnung der Quadraturformel möglichst hoch wird.

Lemma 3.14. *Eine obere Grenze für die Ordnung der Quadraturformel $Q_n(f)$ ist $2n+2$, das heißt es können höchstens alle Polynome $p \in \mathbb{P}_{2n+1}$ exakt integriert werden.*

Beweis. Es sei Q_n exakt für Polynome $p \in \mathbb{P}_{2n+2}$. Dann gilt

$$Q_n(p) = I(p) \quad \text{für} \quad p(x) = \prod_{i=0}^{n}(x - x_i)^2 \in \mathbb{P}_{2n+2}.$$

Daraus ergibt sich aber der Widerspruch

$$0 < \int_a^b p(x)\,\mathrm{d}x = Q_n(p) = 0. \qquad \square$$

Nun wollen wir zeigen, dass es tatsächlich Quadraturformeln zu $n+1$ Stützstellen gibt, die die Maximalordnung $2n+2$ haben.

Dazu seien $p_n \in \mathbb{P}_n$ und $p_{2n+1} \in \mathbb{P}_{2n+1}$ die Interpolationspolynome einer Funktion $f \in C([a,b])$ zu den $n+1$ beziehungsweise $2n+2$ Stützstellen $x_0, \ldots, x_n, x_{n+1}, \ldots, x_{2n+1} \in [a,b]$. Für die zugehörigen Quadraturformeln Q_n und Q_{2n+1} gilt dann

$$\begin{aligned} I(f) - Q_{2n+1}(f) &= I(f) - \int_a^b \sum_{i=0}^{2n+1} [x_0 \ldots, x_i]f \prod_{j=0}^{i-1}(x - x_j)\,\mathrm{d}x \\ &= I(f) - \sum_{i=0}^{2n+1} [x_0 \ldots, x_i]f \int_a^b \prod_{j=0}^{i-1}(x - x_j)\,\mathrm{d}x \\ &= I(f) - Q_n(f) - \sum_{i=n+1}^{2n+1} [x_0 \ldots, x_i]f \int_a^b \prod_{j=0}^{i-1}(x - x_j)\,\mathrm{d}x. \end{aligned}$$

Für $i = n+1, \ldots, 2n+1$ schreiben wir nun

$$\int_a^b \prod_{j=0}^{i-1}(x - x_j)\,\mathrm{d}x = \int_a^b \prod_{j=0}^{n}(x - x_j) \prod_{j=n+1}^{i-1}(x - x_j)\,\mathrm{d}x.$$

Die $n+1$ Polynome

$$\left\{ 1, x - x_{n+1}, (x - x_{n+1})(x - x_{n+2}), \ldots, \prod_{j=n+1}^{2n}(x - x_j) \right\}$$

bilden eine Basis von $\mathbb{P}_n$. Wähle nun die ersten $n+1$ Stützstellen $x_0, \ldots, x_{n+1} \in [a,b]$ so, dass

$$\int_a^b \prod_{j=0}^{n}(x - x_j) q(x)\,\mathrm{d}x = 0 \quad \text{für alle} \quad q \in \mathbb{P}_m. \qquad (*)$$

So folgt

$$I(f) - Q_n(f) = I(f) - Q_{2n+1}(f).$$

Das heißt, die interpolatorische Quadraturformel $Q_n(f)$ ist exakt für Polynome aus $\mathbb{P}_{2n+1}$, also von der Ordnung $2n+2$.

Auf dem Funktionenraum $C([a,b])$ sei

$$(f,g) = \int_a^b f(x)g(x)\,\mathrm{d}x, \quad \|f\|_2 = \sqrt{(f,f)}$$

das L_2-Skalarprodukt und die zugehörige Norm. Dann besagt $(*)$, dass das Polynom

$$p(x) = \prod_{j=0}^{n}(x - x_j) = x^{n+1} + r(x), \quad r(x) \in \mathbb{P}_n$$

bezüglich des L_2-Skalarprodukts orthogonal zum Teilraum $\mathbb{P}_n[a,b] \subset C([a,b])$ sein muss.

Konstruktion von p und somit seiner Nullstellen:

Wende das Gram-Schmidt-Orthogonalisierungsverfahren auf die Monombasis $\{1, x, \ldots, x^{n+1}\}$ an. Dann gilt

$$p_0(x) = 1, \quad p_k(x) = x^k - \sum_{j=0}^{k-1} \frac{(x^k, p_j)}{\|p_j\|_2^2} p_j(x), \quad k = 1, \ldots, n+1$$

und $\{p_0, \ldots, p_{n+1}\}$ bildet ein Orthogonalsystem in $\mathbb{P}_{n+1}[a,b]$. Offensichtlich gilt

$$p_{n+1}(x) = x^{n+1} + r(x), \quad r(x) \in \mathbb{P}_n.$$

Setze also $p(x) = p_{n+1}(x)$. Die $n+1$ Nullstellen $\lambda_0, \ldots, \lambda_n$ von $p_{n+1}(x)$ sind dann mögliche Kandidaten für „optimale“ Integrationspunkte.

Der folgende Satz macht eine Aussage über die Nullstellen von Orthogonalpolynomen. Wir betrachten ein Skalarprodukt der allgemeinen Gestalt

$$(f,g)_w = \int_a^b f(x)g(x)w(x)\,\mathrm{d}x$$

mit einer Gewichtsfunktion $w(x) \geq 0$ für alle $x \in (a,b)$ und höchstens endlich vielen Nullstellen in $[a,b]$. Es seien p_n die mit Hilfe des Gram-Schmidt-Verfahrens aus $\{1, x, x^2, \ldots\}$ gewonnenen bezüglich $(\cdot,\cdot)_w$ orthogonalen Polynome.

Satz 3.15. *Die bezüglich des Skalarprodukts $(\cdot,\cdot)_w$ orthogonalen Polynome p_n besitzen lauter reelle, einfache Nullstellen, die alle im Inneren des Intervalls $[a,b]$ liegen.*

Beweis. Es sei $N_n := \{\lambda \in (a,b) \mid \lambda \text{ Nullstelle ungerader Vielfachheit von } p_n\}$. Setze $q(x) = 1$ für $N_n = \emptyset$ und

$$q(x) = \prod_{i=1}^{m}(x - \lambda_i) \quad \text{für} \quad N_n = \{\lambda_1, \ldots, \lambda_m\}.$$

Dann ist $p_n q \in \mathbb{P}_{n+m}$ reell und hat in (a,b) keinen Vorzeichenwechsel. Es gilt

$$(p_n, q)_w = \int_a^b p_n(x)q(x)w(x)\,\mathrm{d}x \neq 0.$$

Für $m < n$ ist dies im Widerspruch zu $p_n \perp \mathbb{P}_{n-1}$. Also gilt $m = n$. □

Bemerkung. Für die Gewichtsfunktion $w(x) = \frac{1}{\sqrt{1-x^2}}$ erhält man beispielsweise die Tschebyscheff-Polynome.

Die orthogonalen Polynome p_n bezüglich des Skalarprodukts $(\cdot,\cdot)$ auf $[-1,1]$ mit $p_n(x) = x^n + \ldots$ sind Vielfache der *Legendre-Polynome*. Es gibt verschiedene Darstellungen dieser Polynome:

(i) Rodrigueres-Formel:

$$p_n(x) = \frac{n!}{(2n)!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n,$$

wobei hier $0! = 1$ gilt.

(ii) Rekursionsformel:

$$p_0(x) = 1, \quad p_1(x) = x, \quad p_{n+1}(x) = xp_n(x) - \frac{n^2}{4n^2-1}p_{n-1}(x), \quad n \geq 1.$$

Ihre Nullstellen werden für $n > 3$ numerisch bestimmt und können Tabellen entnommen werden. Zum Beispiel gilt

$$p_2(x) = x^2 - \frac{1}{3}, \quad \lambda_0 = -\sqrt{\frac{1}{3}}, \quad \lambda_1 = \sqrt{\frac{1}{3}},$$
$$p_3(x) = x^3 - \frac{3}{5}x, \quad \lambda_0 = -\sqrt{\frac{3}{5}}, \quad \lambda_1 = 0, \quad \lambda_2 = \sqrt{\frac{3}{5}}.$$

Nach Satz 3.15 können wir die Nullstellen $\lambda_0, \ldots, \lambda_n$ des $(n+1)$-ten Legendre-Polynoms p_{n+1} als Stützstellen einer interpolatorischen Quadraturformel auf dem Intervall $[-1,1]$

verwenden und erhalten

$$Q_n(f) = \sum_{i=0}^{n} a_i f(\lambda_i), \quad a_i = \int_{-1}^{1} \prod_{i \neq j}^{n} \frac{x - \lambda_j}{\lambda_i - \lambda_j} \, \mathrm{d}x.$$

Bemerkung. Die Knoten der Gauß-Quadraturformel, also die Nullstellen der Legendre-Polynome, sowie die zugehörigen Gewichte, können durch die Lösung eines tridiagonalen Eigenwertproblems berechnet werden (siehe Übung). Die Nullstellen des n-ten Legendre-Polynoms sind Eigenwerte von

$$T_n = \begin{pmatrix} \alpha_0 & \beta_1 & & \\ \beta_1 & & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_{n-1} \end{pmatrix}, \quad \alpha_i = 0, \quad \beta_i = \frac{i}{\sqrt{4i^2 - 1}}, \quad i = 0, \ldots, n-1.$$

Es sei $T_n = VDV^T$ eine orthogonale Diagonalisierung von T_n mit $V = [v_1 \cdots v_n]$ und $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Dann sind die Gewichte der Gauß-Quadraturformel

$$c_j = 2(v_j)_1^2.$$

```
function I = gauss(f,n)                  % (n+1)-pt Gauss quadrature of f
   beta = .5./sqrt(1-(2*(1:n)).^(-2)); % 3-term recurrence coeffs
   T = diag(beta,1) + diag(beta,-1);    % Jacobi matrix
   [V,D] = eig(T);                      % eigenvalue decomposition
   x = diag(D); [x,i] = sort(x);        % nodes (= Legendre points)
   w = 2*V(1,i).^2;                     % weights
   I = w*feval(f,x);                    % the integral
end
```

MATLAB-Code 9: Berechnung der Gauß-Quadratur mit Hilfe der Eigenwertzerlegung

Satz 3.16 (Gaußsche Quadraturformel). *Es gibt genau eine interpolatorische Quadraturformel zu $n+1$ paarweise verschiedenen Stützstellen über dem Intervall $[-1, 1]$ mit der Ordnung $2n+2$ (das heißt exakt für alle Polynome $p \in \mathbb{P}_{2n+1}$). Ihre Stützstellen sind gerade die Nullstellen $\lambda_0, \ldots, \lambda_n \in [-1, 1]$ des $(n+1)$-ten Legendre-Polynoms $p_{n+1} \in \mathbb{P}_{n+1}$ und ihre Gewichte genügen der Beziehung $a_i > 0$ für $i = 0, \ldots, n$. Für $f \in C^{2n+2}([-1, 1])$ besitzt das Restglied die Darstellung*

$$R_n(f) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^{1} \prod_{j=0}^{n} (x - \lambda_j)^2 \, \mathrm{d}x, \quad \xi \in (-1, 1).$$

Beweis. (i) Zunächst ist p_{n+1} orthogonal zu $\mathbb{P}_n[-1,1]$ und hat die Nullstellen $\lambda_0, \ldots, \lambda_n \in (-1,1)$. Es gilt also

$$p_{n+1}(x) = \prod_{i=0}^{n}(x-\lambda_i) = x^n + \ldots$$

Aufgrund der obigen Betrachtungen ist die zugehörige interpolatorische Quadraturformel von der Ordnung $2n+2$. Die typische Berechnung der Gewichte einer interpolatorischen Quadraturformel ist

$$a_i = \int_{-1}^{1} l_{jn}(x)\,\mathrm{d}x, \quad l_{jn}(x) = \prod_{i\neq j}^{n} \frac{x-\lambda_i}{\lambda_i-\lambda_j}, \quad j = 0, \ldots n.$$

Es gilt $l_{jn} \in \mathbb{P}_n$ und damit $l_{jn}^2 \in \mathbb{P}_{2n}$. Also wird auch l_{jn}^2 von der Quadraturformel exakt integriert und es folgt

$$0 < \int_{-1}^{1} (l_{jn}(x))^2\,\mathrm{d}x = \sum_{i=0}^{n} a_i l_{jn}^2(\lambda_i) = a_i,$$

da $l_{jn}^2(\lambda_i) = \delta_{ij}$. Dies liefert $a_j > 0$ für alle $j = 0, \ldots, n$.

(ii) Um die Eindeutigkeit zu zeigen, nehmen wir an, dass es eine zweite Quadraturformel

$$\tilde{Q}_n = \sum_{i=0}^{n} \tilde{a}_i f(\tilde{\lambda}_i)$$

gibt, die exakt für Polynome $p \in \mathbb{P}_{2n+1}$ ist. Mit den analog gebildeten Polynomen $\tilde{l}_{jn} \in \mathbb{P}_n$ folgt dann wie oben $\tilde{a}_i > 0$. Mit der Orthogonalität von p_{n+1} zu $\mathbb{P}_n[-1,1]$ folgt damit

$$0 = \int_{-1}^{1} \frac{1}{\tilde{a}_j} \tilde{l}_{in}(x) p_{n+1}(x)\,\mathrm{d}x = \sum_{j=0}^{n} \frac{\tilde{a}_j}{\tilde{a}_j} \tilde{l}_{in}(\lambda_j) p_{n+1}(\tilde{\lambda}_j) = p_{n+1}(\tilde{\lambda}_i).$$

Also sind auch $\tilde{\lambda}_i$ für $i = 0, \ldots, n$ Nullstellen von p_{n+1} und wegen der eindeutigen Bestimmtheit dieser gilt folglich $\lambda_i = \tilde{\lambda}_i$ und somit auch $a_i = \tilde{a}_i$, womit die Eindeutigkeit gezeigt ist.

(iii) Schließlich bleibt die Fehlerabschätzung der Gauß-Quadraturformel zu zeigen. Dazu benutzen wir ein Hilfsresultat der Hermite-Interpolation.

gegeben:	Knoten	$x_0 < x_1 < \cdots < x_m$,	$m \geq 0$,
	Werte	$y_i^{(k)} \in \mathbb{R}$,	$i = 0, \ldots, m, k = 0, \ldots, \mu_i$,
			$\mu_i \geq 0$ für $0 \leq i \leq m$,
gesucht:	$p \in \mathbb{P}_n, n = m + \sum_{i=0}^{m} \mu_i$,	mit $p(x_i)^k = y_i^{(k)}$,	$i = 0, \ldots, m, k = 0, \ldots, \mu_i$.

Nach Satz 2.14 besitzt die Hermite-Interpolationsaufgabe eine eindeutige Lösung und nach Satz 2.15 gilt für den Fehler der Hermite-Interpolation für eine Funktion $f \in C^{n+1}([x_0, x_m]$ die Gleichung

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x)) \prod_{i=0}^{m} (x - x_i)^{\mu_i + 1}, \quad \xi(x) \in [x_0, x_m].$$

Also gibt es ein Polynom $h \in \mathbb{P}_{2n+1}$, dass die Hermite-Interpolationsaufgabe

$$h(\lambda_i) = f(\lambda_i), \quad h'(\lambda_i) = f'(\lambda_i), \quad i = 0, \ldots, n$$

löst und für $f \in C^{2n+2}([-1, 1])$ die Restglieddarstellung

$$f(x) - h(x) = \frac{1}{(2n+2)!} f^{(2n+2)}(\xi(x)) \prod_{i=0}^{n} (x - \lambda_i)^2, \quad \xi(x) \in [-1, 1]$$

hat. Anwendung der Gaußschen Quadraturformel auf $h(x)$ ergibt wegen $Q_n(h) = I(h)$ dann

$$\begin{aligned} R_n(f) &= I(f) - Q_n(f) = I(f - h) - Q_n(f - h) \\ &= \int_{-1}^{1} \frac{1}{(2n+2)!} f^{(2n+2)}(\xi(x)) \prod_{i=0}^{n} (x - \lambda_i)^2 \, \mathrm{d}x - \sum_{i=0}^{n} a_i (f(\lambda_i) - h(\lambda_i)) \\ &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^{1} \prod_{i=0}^{n} (x - \lambda_i)^2 \, \mathrm{d}x \end{aligned}$$

aufgrund des Mittelwertsatzes der Integralrechnung. □

Clenshaw-Curtis-Verfahren

Auf $[-1, 1]$ ist das Clenshaw-Curtis-Verfahren die interpolatorische Quadraturformel zu den Tschebyscheff-Knoten 2. Art, also

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad 0 \leq j \leq n.$$

Die Gewichte können effektiv unter Verwendung der schnellen Fouriertransformation (FFT) berechnet werden.

```
function I = clenshaw_curtis(f,n)              % (n+1)-pt C-C quadrature of f
   x = cos(pi*(0:n)'/n);                       % Chebyshev points
   fx = feval(f,x)/(2*n);                      % f evaluated at these points
   g = real(fft(fx([1:n+1 n:-1:2])));          % Fast Fourier Transform
   a = [g(1); g(2:n)+g(2*n:-1:n+2); g(n+1)]; % Chebyshev coefficients
   w = 0*a'; w(1:2:end) = 2./(1-(0:2:n).^2); % weight vector
   I = w*a;                                    % the integral
end
```

MATLAB-Code 10: Clenshaw-Curtis-Verfahren

Bemerkung. Obwohl das Clenshaw-Curtis-Verfahren mit $n+1$ Knoten nur Polynome n-ten Grades exakt integriert, zeigen numerische Rechnungen, dass dieses Verfahren oft vergleichbare Genauigkeit bietet. Die Vorteile dieses Verfahrens liegen bei der trivialen Berechnung der Knoten und der effektiven Berechnung der Gewichte über die schnelle (siehe Abbildung 3.1). Fouriertransformation.

```
% Test Gauss QF
clear all;
close all;

% 1st test:
 f = @(x) x.^20;
 Iex = 2./21;

% 2nd test:
% f = @(x) exp(x);
% Iex = exp(1)-exp(-1);

% 3rd test
% f = @(x) abs(x).^3;
% Iex = 0.5;


for n=1:30
  Inumg = gauss(f,n);
  errorg(n)=abs(Iex-Inumg);
  Inumcc = clenshaw_curtis(f,n);
  errorcc(n) = abs(Iex-Inumcc);
end

semilogy([1:30],errorg,'r-+');
hold on;
semilogy([1:30],errorcc,'bo-');
```

MATLAB-Code 11: Matlab-Testskript zur Gauß-Quadratur und zum Clenshaw-Curtis-Verfahren

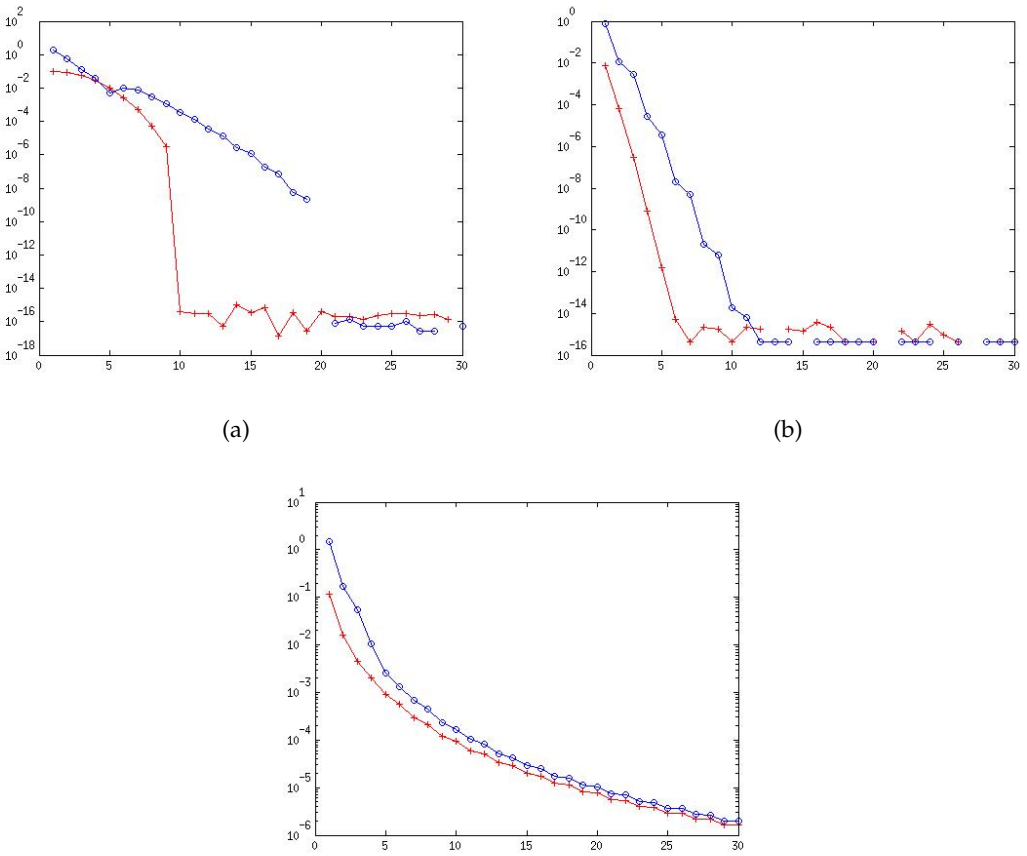

Abbildung 3.1: Fehler bei der Integration mit der Gauß-Quadratur (rot) und dem Clenshaw-Curtis-Verfahren (blau) von a) $f(x) = x^{20}$, b) $f(x) = e^x$ und c) $f(x) = |x|^3$.

In Abbildung 3.1 sind die Fehler der Gauß-Quadratur (rot) und des Clenshaw-Curtis-Verfahrens (blau) für drei verschiedene Funktionen aufgetragen. Dabei wurde jeweils das Integral der Funktion über dem Intervall $[-1, 1]$ berechnet. In Abbildung a) lässt sich deutlich die Ordnung der beiden Verfahren erkennen. In diesem Fall wurde das Polynom $f(x) = x^{20}$ betrachtet. Die Resultate zur Gauß-Quadratur besagen, dass diese die Ordnung $2n + 1$ hat. Entsprechend kann man ablesen, dass der Fehler für $n = 10$ einen deutlichen Sprung macht. Andererseits sieht man für das Clenshaw-Curtis-Verfahren, dass dieser Sprung des Fehlers erst bei $n = 20$ auftritt. Für das Beispiel $f(x) = e^x$ sieht man diesen deutlichen Unterschied nicht mehr. Dort liefern beide Verfahren ab etwa $n = 10$ gleichgute Ergebnisse. Schließlich zeigt das Beispiel $f(x) = |x|^3$, dass das Clenshaw-Curtis-Verfahren trotz der geringeren Ordnung nicht wesentlich schlechter ist als die Gauß-Quadratur.