Numerische Analysis im SS 2019

Frank Wübbeling

15. Juni 2021

Inhaltsverzeichnis

1.1 Einführung in die Numerische Mathematik

Buchübersicht. Grundlegende Begriffe.

Kapitel 2

Interpolation

Als Interpolation bezeichnen wir die Aufgabe, vorgegebene Funktionswerte auf sinnvolle Weise zu einer Funktion zu ergänzen. Die Interpolation tritt in vielen praktischen Varianten auf. Einige Beispiele:

1. Gegeben seien $(N+1)$ Punkte x_i, $i = 0 \ldots N$, in der Ebene. Verbinde die Punkte durch eine glatte Kurve, d.h. finde eine (differenzierbare) Funktion $s : [0, N] \mapsto \mathbb{R}^2$: $s(i) = x_i$, $i = 0 \ldots n$.

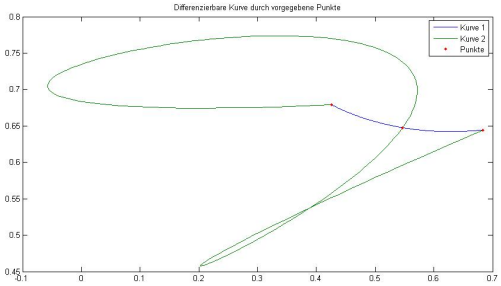


Abbildung 2.1: Interpolationsfunktionen

Klick für Bild interpol

2. Wir nehmen eine Holzlatte (**Straklatte**, engl. **Spline**) und biegen sie so, dass sie durch einige vorgegebene Punkte geht. Welche Form nimmt die Holzlatte an, bzw. welche Funktion stellt sie dar? Mathematisch: Die Energie, die in der Biegung einer Funktion steckt, ist proportional zum Integral über das Quadrat der zweiten Ableitung. Es sollte also die Norm der zweiten Ableitung der Formfunktion minimiert werden.

Sei $x_i \in [a,b]$, $i = 0 \dots N$. Finde $s : [a,b] \mapsto \mathbb{R}$, $s \in C^2([a,b])$, so dass $s(x_i) = y_i$, $i = 0 \dots N$ und

$$\int_a^b (s''(x))^2 \mathrm{dx} \leq \int_a^b g''(x)^2 \mathrm{dx}$$

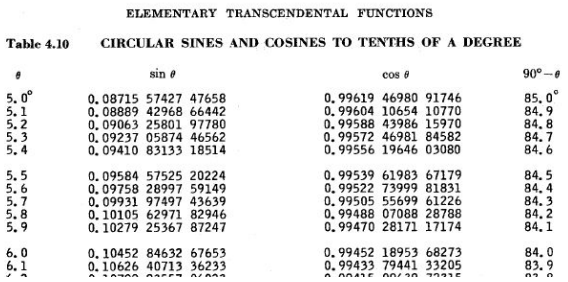
für alle Funktionen $g : [a,b] \mapsto \mathbb{R}$ mit $g(x_i) = y_i$, $i = 0 \dots N$, $g \in C^2([a,b])$.

Abbildung 2.2: Straklatte im Schiffsbau

Klick für Bild straklatte

Bemerkung: Außerhalb des Intervalls, in dem die Stützwerte liegen, ist die Funktion linear.

3. Eine Funktion h sei vertafelt (z.B. in dem Buch von Abramowitz and Stegun [1965]), und es seien die Werte $h(x_i) = y_i$ bekannt. Finde eine möglichst gute Näherung für h an einer Zwischenstelle ξ. Wie groß ist der maximale Fehler?



ELEMENTARY TRANSCENDENTAL FUNCTIONS

Table 4.10 CIRCULAR SINES AND COSINES TO TENTHS OF A DEGREE

θ	$\sin\theta$	$\cos\theta$	$90° - \theta$
5.0°	0.08715 57427 47658	0.99619 46980 91746	85.0°
5.1	0.08889 42968 66442	0.99604 10654 10770	84.9
5.2	0.09063 25801 97780	0.99588 43986 15970	84.8
5.3	0.09237 05874 46562	0.99572 46981 84582	84.7
5.4	0.09410 83133 18514	0.99556 19646 03080	84.6
5.5	0.09584 57525 20224	0.99539 61983 67179	84.5
5.6	0.09758 28997 59149	0.99522 73999 81831	84.4
5.7	0.09931 97497 43639	0.99505 55699 61226	84.3
5.8	0.10105 62971 82946	0.99488 07088 28788	84.2
5.9	0.10279 25367 87247	0.99470 28171 17174	84.1
6.0	0.10452 84632 67653	0.99452 18953 68273	84.0
6.1	0.10626 40713 36233	0.99433 79441 33205	83.9

Abbildung 2.3: Vertafelter sinus im Buch von Abramowitz und Stegun

Klick für Bild Abramo

4. Eine (transzendente) Funktion h soll auf einem Rechner ausgewertet werden, der nur die Grundrechenarten beherrscht. Gesucht ist eine berechenbare Funktion g, so dass

$$||h(x) - g(x)|| \leq \epsilon.$$

Dies ist ausdrücklich **keine** Interpolationsaufgabe. Wir schreiben nicht vor, dass die gesuchte Funktion durch feste Punkte gehen soll, sondern verlangen, dass sie sich einer gegebenen Kurve möglichst gut annähert. Dies ist das Problem der **Approximation** und wird in der Vorlesung Numerische Lineare Algebra ausführlich behandelt.

5. Ein Bild $f : \{1 \dots N\} \times \{1 \dots M\} \mapsto \mathbb{R}$ soll möglichst platzsparend abgespeichert werden. Hierzu bestimmen wir zunächst Koeffizienten a_{ik}, so dass

$$f(l,j) = \sum_{i,k} a_{ik} f_{ik}(l,j)$$

ist (Interpolationsschritt). Hierbei sind die f_{ik} z.B. trigonometrische Funktionen. Beim Abspeichern des Bildes ersetzen wir a_{ik} durch 0, falls sein Betrag klein ist, hierdurch wird eine Komprimierung erreicht. Zum Anzeigen wird die Näherung

$$\widetilde{f}(l,j) = \sum_{i,k} \widetilde{a}_{ik} f_{ik}(l,j)$$

berechnet, hierbei sind $\widetilde{a}$ die abgespeicherten Koeffizienten. Diese Methode ist die Standardmethode zur **Komprimierung in der Bildverarbeitung** (JPEG, MPEG, MP3).

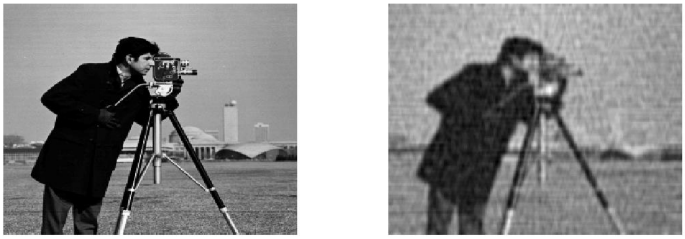


Abbildung 2.4: Original, zu stark komprimiertes Bild
Klick für Bild man
Klick für Bild man2

```
function comprimg
%COMPRIMG take cameraman.tif and delete small
%coefficients in the cosine transform
A=imread('cameraman.tif');
A=double(A);
B=dct2(A);
```

Listing 2.1: Bildkompression (interpolation/comprimg.m)

Klicken für den Quellcode von interpolation/comprimg.m

2.1 Polynominterpolation

Definition 2.1 (Allgemeine Interpolationsaufgabe)
Sei $N \in \mathbb{N}$. Seien $x_0, \ldots, x_N$ paarweise verschiedene Werte in $\mathbb{C}$ oder $\mathbb{R}$. Seien weiter $y_0, \ldots, y_N$ Elemente in $\mathbb{C}$ oder $\mathbb{R}$. Dann ist die **allgemeine Interpolationsaufgabe***:*
Suche eine Funktion f mit der Eigenschaft, dass $f(x_i) = y_i \, \forall i = 0..N$.
Die x_i heißen **Stützpunkte***, die y_i heißen* **Stützwerte***.*

In dieser Allgemeinheit, also ohne Einschränkung des zulässigen Funktionenraums, besitzt die Aufgabe offensichtlich unendlich viele Lösungen. Wir suchen f daher immer in einem angegebenen Funktionenraum, z.B den Polynomen.

Definition 2.2 (Aufgabe der Polynominterpolation, Polynomraum)
Sei $N \in \mathbb{N}$. Dann ist $\mathcal{P}_N$ der Raum der Polynome vom Grad kleiner oder gleich N.
Seien $x_0, \ldots, x_N$ paarweise verschieden, $y_0, \ldots, y_N$ gegeben. Dann ist die Aufgabe der **Polynominterpolation***:*
Finde ein $p \in \mathcal{P}_N$ mit $p(x_i) = y_i \, \forall i = 0 \ldots N$.

Damit gilt:

Satz 2.3 *Die Aufgabe der Polynominterpolation ist* **eindeutig** *lösbar.*

Beweis:

1. Formel von Lagrange, **Existenz** einer Lösung: Sei

$$w_j(x) := \prod_{\substack{k=0 \\ k \neq j}}^{N} \frac{x - x_k}{x_j - x_k}, \; j = 0 \ldots N.$$

   Dann ist $w_j(x) \in \mathcal{P}_N$, und $w_j(x_i) = \delta_{ji}$ für $j, i = 0 \ldots N$, denn für $j \neq i$ ist x_i Nullstelle des Zählers. Dabei ist

$$\delta_{ji} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

   das Kronecker–Delta. Sei

$$p(x) := \sum_{j=0}^{N} y_j w_j(x).$$

Dann ist $p \in P_N$, und es gilt

$$p(x_i) = \sum_{j=0}^{n} y_j w_j(x_i) = y_i w_i(x_i) = y_i$$

für alle $i = 0 \dots N$.

2. **Eindeutigkeit** der Lösung: Seien p_1 und p_2 Lösungen der Polynominterpolationsaufgabe. Sei $p = p_1 - p_2$. Dann ist $p \in \mathcal{P}_N$, und es gilt

$$p(x_i) = p_1(x_i) - p_2(x_i) = y_i - y_i = 0$$

für alle $i = 0 \dots N$. Also ist p ein Polynom vom Grad kleiner oder gleich N mit $N + 1$ Nullstellen, also ist nach dem Fundamentalsatz der Algebra $p = 0$, und damit $p_1 = p_2$.

□

Die Formel von Lagrange sichert die Existenz einer Lösung und gibt sie konstruktiv an, zur Auswertung eignet sie sich aber nicht, denn zur (naiven) Auswertung von $p(\widetilde{x})$ nach dieser Formel werden $N(N+1)$ Divisionen und Multiplikationen benötigt. Alternativ kann man die Koeffizienten des Interpolationspolynoms mit Hilfe der Vandermondematrizen bestimmen.

Definition 2.4 (Vandermondematrizen)
Es seien x_i, $i = 0 \dots N$ paarweise verschieden. Die Matrix $V \in \mathbb{C}^{n \times n}$, $V_{ik} = (x_i)^k$, $i, k = 0 \dots N$, heißt Vandermondematrix zu $x_0, \dots, x_N$.

Also:

$$V(x_0, \dots, x_N) = \begin{pmatrix} x_0^0 & \dots & x_0^N \\ \vdots & \ddots & \vdots \\ x_N^0 & \dots & x_N^N \end{pmatrix}$$

Satz 2.5 (Invertierbarkeit der Vandermondematrizen)
Seien $x_0, \dots, x_N$ paarweise verschiedene Zahlen, $y_0, \dots, y_N$ in $\mathbb{R}$ oder $\mathbb{C}$. Sei $p(x) = \sum_{k=0}^{N} a_k x^k$. Sei $y = (y_0, \dots, y_N)^t$, $a = (a_0, \dots, a_N)^t$, $V = V(x_0, \dots, x_N)$ Vandermonde–Matrix zu $x_0, \dots, x_N$. Dann gilt:

1. *p ist genau dann Lösung des Polynominterpolationsproblems, wenn $Va = y$.*
2. *V ist invertierbar.*

Beweis:

1. Es gilt $(Va)_j = p(x_j)$ und $p \in \mathcal{P}_N$.

2. Die Interpolationsaufgabe besitzt eine eindeutige Lösung nach 2.3, also ist V injektiv und surjektiv, also invertierbar.

□

Bemerkung: Da V quadratisch ist, reicht schon injektiv oder surjektiv allein zum Beweis der Invertierbarkeit. Dies ist einer der einfachsten Beweise der Abschätzung der Zahl der Nullstellen nach oben im Fundamentalsatz der Algebra: Mit Lagrange folgt, dass V surjektiv ist, also injektiv, also gibt es nur ein Polynom in $\mathcal{P}_N$ mit $(N+1)$ Nullstellen, das Nullpolynom.

Damit lassen sich die Koeffizienten eines Interpolationspolynoms durch Lösung eines linearen Gleichungssystems der Ordnung $(N+1)$ bestimmen, der Aufwand dazu beträgt $N^3/2$ Rechenoperationen (siehe Numerische LA), wobei wir eine Addition und eine Multiplikation zu einer Rechenoperation zusammenfassen. Das Polynom kann dann mit N Additionen und N Multiplikationen ausgewertet werden mit Hilfe des **Horner–Schemas**

$$p(x) = a_0 + x(a_1 + x(a_2 + (\ldots + a_n x))).$$

p lässt sich auch rekursiv aufbauen. Dies ist von Vorteil, wenn nachträglich eine Stützstelle hinzugefügt werden soll, bei Berechnung mit der Vandermonde–Matrix müsste in diesem Fall komplett neu gerechnet werden.

Satz 2.6 (Formel von Neville)
Gegeben sei die Polynominterpolationsaufgabe mit Stützstellen $x_0, \ldots, x_N$ und Stützwerten $y_0, \ldots, y_N$. Sei $p_{i\ldots k}$ die Lösung der Aufgabe für die Stützstellen $x_i, \ldots, x_k$ und Stützwerte $y_i, \ldots, y_k$, also

$$p_{i\ldots k} \in \mathcal{P}_{k-i},\ p(x_j) = y_j,\ j = i \ldots k.$$

Dann ist $p = p_{0\ldots N}$ die Lösung der vollen Aufgabe, und es gilt

$$p_{i\ldots k+1}(x) = \frac{1}{x_i - x_{k+1}} \left((x - x_{k+1})\, p_{i\ldots k}(x) + (x_i - x) p_{i+1\ldots k+1}(x) \right).$$

Beweis: Sei q das Polynom auf der rechten Seite. Dann ist $q \in \mathcal{P}_{k+1-i}$, und durch Einsetzen sieht man $q(x_j) = y_j$ für $j = i \ldots k+1$. Also ist q die eindeutige Lösung der Polynominterpolationsaufgabe und damit $q = p_{i\ldots k+1}$. □

Die Formel von Neville erlaubt die rekursive Berechnung von $p_{i\ldots k+1}$ aus $p_{i\ldots k}$ und $p_{i+1\ldots k+1}$ mit Hilfe des folgenden Schemas (N=2):

$$
\begin{array}{llll}
x_0 & y_0 = & p_0 & \\
 & & & p_{01} \\
x_1 & y_1 = & p_1 & \quad p_{012} \\
 & & & p_{12} \\
x_2 & y_2 = & p_2 &
\end{array}
$$

Kommt nun nachträglich eine weitere Stützstelle x_3 mit Stützwert y_3 hinzu, so wird einfach an dieses Schema unten eine weitere Reihe angehängt.

```
function out = neville( x,y )
%NEVILLE
%Compute interpolating polynomial through x,y
%Using cell arrays

if (nargin<1)
```

Listing 2.2: Polynomberechnung mit dem Neville–Schema (interpolation/neville.m)

Klicken für den Quellcode von interpolation/neville.m

Die Formel von Neville kann auch eingesetzt werden, um den Wert des Interpolationspolynoms an einer Stelle $x = z$ direkt auszurechnen (ohne explizite Berechnung der Koeffizienten des Polynoms). Hierzu wird im Neville–Schema jeweils direkt z eingesetzt (s. Beispiele).

```
function [out,out1] = nevilleeval( x,y,z )
%NEVILLEEVAL
%Evaluate interpolating polynomial through x,y at z
if (nargin<1)
    x=[-1 0 2];
    y=[1 2 3];
```

Listing 2.3: Auswertung mit dem Neville–Schema (interpolation/nevilleeval.m)

Klicken für den Quellcode von interpolation/nevilleeval.m

Nachteil bei der Berechnung des Interpolationspolynoms mit dem Neville–Schema ist, dass alle Einträge im Schema Polynome sind. Dies umgehen wir mit der zweiten Art der rekursiven Berechnung des Interpolationspolynoms mit der Form von Newton. Mit den Bezeichnungen aus der Formel von Neville gilt die

Satz 2.7 (Formel von Newton)

$$
p_{i\ldots k}(x) = p_{i\ldots k-1}(x) + \frac{(y_k - p_{i\ldots k-1}(x_k))}{(x_k - x_i)\cdots(x_k - x_{k-1})}(x - x_i)\cdots(x - x_{k-1}).
$$

Beweis: Die rechte Seite hat die richtige Ordnung. Da der zweite Summand verschwindet für $x_i \dots x_{k-1}$ und $p_{i\dots k-1}$ das Interpolationspolynom für $x_i \dots x_{k-1}$ ist, liefert die rechte Seite für diese Stützstellen den korrekten Wert. Der zweite Summand ist dann gerade so gebaut, dass er auch für x_k den richtigen Wert liefert, also ist die rechte Seite das gesuchte Interpolationspolynom $p_{i\dots k}$. □

Definition 2.8 (Dividierte Differenzen)
Der Koeffizient $[y_i, \dots, y_k] = \frac{(y_k - p_{i\dots k-1}(x_k))}{(x_k - x_i)\cdots(x_k - x_{k-1})}$ *in der Formel von Newton heißt dividierte Differenz von* $y_i, \dots, y_k$.

Satz 2.9 (Interpolation nach Newton, Rekursion der dividierten Differenzen)

1. $[y_i, \dots, y_k]$ *ist der Höchstkoeffizient (Koeffizient von* x^{k-i}*) in* $p_{i\dots k}$.
2. *Es gilt* $[y_i] = y_i$ *und*
$$[y_i, \dots, y_{k+1}] = \frac{1}{x_i - x_{k+1}}([y_i, \dots, y_k] - [y_{i+1}, \dots, y_{k+1}].$$
3. *Es gilt*
$$p(x) = \sum_{j=0}^{N} [y_0, \dots, y_j](x - x_0)\cdots(x - x_{j-1}).$$

Beweis:

1. Folgerung aus der Formel von Newton.
2. Folgerung aus 1. und der Formel von Neville.
3. Folgerung aus der Formel von Newton.

□

Ähnlich wie beim Neville–Schema wird auch die Newton–Form rekursiv berechnet:

$$\begin{array}{llll} x_0 & y_0 = [y_0] & & \\ & & [y_0, y_1] & \\ x_1 & y_1 = [y_1] & & [y_0, y_1, y_2] \\ & & [y_1, y_2] & \\ x_2 & y_2 = [y_2] & & \end{array}$$

Hierbei sind die Einträge im Schema, die dividierten Differenzen, anders als im Neville–Schema natürlich nur Zahlen, was die Berechnung erheblich vereinfacht.

```
function out = divdiff( x,y)
%divdiff
%compute divided differences of x and y
if (nargin<1)
    x=[-1 0 2];
    y=[1 2 3];
```

Listing 2.4: Berechung der Dividierten Differenzen (interpolation/divdiff.m)

Klicken für den Quellcode von interpolation/divdiff.m

Die Newton–Form lässt sich ähnlich wie das Horner–Schema auswerten:

$$p(x) = [y_0] + (x - x_0)([y_0, y_1] + (x - x1)([y_0, y_1, y_2] + \ldots)).$$

Beispiel 2.10 *Sei* $N = 2$, $x_0 = -1$, $x_1 = 0$, $x_2 = 2$, $y_0 = 1$, $y_1 = 2$, $y_2 = 3$.

Lagrange

$$\begin{aligned} w_0(x) &= \frac{(x-0)(x-2)}{(-1-0)(-1-2)} \\ w_1(x) &= \frac{(x-(-1))(x-(-2))}{(0-(-1))(0-2)} \\ w_2(x) &= \frac{(x-(-1))(x-0)}{(2-(-1))(2-0)} \\ p(x) &= 1w_0(x) + 2w_1(x) + 3w_2(x) = -x^2/6 + 5/6x + 2 \end{aligned}$$

Neville

$$\begin{array}{lll} -1 & 1 & 1 \\ & & \frac{1}{-1}(x + 2(-1-x) = 2 + x \\ 0 & 2 & 2 \\ & & \frac{1}{-2}(2(x-2) + 3(-x)) = 2 + x/2 \\ 2 & 3 & 3 \end{array} \qquad \frac{1}{-3}((x-2)(x+2) + (-1-x)(2+x/2)) = -x^2/6 + 5/6x + 2$$

Neville, ausgewertet für $x = 1$*:*

$$\begin{array}{lll} -1 & 1 & 1 \\ & & \frac{1}{-1}(1 + (-2) \cdot 2) = 3 \\ 0 & 2 & 2 \qquad\qquad \frac{1}{-3}(-1 \cdot 3 + (-2) \cdot (\frac{5}{2})) = \frac{8}{3} \\ & & \frac{1}{-2}(-1 \cdot 2 + (-1) \cdot 3) = \frac{5}{2} \\ 2 & 3 & 3 \end{array}$$

Newton

$$\begin{array}{lll} -1 & 1 & 1 \\ & & \frac{1}{-1}(1 - 2) = 1 \\ 0 & 2 & 2 \qquad\qquad \frac{1}{-3}(\frac{1}{2}) = -1/6 \\ & & \frac{1}{-2}(2 - 3) = 1/2 \\ 2 & 3 & 3 \end{array}$$

und damit

$$p(x) = p_{012}(x) = 1 + (x + 1) - 1/6x(x + 1) = -x^2/6 + 5/6x + 2.$$

Vandermonde

$$V(-1, 0, 2) = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 2 & 4 \end{pmatrix}, \; y = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

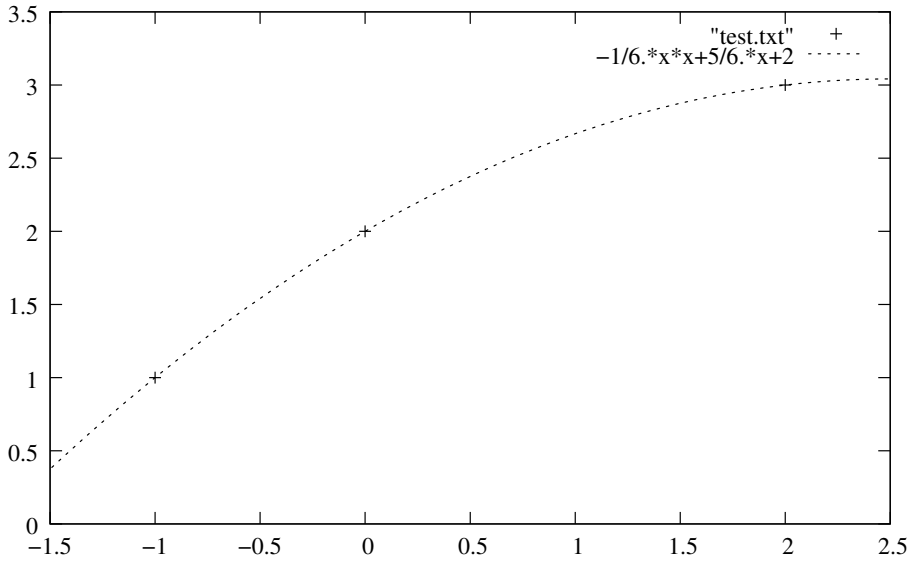
Es ergibt sich sofort $a_0 = 2$ *und*

$$-2a_1 + 2a_2 = -2, \; 2a_1 + 4a_2 = 1$$

und damit $a_2 = -1/6$*,* $a_1 = 5/6$*.*

Natürlich erzeugen alle Rechenvorschriften dasselbe Interpolationspolynom.

Im Diagramm erkennen wir, dass das Polynom die Punkte glatt verbindet (natürlich, denn es ist vom Grad 2).



Matlab-Code zur Berechnung der Koeffizienten des Interpolationspolynoms:

```
function p=interpolate(x,y)

%Find coefficients of interpolating polynomial
%by solving the linear equation with the Vandermonde matrix.
%Plot the result.
n=numel(x);
```

Listing 2.5: Polynominterpolation (interpolation/interpolate.m)

Klicken für den Quellcode von interpolation/interpolate.m

2.2 Interpolationsfehler bei Polynomen

Wir können die Polynominterpolation nutzen, um Funktionen zu approximieren. Sei in diesem Abschnitt f eine Funktion auf dem Intervall $[a, b] \subset \mathbb{R}$ nach $\mathbb{R}$. Wir werten f an den paarweise verschiedenen Stellen x_i, $i = 0 \dots N$, aus und betrachten das Interpolationspolynom p_N mit den Stützstellen x_i und Stützwerten $f(x_i)$, $i = 0 \dots N$.

$$p_N - f$$

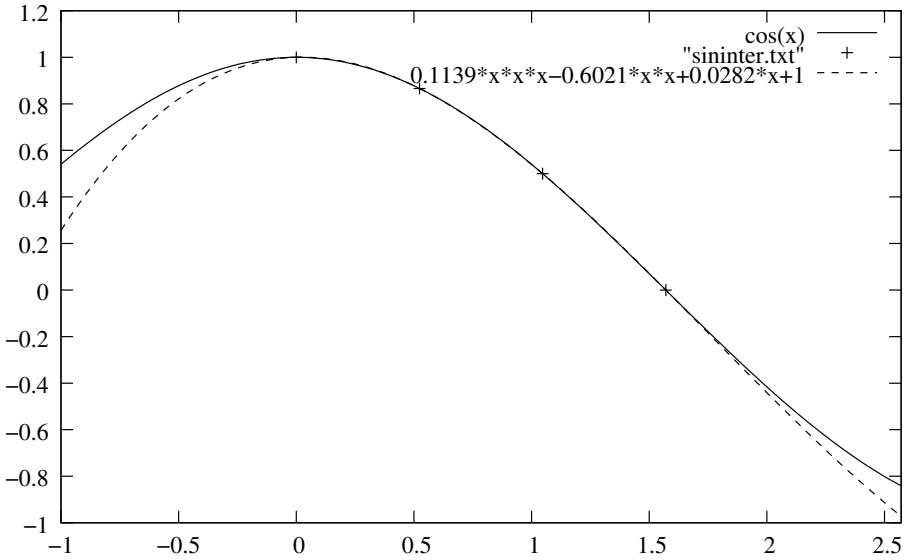
heißt **Interpolationsfehler**.

Ist p_N eine gute Approximation an f, also der Betrag des Interpolationsfehlers klein? Konvergiert p_N für wachsendes N und im Intervall $[a, b]$ gleichverteilte (äqui-

distante) Stützstellen $x_0, \ldots, x_N$ punktweise gegen f, wie wir es sicherlich erwarten?

Leider ist die Antwort häufig negativ (abhängig von f), die Polynominterpolation eignet sich nur begrenzt zur Approximation von Funktionen. Zwei typische Beispiele für äquidistante Stützstellen $x_k = a + k * (b - a)/N$, $k = 0 \ldots N$, folgen.

Beispiel 2.11 (Interpolation des Cosinus)

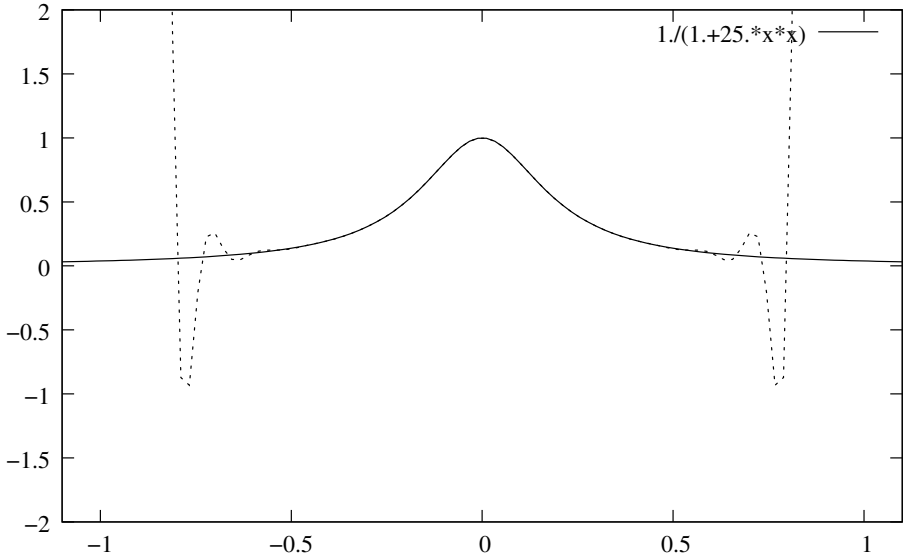


Interpolation des Cosinus auf $[0, \pi/2]$ mit vier Stützpunkten. Die Approximation ist bereits so exakt, dass innerhalb des von den Stützstellen abgedeckten Intervalls kaum ein Unterschied zwischen dem Cosinus und dem Interpolationspolynom vom Grade 3 sichtbar ist. Außerhalb steigt dagegen der Fehler schnell dramatisch an.

Beispiel 2.12 (Runge–Beispiel) *Runge and König [1925]*
Leider sind die Verhältnisse nicht immer so gut. Von Carl Runge stammt das Beispiel der Funktion

$$f(x) = \frac{1}{1 + 25x^2}$$

auf dem Intervall $[-1, 1]$: Für steigende Zahl der Stützstellen nimmt der maximale Fehler schnell zu.



Interpolation von $f(x) = 1/(1 + 25x^2)$ auf dem Einheitsintervall mit 30 äquidistanten Stützstellen. Die Approximation in der Nähe der o ist gut, am Rand beliebig schlecht.

Beispiel 2.13 Interpolation des Cosinus mit Auswertungsfehlern
Wir betrachten noch einmal die Interpolation des Cosinus. Diesmal nehmen wir aber an, dass wir den Cosinus nicht exakt berechnen, sondern dabei einen kleinen Fehler machen (wie es bei Messungen notwendig der Fall ist).

Bei niedrigen Polynomgraden hat dieser Fehler nur geringe Auswirkungen, bei höheren Polynomgraden bekommen wir dagegen Oszillationen wie im Beispiel von Runge.

Also: (Kleine) Störungen des Cosinus führen dazu, dass auch hier hohe Polynomgrade zu keiner vernünftigen Interpolation führen. Dies lässt bereits vermuten, dass die guten Eigenschaften des Cosinus bei der Polynominterpolation eine Ausnahme bilden.

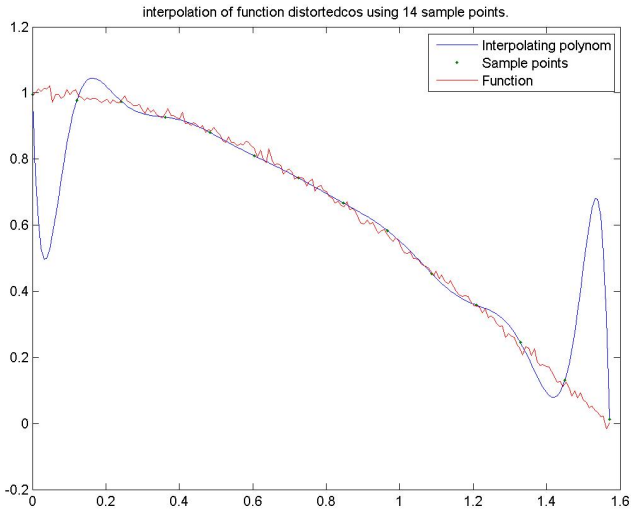


Abbildung 2.5: Interpolation des Cosinus mit kleinem Messfehler, mit äquidistanten Stützstellen

Klick für Bild distortedcos
Klick für Matlab Figure distortedcos

```
function p = polapprox( f,a,b,n,x )
%POLAPPROX Approximate a function f by polynomial
%interpolation in (n+1) equidistant sample points
if (nargin<5)
x=(0:n)/n*(b-a)+a;
end
```

Listing 2.6: Polynomapproximation (interpolation/polapprox.m)

Klicken für den Quellcode von interpolation/polapprox.m

```
function rungebeispiel(N)
%RUNGEBEISPIEL
if (nargin<1)
N=10;
end
close all;
```

Listing 2.7: Cosinus und Runge–Beispiel (interpolation/rungebeispiel.m)

Klicken für den Quellcode von interpolation/rungebeispiel.m

Wir stellen fest:

Bei steigender Zahl der Stützstellen sinkt der Betrag des Interpolationsfehlers nicht notwendig monoton. Er konvergiert nicht einmal notwendig gegen Null, wie wir es heuristisch erwarten würden.

Dieses Phänomen sorgte zu Beginn des 20. Jahrhunderts für großes Aufsehen. Insbesondere, weil es nicht eine in der Praxis nie auftauchende, obskure Funktion benutzt, sondern eine völlig unscheinbare, extrem glatte gebrochenrationale Funktion.

Polynominterpolation ist für uns das erste Beispiel eines Algorithmus, der scheinbar sinnvoll ist, aber tatsächlich viel größere Fehler liefert als er müsste. Es gibt nämlich einen viel einfacheren, konkurrierenden Algorithmus, bei dem offensichtlich die Interpolationsfunktion mit steigender Zahl der Nullstellen gegen die (stetige) zu interpolierende Funktion konvergiert: Dies ist die lineare Interpolation benachbarter Stützstellen, die wir mit ihren verwandten Algorithmen im Abschnitt über Splines noch betrachten werden.

Es gibt auch noch ein zweites Problem. Scheinbar liefert der Interpolationsalgorithmus für den Cosinus gute Ergebnisse (dies werden wir durch den nächsten Satz auch erklären). Tatsächlich ist er aber auch hier völlig unbrauchbar: Macht man kleine Fehler bei der Berechnung des Cosinus, wirken diese sich extrem auf das Ergebnis aus, und wir erhalten wieder hochoszillierende Funktionen. Auch hier würde die einfache lineare Interpolation wieder Abhilfe schaffen.

Algorithmen mit diesen Eigenschaften (das Ergebnis ist erheblich schlechter, als es sein müsste, ist typischerweise oszillierend, liefert betragsmäßig große, unsinnige Ergebnisse, reagiert sehr sensitiv auf Eingabefehler) nennen wir instabil. Eine genauere Definition werden wir im Abschnitt über Mehrschrittverfahren bei gewöhnlichen Differentialgleichungen kennenlernen.

Der folgende Satz schätzt den maximal zu erwartenden Interpolationsfehler ab.

Satz 2.14 (Abschätzung des Interpolationsfehlers)
Sei $f \in \mathbb{C}^{N+1}([a,b])$, $f : [a,b] \mapsto \mathbb{R}$. Seien x_i paarweise verschieden in $[a,b]$, $i = 0 \dots N$, und sei $p \in \mathcal{P}_N$ das zugehörige Interpolationspolynom mit $p(x_i) = f(x_i)$. Dann gilt:

$$\forall \overline{x} \in [a,b] \exists \widetilde{x} \in [a,b] \text{ mit } f(\overline{x}) - p(\overline{x}) = w(\overline{x}) \frac{f^{(N+1)}(\widetilde{x})}{(N+1)!},\ w(x) := \prod_{j=0}^{N} (x - x_j).$$

Insbesondere gilt

$$|f(x) - p(x)| \leq |w(x)| \frac{||f^{(N+1)}||_\infty}{(N+1)!}$$

und

$$||f - p||_\infty \leq ||w||_\infty \frac{||f^{(N+1)}||_\infty}{(N+1)!}$$

mit der Maximumnorm $||f||_\infty = \max_{x \in [a,b]} |f(x)|$.

Beweis:

1. Sei $\overline{x} = x_i$ für ein i. Dann ist $f(\overline{x}) = p(\overline{x})$, $w(\overline{x}) = 0 \Rightarrow$ Behauptung.
2. Sei $\overline{x} \neq x_i$ für alle $i = 0 \dots N$, also $w(\overline{x}) \neq 0$. Wir betrachten den Interpolationsfehler. Dieser hat bereits $(N+1)$ Nullstellen an den interpolierenden Punkten. Wir modifizieren die Fehlerfunktion nun leicht so, dass sie noch eine zusätzliche Nullstelle bei $\overline{x}$ hat. Sei also

   $$F(x) := (f(x) - p(x)) - Kw(x),\ K = \frac{f(\overline{x}) - p(\overline{x})}{w(\overline{x})}.$$

   F hat mindestens die $(N+2)$ verschiedenen Nullstellen $\overline{x}$ und x_i, $i = 0 \dots N$. Nach dem Satz von Rolle hat F' mindestens $(N+1)$ verschiedene Nullstellen, F'' mindestens N Nullstellen und $F^{(N+1)}$ hat mindestens eine Nullstelle $\widetilde{x}$ im Intervall $[a, b]$. $p \in \mathcal{P}_N$, also verschwindet $p^{(N+1)}$. Der Höchstkoeffizient von $x^{(N+1)}$ in w ist 1, also gilt

   $$p^{(N+1)}(x) = (N+1)!$$

   und damit insgesamt

   $$0 = F^{(N+1)}(\widetilde{x}) = f^{(N+1)}(\widetilde{x}) - K(N+1)! \Rightarrow K = \frac{f^{(N+1)}(\widetilde{x})}{(N+1)!}$$

   und damit

   $$0 = F(\overline{x}) = f(\overline{x}) - p(\overline{x}) - \frac{f^{(N+1)}(\widetilde{x})}{(N+1)!} w(\overline{x}).$$

□

Dieser Satz erklärt unsere einführenden Beispiele komplett. Alle Ableitungen des $\cos$ sind beschränkt, deshalb fällt hier der maximale Approximationsfehler schnell. Außerhalb des Intervalls $[a, b]$ steigt die Funktion w schnell an, deshalb bekommen

wir dort keine guten Approximationen. Die Ableitungen des Runge–Beispiels wachsen wie 50^n auch für kleine n rasant an, deshalb bekommen wir hier keine guten Approximationen. Wenn wir zufällige Auswertungsfehler berücksichtigen, so sind die dadurch entstehenden Funktionen sicherlich nicht mehr differenzierbar, und wir können den Satz nicht einmal mehr anwenden.

Bemerkung: Der Interpolationsfehler kann also durch eine Schranke abgeschätzt werden, die von der $(N+1)$. Ableitung von f und der Verteilung der Stützstellen (durch die Funktion $w(x)$) abhängt. Da wir f nicht beeinflussen können, sollten wir die x_i so wählen, dass $||w||_\infty$ möglichst klein wird.

Bemerkung: Die Voraussetzung, dass f glatt ist (also die $(N+1)$. Ableitung existiert), ist notwendig. Falls f nur k–mal differenzierbar ist, bringt eine Erhöhung des Polynomgrads jenseits von $k-1$ nichts mehr (siehe Übungen).

Beispiel 2.15

1. *Äquidistante Stützstellenwahl.*

   Ohne Einschränkung sei $[a,b] = [0,1]$ *und* $h = 1/N$ *der Abstand zwischen zwei Punkten. Sei* $f \in C^\infty([0,1]$. *Dann ist für* $j = \lfloor \frac{x}{h} \rfloor$ *bzw.* $j = \lceil \frac{x}{h} \rceil$

   $$\frac{|w(x)|}{(N+1)!} = h^{N+1} \frac{\prod_{i=0}^{N} |x/h - i|}{(N+1)!} < h^{N+1} \frac{j!(N+1-j)!}{(N+1)!} \leq \frac{1}{N^{N+1}}$$

   (Beweis durch Ausmultiplizieren des Zählers).

   Damit man für große N **keine** *Konvergenz des Interpolationsfehlers gegen* 0 *bekommt, muss die Supremumsnorm der* n*. Ableitung also schneller wachsen als* N^N.

   w *nimmt sein Betragsmaximum zwischen* x_0 *und* x_1 *bzw. zwischen* x_{N-1} *und* x_N *an, die Abschätzung wird also am Rand besonders schlecht.*

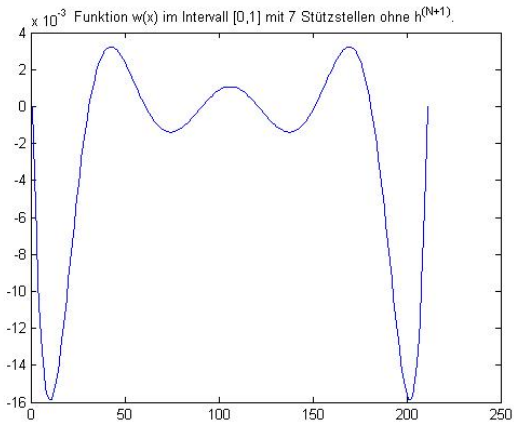


Abbildung 2.6: $w(x)$ ohne den Faktor h^{N+1} für $N = 7$

Klick für Bild approtest

Die inhaltliche Erklärung dafür, dass an den Rändern die Approximation schlecht ist: Bei äquidistanter Verteilung sind in der Nähe der Mitte des Intervalls doppelt so viele Stützpunkte wie am Rand. Für eine gleichmäßige Approximation sollten wir also die Stützstellen in der Nähe des Randes verdichten.

Diese Betrachtung gilt nicht bei der Interpolation periodischer Funktionen über die gesamte Periode: Hier sind alle Punkte des Intervalls gleichberechtigt, und tatsächlich zeigt eine einfache Symmetriebetrachtung, dass in diesem Fall die äquidistante Verteilung optimal ist.

Trefethen zeigt in einem Artikel, in dem er die Stützstellen als elektrisch geladene Teilchen interpretiert, die Optimalität der Dichteverteilung $1/(1-x^2)$ (Trefethen [2000]).

Unabhängig von all dem bedeutet ein hoher Polynomgrad ja auch noch einen hohen Aufwand bei der Auswertung des Polynoms. Grundsätzlich gilt die Regel: Ein hoher Polynomgrad bei der Interpolation (jenseits von ca. 8) ist im allgemeinen nicht zu empfehlen.

```
function out = approtest( N )
%APPROTEST
if (nargin<1)
    N=7;
end
close all;
```

Listing 2.8: Approximationsfehler (interpolation/approtest.m)

Klicken für den Quellcode von interpolation/approtest.m

2. *Wir betrachten wieder das Intervall $[a, b]$, teilen es aber in M Teile auf. In jedem Teilintervall führen wir eine Polynominterpolation an N äquidistanten Stützstellen durch, N fest. Um den Wert der Approximation an einer Stelle z zu berechnen, stellen wir zunächst fest, in welchem Teilintervall z liegt, und werten dort das Interpolationspolynom in diesem Intervall aus. In diesem Fall konvergiert für $M \mapsto \infty$ die Approximation gegen die Originalfunktion punktweise, und es gilt*

$$||f - p||_\infty \leq C \frac{||f^{(N+1)}||_\infty}{M^N}.$$

Für $N = 1$ wird in jedem Intervall durch eine Zahl approximiert, für $N = 2$ erhält man den Polygonzug.

Vorteil: Wir erhalten garantierte Konvergenz, der Aufwand zur Auswertung bleibt konstant.

Nachteil: Die entstehende Interpolationsfunktion ist zwar stetig (für $N > 1$), aber nicht mehr differenzierbar.

Diese Idee (Aufteilung der Approximationsaufgabe auf kleine Intervalle) wird die zentrale Idee für die Spline-Interpolation sein.

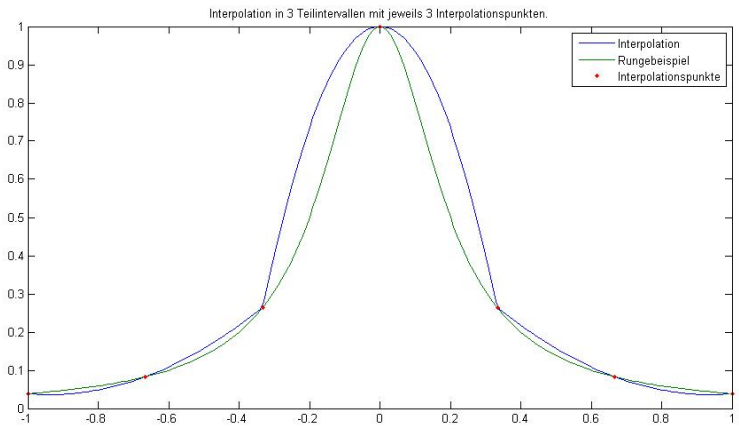


Abbildung 2.7: Interpolation in Teilintervallen

Klick für Bild partinter

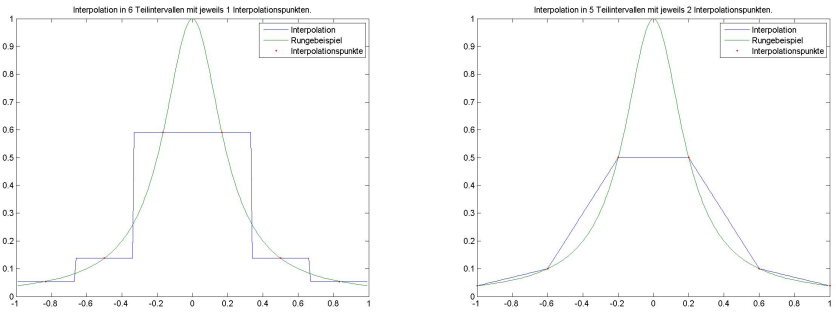


Abbildung 2.8: Auswertung und Polygonzug–Approximation
Klick für Bild partintero
Klick für Bild partinter1

```
function  partinter (M,N)
%PARTINTER
    function y=interpval(x)
        k=floor((x-a)/(b-a)*M)+1;
        y=polyval(Q(k,:),x);
    end
```

Listing 2.9: Interpolation in Teilintervallen (interpolation/partinter.m)

Klicken für den Quellcode von interpolation/partinter.m

Wir halten noch als ein wichtiges Ergebnis dieses Kapitels fest:

Korollar 2.16 *Die Polynominterpolation konvergiert bei steigendem Polynomgrad und äquidistanter Stützstellenwahl nicht notwendig gegen die zu interpolierende Funktion.*

Vorlesungsnotiz: 12.4.2013

2.3 Optimale Wahl der Stützstellen, Tschebyscheff–Interpolation

Im Licht von Satz 2.14 stellt sich die Frage: Falls wir frei sind in der Wahl der Stützstellen, welche Wahl liefert die beste Fehlerabschätzung, also den kleinsten Wert für $||w||_\infty$? Hierzu bestimmen wir das in der Maximumnorm kleinste Polynom p in $\mathcal{P}_{n+1}$ mit Höchstkoeffizient 1.

Definition 2.17 *(***Tschebyscheff–Polynome)**

$$T_n : [-1,1] \mapsto \mathbb{R},\ T_n(x) := \cos(n \arccos x),\ n \in \mathbb{N}$$

heißt **Tschebyscheff–Polynom** *der Ordnung* n*.*

Satz 2.18 Eigenschaften der Tschebyscheff–Polynome
Für die Tschebyscheff–Polynome T_n *gilt:*

1. $T_n \in \mathcal{P}_n$*. Für* $n > 0$ *hat* $T_n(x)/2^{n-1}$ *den Höchstkoeffizienten 1.*
2. *Die* T_n *bilden ein Orthogonalsystem im Vektorraum der stetigen Funktionen auf dem Intervall* $[-1,1]$ *bezüglich des Skalarprodukts*

$$(p,q) = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} p(x)q(x)dx.$$

3. *Die Nullstellen von* T_{n+1} *sind*

$$x_k^n = \cos\left(\frac{2k+1}{2(n+1)}\pi\right), k = 0 \ldots n.$$

4. *Es gilt*

$$||\frac{1}{2^{n-1}} T_n(x)||_\infty = \frac{1}{2^{n-1}} \leq ||p||_\infty$$

   für alle $p \in \mathcal{P}_n$ *mit Höchstkoeffizient 1.*

5. *Wählt man für eine Polynominterpolation vom Grad* n *die Stützstellen* x_k^n*,* $k = 0 \ldots n$*, so ist*

$$w(x) = \prod_{k=0}^{n} (x - x_k^n) = \frac{1}{2^n} T_{n+1}(x).$$

Beweis: Siehe Numerische Lineare Algebra .
Kurzer Beweis zu 4.: Sei

$$q(x) = T_n(x)/2^{n-1}.$$

Angenommen, $p \in \mathcal{P}_n$ mit Höchstkoeffizient 1 und

$$||p||_\infty < 1/2^{n-1} = ||q||_\infty.$$

Dann ist

$$r := q - p \in \mathcal{P}_{n-1}.$$

q nimmt sein Betragsmaximum mit wechselnden Vorzeichen an an den Stellen

$$z_k = \cos(k\pi/n),\ k = 0 \ldots n.$$

Wegen

$$|p(z_k)| \leq ||p||_\infty < ||q||_\infty = |q(z_k)|$$

gilt

$$\operatorname{sgn}(r(z_k)) = \operatorname{sgn}(q(z_k) - p(z_k)) = \operatorname{sgn}(q(z_k)).$$

r wechselt also ebenfalls mindestens n-Mal sein Vorzeichen, hat also n Nullstellen, also gilt $r = 0$ und damit $p = q$ im Widerspruch zur Annahme. □

Die Polynominterpolation, bei der wir die Stützstellen $x_0 \dots x_N$ als Nullstellen des Tschebyscheff–Polynoms T_{N+1} wählen, nennen wir **Tschebyscheff–Interpolation**. Wir erhalten für die Tschebyscheff–Interpolation nach 2.18 und 2.14 die Abschätzung

$$||f - p||_\infty \leq \frac{||f^{(n+1)}||_\infty}{2^n(n+1)!}.$$

Bemerkung: Durch die Abbildung

$$x \mapsto a + (x+1)\frac{b-a}{2}$$

wird das Intervall $[-1, 1]$ auf $[a, b]$ abgebildet. Für ein allgemeines Intervall $[a, b]$ lauten die Tschebyscheff–Stützstellen also

$$\widetilde{x_k^n} = a + (x_k^n + 1)\frac{b-a}{2}.$$

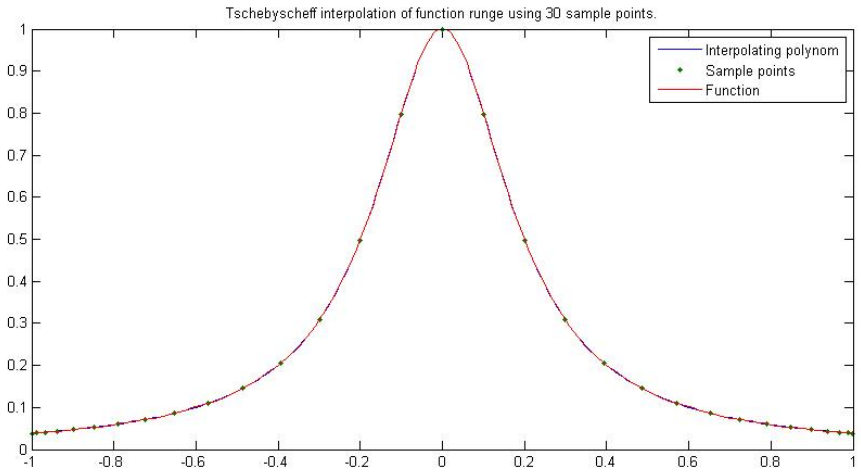


Abbildung 2.9: Tschebyscheff–Interpolation für das Runge–Beispiel

Klick für Bild tschebyscheff

```
function tscheb( N )
%TSCHEB
a=-1;
b=1;
f=@runge;
if (nargin<1)
```

Listing 2.10: Tschebyscheff–Interpolation (interpolation/tscheb.m)

Klicken für den Quellcode von interpolation/tscheb.m

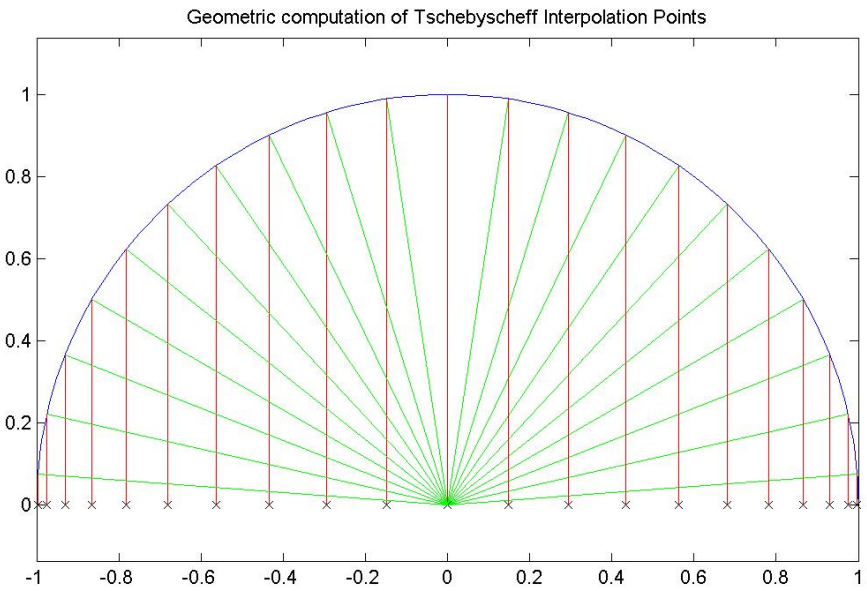


Abbildung 2.10: Geometrische Interpretation der Tschebyscheff–Interpolationspunkte als Abszissen der $n.$ komplexen Einheitswurzeln

Klick für Bild tschebgeom
Klick für Matlab Figure tschebgeom

```
function [ output_args ] = drawtscheb( input_args )
%DRAWTSCHEB
```

```
x=0:0.01:1;
hold off;
plot(cos(x*pi),sin(x*pi));
hold on;
```

Listing 2.11: Tschebyscheff: Geometrische Interpretation (integration/drawtscheb.m)

Klicken für den Quellcode von integration/drawtscheb.m

2.4 Hermite–Interpolation

Ein weiterer Spezialfall der Polynominterpolation ist die Hermite–Interpolation. Hier werden nicht nur die Werte der Funktion, sondern auch ihre Ableitungen spezifiziert. Es gilt der Satz:

Satz 2.19 *Gegeben seien die Stützstellen x_k, $k = 0 \dots N$, und die Stützwerte y_k^i, $k = 0 \dots N$, $i = 0 \dots N_k$. Weiter sei $M = \sum_{k=0}^{N}(N_k + 1) - 1$. Dann gibt es genau ein Polynom $p \in \mathcal{P}_M$ mit*

$$p^{(i)}(x_k) = y_k^i,\ k = 0 \dots N,\ i = 0 \dots N_k.$$

Beweis: Übungen. □

Definition 2.20 *Interpolationsaufgaben, bei denen an einigen Stützstellen zusätzlich zum Wert der Funktion auch der Wert von Ableitungen vorgeschrieben ist, nennen wir* **Hermite–Interpolationsaufgaben**.

Bemerkung: Für die Hermite–Interpolation gilt die Fehlerabschätzung 2.14 mit $w(x) = \prod_k (x - x_k)^{N_k}$, der Beweis verläuft wörtlich wie der Beweis zu 2.14.

Beispiel 2.21 *Sei $N = 2$, $x_0 = -1$, $x_1 = 0$, $x_2 = 1$, $y_0^0 = 1$, $y_0^1 = 0$, $y_1^0 = 0$, $y_2^0 = 0$, $y_2^1 = 0$. Wir suchen also ein Interpolationspolynom durch $(-1, 1)$, $(0, 0)$ und $(1, 1)$, dessen Ableitung an den Rändern verschwindet. Die Koeffizienten lassen sich durch eine entsprechend angepasste Vandermonde–Matrix berechnen, das resultierende Polynom ist*

$$p(x) = x^2(2 - x^2).$$

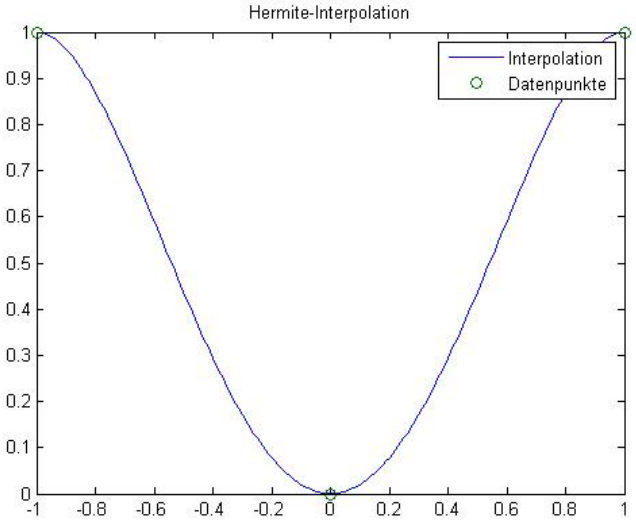


Abbildung 2.11: Beispiel zur Hermite–Interpolation

Klick für Bild hermite

```
function hermitebeispiel(x,N,y)
%HERMITEBEISPIEL
if (nargin<1)
    x=[-1 0 1];
    N=[2 1 2];
    y={[1 0] [0] [1 0]};
```

Listing 2.12: Programm zur Hermite–Interpolation (interpolation/hermitebeispiel.m)

Klicken für den Quellcode von interpolation/hermitebeispiel.m

2.5 Richardson–Extrapolation

Es sei der Grenzwert a_0 einer Funktion f für $h \mapsto 0$ zu bestimmen, die an der Stelle 0 nicht direkt auswertbar ist. Zur Verfügung stehen Auswertungen der Form $y_k = f(h_k)$, $k = 0 \ldots N$. Es sei bekannt, dass f mindestens eine Taylorreihe bis zum Grad N in h^p besitzt, also

$$f(h) = a_0 + \sum_{k=1}^{N} a_j h^{pk} + C(h)h^{p(N+1)} \tag{2.1}$$

mit (unbekannten) Koeffizienten a_j und einer (unbekannten) beschränkten Funktion $C(h)$ mit $|C(h)| \leq C$, aber bekanntem p (typischerweise ist $p = 1$ oder $p = 2$). Dies ist sicher richtig, falls f $(N+1)$-mal stetig differenzierbar ist auf einem Intervall um die 0. Wie erreicht man eine möglichst gute Approximation an den Grenzwert?

Wir bemerken zunächst, dass für kleine h der Restterm hinter der Summe keine Rolle mehr spielt, denn er geht auf der rechten Seite am schnellsten gegen Null. Vernachlässigen wir ihn für den Augenblick, so ist $p(h) = f(h^{1/p})\,(h > 0) \in \mathcal{P}_N$. p ist als Polynominterpolation der Punkte $(h_k^p, f(h_k))$ eindeutig bestimmt. Den gesuchten Wert $f(0)$ approximieren wir durch $p(0)$. Aufgrund unserer Herleitung erwarten wir dabei den Fehler $Ch_{\max}^{p(N+1)}$.

Kurz: Wir nehmen einige Auswertepunkte, legen ein Polynom hindurch, und werten das Polynom an der gesuchten, nicht berechenbaren Stelle aus. Dazu nutzen wir natürlich die Formel und das Schema von Neville 2.6.

Beispiel 2.22 (Richardson–Extrapolation)
Gesucht sei der Grenzwert 1 *der Funktion* $\operatorname{sinc}(x) = sinx/x$ *für* $x \mapsto 0$. *Der* sinc *ist gerade, hat also eine Taylorreihenentwicklung in* x^2, *also gilt* $p = 2$. *Wir interpolieren also an den Stellen* $(h_k^2, f(h_k)$ *durch Polynome. Zur Verfügung stehen Auswertungen von* $\operatorname{sinc}(2^{-k})$, $k = 0 \ldots N$. *Wir werten aus für* $x = 0$ *mit dem Schema von Neville 2.6. Angegeben ist jeweils der Fehler, also die Differenz des berechneten Werts zum korrekten Ergebnis 1:*

$$
\begin{array}{llllll}
-1.5853e-001 & -2.0222e-003 & -3.0442e-006 & -6.6472e-010 & -2.3759e-014 & -1.1102e-016 \\
-4.1149e-002 & -1.2924e-004 & -4.8220e-008 & -2.6202e-012 & -1.1102e-016 & \\
-1.0384e-002 & -8.1229e-006 & -7.5602e-010 & -1.0325e-014 & & \\
-2.6021e-003 & -5.0839e-007 & -1.1823e-011 & & & \\
-6.5091e-004 & -3.1785e-008 & & & & \\
-1.6275e-004 & & & & &
\end{array}
$$

```
function [ output_args ] = richardson1( input_args )
%RICHARDSON1
f=@sinc;
N=5;

format short e;
```

Listing 2.13: Richardson–Extrapolation der Sinc–Funktion (interpolation/richardson1.m)

Klicken für den Quellcode von interpolation/richardson1.m

Aus Folgegliedern mit der Genauigkeit 10^{-4} wird hier also durch die Richardson–Extrapolation ein Grenzwert mit der Genauigkeit 10^{-16} berechnet. Andere Beispiele sind etwa Differenzenquotienten (Grenzwert von $(f(x+h) - f(x))/h$ für $h \mapsto 0$)

oder die näherungsweise Berechnung von Integralen (siehe **??**). Zum Verständnis dieses Verhaltens führen wir zunächst das Landau–Symbol $O()$ ein.

Definition 2.23 (Landau–Symbol)
Seien $f, g : X \mapsto \mathbb{R}$, $X = \mathbb{R}$ oder $\mathbb{C}$. f heißt von der Ordnung g ($f = O(g)$) genau dann wenn:

1. $f(N) = O(g(N))$ *für große N, falls es Konstanten N_0 und C gibt mit*

$$|f(N)| \leq C|g(N)| \, \forall N > N_0.$$

2. $f(h) = O(g(h))$ *für kleine h, falls es Konstanten h_0 und C gibt mit*

$$|f(h)| \leq C|g(h)| \, \forall h < h_0.$$

Beispiel 2.24 *(zum Landau–Symbol)*

1. $N^2 = O(N^3)$, *allgemein* $N^a = O(N^b)$ *für* $a < b$.
2. $h^3 = O(h^2)$, *allgemein* $h^a = O(h^b)$ *für* $a > b$.
3. $\sin(x) = O(1)$ *für alle* $x \in \mathbb{R}$.

Die Ordnung schätzt ab, wie sich ein Term für sehr große N oder sehr kleine h verhält. Für h gilt insbesondere: $f(h) = a_0 + O(h^n)$ konvergiert umso schneller gegen a_0, je größer n ist.

Für die Richardson–Extrapolation, also die erste Zeile im Neville–Schema, gilt: Die erste Spalte hat eine Genauigkeit von $O(h^p)$, die zweite von $O(h^{2p})$ usw.

Satz 2.25 (Richardson–Extrapolation)
$f(x)$ besitze an der Stelle 0 eine Taylorreihe bis zum Grad N in x^p. Sei $g(x) = f(x^{1/p})$, $x \geq 0$. Berechne das Neville–Schema für die Stützstellen $x_k = (a^k h)^p$, $0 < a < 1$, $h > 0$, und Stützwerte $y_k = g(x_k) = f(a^k h)$. Dann gilt für die Einträge $p_{i\ldots i+k-1}(0)$ in der k. Spalte des Tableaus

$$p_{i\ldots i+k-1}(0) = a_0 + \sum_{j=k}^{N} a_{k,j}((a^i h)^p)^j$$

mit Zahlen $a_{k,j}$. Insbesondere gilt für die Zahlen in der ersten Zeile des Tableaus

$$p_{0\ldots k} = a_0 + O(h^{pk)}).$$

Beweis: Es sei zunächst $p = 1$, also $f = g$. In der ersten Spalte des Tableaus stehen die Stützwerte

$$p_i(0) = y_i = f(a^i h) = a_0 + \sum_{j=0}^{N} a_j (a^i h)^j + O(h^{N+1}),$$

das ist die Aussage für $k = 1$.
Sei der Satz nun richtig für die Spalte k. Wir zeigen, dass in der ersten Zeile in Spalte $k + 1$ die Exponenten bis h^k herausfallen. Der Eintrag wird nach der Formel von Neville 2.6 berechnet durch

$$\begin{aligned}
&p_{0\ldots k}(0) \\
&= \frac{1}{x_0 - x_k}((0 - x_k)p_{0\ldots k-1}(0) + (x_0 - 0)p_{1\ldots k}(0)) \\
&= \frac{1}{h(1 - a^k)}((-ha^k)(a_0 + a_{k,k}h^k + O(h^{k+1}) + (h)(a_0 + a_{k,k}(ah)^k + O(h^{k+1})))) \\
&= a_0 + O(h^{k+1}).
\end{aligned}$$

Die Rechnung kann so für alle Zeilen durchgeführt werden, das gibt die Aussage des Satzes für $p = 1$.
Für $p \neq 1$ wird die Rechnung für g statt f durchgeführt. Nur für $p \neq 1$ benötigen wir die Voraussetzung $h > 0$. □
Die Richardson–Interpolation berechnet also eine Linearkombination der bekannten Werte, so dass sich die Ordnung der Konvergenz erhöht. Wichtig: Die Koeffizienten der Taylorentwicklung a_k müssen nicht bekannt sein (ansonsten könnte man den Fehler auch gleich direkt abziehen), geschickte Linearkombination lässt die Fehler herausfallen.

Man kann sich sehr schnell klarmachen, wie die Richardson–Extrapolation funktioniert. Nehmen wir an, dass uns die Auswertungen $y_0 = f(h)$ und $y_1 = f(h/2)$ zur Verfügung stehen, um $f(0)$ auszurechnen, und dass f eine Taylorentwicklung in h^2 hat. Es gilt also:

$$\begin{aligned}
y_0 &= f(0) + a_0 h^2 && +O(h^4) \\
y_1 &= f(0) + a_0 (h/2)^2 && +O(h^4)
\end{aligned}$$

Wir bilden nun eine günstige Linearkombination von y_0 und y_1, so dass der störende Term in h^2 herausfällt. Also

$$\tilde{y} = 4y_1 - y_0 = 3f(0) + O(h^4)$$

und damit

$$\frac{1}{3}\tilde{y} = f(0) + O(h^4)$$

und wir haben eine Formel zur Berechnung von $f(0)$ mit der Ordnung h^4 gefunden, dies ist gerade die, die auch Richardson liefert.

Wir wollen Richardson an zwei ganz kleinen Beispielen testen. Es soll die Ableitung der Funktion f mit Taylorentwicklung an der Stelle 0 ausgerechnet werden. Hierzu betrachten wir die Funktion

$$F(h) = \begin{cases} \frac{f(h)-f(0)}{h} & h \neq 0 \\ f'(0) & h = 0 \end{cases}.$$

$f(h)$ besitze um 0 eine Taylorentwicklung in h mit Koeffizienten a_k. Dann besitzt auch $F(h)$ um 0 eine Taylorentwicklung in h. Wir suchen $F(0)$, das nicht direkt ausgerechnet werden kann. Statt dessen werten wir F an einigen Stellen aus und versuchen, darüber $F(0)$ mit Richardson zu approximieren.

Wir wählen zunächst $x_0 = -h$, $x_1 = h$. Es stehen also die Werte $y_0(h) = F(-h)$ und $y_1(h) = F(h)$ zur Verfügung. Neville liefert die Auswertung der interpolierenden Geraden an der Stelle 0, d.h.

$$\tilde{y}(h) = \frac{1}{2}(y_1(h) + y_0(h)) = \frac{f(h) - f(-h)}{2h}.$$

$\tilde{y}(h)$ hat die Taylorentwicklung

$$\tilde{y}(h) = f'(0) + \sum_{k=1}^{\infty} a_{2k+1} h^{2k}$$

und damit einen Fehler von $O(h^2)$, während $y_0(h)$ und $y_1(h)$ für sich jeweils einen Fehler in $O(h)$ aufweisen.

Jetzt wählen wir $x_0 = h/2$, $x_1 = h$. Das interpolierende Polynom ist in diesem Fall mit Lagrange

$$p_h(x) = y_0 \frac{x - h}{h/2 - h} + y_1 \frac{x - h/2}{h - h/2}$$

und wir erhalten für $x = 0$

$$\tilde{y}(h) = 2y_0 - y_1 = \frac{1}{h}\left(4f\left(\frac{h}{2}\right) - f(h) - 3f(0)\right) = f'(0) + O(h^2).$$

Bemerkung: Ein Spezialfall der Richardson–Interpolation ist die Romberg–Integration 3.3.

2.6 Rationale Interpolation

Natürlich können wir auch mit gebrochenrationalen Funktionen interpolieren.

Definition 2.26 (Rationale Interpolation)
Seien $x_k \in \mathbb{C}$ paarweise verschiedene Stützstellen, y_k Stützwerte, $k = 0 \dots N$. Es sei $P \geq 0$, $Q \geq 0$. Die Aufgabe

$$\textit{Bestimme } p \in \mathcal{P}_P,\ q \in \mathcal{P}_Q \textit{ mit } \frac{p(x_k)}{q(x_k)} = y_k,\ k = 0 \dots N$$

heißt rationale Interpolationsaufgabe.

Erste Frage ist natürlich, wie N zu wählen ist. p hat $P + 1$ Koeffizienten, q hat $Q + 1$ Koeffizienten, also insgesamt $P + Q + 2$ Freiheitsgrade, man könnte also $N = P + Q + 1$ wählen, um $P + Q + 2$ Bedingungen zu erfüllen.

Offensichtlich kann man aber mit einer Zahl erweitern, ohne dass sich der Wert des Bruchs ändert, das reduziert die Zahl der Freiheitsgrade um 1. Wir wählen also $N = P + Q$. Selbst dann kann man aber noch mit Polynomen von höherem Grad in Zähler und Nenner erweitern, ohne den Wert zu ändern. Existenz und Eindeutigkeit sind für die rationale Interpolation mit dieser Wahl also nicht gesichert (siehe Übungen).

Rationale Interpolation hat häufig bessere Interpolationseigenschaften als die normale Polyominterpolation (das Runge–Beispiel etwa kann sie natürlich exakt approximieren!).

Zur **Berechnung** von p und q multiplizieren wir 2.26 mit $q(x_k)$ und erhalten

$$y_k q(x_k) = p(x_k),\ k = 0 \dots N.$$

Häufig normiert man der Einfachheit halber den Koeffizienten von 1 in p auf 1, also $p(0) = 1$. Korrekter wäre hier: Sei k_0 ein Index mit $y_{k_0} \neq 0$. Dann setzen wir $p(x_{k_0}) = 1$ und können hieraus den Koeffizienten der 1 eliminieren.

In jedem Fall erhalten wir ein lineares Gleichungssystem für die restlichen $P + Q + 1 = N + 1$ Koeffizienten mit $N + 1$ Gleichungen, das wir zur Berechnung der Koeffizienten nutzen, sofern die Aufgabe lösbar ist.

Wegen seiner guten Glattheitseigenschaften ist die rationale **Approximation** mit Splines die Grundlage vieler Algorithmen in der Computergrafik (etwa NURBS, nonuniform rational B-Splines).

```
function [a,b] = ratinterp( x,y,n,m )
%RATINTERP rational interpolation
%assumes that po\not=o.
M=n+m+1;
if M~=numel(x)
    'Incorrect Size.'
```

Listing 2.14: Rationale Interpolation (interpolation/ratinterp.m)

Klicken für den Quellcode von interpolation/ratinterp.m

```
function y = rateval( a,b,x )
%RATEVAL
y=polyval(a,x)./polyval(b,x);
end
```

Listing 2.15: Rationale Auswertung (interpolation/rateval.m)

Klicken für den Quellcode von interpolation/rateval.m

```
function [ output_args ] = ratdemo( n,m )
%RATDEMO
if (nargin<1)
    n=2;
end
if (nargin<2)
```

Listing 2.16: Beispiel zur rationalen Interpolation (interpolation/ratdemo.m)

Klicken für den Quellcode von interpolation/ratdemo.m

2.7 Interpolation mit allgemeinen Ansatzfunktionen

Bei der Polynominterpolation versucht man, eine Funktion f durch eine Linearkombination der Monome x^k, $k = 0 \dots N$, zu approximieren. Diese Wahl ist nicht immer optimal. Falls etwa bekannt ist, dass die Funktion f stark (exponentiell) wächst, sollte man f durch eine Linearkombination von stark wachsenden Funktionen approximieren. Entsprechend: Falls f periodisch ist, etwa mit der Periode 2π, sollte es durch eine Linearkombination von periodischen Funktionen approximiert werden, also am einfachsten durch e^{ikx} in $\mathbb{C}$ oder $\sin(kx)$, $\cos(kx)$ in $\mathbb{R}$. Das Interpolationsproblem mit allgemeinen Ansatzfunktionen lautet

Definition 2.27 (Interpolation mit allgemeinen Ansatzfunktionen)
Seien $f_j : I \mapsto \mathbb{C}$, $j = 0 \dots N$, Ansatzfunktionen. Gegeben seien die paarweise verschiedenen Stützstellen x_k und Stützwerte y_k, $k = 0 \dots N$. Finde Koeffizienten a_j so dass

$$y_k = \sum_{j=0}^{N} a_j f_j(x_k),\ k = 0 \dots N.$$

Die Wahl der Ansatzfunktionen bietet also eine Möglichkeit, zusätzliches Wissen (a priori-Wissen) über die zu approximierende Funktion mit einzubringen: Der von den Ansatzfunktionen aufgespannte Raum sollte dieselben Eigenschaften haben wie die Funktion f. Weiß man etwa, dass f unstetig ist im Punkt z, so sollte zumindest eine der Ansatzfunktionen ebenfalls unstetig sein in z.

Das zugehörige Gleichungssystem für die a_j ergibt sich mit der Vandermonde–ähnlichen Matrix

$$V = \begin{pmatrix} f_0(x_0) & \cdots & f_N(x_0) \\ \vdots & \ddots & \vdots \\ f_0(x_N) & \cdots & f_N(x_N) \end{pmatrix} \tag{2.2}$$

und das allgemeine Interpolationsproblem ist eindeutig lösbar, falls V invertierbar ist. Für $f_k(x) = x^k$ erhalten wir die Polynominterpolation zurück.

Wir betrachten als Spezialfälle die Interpolation mit trigonometrischen Funktionen und die Splines.

2.8 Trigonometrische Interpolation: Diskrete Fouriertransformation

2.8.1 Eindimensionale diskrete Fouriertransformation

Als einen Spezialfall der Polynominterpolation und der Interpolation mit allgemeinen Ansatzfunktionen betrachten wir die trigonometrische Interpolation. Bei der Definition der Polynominterpolation hatten wir ausdrücklich Stützstellen und –werte in $\mathbb{C}$ zugelassen. Für die trigonometrische Interpolation wählen wir als Stützstellen x_k die $n.$ Einheitswurzeln, also

$$x_k = e^{2\pi i k/N} = \omega_N^k,\ \omega_N = e^{2\pi i/N},\ (x_k)^N = 1,\ k = 0 \dots N-1.$$

Die x_k liegen äquidistant auf dem Rand des Einheitskreises in der komplexen Ebene (deshalb auch “Interpolation am Kreis”).

Satz 2.28 Inverse der Vandermondematrix für Einheitswurzeln
Seien $(x_k) = \omega_N^k$, $k = 0 \dots N-1$, $W = V(x_0, \dots, x_{N-1})$ *die zugehörige Vandermonde–Matrix. Dann gilt*

$$W\overline{W} = NI \text{ und damit } W^{-1} = \frac{1}{N}\overline{W}.$$

Beweis: Durch Ausrechnen. Der Einfachheit halber wählen wir für W die Indizierung von 0 bis $N-1$.

$$\begin{aligned}
(W\overline{W})_{jk} &= \sum_{l=0}^{N-1} W_{jl}\overline{W}_{lk} \\
&= \sum_{l=0}^{N-1} \omega_N^{jl}\omega_N^{-lk} \\
&= \sum_{l=0}^{N-1} (\omega_N^{j-k})^l \\
&= \begin{cases} N & j = k \\ \frac{(\omega_N^{j-k})^N - 1}{\omega_N^{j-k} - 1} = 0 & j \neq k \end{cases} \\
&= n\delta_{jk}
\end{aligned}$$

□

Bemerkung:

$$W = \left(\omega_N^{jk}\right)_{jk} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_n^{2(N-1)} \\ \vdots & \vdots & & \vdots & \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}.$$

W ist symmetrisch, aber nicht selbstadjungiert.

Mit Hilfe dieser Formel für die Inverse der Vandermonde–Matrix lassen sich die Koeffizienten des Interpolationspolynoms der trigonometrischen Interpolation sofort angeben.

Korollar 2.29 (Interpolation am Kreis)
Sei $x_k = \omega_N^k$, $\omega_N = e^{2\pi i/N}$, $y_k \in \mathbb{C}$, $k = 0 \dots N-1$. *Dann ist das Interpolationspolynom* $p \in P_{N-1}$ *mit* $p(x_k) = y_k$ *gegeben durch*

$$p(x) = \frac{1}{N}\sum_{k=0}^{N-1} \widehat{y}_k x^k,\ \widehat{y}_k = \sum_{j=0}^{N-1} y_j \overline{\omega}_N^{kj} = \sum_{j=0}^{N-1} y_j e^{-2\pi i kj/N},\ k = 0 \dots N-1. \qquad (2.3)$$

Umgekehrt gilt

$$y_j = p(x_j) = \frac{1}{N}\sum_{k=0}^{N-1} \widehat{y}_k x_j^k = \frac{1}{N}\sum_{k=0}^{N-1} \widehat{y}_k \omega_N^{kj} = \frac{1}{N}\sum_{k=0}^{N-1} \widehat{y}_k e^{2\pi i kj/N},\ j = 0 \dots N-1. \tag{2.4}$$

Definition 2.30 (Diskrete Fouriertransformation)
Es sei $y_k \in \mathbb{C}$, $k = 0 \dots N-1$, *und*

$$\widehat{y}_k := \sum_{j=0}^{N-1} y_j e^{-2\pi i kj/N},\ k = 0 \dots N-1.$$

Dann heißt $(\widehat{y}_0, \dots, \widehat{y}_{N-1})$ *diskrete Fouriertransformation von* $(y_0, \dots, y_{N-1})$.

Es sei

$$\widetilde{y}_k := \frac{1}{N}\sum_{j=0}^{N-1} y_j e^{2\pi i kj/N},\ k = 0 \dots N-1.$$

Dann heißt $(\widetilde{y}_0, \dots, \widetilde{y}_{N-1})$ *inverse diskrete Fouriertransformation von* $(y_0, \dots, y_{N-1})$. *Üblicherweise wird die Fouriertransformation eines Vektors* y *mit* $\widehat{y}$ *bezeichnet, die inverse Fouriertransformation mit* $\widetilde{y}$. *Damit gilt*

$$\widetilde{\widehat{y}} = \widehat{\widetilde{y}} = y.$$

Bemerkung: In der Literatur finden sich viele Varianten: In der Ingenieurliteratur sind häufig die Definition von $\widehat{y}$ und $\widetilde{y}$ vertauscht. Den Faktor N kann man natürlich auch der jeweils anderen Transformation zuschlagen, oder als $1/\sqrt{N}$ auf beide Transformationen aufteilen. Im letzten Fall wird W unitär.

Beispiel 2.31
$N = 2$, $\omega_2 = -1$, $\overline{\omega}_2 = -1$:

$$\begin{aligned} \widehat{y}_0 &= y_0 + y_1 \\ \widehat{y}_1 &= y_0 - y_1 \end{aligned}$$

$N = 4$, $\omega_4 = i$, $\overline{\omega}_4 = -i$:

$$\begin{array}{llllll} \widehat{y}_0 & = y_0 & +y_1 & +y_2 & +y_3 \\ \widehat{y}_3 & = y_0 & -iy_1 & -y_2 & +iy_3 \\ \widehat{y}_2 & = y_0 & -y_1 & +y_2 & -y_3 \\ \widehat{y}_1 & = y_0 & +iy_1 & -y_2 & -iy_3 \end{array}$$

Bemerkung: Die naive Berechnung der Fouriertransformierten anhand der Formel benötigt N^2 Additionen und Multiplikationen.

Die diskrete Fouriertransformation wird genutzt zur Interpolation von periodischen Funktionen. Sei etwa

$$f : \mathbb{R} \mapsto \mathbb{C},\ f \in C^{(N)}(\mathbb{R}),$$

periodisch mit der Periode 2π. Seien

$$x_k = 2\pi k/N,\ k = 0 \ldots N-1,$$

gleichverteilte Stützstellen im Intervall $[0, 2\pi]$ (wegen $f(x_0) = f(0) = f(2\pi) = f(x_N)$ bringt die $N.$ Stützstelle keine zusätzliche Information). Als Ansatzfunktionen für die allgemeine Interpolationsaufgabe wählen wir die periodischen Funktionen $f_j(x) = e^{ijx}$. Dann liefert die Fouriertransformation die zugehörigen Entwicklungskoeffizienten der Interpolationsfunktion, bis auf den Faktor $1/N$.

Für praktische Rechnungen ist es ungünstig, dass sogar für reelle y_k die Fouriertransformierte komplexe Werte annimmt. In der Praxis rechnet man daher eher mit der Cosinus– oder Sinus–Transformation, die eng mit der Fouriertransformation zusammenhängen.

Sei also y reell und N z.B. gerade. Dann gilt

$$\begin{aligned}
\widehat{y}_k &= \sum_{j=0}^{N-1} y_j e^{-2\pi ijk/N} \\
&= \sum_{j=0}^{N-1} (y_j \cos(2\pi kj/N) - iy_j \sin(2\pi kj/N)) \\
&= y_0 + (-1)^k y_{N/2} + \sum_{j=1}^{N/2-1} (y_j \cos(2\pi kj/N) + y_{N-j} \cos(2\pi k(N-j)/N)) \\
&\quad - i \sum_{j=1}^{N/2-1} (y_j \sin(2\pi kj/N) - y_{N-j} \sin(2\pi k(N-j)/N)) \\
&= y_0 + (-1)^k y_{N/2} + \sum_{j=1}^{N/2-1} (y_j + y_{N-j}) \cos(2\pi kj/N) \\
&\quad - i \sum_{j=1}^{N/2-1} (y_j - y_{N-j}) \sin(2\pi kj/N)
\end{aligned}$$

(denn $\cos(x_k) = \cos(x_{N-k})$ und $\sin(x_k) = -\sin(x_{N-k})$).

Ist y gerade, also $y_j = y_{N-j}$, so ist in diesem Fall $\widehat{y}_k$ reell. Damit ist

$$\gamma_k := \widehat{y}_k/2 = \frac{y_0 + (-1)^k y_{N/2}}{2} + \sum_{j=1}^{N/2-1} \cos(2\pi kj/N) y_j,\ k = 0 \dots N/2.$$

$(\gamma_0, \dots, \gamma_{N/2})$ heißt dann **Cosinustransformation** von $(y_0, \dots, y_{N/2})$.

Ist y ungerade, also $y_j = -y_{N-j}$ so ist in diesem Fall $\widehat{y}_k$ rein imaginär, Damit ist

$$\sigma_k := \frac{i\widehat{y}_k}{2} = \sum_{j=1}^{N/2-1} \sin(2\pi kj/N) y_j,\ k = 1 \dots N/2 - 1.$$

$(\sigma_1, \dots, \sigma_{N/2-1})$ heißt dann **Sinustransformation** von $(y_1, \dots, y_{N/2-1})$.

Die genauen Definitionen der Cosinustransformation und der Sinustransformation sind noch stärker vom Anwendungsfall abhängig als die Fouriertransformierte, siehe etwa die Definition in Wikipedia mit der DCT I-IV. Die hier vorgestellte entspricht der DCT I.

Vorlesungsnotiz: 18. April 2013

2.8.2 Höherdimensionale Fouriertransformation

Häufig wird die diskrete Fouriertransformation auf Bilder angewandt, d.h. y ist eine (n_1, n_2)–Matrix. Hierzu wird die Multiplikation im Exponenten von 2.3 als Skalarprodukt interpretiert.

Definition 2.32 (Zweidimensionale Fouriertransformation)
Sei $k = (k_1, k_2)$, $j = (j_1, j_2)$, $0 \le k_1, j_1 \le N_1 - 1$, $0 \le k_2, j_2 \le N_2 - 1$, *und* $y \in \mathbb{C}^{N_1 \times N_2}$. *Dann ist die Fouriertransformation* $\widehat{y}$ *von* y *definiert durch*

$$\begin{aligned}\widehat{y}_k &= \sum_{j_1=0}^{N_1-1}\sum_{j_2=0}^{N_2-1} y_{j_1,j_2} e^{-2\pi i(k_1 j_1/N_1 + k_2 j_2/N_2)} \\ &= \sum_{j_1=0}^{N_1-1} e^{-2\pi i k_1 j_1/N_1} \sum_{j_2=0}^{N_2-1} y_{j_1,j_2} e^{-2\pi i k_2 j_2/N_2}.\end{aligned}$$

Die zweidimensionale Fouriertransformation entspricht einer Fouriertransformation auf den Zeilen von y, gefolgt von einer Fouriertransformation auf den Spalten. Die Formeln für die inverse Fouriertransformation, $\cos$– und $\sin$–Transformationen gelten entsprechend, ebenso wird die FT für höhere Dimensionen erweitert.

2.8.3 FT in der Bild- und Signalverarbeitung: Faltungssatz

Einer der wichtigsten Sätze über die Fouriertransformation ist der Faltungssatz.

Definition 2.33 (Faltung von Vektoren)
Seien $y \in \mathbb{C}^N$, $z \in \mathbb{C}^M$. Dann ist die **diskrete Faltung** $(y\widehat{*}z) \in \mathbb{C}^{N+M-1}$ *von y und z definiert durch*

$$(y\widehat{*}z)_k = \sum_{j=0}^{N-1} y_j z_{k-j} = \sum_{l=k-N+1}^{k} y_{k-l} z_l, k = 0 \ldots N+M-2$$

wobei $z_p := 0$ für $p < 0$.

Seien $y, z \in \mathbb{C}^N$. Dann ist die **symmetrische diskrete Faltung** $(y * z) \in \mathbb{C}^N$ *von y und z definiert durch*

$$(y * z)_k = \sum_{j=0}^{N-1} y_j z_{k-j} = \sum_{l=0}^{N-1} y_l z_{k-l},\ k = 0 \ldots N-1$$

wobei für z der Index modulo N genommen wird, also $z_{-1} = z_{N-1}$ usw.

Die Wirkung der Faltung auf einen Vektor machen wir uns an einem kleinen Beispiel klar. Sei $y \in \mathbb{R}^N$, $z \in \mathbb{R}^2 = (1, -1)$. Dann gilt

$$(y\widehat{*}z)_k = y_k z_0 + y_{k-1} z_1 = y_k - y_{k-1},\ k = 0 \ldots N.$$

Dabei ist $y_{-1} = 0$, $y_N = 0$ für die echte Faltung und $y_{-1} = y_{n-1}$, $y_N = y_0$ für die symmetrische Faltung. Die Faltung bildet also bei festem Faltungsvektor z für jeden Wert im Ergebnis immer dieselbe Linearkombination von Werten im Signalvektor y, nur der Punkt, an dem diese Linearkombination genommen wird, wird jeweils verschoben. Die Koeffizienten sind dabei die Einträge von z.

Für unseren einfachen Vektor z bedeutet das: Bilde die Differenz aus dem aktuellen Wert y_k und dem letzten Wert y_{k-1}.

Die Faltung kann durch Anhängen von Nullen an y und z mit Hilfe der symmetrischen Faltung berechnet werden (Übungen).

Wieder ist die höherdimensionale Faltung entsprechend definiert, indem man die Indizes als Vektoren interpretiert.

Faltungen spielen eine große Rolle in der Signal- und Bildanalyse. Meist wählt man z fest aus einem kleinen Raum (Filter) und faltet dann ein Eingangssignal y mit z. Einige Beispiele:

1. $z = (-1, 0, 1)$: Approximation der Ableitung, verstärkt Kanten im Signal.
2. $z = (1, -2, 1)$: Approximation der 2. Ableitung, verstärkt Krümmung.
3. $z = (1, 2, 1)$: Glättung, Kanten werden geschwächt.
4. $z = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$: Kantenschärfer in 2D

```
function faltung1D
%FALTUNG1D
N=128;
x=(0:N)/N*2*pi;
y=cos(x)+0.2*randn(size(x));
z=[1 2 3 2 1];
```

Listing 2.17: Eindimensionale Faltung (interpolation/faltung1D.m)

Klicken für den Quellcode von interpolation/faltung1D.m

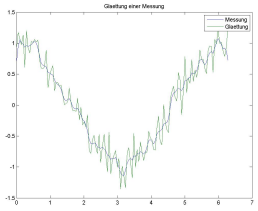


Abbildung 2.12: 1D–Faltung mit glattem Vektor, Glättung

Klick für Bild glatt1D

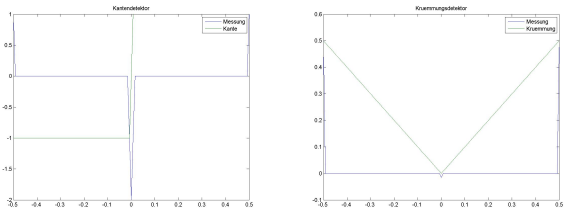


Abbildung 2.13: 1D–Faltung mit einem Kanten– bzw. Krümmungsdetektor
Klick für Bild kant1D
Klick für Bild kruem1d

```
function faltung2D
%FALTUNG2D
Y=imread('cameraman.tif');
Y=double(Y);
Z=[0 -1 0;-1 0 1; 0 1 0];
F=conv2(Y,Z);
```

Listing 2.18: Zweidimensionale Faltung (interpolation/faltung2D.m)

Klicken für den Quellcode von interpolation/faltung2D.m

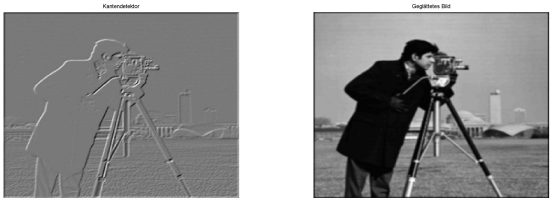


Abbildung 2.14: 2D Kantendetektor, Glättung
Klick für Bild kante2d
Klick für Bild glatt2d

Alle diese Filter sind aus Bildbearbeitungsprogrammen wie Photoshop bekannt. Dort gibt es auch die Option, Filter rückgängig zu machen. Dahinter steckt die Idee, ein unscharfes Foto nachträglich zu schärfen. Die mathematische Grundlage dafür und für die Realisierung der Faltung mit Hilfe von Fouriertransformationen gibt der Faltungssatz:

Satz 2.34 *Seien $y, z \in \mathbb{C}^N$. Dann gilt für die symmetrische diskrete Faltung von y und z*

$$\widehat{y}_p \widehat{z}_p = \widehat{(y * z)}_p.$$

Beweis: Sei $\omega_N = e^{-2\pi i/N}$ (beachte das Vorzeichen im Exponenten!).

$$\begin{aligned}
\widehat{y}_p \widehat{z}_p &= \left(\sum_{j=0}^{N-1} y_j \omega_N^{pj} \right) \left(\sum_{l=0}^{N-1} z_l \omega_N^{pl} \right) \\
&= \sum_{j=0}^{N-1} \sum_{l=0}^{N-1} y_j z_l \omega_N^{p(j+l)} \quad k = j + l \\
&= \sum_{j=0}^{N-1} \sum_{k=j}^{j+N-1} y_j z_{k-j} \omega_N^{pk} \\
&= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} y_j z_{k-j} \omega_N^{pk} \qquad \text{(Summand ist } N\text{-periodisch in } k\text{)} \\
&= \sum_{k=0}^{N-1} \omega_N^{pk} \sum_{j=0}^{N-1} y_j z_{k-j} \\
&= \sum_{k=0}^{N-1} \omega_N^{pk} (y * z)_k \\
&= \widehat{(y * z)}_p.
\end{aligned}$$

□

Diser Satz ist sehr wichtig in der Bildverarbeitung. Falls ein unscharfes Bild $y * z$ und die Unschärfefunktion z bekannt sind, so können wir deren Fouriertransformierte berechnen, und es gilt

$$\widehat{y}_p = \widehat{(y * z)}_p / \widehat{z}_p.$$

Hieraus lässt sich das scharfe Bild y durch inverse Fouriertransformation berechnen. Dies ist gültig, solange die Fourierkoeffizienten von z nicht verschwinden.

Theoretisch lassen sich also mit dieser Formel unscharfe Bilder scharfrechnen, wenn man die Filterfunktion z kennt (unscharf maskieren, s. Übungen). Praktisch funktioniert das nur sehr eingeschränkt, da die Fourierkoeffizienten von z sehr schnell gegen 0 gehen und man durch sehr kleine Zahlen teilen muss.

2.8.4 FT in der Signalkompression

Aus der Analysis 1 (oder dem Forster) ist bekannt, dass sich eine stetige 2π-periodische Funktion f auf $\mathbb{R}$ als Linearkombination der trigonometrischen Funktionen schreiben lässt, also

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

mit den Fourierkoeffizienten

$$\begin{aligned} c_k &= \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} \mathsf{dx},\ k = -\infty \dots \infty, \\ a_k &= \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) \mathsf{dx},\ k = 0 \dots \infty, \\ b_k &= \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(kx) \mathsf{dx},\ k = 1 \dots \infty \end{aligned}$$

(Fourierreihe). Die Fourierreihe stellt also periodische Signale als Linearkombination von periodischen Funktionen unterschiedlicher Frequenz dar.

Sei nun f ein aufgenommener, periodischer Ton. Dann sind typischerweise nur wenige c_k im Betrag groß. In umfangreichen Studien (zuerst wohl in Mayer [1896], Originalveröffentlichung 1894 (!Doubtful!)) wurden Aussagen etabliert wie: Falls für eine Frequenz k $|c_k|$ groß ist, die Frequenz k also mit großem Anteil im Signal vorkommt, und die Frequenzen k' in der Nähe von k mit $|k' - k| < d$ mit sehr kleinem Anteil, so sind die Anteile in k' nicht hörbar.

Konsequenterweise müssen diese Frequenzen k' bei der Speicherung oder Übertragung von Tönen auch nicht berücksichtigt werden.

Natürlich ist in diesen Fällen nicht die Funktion selbst bekannt, es stehen nur äquidistante Messwerte zu Zeitpunkten x_j zur Verfügung. Approximieren wir das Integral in der Formel aber etwa durch die Riemannsumme

$$c_k \sim \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-ikx_j},$$

so erhalten wir (bis auf Konstante) die Formel für die diskrete Fouriertransformation. Falls alle Werte reell sind, kann man alternativ auch die diskrete Cosinustransformation nutzen.

Beispiel 2.35 *Sei $f(x) = |x|$ im Intervall $[-\pi, \pi]$, periodisch fortgesetzt auf $\mathbb{R}$. Die Abbildung zeigt die Approximation an f für verschiedene Interpolationsgrade.*

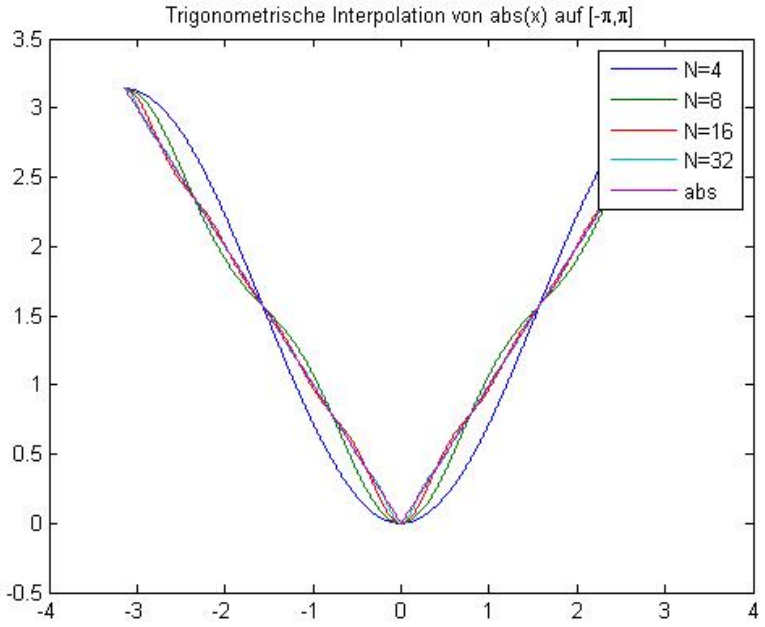


Abbildung 2.15: Trigonometrische Interpolation

Klick für Bild simplefour

```
function simplefour
%SIMPLEFOUR

    function y1=compute(N)
        x=(0:N-1)/N*2*pi;
        y=f(x-pi);
```

Listing 2.19: Trigonometrische Interpolation (interpolation/simplefour.m)

Klicken für den Quellcode von interpolation/simplefour.m

Beispiel 2.36 *Das gleiche Beispiel mit der Cosinus–Transformation. Das Ergebnis ist natürlich dasselbe, nur die Programme unterscheiden sich.*

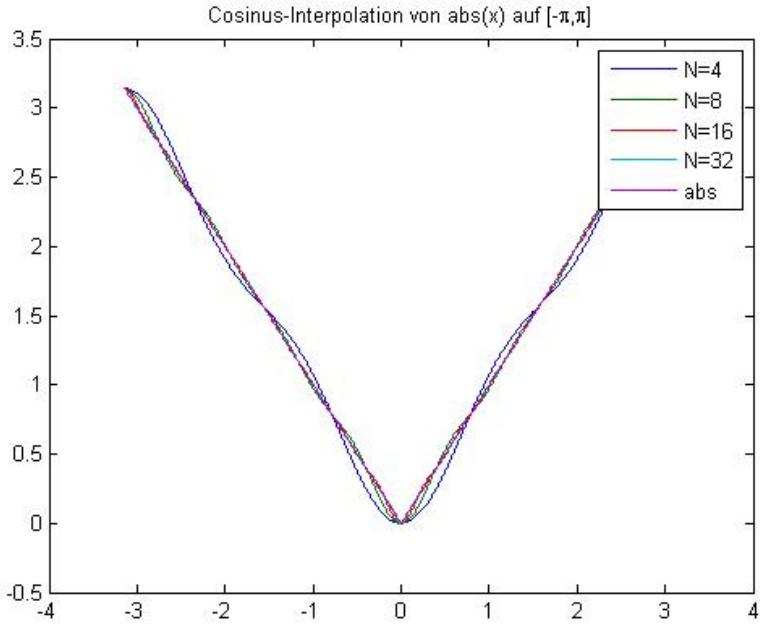


Abbildung 2.16: Interpolation mit der Cosinus-Transformation

Klick für Bild simplecos

```
function simplecostrans
%SIMPLECOSTRANS

    function y1=compute(N)
        x=(0:N-1)/N*pi;
        y=f(x-pi);
```

Listing 2.20: Cosinus-Transformation (interpolation/simplecostrans.m)

Klicken für den Quellcode von interpolation/simplecostrans.m

Technisch passiert nun das folgende:

1. Ein Signal wird aufgenommen.
2. Das Signal wird in seine Frequenz–Bestandteile zerlegt mit der diskreten Cosinus–Transformation.
3. Unhörbare Bestandteile werden identifiziert und $= 0$ gesetzt.
4. Die restlichen Koeffizienten der Cosinus-Transformation werden übermittelt.

5. Das Signal wird mit der inversen Cosinus–Transformation wieder zusammengesetzt.

Der letzte Schritt passiert dabei im MP3–Player auf sehr bescheidener Hardware, das muss also sehr einfach durchzuführen sein. Die Formeln in 2.3 und 2.4 sind dazu nicht geeignet, der Aufwand zur Transformation von N Werten wäre N^2 Additionen und Multiplikationen. Wir werden deutlich einfachere Formeln herleiten, die mit $O(N \log N)$ Rechenoperationen auskommen.

Der Ehrlichkeit halber sei gesagt, dass der tatsächliche Prozess im Größenordnungen komplizierter ist. Tatsächlich bildet aber die DCT die Basis dieser Komprimierungen.

Diese Idee lässt sich auch für Bilder durchführen. Mit Hilfe der zweidimensionalen Fouriertransformation schreiben wir unsere Funktion als Linearkombination von trigonometrischen Funktionen, wieder ist die Idee, dass der Eindruck des Bildes durch relativ wenige Entwicklungskoeffizienten beschrieben wird. Dies ist die Grundlage der JPEG– und MPEG–Kompression (wieder mit der Cosinustransformation an Stelle der Fouriertransformation).

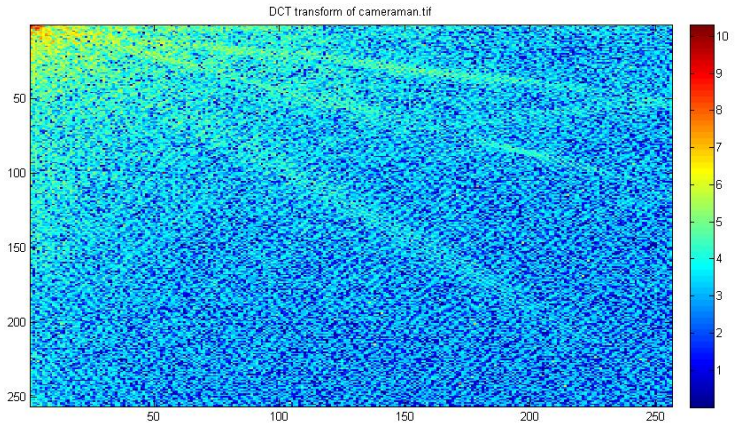


Abbildung 2.17: DCT des Kameramann-Bildes (abs val of log)

Klick für Bild cameramandct

Im Beispielbild wurde die zweidimensionale Cosinus–Transformation (DCT entlang Zeilen, dann Spalten) auf dem Kameramann–Bild ausgeführt. Der Betrag des log des Betrags der DCT wurde geplottet. Deutlich sichtbar ist, dass die relevanten Koeffizienten in der linken oberen Ecke stehen (niederfrequente Anteile).

2.8.5 FFT: Schnelle Fourier–Transformation nach Cooley und Tukey

Wir beginnen mit zwei kurzen Beispielen. Sei wieder $N = 2$, also $\omega_N = -1$. Damit ergibt sich für die $\widehat{y}_k$:

$$\begin{array}{rcrcl} \widehat{y}_0 & = & y_0 & + & y_1 \\ \widehat{y}_1 & = & y_0 & - & y_1. \end{array}$$

Für $N = 4$ ist $e^{-2\pi i/4} = -i$, also

$$\begin{array}{rcrcrcrcrcrcrcr} \widehat{y}_0 & = & y_0 & + & & y_1 & + & & y_2 & + & & y_3 & = & (y_0+y_2) & + & & (y_1+y_3) \\ \widehat{y}_1 & = & y_0 & + & (-i) & y_1 & + & (-1) & y_2 & + & (i) & y_3 & = & (y_0-y_2) & + & (-i) & (y_1-y_3) \\ \widehat{y}_2 & = & y_0 & + & (-1) & y_1 & + & & y_2 & + & (-1) & y_3 & = & (y_0+y_2) & - & & (y_1+y_3) \\ \widehat{y}_3 & = & y_0 & + & i & y_1 & + & (-1) & y_2 & + & (-i) & y_3 & = & (y_0-y_2) & - & (-i) & (y_1-y_3) \end{array}$$

Durch naives Ausrechnen der Formel in 2.3 für $N = 4$ bräuchten wir 12 (komplexe) Additionen. Offensichtlich lässt sich diese Anzahl verringern, indem man zunächst jeweils eine Fouriertransformation halber Länge auf den Fourierkoeffizienten mit geradem und ungeradem Index durchführt, und auf den Ergebnissen wieder zwei Fouriertransformationen der Länge zwei ausführt.

Dies ist die Grundidee der schnellen Fouriertransformation (FFT) von Cooley und Tukey (1965), einer der Top 10-Algorithms of the century.

Satz 2.37 (FFT nach Cooley und Tukey)
Sei $N = pq$. Dann kann die Fouriertransformation eines Vektors $(y_0, \ldots, y_{N-1})$ durch p Fouriertransformationen der Länge q, q Fouriertransformationen der Länge p und N Multiplikationen berechnet werden.

Beweis: Sei wieder

$$\omega_N = e^{-2\pi i/N}.$$

Wir bemerken zunächst, dass

$$\omega_N^p = \omega_q,\ \omega_N^q = \omega_p.$$

Sei $j \in \{0, \ldots, N-1\}$. Dann können wir j eindeutig schreiben als

$$j = ps + r,\ s \in \{0, \ldots, q-1\},\ r \in \{0, \ldots, p-1\}.$$

Hierbei ist $r = j \mod p$ und $s = (j - r)/p$. Die Abbildung $j \mapsto (r, s)$ ist bijektiv auf den angegebenen Mengen.
Sei nun $k \in \{0, \ldots, N-1\}$. Dann können wir k entsprechend schreiben als

$$k = k_1 q + k_2,\ k_1 = 0 \ldots p-1,\ k_2 = 0 \ldots q-1$$

und es gilt

$$\begin{aligned}
\widehat{y}_k &= \sum_{j=0}^{N-1} \omega_N^{kj} y_j \\
&= \sum_{r=0}^{p-1} \sum_{s=0}^{q-1} \omega_N^{k(ps+r)} y_{ps+r} \\
&= \sum_{r=0}^{p-1} \omega_N^{kr} \sum_{s=0}^{q-1} \omega_q^{ks} y_{ps+r} \\
&= \sum_{r=0}^{p-1} \omega_p^{k_1 r} \underbrace{\omega_N^{k_2 r} \underbrace{\sum_{s=0}^{q-1} \omega_q^{k_2 s} \underbrace{y_{ps+r}}_{=:C_{r,s}}}_{=:A_{r,k_2}}}_{=:B_{r,k_2}}.
\end{aligned}$$

Diese Summe berechnen wir nach der folgenden Vorschrift:

1. Setze
$$C_{r,s} = y_{ps+r},\ r = 0 \ldots p-1,\ s = 0 \ldots q-1.$$

2. Für jedes r von $0 \ldots p-1$ und k_2 von $0 \ldots q-1$ berechne
$$A_{r,k_2} = \sum_{s=0}^{q-1} \omega_q^{k_2 s} y_{ps+r} = \sum_{s=0}^{q-1} \omega_q^{k_2 s} C_{r,s}.$$
Für festes r ist das jeweils eine FT der Länge q. Insgesamt sind das p Fouriertransformationen der Länge q, ausgeführt auf den Vektoren C_r.

3. Für jedes r von $0 \ldots p-1$ und jedes k_2 von $0 \ldots q-1$ berechne
$$B_{r,k_2} = \omega_N^{k_2 r} A_{r,k_2}.$$
Das sind N Multiplikationen.

4. Für jedes k_1 von $0 \dots p-1$ und k_2 von $0 \dots q-1$ berechne

$$\widehat{y}_{k_1 q + k_2} = \sum_{r=0}^{p-1} \omega_p^{k_1 r} B_{r,k_2}.$$

Für jedes feste k_2 ist das eine FT der Länge p. Insgesamt sind das q Fouriertransformationen der Länge p, ausgeführt auf den Vektoren $B_{\cdot,k_2}$.

□

Beispiel 2.38 (Fouriertransformation von gerader Länge)
Wir betrachten den Fall $q = 2$, also $N = p \cdot 2$. Dann sind $s, k_2 \in \{0, 1\}$. Wir erhalten

$$C_{r,0} = y_r,\ C_{r,1} = y_{r+p}.$$

Auf diesen ist nun eine Fouriertransformation durchzuführen, also

$$A_{r,0} = C_{r,0} + C_{r,1},\ A_{r,1} = C_{r,0} - C_{r,1}.$$

Weiter

$$B_{r,0} = A_{r,0},\ B_{r,1} = \omega_N^r A_{r,1}.$$

Auf den zwei Vektoren $B_{r,0}$ und $B_{r,1}$, $r = 0 \dots p-1$, führen wir nun jeweils eine Fouriertransformation der Länge p durch und erhalten $\widehat{y}_{2r}$ bzw. $\widehat{y}_{2r+1}$.

Falls p wieder gerade ist, so können wir die Formel rekursiv anwenden.

Nun betrachten wir den Fall $p = 2$, also $N = 2q$. Dann sind $r, k_1 \in \{0, 1\}$. Wir erhalten

$$C_{0,s} = y_{2s},\ C_{1,s} = y_{2s+1}.$$

C_0 enthält also die Elemente von y mit geradem Index, C_1 die mit ungeradem Index. Auf diesen führen wir nun eine Fouriertransformation der Länge q durch und erhalten A_{0,k_2} bzw. A_{1,k_2}. Weiter

$$B_{0,k_2} = A_{0,k_2},\ B_{1,k_2} = \omega_N^{k_2} A_{1,k_2}.$$

Unser Ergebnis erhalten wir nun durch

$$y_{k_2} = B_{0,k_2} + B_{1,k_2},\ y_{k_2+q} = B_{0,k_2} - B_{1,k_2}.$$

Wir erhalten zwei grundsätzlich unterschiedliche Implementierungen. Wieder können wir natürlich, falls q gerade ist, die Formel rekursiv nutzen.

Damit können Fouriertransformationen schnell berechnet werden, falls N ein Produkt kleiner Primzahlen ist, weil dann dieser Satz möglichst oft rekursiv genutzt werden kann. Beispielsweise gilt für $N = 2^p$:

Satz 2.39 Aufwand der diskreten Fouriertransformation für Zweierpotenzen
Sei M_p die Anzahl der Multiplikationen, A_p die Anzahl der Additionen für eine Fouriertransformation der Länge $N = 2^p$. Dann gilt

$$M_p \leq p2^p = N \log_2 N,\ A_p \leq p2^p = N \log_2 N.$$

Beweis: Zur Berechnung einer Fouriertransformation der Länge 2^{p+1} werden nach Satz 2.37 höchstens 2 Fouriertransformationen der Länge 2^p, 2^p Fouriertransformationen der Länge 2 und 2^{p+1} Multiplikationen benötigt.
Es gilt $M_1 = 0$, $A_1 = 2$.

$$\begin{aligned} M_{p+1} &\leq 2M_p + 2^{p+1} \\ &\leq 2p2^p + 2^{p+1} \\ &= 2^{p+1}(p+1). \end{aligned}$$

$$\begin{aligned} A_{p+1} &\leq 2A_p + 2^p A_1 \\ &\leq p2^{p+1} + 2^{p+1} \\ &= (p+1)2^{p+1} \end{aligned}$$

□

Machen wir uns klar, was dieser Satz bedeutet: Mit naivem Ausrechnen würde eine Fouriertransformation der typischen Länge $N = 1024 = 2^{10}$ ca. $N^2 \sim 10^6$ Additionen und Multiplikationen benötigen. Durch Nutzung der Cooley–Tukey–FFT benötigen wir nur noch $N \log_2 N = 10 \cdot 1024 \sim 10^4$ Additionen und Multiplikationen, wir haben also den benötigten Aufwand um den Faktor 100 reduziert. Dies ist tatsächlich auch für eingebettete Systeme mit sehr schwacher Rechenleistung berechenbar.

Damit ist die Theorie der schnellen FT aber noch nicht beendet. Völlig unklar etwa ist, ob es auch schnelle Algorithmen gibt, wenn N eine Primzahl ist und wir den Algorithmus nicht anwenden können.

Tatsächlich gibt es für alle Längen spezielle Algorithmen, die eine schnelle Ausführung ermöglichen (siehe etwa Beth [1984] oder die Primzahl–FFT in Winograd [1978]). Die schnelle Fouriertransformation ist beispielsweise in der quelloffenen Bibliothek FFTW, Fastest Fourier Transform in the West, implementiert, sie gilt

allgemein als die effizienteste Implementation. Einer der dort angewandten Tricks ist, dass die Cooley–Tukey–FFT mit unterschiedlichen Zerlegungen angewandt und auf dem jeweiligen Prozessor getestet wird, welche die schnellste ist.

Eine schöne Übersicht über FFT–Algorithmen findet sich im aktuellen Buch Burrus. Die Ideen der Fouriertransformation können auf allgemeinere orthogonale (Wavelet–) Transformationen erweitert werden, siehe hierzu die klassischen Bücher Daubechies et al. [1992] oder Louis et al. [1998]. Von großer Bedeutung für die praktische Arbeit ist die Nicht–äquidistante schnelle Fouriertransformation NFFT, die sich ebenfalls aus der schnellen Fouriertransformation ableiten lässt. Eine Übersicht über mehrere Verfahren zu effizienten Algorithmen, auch zu schnellen Matrix–Multiplikationen und insbesondere den engen Beziehungen zur Algebra, finden sich im lesenswerten (alten) Vorlesungsskript Natterer [1994].

```
function yhat = myfft( y )
%MYFFT
%This is not really fast - it shows how the FFT
%computes its way. Do not use for real computations!
%This is he Cooley-Tukey algorithm.
format compact;
```

Listing 2.21: Einfache Implementierung der FFT (interpolation/myfft.m)

Klicken für den Quellcode von interpolation/myfft.m

2.9 Spline–Interpolation

Die kurze Behandlung innerhalb dieses Abschnitts wird der tatsächlichen Bedeutung der Spline–Interpolation nicht gerecht, tatsächlich ist sie das Standardwerkzeug zur Interpolation und in jedem Werkzeugkasten zur Numerik implementiert. Für eine umfassende Behandlung verweisen wir auf de Boor [2001]. Eine umfassende Würdigung von de Boor und Splines findet sich in DeVore and Ron [2005].

2.9.1 Eindimensionale Splines und B–Splines

Im Kapitel über Interpolationsfehler haben wir gesehen, dass es von Vorteil ist, das zugrundeliegende Intervall für die Interpolation aufzuteilen und die Interpolation mit niedrigem Polynomgrad auf jedem Teilintervall einzeln durchzuführen. Leider geht dabei die Stetigkeit/Differenzierbarkeit an den Intervallgrenzen verloren.

Splines beheben diesen Mangel: Sie teilen zunächst das Intervall $[a, b]$ an Knotenpunkten s_i auf. Auf jedem Einzelintervall $[s_i, s_{i+1}]$ sind die Splines (der Ordnung k)

Polynome p_i vom Grad $k-1$, mit der zusätzlichen Forderung, dass an den Knoten die zusammengesetzte Funktion $(k-2)$–mal differenzierbar ist, die Polynome von links und rechts also bis zur $(k-2)$–ten Ableitung übereinstimmen.

Beispiel 2.40
Gegeben seien die Stützstellen $0,1,2,3$ *und die Stützwerte* $0,0,0,1$.
Das interpolierende Polynom ist mit Lagrange

$$p(x) = \frac{x(x-1)(x-2)}{6}.$$

Der lineare interpolierende Polygonzug ist nicht differenzierbar.
Wir wählen nun die Knotenpunkte $s_k = x_k$. *Dann gilt: Die Funktion*

$$s(x) = \max(x-2,0)^2$$

interpoliert die Stützwerte und ist stetig differenzierbar auf dem gesamten Intervall $[0,3]$, *in jedem Teilintervall* $[s_i, s_{i+1}]$ *liegt sie in* $\mathcal{P}_2$, *sie ist also ein Spline der Ordnung* 3.

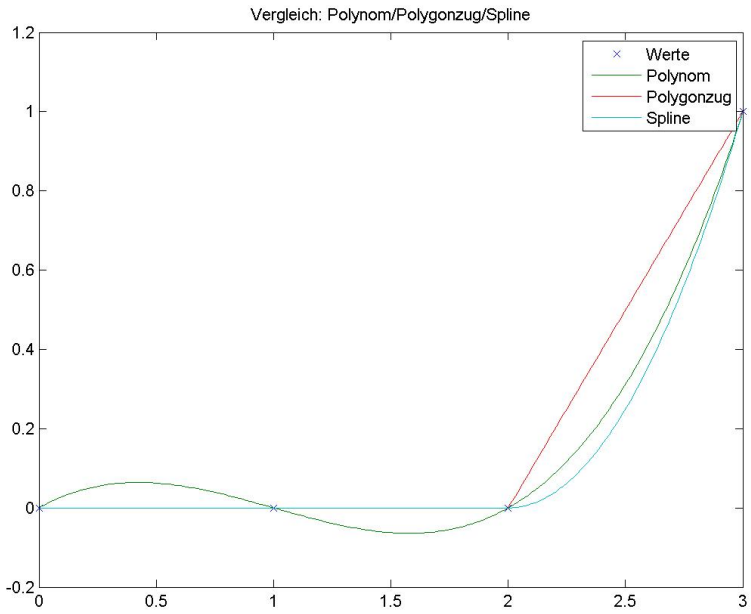


Abbildung 2.18: Vergleich Polynom/Polygon/Spline

Klick für Bild splinetest
Klick für Matlab Figure splinetest

Definition 2.41 (Splines)
Seien $s_0 < s_1 < \ldots < s_n$ reelle Zahlen. Eine Funktion

$$s : [s_0, s_n] \mapsto \mathbb{R}$$

heißt Spline der Ordnung k (zu den Knoten $s_0 \ldots s_n$), falls

1. $s \in C^{(k-2)}([s_0, s_n])$ *für* $k > 1$.
2. $s|_{[s_i, s_{i+1})} \in \mathcal{P}_{k-1}$, $i = 0 \ldots n-1$.

Üblicherweise wird der Spline über sein eigentliches Definitionsgebiet hinaus fortgesetzt, z.B. linear oder periodisch.

Beispiel 2.42

1. *Sei $p \in \mathcal{P}_{k-1}$. Dann ist p Spline der Ordnung k.*
2. *Sei s_i ein Knotenpunkt, k fest. Sei*

   $$s_i^+(x) = (\max(x - s_i, 0))^{k-1}$$

   für $k > 1$ und für $k = 1$

   $$s_i^+(x) = \begin{cases} 0 & x < s_i \\ 1 & x \geq s_i. \end{cases}$$

   Dann gilt

   $$s_i^+ \in C^{(k-2)}([s_0, s_n]), s_i^+ \notin C^{(k-1)}([s_0, s_n]) \, (k \geq 2).$$

   $s_i^+ \in \mathcal{P}_{k-1}$ auf jedem Intervall $[s_j, s_{j+1})$, also ist s_i^+ Spline der Ordnung k.
3. *Splines der Ordnung 1 sind genau die Funktionen, die konstant sind in jedem Intervall $[s_i, s_{i+1})$. Sie sind nicht notwendig stetig.*
4. *Sei s linear in jedem Intervall $[s_i, s_{i+1}]$ und stetig auf $[s_0, s_n]$, also ein Polygonzug. Genau dann ist s Spline der Ordnung 2.*

Sei s ein Spline der Ordnung k. Wir haben n Intervalle. Auf jedem Intervall ist $s \in \mathcal{P}_{k-1}$, also gibt es insgesamt nk Polynomkoeffizienten. An den Schnittstellen s_1 bis s_{n-1} müssen die Ableitungen links und rechts bis zum Grad $(k-2)$ übereinstimmen, macht insgesamt $(k-1)(n-1)$ lineare Bedingungen, von denen man mit dem Satz über die Eindeutigkeit der Hermite–Interpolation sieht, dass sie unabhängig sind. Es gilt also für die Dimension des Vektorraums V_s der Splines zu gegebener Knotenmenge $\{s_0, \ldots, s_n\}$ vom Grad $k-1$

$$\dim V_S \leq nk - (n-1)(k-1) = n + k - 1.$$

Wir können sofort eine Basis für diesen Vektorraum angeben: Nach Vorbemerkung sind alle Polynome in $\mathcal{P}_{k-1}$ Splines, dies ist ein Vektorraum der Dimension k. Für $i = 1 \dots n-1$ sind die s_i^+ linear unabhängige Splines, aber keine Polynome. Die Monome zusammen mit s_i^+, $i = 1 \dots n-1$, bilden also eine linear unabhängige Menge mit $(n+k-1)$ Elementen und damit eine Basis des Splineraums.

Dies gibt uns bereits eine einfache Möglichkeit, das Interpolationsproblem für Splines zu lösen: Wir wählen

$$N + 1 = n + k - 1$$

Stützstellen und Stützwerte und lösen das Interpolationsproblem mit allgemeinen Ansatzfunktionen 2.27 mit den Ansatzfunktionen in dieser Basis. Wir nutzen diesen Algorithmus zur Interpolation des Runge–Beispiels.

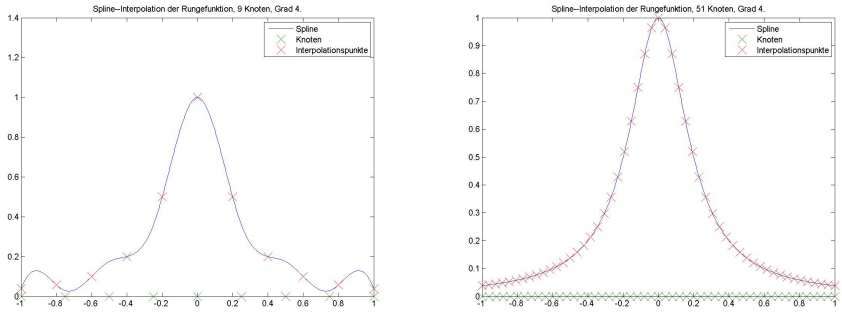


Abbildung 2.19: Interpolation des Rungebeispiels mit einfachen Ansatzfunktionen
Klick für Bild simplesplinerunge8-4
Klick für Matlab Figure simplesplinerunge8-4
Klick für Bild simplesplinerunge50-4
Klick für Matlab Figure simplesplinerunge50-4

```
function y = simplesplinebasisval( s,k,l,x )
%SIMPLESPLINEBASISVAL evaluate simple basis splines
%s – knots
%x – evaluation point
%k – degree of spline
%l – index of basis function, starting at o
```

Listing 2.22: Einfache Spline–Ansatzfunktionen (interpolation/simplesplinebasisval.m)

Klicken für den Quellcode von interpolation/simplesplinebasisval.m

```
function p = simplesplines( s,x,y,k )
%SIMPLESPLINES compute simple spline coefficients
%knots in s, evaluation points in x, values in y,
%order of spline in k (k=2 means linear)
%Returns vector of polynomial coefficients.
if (nargin==0)
```

Listing 2.23: Berechnung von Spline–Koeffizienten mit allgemeinen Ansatzfunktionen (interpolation/simplesplines.m)

Klicken für den Quellcode von interpolation/simplesplines.m

```
function [ output_args ] = simplesplinedemo( input_args )
%SIMPLESPLINEDEMO Summary of this function goes here
%   Detailed explanation goes here

    n=50;
    k=4;
```

Listing 2.24: Demo zu einfachen Spline–Ansatzfunktionen (interpolation/simplesplinedemo.m)

Klicken für den Quellcode von interpolation/simplesplinedemo.m

Dies ist leider doppelt ineffizient: Typischerweise ist haben wir viele Knoten, d.h. n ist groß, k klein. Mit diesen Ansatzfunktionen wäre die zugehörige Vandermonde–ähnliche Matrix 2.2 fast voll besetzt, d.h. das zugehörige Gleichungssystem zur Bestimmung der Koeffizienten wäre schwierig zu lösen. Zusätzlich verschwindet z.B. auf dem Intervall $[s_{n-1}, s_n]$ keine einzige Ansatzfunktion. Zur Auswertung des Splines müssten wir also alle Ansatzfunktionen auswerten und aufaddieren, der Aufwand dazu wäre mindestens $O(n)$, obwohl der Spline in diesem Intervall nur den Polynomgrad $k << n$ hat.

Wir werden eine günstigere Basis bestimmen, bei der die Ansatzfunktionen nur einen kleinen Träger haben, so dass die Matrix 2.2 nur wenige von Null verschiedene Zahlen enthält, und zusätzlich die resultierende Splinefunktion einfach auswertbar ist.

Definition 2.43 (B-Splines)
Seien $s_0 < s_1 < \ldots < s_n$ reelle Zahlen. Dann heißen

$$B_{i,1}(x) = \begin{cases} 1 & s_i \leq x < s_{i+1} \\ 0 & sonst \end{cases}$$

$$B_{i,k}(x) = \frac{x - s_i}{s_{i+k-1} - s_i} B_{i,k-1}(x) + \frac{s_{i+k} - x}{s_{i+k} - s_{i+1}} B_{i+1,k-1}(x)$$

B–Splines der Ordnung k.

Beispiel 2.44

1. *B-Splines der Ordnung $k = 1$ sind stückweise konstant, also Splines der Ordnung 1.*
2. *Sei $k = 2$. Für $x < s_i$ und $x > s_{i+2}$ ist*

   $$B_{i,1}(x) = B_{i+1,1}(x) = 0,$$

   daher ist auch $B_{i,2}(x) = 0$. Weiter gilt

   $$B_{i,2}(x) = \begin{cases} \frac{x - s_i}{s_{i+1} - s_i}, & s_i \leq x < s_{i+1} \\ \frac{s_{i+2} - x}{s_{i+2} - s_{i+1}}, & s_{i+1} \leq x < s_{i+2}. \end{cases}$$

   s ist also stetig und linear in jedem Intervall $[s_i, s_{i+1}]$. Also ist $B_{i,2}$ der lineare Polygonzug mit

   $$B_{i,2}(s_j) = \delta_{i+1,j}.$$

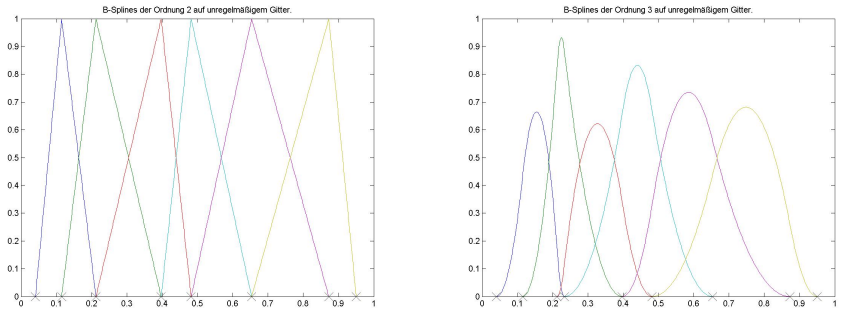


Abbildung 2.20: B–Splines der Ordnungen 2 und 3 auf unregelmäßigen Gittern

Klick für Bild bspline-8-2
Klick für Matlab Figure bspline-8-2
Klick für Bild bspline-9-3
Klick für Matlab Figure bspline-9-3

Bemerkung: Diese Definition kann auch für mehrfache Knoten ($s_i = s_{i+1}$) erweitert werden, in diesem Fall ist B definiert als der Grenzwert für $s_{i+1} \mapsto s_i$.

Das Beispiel lässt sich fortsetzen, es gilt der Satz:

Satz 2.45 *(Eigenschaften der B–Splines)*

1. $B_{i,k}$ *ist Spline der Ordnung* k.
2. *Der Träger von* $B_{i,k}$ *ist* (s_i, s_{i+k}) *(*$[s_i, s_{i+k})$ *für* $k = 1$*).*
3. *Falls links von* s_0 *und rechts von* s_n *noch jeweils* $(k-1)$ *zusätzliche paarweise verschiedene Knoten* s_{-k+1} *bis* s_{-1} *bzw.* $s_{n+1} \dots s_{n+k-1}$ *eingefügt werden, so sind die B–Splines* $B_{j,k}$*,* $j = -k-1 \dots n-1$*, Basis des Spline–Ansatzraums.*

Beweis: Durch Darstellung der B–Splines als dividierte Differenzen, siehe z.B. de Boor [2001], S. 87ff, oder Freund and Hoppe [2007].
Zum letzten Punkt: Zusammen mit den eingefügten Knoten gibt es gerade $(n+k-1)$ B–Splines $B_{i,k}$, $i = -k+1 \dots n-1$, mit Träger in (s_0, s_n). Wegen der Träger–Bedingung sind sie linear unabhängig. □

```
function y = bspline( i,k,xo,x )
%BSPLINE
    function y=Bint(i,k,xo)
        if (k==1)
            y=zeros(size(xo));
            y((xo>x(i))&(xo<=x(i+1)))=1;
```

Listing 2.25: Berechnung von B–Splines (interpolation/bspline.m)

Klicken für den Quellcode von interpolation/bspline.m

```
function y = dbspline( i,k,xo,x,j )
%DBSPLINE
% j is order of differentiation
if (j==0)
    y=bspline(i,k,xo,x);
    return
```

Listing 2.26: Berechnung der Ableitung (interpolation/dbspline.m)

Klicken für den Quellcode von interpolation/dbspline.m

```
function bsplinedemo2
%BSPLINEDEMO2
x=0:10;
x=[0 0.5 1 2.5 3.5 4 4.5 7 8 8.2 10];
P=100;
xo=(0:P*10)/P;
```

Listing 2.27: B–Splines (interpolation/bsplinedemo2.m)

Klicken für den Quellcode von interpolation/bsplinedemo2.m

```
function y = splinecoeff( x,i,k,j )
%SPLINECOEFF Coefficients of spline B(i,k)
%in interval starting at x_j
if (k==1)
    if (i==j)
        y=1;
```

Listing 2.28: Berechnung der Koeffizienten von B–Splines (interpolation/splinecoeff.m)

Klicken für den Quellcode von interpolation/splinecoeff.m

```
function [ output_args ] = testsplinecoeff( input_args )
%TESTSPLINECOEFF
N=50;
i=1;
k=4;
p=k+1;
```

Listing 2.29: Auswertung von B–Splines anhand der Koeffizienten (interpolation/testsplinecoeff.m)

Klicken für den Quellcode von interpolation/testsplinecoeff.m

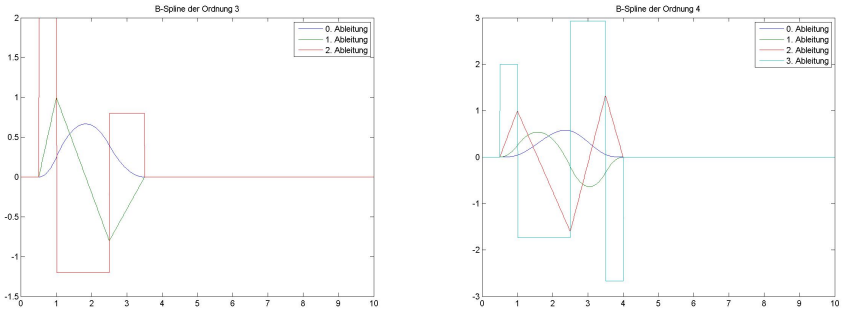


Abbildung 2.21: Splines der Ordnung 3 und 4
Klick für Bild spline3
Klick für Bild spline4

Es bleibt noch zu beweisen, dass das Interpolationsproblem eindeutig lösbar ist, d.h. ob die zugehörige Vandermonde–artige Matrix V invertierbar ist. Dies muss natürlich von der Wahl der Stützstellen abhängen: Es ist sicherlich nicht möglich, alle Stützstellen in einem einzigen Intervall $[s_i, s_{i+1}]$ zu wählen, die Stützstellen müssen sich gleichmäßig auf $[s_0, s_n]$ verteilen.

Tatsächlich gilt der Satz:

Satz 2.46 *(B–Splines sind Basis des Splineraums)*
Seien $s_0 < s_1 < \ldots < s_n$ reelle Zahlen. Gegeben seien Stützstellen x_j mit Stützwerten y_j, $j = 0 \ldots n-k$ und es gelte

$$x_j \in (s_j, s_{j+k}),\ j = 0 \ldots n-k.$$

Dann gibt es genau einen Spline s der Ordnung k,

$$s(x) = \sum_{i=0}^{n-k} a_i B_{ik}(x)$$

mit Koeffizienten $a_i \in \mathbb{R}$, so dass

$$s(x_j) = y_j,\ j = 0 \ldots n-k.$$

Die Koeffizienten lassen sich wieder mit Hilfe der Vandermonde–ähnlichen Matrix 2.2 berechnen.

Beweis: Schoenberg and Whitney [1953] mit der Definition der B–Splines über dividierte Differenzen, De Boor [1976] mit linearer Algebra, einfachster Beweis in de Boor [2001], Seite 97f. □

Die Dimension des Ansatzraums unterscheidet sich von der bereits berechneten, denn die B–Splines sind nur dann eine Basis, wenn wir links und rechts vom Interpolationsintervall noch jeweils $(k-1)$ Knoten einfügen. Tun wir das, kommen wir auf $n-k+1+2(k-1) = n+k-1$ Ansatzfunktionen, wie bereits berechnet.

Wir haben also eine alternative Basis des Spline–Ansatzraums gefunden, und können eine Interpolation mit allgemeinen Ansatzfunktionen durchführen. Dies ist jetzt deutlich effizienter als in unserem ersten Ansatz: Wählen wir die Knoten wie im Satz, so befinden sich genau k Knoten im Intervall $[s_i, s_{i+k}]$. Die k. Spalte von V enthält also höchstens k von Null verschiedene Einträge, es ist eine Bandmatrix. Das zugehörige lineare Gleichungssystem ist also leicht auflösbar.

Sei nun $x \in [s_0, s_n]$. Dann sind höchstens k Ansatzfunktionen an dieser Stelle ungleich Null, auch die Auswertung von $s(x)$ ist also einfach möglich.

Oft wählt man als Stützstellen gleich einige der Knoten, dies ist nach dem Satz jedoch nicht notwendig.

De Boor berechnet auch die Kondition der Matrizen, sie ist klein, falls die Interpolationspunkte einigermaßen gleichmäßig verteilt sind.

Vorlesungsnotiz: 25.4.2013

Wir wollen nun noch eine wichtige Minimaleigenschaft der kubischen Splines beweisen und beginnen mit

Lemma 2.47 *(Eine unendlich oft differenzierbare Funktion mit beschränktem Träger) Seien $\widetilde{x} \in \mathbb{R}$, $\epsilon > 0$. Dann ist die Funktion*

$$g_{\epsilon,\widetilde{x}}(x) = \begin{cases} \exp(\frac{1}{(x-\widetilde{x})^2-\epsilon^2}) & |x-\widetilde{x}| < \epsilon \\ 0 & \textit{sonst} \end{cases}$$

unendlich oft stetig differenzierbar. Der Träger der Funktion ist $(\widetilde{x}-\epsilon, \widetilde{x}+\epsilon)$.

Beweis: An den Enden des Intervalls geht die Exponentialfunktion von innen mit allen ihren Ableitungen exponentiell gegen 0, bei den Ableitungen kommen gebrochenrationale Funktionen hinzu, die polynomial gegen ∞ konvergieren. Insgesamt haben die Ableitungen also den Wert 0 auf dem Rand, und $g_{\epsilon,\widetilde{x}}$ ist unendlich oft differenzierbar. □

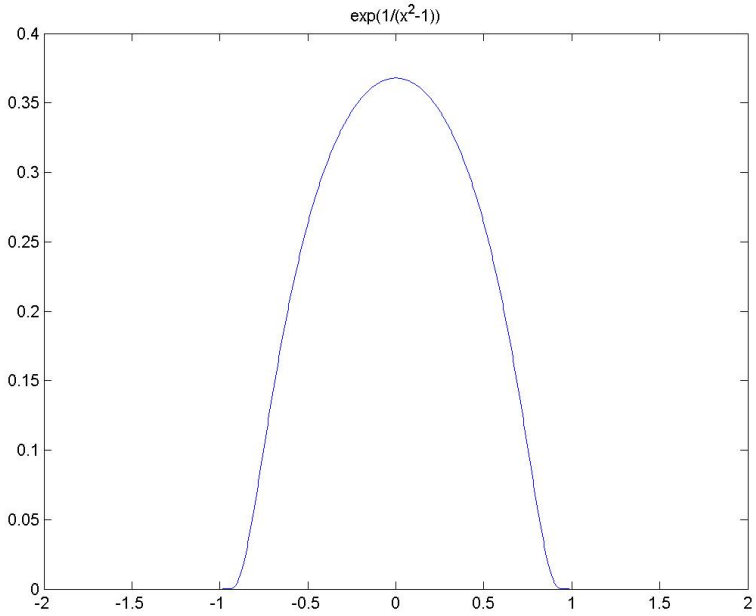


Abbildung 2.22: C^∞–Funktion mit kompaktem Träger

Klick für Bild expfrac

Die Funktion $g_{\epsilon,\widetilde{x}}$ gibt uns die Möglichkeit, eine Funktion in einem sehr kleinen Intervall um $\widetilde{x}$ zu ändern, ohne die Glattheitseigenschaften zu beeinflussen.

Die am häufigsten benutzten Splines sind die kubischen Splines der Ordnung 4. Sie sind zweimal stetig diffbar, also hinreichend glatt, und lösen unser Straklattenproblem aus der Einleitung.

Die Biegeenergie einer Latte mit Formfunktion v, die an den Punkten (x_i, y_i), $i = 0 \dots n$, eingespannt ist, kann approximiert werden durch (siehe z.B. Hanke-Bourgeois [2006], p. 372f)

$$E(u) := (u,u)^{1/2},\ (u,v) := \int_{x_0}^{x_n} u''(x)v''(x)\mathrm{dx}.$$

Sei V der Raum der möglichen Formfunktionen

$$\begin{aligned} V \quad = \{ \quad & v \in C^2(\mathbb{R}) : v \in C^\infty(\,(x_j, x_{j+1})\,),\ v(x_j) = y_j,\ j = 0 \dots n, \\ & v \text{ linear außerhalb } [x_0, x_n] \}\,. \end{aligned}$$

Dies beinhaltet insbesondere, dass die zweite Ableitung der Funktionen in V in x_0 und x_n verschwindet.

Ohne zusätzliche Kräfte nimmt die Latte eine Form $u \in V$ an, die diese Energie unter allen zugelassenen Formfunktionen minimiert, also

$$E(u) \leq E(v) \, \forall v \in V.$$

Wir behaupten: u ist ein Spline.

Satz 2.48 *(Minimaleigenschaft der kubischen B–Splines) Seien $x_0 \dots x_n$ der Größe nach geordnete, paarweise verschiedene Stützstellen und $y_0 \dots y_n$ Stützwerte. Sei*

$$\begin{aligned} V \quad = \{ \quad & v \in C^2(\mathbb{R}) : v \in C^\infty(\,(x_j, x_{j+1}\,),\, v(x_j) = y_j,\, j = 0 \dots n, \\ & v \text{ linear außerhalb } [x_0, x_n]\} \,. \end{aligned}$$

Auf $C^2(\mathbb{R})$ (linear außerhalb $[x_0, x_n]$) seien das Halbskalarprodukt (positiv semidefinite Bilinearform)

$$(u, v) = \int_{\mathbb{R}} u''(x) v''(x) dx$$

und die Halbnorm

$$E(v) = (v, v)^{1/2}$$

definiert. Sei $u \in V$ und

$$E(u) \leq E(v) \forall v \in V.$$

Dann ist u kubischer Spline der Ordnung $k = 4$ zu den Knoten $x_0 \dots x_n$.

$v''(x_0) = v''(x_n) = 0$ liefert uns zusammen mit der Interpolationsbedingung gerade $n + 1 + 2 = n + 3 = n + k - 1$ Bedingungen, d.h. die Funktion ist auch eindeutig bestimmt.

Beweis: Wir setzen $s_i := x_i$. Zu zeigen ist nur noch: u ist auf jedem Intervall (s_i, s_{i+1}) ein Polynom in $\mathcal{P}_3$.
Wir machen den üblichen Ansatz: Bei Minimierungsproblemen über (Halb–) Skalarprodukte wird immer wieder dieselbe Orthogonalitätseigenschaft genutzt (z.B. auch in der Vorlesung Numerische Lineare Algebra bei überbestimmten linearen Gleichungssystemen).
Sei $v \in V$ beliebig. Dann ist auch $u + \alpha(v - u) \in V$ für jedes $\alpha \in \mathbb{R}$. Da u $E(u)$ auf V minimiert, hat die Funktion

$$g(\alpha) := E(u+\alpha(v-u))^2 = (u+\alpha(v-u), u+\alpha(v-u)) = E(u)^2 + 2\alpha(u, v-u) + \alpha^2 E(v-u)^2$$

ein Minimum bei $\alpha = 0$. Also ist

$$g'(0) = 0$$

und damit

$$(u, v - u) = 0 \, \forall v \in V.$$

Dies ist die notwendige Bedingung. Hinreichend wäre, dass die zweite Ableitung (also $2E(v-u)^2$) positiv ist. Man kann wieder zeigen de Boor [2001]: Genau ein Spline löst die Interpolationsaufgabe und ist linear fortsetzbar, d.h. V enthält genau einen Spline u. Wir zeigen: u minimiert das Funktional E.
Zunächst gilt: Dann ist tatsächlich $E(v-u)$ positiv für $v \in V$, $u \neq v$, denn aus $E(v-u) = 0$ folgt $v'' = u''$. Also unterscheiden sich u und v nur um eine lineare Funktion in jedem Intervall, damit ist v ebenfalls ein Spline, und da V nur einen enthält gilt $u = v$.
Zu zeigen ist also noch: $(u, v - u) = 0 \, \forall v \in V$. Mit partieller Integration auf jedem Intervall gilt, da u'' und v'' stetig sind

$$(u, v - u) = \int_{x_0}^{x_n} u''(x)(v-u)''(x)\, dx = -\int_{x_0}^{x_n} u'''(x)(v-u)'(x)\, dx.$$

Die Randterme $u''(x)(v-u)'(x)$ bei x_0 und x_n fallen weg, denn da u und v linear fortsetzbar sind in C^2, gilt $u''(x_0) = v''(x_0) = 0$.
Wir können jetzt nochmals partiell integrieren und erhalten, wieder durch Anwendung auf jedes Einzelintervall,

$$(u, v - u) = \int_{x_0}^{x_n} u''''(x)(v-u)(x)\, dx.$$

Diesmal fallen die Randterme $u'''(x)(v-u)(x)$ weg, weil v und u beide Interpolationsfunktionen sind, d.h. an allen x_k haben sie den gleichen Wert und damit $(v-u)(x_k) = 0$.
Da aber u auf jedem Teilintervall ein kubisches Polynom ist, gilt $u'''' = 0$ und damit endgültig

$$(u, v - u) = 0 \, \forall v \in V.$$

Also ist u ein lokales Minimum von E. Wir zeigen nun: Weitere Minima gibt es nicht. Angenommen, u ist kein kubischer Spline, also nicht in $\mathcal{P}_3$ auf (s_i, s_{i+1}). Dann

$$\exists \widetilde{x} \in (s_i, s_{i+1}) : \; u^{(4)}(\widetilde{x}) \neq 0.$$

Wir zeigen: Dann ist u kein lokales Minimum von E.
Sei also $\epsilon > 0$ so klein, dass die ϵ–Umgebung von $\widetilde{x}$ noch ganz in (s_i, s_{i+1}) liegt, und so klein, dass $u^{(4)}$ sein Vorzeichen in der ϵ–Umgebung nicht ändert ($u^{(4)}$ ist stetig auf (s_i, s_{i+1})!). Wir setzen

$$v = u + g_{\epsilon, \widetilde{x}}.$$

Dann ist wegen $u \in V$ auch $v \in V$. Angenommen, u wäre ein lokales Minimum. Dann gilt wieder nach der Rechnung von oben

$$\begin{aligned} 0 &= (u, v-u) \\ &= \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} u^{(4)} g_{\epsilon,\widetilde{x}} \mathrm{dx}. \end{aligned}$$

Die Randterme fallen dabei jeweils weg, weil $g_{\epsilon,\widetilde{x}}$ mit allen seinen Ableitungen bei $\widetilde{x}-\epsilon$ und $\widetilde{x}+\epsilon$ verschwindet.
Nun ändert aber $u^{(4)}$ sein Vorzeichen nicht in $(\widetilde{x}-\epsilon, \widetilde{x}+\epsilon)$, und $g_{\epsilon,\widetilde{x}}$ ist dort positiv. Das Integral kann also nicht verschwinden, die Annahme war falsch, es gilt $u^{(4)} = 0$ auf (s_i, s_{i+1}), und damit ist $u \in \mathcal{P}_3$ auf diesem Intervall. □

```
function b = splineinterpol4
%SPLINEINTERPOL natural cubic splines
K=4;  %Splinegrad
N=8; %Stuetzstellen
M=N+K-1;
P=100;
```

Listing 2.30: Interpolation mit B–Splines (interpolation/splineinterpol4.m)

Klicken für den Quellcode von interpolation/splineinterpol4.m

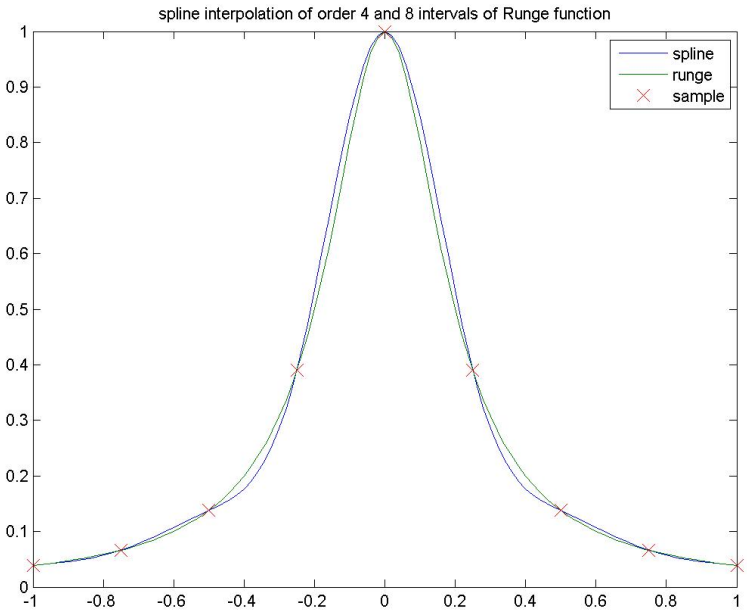


Abbildung 2.23: Spline–Interpolation der Runge–Funktion

Klick für Bild splineinterpol

2.9.2 Spline–Interpolation in höheren Dimensionen

Falls die Auswertepunkte in höheren Dimensionen auf einem rechteckigen Gitter liegen, kann man in so vorgehen wie bei der Fouriertransformation: Für ein Bild, das auf eine höhere Auflösung hochinterpoliert werden soll, wird erst entlang der Zeilen, dann entlang der Spalten mit Splines interpoliert. Dieses Vorgehen führt nicht zum Erfolg, falls die Auswertepunkte unregelmäßig verteilt sind. In diesem Fall benötigt man echte Splines in höheren Dimensionen. In der Numerik der partiellen Differentialgleichungen wird genau dies benutzt, wir geben nur einen ganz kurzen, informellen Ausblick.

Eine natürliche Erweiterung von Splines in höhere Dimensionen sind Finite Elemente. Wir betrachten ein Beispiel im $\mathbb{R}^2$. Es sei eine Funktion auf einem Gebiet $G \subset \mathbb{R}^2$ zu approximieren. Hierzu teilen wir G zunächst in endlich viele Teilgebiete G_k auf. Für jedes Teilgebiet wählen wir einen linearen Raum von Ansatzfunktionen V_k, etwa lineare Funktionen in den Raumkoordinaten x_1, x_2. Wir definieren Funktionen f auf G mit $f|G_k \in V_k \, \forall k$. Damit wäre allerdings nicht einmal die Stetigkeit gesichert. Hierzu fordern wir noch zusätzliche lineare Bedingungen, die die Glattheit sicherstellen.

Beispiel 2.49 Lineare Dreiecke
Sei $G = \bigcup_k \overline{G}_k$, G_k jeweils das Innere eines Dreiecks, und keine Ecke eines Dreiecks falle auf eine Seite eines anderen Dreiecks. Als Ansatzraum auf jedem G_k wählen wir die linearen Funktionen.
Sei f eine Funktion auf $\bigcup_k G_k$, die linear ist auf jedem G_k. Für alle Ecken P, an denen mehrere Dreiecke zusammenstoßen, gelte, dass die Fortsetzungen von $f|_{G_k}$ in P für alle Dreiecke denselben Wert hat. Dann hat f eine stetige Fortsetzung. f heißt Finite–Elemente–Funktion für lineare Dreiecke.

Beweis: *Zu zeigen ist: Die Fortsetzung von f ist entlang aller Kanten stetig. Sei K eine Kante, die durch P und Q begrenzt ist und die Dreiecke G_k und G_j trennt. $f|G_k$ ist linear, also durch die Werte $f(P)$ und $f(Q)$ bereits eindeutig bestimmt. Gleiches gilt aber auf für $f|G_j$, also sind die Fortsetzungen von beiden Seiten gleich.* □

Die Finite–Elemente–Funktionen können wir nun zur Approximation einer Funktion g auf G nutzen. Seien P_k alle Ecken. Dann gibt es genau eine Finite–Elemente–Funktion f mit $f(P_k) = g(P_k)$.

2.10 Approximation und Ausgleichspolynome

In vielen Situationen ist es günstiger, statt einer Interpolation eine Approximation zu wählen, bei der wir die Bedingung fallen lassen, dass die resultierende Funktion exakt durch die vorgegebenen Punkte gehen muss. Dies ist die Methode der Ausgleichspolynome: Wir suchen ein Polynom in $\mathcal{P}_M$ mit $p(x_k) = y_k$, $k = 0 \dots N$, mit $M < N$. Diese Aufgabe ist so offensichtlich im allgemeinen nicht lösbar, mit der Methode der kleinsten Quadrate lassen sich aber optimale Näherungen finden (siehe Numerische Lineare Algebra), siehe Beispiel für $N = 2M$ und das Rungebeispiel.

```
function ausgleichspol( N,M )
%AUSGLEICHSPOL
if (nargin<1)
    N=100;
end
close all;
```

Listing 2.31: Approximation durch Ausgleichspolynome (interpolation/ausgleichspol.m)

Klicken für den Quellcode von interpolation/ausgleichspol.m

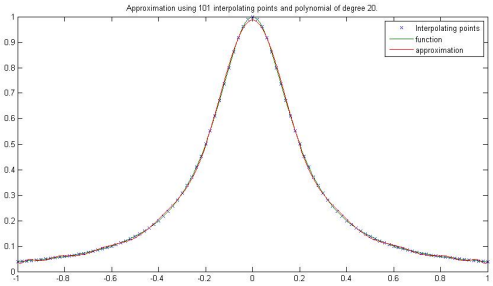


Abbildung 2.24: Approximation durch Ausgleichspolynome

Klick für Bild ausgleichspol

Alternativ können Methoden der Funktionsapproximation gewählt werden. Falls die komplette Funktion f bekannt ist, definieren wir als die Bestapproximation von f bezüglich der ∞–Norm in $\mathcal{P}_N$ das Polynom p_N in $\mathcal{P}_N$, für das $||f - p||_\infty$ minimal ist. Der Satz von Weierstrass garantiert die punktweise Konvergenz von p_N gegen f.

```
> with(numapprox);

  [chebdeg, chebmult, chebpade, chebsort, chebyshev, confracform,

        hermite_pade, hornerform, infnorm, laurent, minimax, pade,
```

Listing 2.32: Bestapproximation in Maple (interpolation/minimax.m)

Klicken für den Quellcode von interpolation/minimax.m

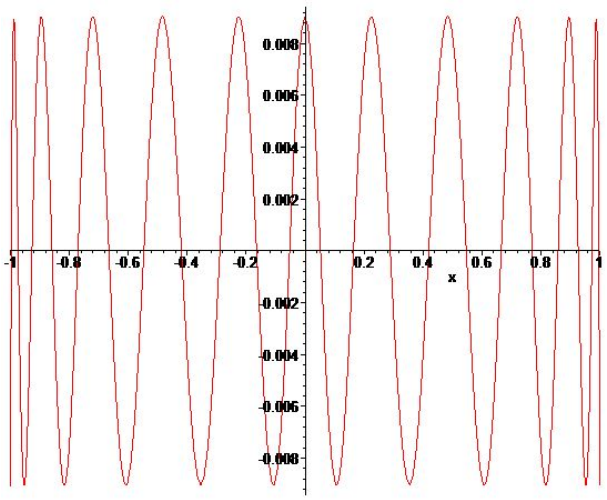


Abbildung 2.25: Fehler der Bestapproximation des Rungebeispiels, berechnet mit Maple

Klick für Bild minimax

```
function y1=matlabminmax(n)
%MATLABMINMAX
%Achtung: Ab Matlab R2009 benoetigt dieses
%Programm die maple-Toolbox
%(wird beim Installieren von Maple nach Matlab
% gleich mitinstalliert)
```

Listing 2.33: Bestapproximation in Matlab (interpolation/matlabminmax.m)

Klicken für den Quellcode von interpolation/matlabminmax.m

Diese Methode wird ausführlich in der Vorlesung Numerische Lineare Algebra behandelt.

Zum Abschluss noch ein Hinweis auf eine Methode, die wir nicht behandelt haben: In unserem Sinn ist der Bèzier–Spline der Computergrafik kein Spline, denn die dort benutzten Kontrollpunkte fordern keine Interpolation, sondern nur eine Approximation.

Kapitel 3

Numerische Integration und Differentiation

In diesem Kapitel wollen wir für $f \in C^k(a,b)$ und eine feste Gewichtsfunktion $w \in L^1(a,b)$, $w(x) > 0$, bestimmte Integrale der Form

$$I(f) = \int_a^b w(x) f(x) \mathrm{dx}$$

sowie die Ableitungen von f numerisch approximieren. Für die normale Integration gilt einfach $w = 1$.

Die einfache zentrale Idee für beide Aufgaben ist: Werte f an einigen Stützstellen aus und integriere bzw. differenziere statt f eine Interpolationsfunktion.

3.1 Klassische Quadraturformeln durch Polynominterpolation

Definition 3.1 *(Quadraturformeln)*
Sei $f \in C([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ Stützstellen. Dann heißt das Funktional

$$I_n(f) = \sum_{j=0}^{n} A_j f(x_j)$$

Quadraturformel *mit* **Gewichten** A_j.

$$|I_n(f) - I(f)|$$

heißt **Quadraturfehler**. I_n *heißt* **exakt von der Ordnung** m**,** *falls*

$$I_n(p) = I(p) \ \forall p \in \mathcal{P}_m.$$

Mit Hilfe der Lagrange–Interpolation können wir leicht Quadraturformeln I_n herleiten, die exakt sind von der Ordnung n. Wir ersetzen einfach die Integration über f durch die Interpolation über das Interpolationspolynom

$$p \in \mathcal{P}_n,\ p(x_j) = f(x_j),\ j = 0 \ldots n,$$

und berechnen p mit der Lagrange–Formel.

Satz 3.2 *(Quadratur durch Lagrange–Interpolation,* **Newton–Cotes***)*
Seien $x_0, \ldots, x_n$ *paarweise verschiedene Stützstellen in* $[a, b]$*, und*

$$A_j := I(w_j) = \int_a^b w(x) w_j(x) dx$$

mit den Lagrange–Polynomen w_j *(2.3)*

$$w_j(x) := \prod_{\substack{k=0 \\ k \neq j}}^{N} \frac{x - x_k}{x_j - x_k},\ j = 0 \ldots N.$$

Dann ist die Newton–Cotes–Quadraturformel der Ordnung n

$$I_n(f) := \sum_{j=0}^{n} A_j f(x_j)$$

exakt von der Ordnung n*. Dies ist die einzige Quadraturformel mit den Stützstellen* $x_0, \ldots, x_n$*, die exakt ist von der Ordnung* n*.*

Beweis: Sei $p \in \mathcal{P}_n$. Dann ist p sein eigenes Interpolationspolynom der Ordnung n, d.h. mit der Formel von Lagrange

$$p(x) = \sum_{j=0}^{n} p(x_j) w_j(x).$$

Begründung: Auf der linken Seite steht ein Polynom in $\mathcal{P}_n$, das an den Stellen x_j den Wert $p(x_j)$ annimmt, auf der rechten Seite auch, und dieses Polynom ist eindeutig

nach 2.3.
Also gilt

$$\begin{aligned} I(p) &= \int_a^b w(x)p(x)\mathsf{dx} \\ &= \sum_{j=0}^{n} p(x_j) \int_a^b w(x)w_j(x)\mathsf{dx} \\ &= \sum_{j=0}^{n} A_j p(x_j) \\ &= I_n(p). \end{aligned}$$

Zur Eindeutigkeit setze $p := w_j \in \mathcal{P}_n$, damit gilt $A_j = I_n(w_j) = I(w_j)$. □

Definition 3.3 *(**offene und geschlossene Newton–Cotes–Formeln**)*

1. *Seien $x_0, \ldots, x_n$ gleichverteilt im abgeschlossenen Intervall $[a, b]$, also $x_k = a+kh$, $h = (b-a)/n$. Dann heißen die zugehörigen exakten Quadraturformeln der Ordnung n* **geschlossene Newton–Cotes–Formeln der Ordnung** n.
2. *Seien $x_0, \ldots, x_n$ gleichverteilt im offenen Intervall (a, b), also $x_k = a+(k+1)h$, $h = (b-a)/(n+2)$. Dann heißen die zugehörigen exakten Quadraturformeln der Ordnung n* **offene Newton–Cotes–Formeln der Ordnung** n.

Üblicherweise gibt man die Formeln auf einem Standardintervall an und transformiert linear auf $[a, b]$. Sei also etwa eine Formel mit Stützstellen t_j auf $[0, N]$ und Gewichten A_j gegeben, dann erhält man durch

$$x = a + th,\ h = \frac{b-a}{N}$$

$$\int_a^b w(N\frac{x-a}{b-a})f(x)\mathsf{dx} = h\int_0^N w(t)f(a+th)\mathsf{dt} \sim h\sum_{j=0}^{n} A_j f(x_j)$$

mit

$$x_j = a + t_j h,\ j = 0 \ldots n$$

eine Quadraturformel für $[a, b]$.

Beispiel 3.4 *(**Gewichte für äquidistante Newton–Cotes–Formeln**)*
Die geschlossenen Newton–Cotes–Formeln der Ordnung n geben wir auf dem Intervall $[0, n]$. Für die Stützstellen t_j gilt dann gerade

$$t_j = j,\ j = 0 \ldots n.$$

Es gilt

$$\begin{aligned} A_j &= \int_0^n w(t) w_j(t) dt \\ &= \int_0^n w(t) \prod_{\substack{l=0 \\ l \neq j}}^{n} \frac{t-l}{j-l} dt,\ j = 0 \dots n \end{aligned}$$

und damit

$$I_n(f) = h \sum_{j=0}^{n} A_j f(a + jh),\ h = \frac{b-a}{n}.$$

Die offenen Formeln der Ordnung n geben wir auf dem Intervall $[0, n+2]$ an. Es gilt für dieses Intervall (Index beachten!)

$$t_j = j,\ j = 1 \dots n+1.$$

$$A_j = \int_0^{n+2} w(t) \prod_{\substack{l=1 \\ l \neq j}}^{n+1} \frac{t-l}{j-l} dt,\ j = 1 \dots n+1$$

und damit

$$I_n(t) = h \sum_{j=1}^{n+1} A_j f(a + jh),\ h = \frac{b-a}{n+2}.$$

Beispiel 3.5 *(Newton–Cotes für $w = 1$)*

1. *Offene Newton–Cotes–Formel, $n = 0$, $h = (b-a)/2$:*

   $$x_0 = (a+b)/2,\ A_1 = \int_0^2 1 dx = 2,\ I_0(f) = f\left(\frac{a+b}{2}\right)(b-a).$$

   Dies ist die Mittelpunktsregel.

2. *Geschlossene Newton–Cotes–Formel, $n = 1$:*

   $$h = (b-a),\ x_0 = a,\ x_1 = b$$

   $$\begin{aligned} a_0 &= \int_0^1 \frac{t-1}{0-1} dt = \frac{1}{2} \\ a_1 &= \int_0^1 \frac{t-0}{1-0} dt = \frac{1}{2} \end{aligned}$$

$$I_1(f) = \frac{b-a}{2}(f(a)+f(b))$$

Dies ist die Trapezregel. Bei der Trapezregel wird die Integration über die Funktion durch die Integration über ein Trapez ersetzt, das aus der x-Achse und den Funktionswerten in a und b gebildet wird.

Die folgende Tabelle gibt die Gewichte A_j für die geschlossenen Newton–Cotes–Formeln

$$I_n(f) := h\sum_{j=0}^{n} A_j f(a+jh),\ h = \frac{b-a}{n}$$

und ihre üblichen Namen an:

n	A_0	A_1	A_2	A_3	A_4		Name
1	1	1				$\cdot\frac{1}{2}$	Trapezregel
2	1	4	1			$\cdot\frac{1}{3}$	Simpson–Regel, Keplersche Fassregel
3	1	3	3	1		$\cdot\frac{3}{8}$	Newtonsche $\frac{3}{8}$–Regel
4	7	32	12	32	7	$\cdot\frac{2}{45}$	Milne–Regel

Für $n \geq 8$ werden die Gewichte teils negativ, was zu Instabilität führt, die Formeln sind dann unbrauchbar, wie wir gleich zeigen werden (3.6).

```
function A = quadratur( a,b,x,w )
%QUADRATUR
if (nargin<4)
    w=@one;
end
n=numel(x);
```

Listing 3.1: Gewichte für Quadraturformeln (integration/quadratur.m)

Klicken für den Quellcode von integration/quadratur.m

```
function [A,x] = newtoncotesclosed( n,a,b,w )
%NEWTONCOTESCLOSED
if (nargin<3)
    a=0;
    b=n;
end
```

Listing 3.2: Geschlossene Newton–Cotes–Formeln (integration/newtoncotesclosed.m)

Klicken für den Quellcode von integration/newtoncotesclosed.m

```
function [A,x] = newtoncotesopen( n,a,b,w )
%NEWTONCOTESCLOSED
if (nargin<3)
    a=0;
    b=n+2;
end
```

Listing 3.3: Offene Newton–Cotes–Formeln (integration/newtoncotesopen.m)

Klicken für den Quellcode von integration/newtoncotesopen.m

Wir vermuten, dass Formeln höherer Ordnung zu besserer Approximation führen, und testen dies anhand der Exponentialfunktion:

Beispiel 3.6 *(Geschlossene Newton–Cotes für die Exponentialfunktion)*
Sei $f(x) := \exp(x)$.

$$\begin{array}{lllll} I(f) = \int_0^1 \exp(x)dx & = & e-1 & \sim & 1.7183 \\ I_1(f) & = & \frac{1}{2}(1+e) & \sim & 1.8591 \\ I_2(f) & = & \frac{1}{6}(1+4e^{1/2}+e) & \sim & 1.7189 \\ I_3(f) & = & \frac{1}{8}(1+3e^{1/3}+3e^{2/3}+e) & \sim & 1.7185 \end{array}$$

Offensichtlich bringt die Erhöhung der Stützstellen von 1 nach 2 sehr viel, die von 2 nach 3 fast überhaupt nichts. Die Erklärung liefert der folgende Satz.

Satz 3.7 *(Fehlerabschätzung für Quadraturformeln)*
Sei I_n *eine Quadraturformel, die exakt ist von der Ordnung* n.

1. *Sei*

$$f \in C^{n+1}([a,b]),\ W(x) = \prod_{j=0}^{n}(x-x_j).$$

Dann gilt

$$|I(f) - I_n(f)| \leq C_n \max_{x\in[a,b]} |f^{(n+1)}(x)|$$

mit

$$\begin{aligned} C_n &= \frac{1}{(n+1)!}\int_a^b w(x)|W(x)|dx \\ &\leq \frac{||w||_\infty}{(n+1)!}\int_a^b |W(x)|dx \\ &\leq ||w||_\infty \, ||W||_\infty \frac{b-a}{(n+1)!} \\ &\leq ||w||_\infty \frac{(b-a)^{n+2}}{(n+1)!}. \end{aligned}$$

2. *Es sei n gerade, also*

$$n = 2m, f \in C^{n+2}([a,b]).$$

Es seien die Gewichtsfunktion w und die Stützpunkte $x_0, \ldots, x_n$ gerade bezüglich der Intervallmitte, d.h.

$$w(a+x) = w(b-x),\ x \in [0, b-a],$$

und

$$x_j - a = b - x_{n-j},\ k = 0 \ldots n.$$

Letzteres ist etwa für die offenen und geschlossenen Newton–Cotes–Formeln erfüllt. Dann ist I_n sogar exakt von der Ordnung $n+1$, und es gilt

$$|I(f) - I_n(f)| \leq C_n^* \max_{x\in[a,b]} |f^{(n+2)}(x)|$$

mit

$$C_n^* = \frac{b-a}{2} C_n.$$

Insbesondere ist I_n von der Ordnung $n+1$.

Der Satz sagt also; Für gerade n ist die Interpolationsformel bei symmetrisch verteilten Stützstellen sogar noch für eine Ordnung mehr exakt, als wir eigentlich gefordert haben. Deshalb bekommen wir hier bereits fast dieselbe Abschätzung wie bei Formeln der Ordnung $n+1$.

Beweis: Sei p das Interpolationspolynom

$$p \in \mathcal{P}_n,\ p(x_j) = f(x_j),\ j = 0 \ldots n.$$

Nach (2.14) gilt

$$\begin{aligned} |I(f) - I_n(f)| &\leq \int_a^b w(x)|(f-p)(x)|\mathsf{dx} \\ &= \int_a^b w(x)|f^{n+1}(\xi(x))|\frac{|W(x)|}{(n+1)!}\mathsf{dx} \\ &\leq \int_a^b \frac{w(x)|W(x)|}{(n+1)!}\mathsf{dx}||f^{(n+1)}||_\infty. \end{aligned}$$

Da $|x - x_j| \leq (b-a)$, gilt $W(x) \leq (b-a)^{n+1}$, daraus folgt Teil 1. Dies ist natürlich eine extreme Worst–Case–Abschätzung, sie geht davon aus, dass alle Stützstellen nah am linken oder rechten Rand liegen. Für gleichverteilte Stützstellen bekommt man die viel bessere Abschätzung aus 2.15.
Seien nun die Voraussetzungen aus Teil 2 erfüllt. Sei

$$c = \frac{a+b}{2}.$$

$$\begin{aligned} W(c+x) &= \prod_{j=0}^{n}((c+x) - x_j) \\ &= \prod_{j=0}^{n}((c+x) - (a+b-x_{n-j})) \\ &= \prod_{j=0}^{n}(-c+x+x_j) \\ &= (-1)^{n+1}\prod_{j=0}^{n}((c-x) - x_j) \\ &= -W(c-x). \end{aligned}$$

W ist also ungerade bezüglich der Intervallmitte c. und w ist gerade. Also gilt

$$\int_a^b w(x)W(x)\mathsf{dx} = 0.$$

Da $f \in C^{(n+2)}$, können wir f^{n+1} in die Taylorreihe der Ordnung 1 um c entwickeln,

es gilt also

$$\begin{aligned} I(f) - I_n(f) &= \int_a^b w(x)(f-p)(x)\mathsf{dx} \\ &= \int_a^b \frac{w(x)W(x)}{(n+1)!} f^{(n+1)}(\xi(x))\mathsf{dx} \\ &= \int_a^b \frac{w(x)W(x)}{(n+1)!} \{f^{(n+1)}(c) + \underbrace{(\xi(x)-c)}_{|\cdot| \leq \frac{b-a}{2}} f^{(n+2)}(\mu(x))\}\mathsf{dx} \end{aligned}$$

und damit

$$|I(f) - I_n(f)| \leq \frac{b-a}{2} \int_a^b \frac{w(x)|W(x)|}{(n+1)!}\mathsf{dx}||f^{(n+2)}||_\infty$$

und dies ist Aussage 2.
Für $p \in P_{n+1}$ ist $p^{(n+2)} = 0$, also ist $I(p) - I_n(p) = 0$. □

Für die geschlossenen Newton–Cotes–Formeln gilt damit

Satz 3.8 *(Fehlerabschätzung für die geschlossenen Newton–Cotes–Formeln)*
Sei $f \in C^{(n+1)}([a,b])$. Für die geschlossenen Newton–Cotes–Formeln gilt

$$|I(f) - I_n(f)| \leq h^{n+2} c_n ||f^{(n+1)}||_\infty,\ h = \frac{b-a}{n}$$

mit

$$c_n = \frac{1}{(n+1)!} \int_0^n \prod_{k=0}^n |t-k|\mathsf{dt}\, ||w||_\infty.$$

Falls n und w gerade sind und $f \in C^{(n+2)}([a,b])$, so gilt

$$|I(f) - I_n(f)| \leq h^{n+3} c_n^* ||f^{(n+2)}||_\infty$$

mit $c_n^* = \frac{n}{2} c_n$.

Beweis: Wir zeigen den Satz für $w = 1$. Wieder gilt mit der Substitution $x = a +$

th

$$\begin{aligned}\int_a^b |W(x)|\mathrm{dx} &= \int_a^b \prod_{j=0}^{n} |x - x_j|\mathrm{dx} \\ &= \int_a^b \prod_{j=0}^{n} |x - (a + jh)|\mathrm{dx} \\ &= h\int_0^n \prod_{j=0}^{n} (|t - j|h)\mathrm{d}th \\ &= h^{n+2}\int_0^n \prod_{j=0}^{n} |t - j|\mathrm{dt}.\end{aligned}$$

Einsetzen in die obige Formel liefert das Gewünschte. Für den zweiten Teil beachte $(b - a) = nh$. □

Bemerkung: c_n und c_n^* hängen nicht vom Intervall ab.

Bemerkung: Eine etwas genauere Rechnung zeigt, dass man die Konstanten noch verbessern kann (Freund and Hoppe [2007], p. 175). Damit erhält man die endgültigen Fehlerformeln

Fehler	Integrationsformel
$\frac{h^3}{12}\|\|f^{(2)}\|\|_\infty$	Trapezregel
$\frac{h^5}{90}\|\|f^{(4)}\|\|_\infty$	Simpson–Regel
$\frac{3h^5}{80}\|\|f^{(4)}\|\|_\infty$	Newtonsche 3/8–Regel
$\frac{8h^7}{945}\|\|f^{(6)}\|\|_\infty$	Milne–Regel

3.2 Zusammengesetzte Formeln

Bei der Konstruktion der Newton–Cotes–Formeln haben wir bereits bemerkt, dass für großes n Schwierigkeiten auftreten. Dieses Verhalten ist uns bereits aus der Polynominterpolation bekannt, und ähnlich wie dort geht der Quadraturfehler nicht monoton gegen 0. Wir sollten daher denselben Weg wie bei der Polynominterpolation gehen und die Integration auf Teilintervalle aufteilen. Wir betrachten wieder die geschlossenen Newton–Cotes–Formeln.

Definition 3.9 *(Zusammengesetzte Formeln)*
Sei I_n eine Quadraturformel auf einem Referenzintervall (etwa $[0, 1]$). Dann erhält

man die zusammengesetzte Formel $\tilde{I}_n$ zur Integration auf dem Intervall $[a,b]$, indem man $[a,b]$ in Teilintervalle aufteilt, jedes auf das Referenzintervall transformiert und dann dort das Integral mit I_n approximiert. Dies geht im allgemeinen nur für die Gewichtsfunktion $w = 1$.

Beispiel 3.10 *(Zusammengesetzte geschlossene Newton–Cotes–Formeln)*
Sei $N = np$, $h = (b-a)/N$, I_n die geschlossene Newton–Cotes–Formel der Ordnung n. Wir teilen $[a,b]$ in p gleichgroße Intervalle $[x_{nj}, x_{n(j+1)}]$ auf. Dann gilt für die zusammengesetzten Formeln $\tilde{I}_n$ und die Gewichtsfunktion $w = 1$:

1. *(Zusammengesetzte Trapezregel)*

$$\begin{aligned}\tilde{I}_1(f) &= \frac{h}{2}\left(((f(x_0)+f(x_1))+(f(x_1)+f(x_2))+\ldots+(f(x_{N-1})+f(x_N))\right)\\ &= \frac{h}{2}(f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{N-1})+f(x_N))\\ &= \frac{h}{2}\left(f(x_0)+2\sum_{j=1}^{N-1}f(x_j)+f(x_N)\right)\end{aligned}$$

2. *(Zusammengesetzte Simpson–Regel)*

$$\begin{aligned}\tilde{I}_2(f) &= \frac{h}{3}((f(x_0)+4f(x_1)+f(x_2))+(f(x_2)+4f(x_3)+f(x_4))\ldots)\\ &= \frac{h}{3}\left(f(x_0)+4\sum_{j=1}^{p}f(x_{2j-1})+2\sum_{j=1}^{p-1}f(x_{2j})+f(x_N)\right)\end{aligned}$$

Satz 3.11 *(Fehler von zusammengesetzten Newton–Cotes–Formeln)*
Sei $f \in C^{n+1}([a,b])$, $w = 1$. Dann gilt für die zusammengesetzten Newton–Cotes–Formeln der Ordnung n in p Intervallen

$$|\tilde{I}_n(f) - I(f)| \leq (b-a)\frac{c_n}{n}h^{n+1}||f^{(n+1)}||_\infty,\ h = \frac{b-a}{N}$$

mit den Konstanten c_n aus Satz 3.8. Ist n gerade, so gilt sogar

$$|\tilde{I}_n(f) - I(f)| \leq (b-a)\frac{c_n^*}{n}h^{n+2}||f^{(n+2)}||_\infty.$$

Beweis: Folgt direkt aus 3.8. Etwa für den ersten Teil:

$$\begin{aligned}\left|\tilde{I}_n(f) - I(f)\right| &\leq \sum_{j=0}^{p-1}\left|I_n(f|_{[x_{jn},x_{(j+1)n}]}) - I(f|_{[x_{jn},x_{(j+1)n}]})\right|\\ &\leq pc_nh^{n+2}||f^{(n+1)}||_\infty\\ &\leq (b-a)\frac{c_n}{n}h^{n+1}||f^{(n+1)}||_\infty.\end{aligned}$$

□

Korollar 3.12 *(Fehler der zusammengesetzten Trapez– und Simpsonregel)*

Für die zusammengesetzte Trapezregel gilt für $w = 1$ *und* $f \in C^2$

$$|I(f) - \tilde{I}_1(f)| \leq \frac{b-a}{12} h^2 ||f''||_\infty.$$

Für die zusammengesetzte Simpson–Regel gilt für $w = 1$ *und* $f \in C^4$

$$|I(f) - \tilde{I}_2(f)| \leq \frac{b-a}{2880} h^4 ||f^{(4)}||_\infty.$$

(Dies nutzt aber bereits die verbesserte Konstante, bei Anwendung unserer Formeln kommt man im Nenner nur auf 180.)

3.3 Romberg–Verfahren

Zu berechnen sei das Integral

$$I = \int_a^b f(x) \mathrm{dx}.$$

Wir approximieren es mit der Trapezregel und der Schrittweite h und erhalten eine Näherung $\tilde{I}(h)$. Anschließend halbieren wir die Schrittweite und approximieren wieder mit der Trapezregel, dies gibt ein $\tilde{I}(h/2)$. Zur Berechnung von $\tilde{I}(h/2)$ können wir die alten Funktionsauswertungen aus dem letzten Schritt wiederverwenden. Wir halbieren wieder die Schrittweite usw.

Wir erhalten dadurch Näherungen

$$\tilde{I}(h),\ \tilde{I}(h/2),\ \tilde{I}(h/4),\ \ldots$$

an I. Der Grenzwert von $\tilde{I}$ für $h \to 0$ ist 0. Zur Verbesserung der Konvergenzordnung bietet sich also die Richardson–Extrapolation 2.25 an – dies ist das **Romberg–Verfahren**.

Es ist umso effizienter, je größer der Exponent in der Taylorentwicklung des Fehlers ist. Für die zusammengesetzte Trapezregel $\tilde{I}(h)$ besitzt der Quadraturfehler eine Entwicklung in h^2, es ist also bei Richardson $p = 2$. Zum Beweis benötigen wir das Hilfsmittel der Bernoulli–Polynome.

Definition 3.13 *Seien $B_k(x)$ für $x \in [0,1]$ die Bernoulli–Polynome, d.h.*

$$B_0(x) = 1,\ B_k'(x) = B_{k-1}(x),\ \int_0^1 B_k(x)dx = 0,\ k = 1, \ldots, .$$

Dann sind die Bernoulll–Zahlen B_k definiert durch

$$B_k = k!B_k(0).$$

Bemerkung: $B_1(x) = x - 1/2$, $B_2(x) = 1/2(x^2 - x + 1/3)$, $B_3(x) = 1/3x(x - 1/2)(x - 1)$.

Satz 3.14 *Sei $f \in C^{(2m+2)}([a,b])$. Dann gilt für die zusammengesetzte Trapezregel $\widetilde{I}_1(f)$ aus 3.10, $h = (b-a)/N$*

$$\tilde{I}_1(f) = I(f) + c_2h^2 + c_4h^4 + \ldots + c_{2m}h^{2m} + O(h^{2m+2})$$

mit

$$c_k = \frac{B_k}{k!}(f^{(k-1)}|_a^b)$$

und den Bernoulli–Zahlen B_k.

Zum Beweis könnten wir genau so vorgehen wie in 3.7, da sieht man sofort die Entwicklung des Fehlers in h^2. Um die Konstanten in der Entwicklung genau angeben zu können, zeigen wir zunächst eine vereinfachte Form der Euler–MacLaurinschen Summenformel:

Satz 3.15 *Sei B_k periodisch fortgesetzt auf ganz $\mathbb{R}$ mit der Periode 1 (also $B_k(x+1) = B_k(x)$, periodische Bernoulli–Polynome). Sei $g \in C^{2m+2}$.Dann gilt*

$$\begin{aligned}\int_0^N g(x)dx &= \frac{1}{2}g(0) + g(1) + \cdots + g(N-1) + \frac{1}{2}g(N) + \\ &\quad \int_0^N B_{(2m+2)}(x)g^{2m+2}(x)dx - \sum_{k=0}^{m} \frac{B_{2k+2}}{(2k+2)!}(g^{(2k+1)}|_0^N).\end{aligned}$$

Beweis: B_k ist stetig für $k \geq 2$ und $B_{2k+1} = 0$ für $k \geq 1$ (Übungen). Dann gilt

$$\begin{aligned}
\int_0^N B_1(x)g'(x)\mathsf{dx} &= \sum_{i=0}^{N-1} \int_i^{i+1} B_1(x)g'(x)\mathsf{dx} \\
&= \sum_{i=0}^{N-1} B_1 g|_i^{i+1} - \int_i^{i+1} B_1'(x)g(x)\mathsf{dx} \\
&= \sum_{i=0}^{N-1} \frac{g(i+1)+g(i)}{2} - \int_i^{i+1} g(x)\mathsf{dx} \\
&= \frac{1}{2}g(0) + g(1) + \cdots + g(N-1) + \frac{1}{2}g(N) - \int_0^N g(x)\mathsf{dx}.
\end{aligned}$$

Andererseits ist

$$\begin{aligned}
\int_0^N B_1(x)g'(x)\mathsf{dx} &= \int_0^N B_2'(x)g'(x)\mathsf{dx} \\
&= -\int_0^N B_2(x)g''(x)\mathsf{dx} + (B_2(x)g'(x))|_0^N, \text{ denn } B_2 \text{ ist stetig} \\
&= -\int_0^N B_3'(x)g''(x)\mathsf{dx} + (B_2(x)g'(x))|_0^N \\
&= \int_0^N B_3(x)g'''(x)\mathsf{dx} - \underbrace{(B_3(x)g''(x))|_0^N}_{=0} + (\frac{B_2}{2!}g'(x))|_0^N
\end{aligned}$$

usw. Insgesamt erhält man also

$$\int_0^N B_1(x)g'(x)\mathsf{dx} = -\int_0^N B_{2m+2}(x)g^{(2m+2)}(x)\mathsf{dx} + \sum_{k=0}^{m} \frac{B_{2k+2}}{(2k+2)!}(g^{(2k+1)}|_0^N)$$

und daraus folgt die Behauptung. □
Hieraus folgt 3.14 durch die Substitution $g(x) = f(a + xh)$.

Vorlesungsnotiz: 9.Mai 2011

Wir fassen die Idee der Romberg–Integration kurz zusammen.

Definition 3.16 *(***Romberg–Verfahren***)*
Sei f eine ausreichend häufig differenzierbare Funktion auf dem Intervall $[a, b]$. Zu berechnen sei das Integral

$$I(f) = \int_a^b f(x)\mathsf{dx}.$$

Sei $I(h)$ mit $h = (b-a)/N$ die Näherung, die die zusammengesetzte Trapezregel 3.10 mit Schrittweite h liefert (mit $(N+1)$ Funktionsauswertungen). Dann kann die Näherung $I(h/2)$ mit N weiteren Funktionsauswertungen berechnet werden, entsprechend für $I(h/4)$ usw.
Der Grenzwert für $h \mapsto 0$ wird aus diesen Näherungen mit Hilfe der Richardson–Extrapolation 2.1 approximiert. Der Fehler besitzt dabei eine quadratische Entwicklung in h, es gilt also $p = 2$.
Diese Methode heißt Romberg–Verfahren.

Bemerkung: Die Romberg–Integration wird durchgeführt auf der Basis des Schemas, das schon aus der Richardson–Extrapolation 2.22 bzw. aus dem Neville–Schema 2.6 bekannt ist, und heißt hier Romberg–Schema oder Romberg–Tableau. In den ersten drei Spalten des Romberg–Schemas erhält man bekannte Newton–Cotes–Formeln zurück (s. Übungen).

```
function [ y,schema ] = symbolicromberg( level )
%SYMBOLICROMBERG
if (nargin<1)
    level=3;
end
syms h;
```

Listing 3.4: Symbolisches Romberg–Verfahren (integration/symbolicromberg.m)

Klicken für den Quellcode von integration/symbolicromberg.m

Beispiel 3.17 (Romberg–Schema (symbolisch))
Zu berechnen sei

$$\int_a^b f(x)dx,\ f \in C^4.$$

Es stehen mit der zusammengesetzten Trapezregel berechnete Näherungen $I_h(f)$ und $I_{h/2}(f)$ mit den Schrittweiten $h = (b-a)/N$ und $h/2 = (b-a)/(2N)$ zur Verfügung. Wir berechnen wie in 2.22 das Neville– bzw. Romberg–Schema.
Aufgrund der Vorüberlegungen gilt wieder $p = 2$.

$$\begin{array}{ll} h^2 & I_0 = I_h(f) \quad I_{01} \\ \left(\frac{h}{2}\right)^2 & I_1 = I_{h/2}(f) \end{array}$$

Gemäß den Rechenregeln für das Neville–Schema 2.6 gilt

$$I_{01} = \frac{1}{h^2 - (h/2)^2}((-h/2)^2 I_0 + h^2 I_1) = \frac{1}{3}(4I_1 - I_0).$$

Nach Einsetzen der Definition für die zusammengesetzte Trapezregel ist dies die zusammengesetzte Simpsonregel aus 3.10 für die Schrittweite $h = (b-a)/N$.

3.4 Harmonische Analyse und Fouriertransformation

Sei $f \in C(\mathbb{R})$ eine 2π–periodische Funktion. Dann gilt

$$f(x) = \sum_k c_k e^{ikx}$$

mit $c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) \exp(-ikx)$dx. Wir werten die Integrale durch die zusammengesetzte Trapezregel aus, dann gilt offensichtlich $\tilde{c}_k = h/(2\pi)\widehat{y}_k$ mit $y_j = f(jh)$, $j = 0 \ldots n-1$, und $h = 2\pi/n$, $k \in \mathbb{Z}$. Für die Genauigkeit:

Satz 3.18 *(**Harmonische Analyse**)*

1. *Sei $f \in C^{2m}(\mathbb{R})$ periodisch, $I(f)$ das Integral über eine ganze Periode, $\widetilde{I}_1(f)$ die Approximation durch die zusammengesetzte Trapezregel mit der Schrittweite h. Dann gilt*
$$I(f) - \tilde{I}_1(f) = O(h^{2m}).$$
2. *Falls sogar $f \in C^\infty$ und f periodisch, so geht $I(f) - \tilde{I}_1(f)$ schneller mit h gegen 0 als jede Potenz von h.*

Beweis: Satz 3.14. □

Die Auswertung der Trapezregel ist natürlich nichts anderes als die diskrete Fouriertransformation 2.8.4.

Periodische Funktionen können also extrem gut mit der Trapezregel über das gesamte Intervall integriert werden. Dies ist leicht verständlich: Die ungleichmäßige Verteilung der Stützstellen wird gerechtfertigt durch den Einfluss der Intervallenden (siehe die Diskussion in 2.15 zur Polynominterpolation). Bei periodischen Funktionen gibt es keine Intervallenden, also ist eine gleichmäßige Verteilung der Auswertepunkte optimal.

3.5 Gauss–Formeln

Bisher haben wir immer die Stützstellen als gegeben angenommen (und meist äquidistant gewählt). Schon bei der Interpolation haben wir aber gesehen, dass eine nicht–äquidistante, optimale Wahl der Lage der Stützstellen (etwa als Nullstellen der Tschebyscheff–Polynome) günstiger sein kann als eine gleichverteilte. Dies führt uns auf die Gauss–Integrationsformeln.

Die Stützstellen werden also jetzt als Parameter angesetzt und für ein Referenzintervall optimal bestimmt, dann wie in 3.4 auf ein gegebenes Intervall transformiert. Tatsächlich werden wir eine erheblich höhere Genauigkeit als für die Newton–Cotes–Formeln nachweisen.

Der größte Nachteil dieser Formeln sei aber hier gleich von Anfang an vermerkt: Falls wir bei den zusammengesetzten Newton–Cotes–Formeln merken, dass die Genauigkeit nicht ausreicht, können wir die Zahl der Teilintervalle erhöhen und alle bisher berechneten Teilergebnisse wiederverwenden. Es werden lediglich einige zusätzliche Stützpunkte eingefügt. Insbesondere können wir auch Romberg bzw. Richardson nutzen. Bei den zusammengesetzten Gauss–Formeln erhalten wir bei einer Intervallaufteilung völlig neue Stützstellen und müssen alles neu rechnen.

Es stellt sich die Frage, welche Ordnung man durch geschickte Wahl der Stützstellen maximal erreichen kann. Durch unsere neue Sichtweise haben wir jetzt $2n+2$ Freiheitsgrade ($n+1$ Gewichte, $n+1$ Stützstellen). Wir können also hoffen, Polynome vom Grad $(2n+1)$ korrekt zu integrieren.

Seien $x_0 < \ldots < x_n$ Stützstellen für eine Integrationsformel

$$I_n(f) := \sum_{j=0}^{n} A_j f(x_j),\ I_n(f) \sim I(f) := \int_a^b w(x) f(x) \mathsf{dx},$$

die exakt ist von der Ordnung n (3.2).

Sei

$$v_{n+1}(x) := \prod_{j=0}^{n} (x - x_j),\ v_{n+1} \in \mathcal{P}_{n+1}.$$

Sei $q \in \mathcal{P}_n$ beliebig. Dann ist

$$p = v_{n+1} q \in \mathcal{P}_{2n+1}.$$

Es gilt

$$I_n(p) = I_n(v_{n+1}) q = 0,$$

denn die x_j sind Nullstellen von v_{n+1}. Damit die Formel sogar von der Ordnung $2n+1$ exakt ist, muss also gelten

$$\int_a^b w(x) v_{n+1}(x) q(x) \mathsf{dx} = I(p) = I_n(p) = 0. \tag{3.1}$$

Üblicherweise betrachtet man auf dem Vektorraum der stetigen Funktionen Skalarprodukte der Form

$$(p, q)_w := \int_a^b w(x) p(x) q(x) \mathsf{dx}$$

mit positiver, fester Gewichtsfunktion w.

Dann bedeutet 3.1: v_{n+1} muss senkrecht stehen auf den Polynomen aus dem Polynomraum $\mathcal{P}_n$. v_{n+1} bestimmen wir mit dem Gram–Schmidtschen Orthogonalisierungsverfahren.

Lemma 3.19 Gram–Schmidtsches Orthogonalisierungsverfahren
Sei V ein Vektorraum mit Skalarprodukt $(\cdot,\cdot)_w$. Seien

$$p_j \in V,\ j = 0 \dots n,$$

linear unabhängig.
Es sei

$$v_j := \begin{cases} p_0, & j = 0 \\ p_j - \sum_{k=0}^{j-1} \frac{(p_j,v_k)_w}{(v_k,v_k)_w} v_k, & j = 1 \dots n. \end{cases}$$

Dann stehen die v_j senkrecht aufeinander, und

$$\mathit{span}(p_0,\dots,p_j) = \mathit{span}(v_0,\dots,v_j)\ \forall\, j = 0 \dots n.$$

Beweis: Durch vollständige Induktion. Sei der Satz bereits richtig für $j-1$. Sei $l < j$.

$$\begin{aligned}(v_l, v_j)_w &= (v_l, p_j - \sum_{k=0}^{j-1} \frac{(p_j,v_k)_w}{(v_k,v_k)_w} v_k)_w \\ &= (v_l, p_j)_w - \sum_{k=0}^{j-1} \frac{(p_j,v_l)_w}{(v_l,v_l)_w} \underbrace{(v_l,v_k)_w}_{=0 \text{ für } l\neq k (IV)} \\ &= 0.\end{aligned}$$

□

Definition 3.20 *(**orthogonale Polynome**)*
Sei $(\cdot,\cdot)_w$ ein Skalarprodukt auf dem Vektorraum der stetigen Funktionen auf dem Intervall $[a,b]$. Dann liefert das **Gram–Schmidtsche Orthogonalisierungsverfahren,** *angewandt auf die Folge der Monome x^n, eine Folge von Polynomen $v_n \in P_n$, die orthogonal sind bezüglich dieses Skalarprodukts.*

Die v_n heißen orthogonale Polynome. v_n ist nur bis auf einen Faktor eindeutig bestimmt.

Im Folgenden seien immer v_n die orthogonalen Polynome zu $(\cdot,\cdot)_w$.

Beispiel 3.21 *(orthogonale Polynome)*

1. $w(x) = 1$*:* v_n *heißen* **Legendre–Polynome**.
2. $w(x) = \frac{1}{\sqrt{1-x^2}}$*:* v_n *heißen* **Tschebyscheff–Polynome**.

Bemerkung:

1. Da $\{v_0, \dots v_n\}$ den $\mathcal{P}_n$ aufspannen, gilt
$$\text{grad}\, v_n = n.$$
2. $v_0, \dots, v_n$ sind eine ONB von $\mathcal{P}_n$ und stehen senkrecht auf v_{n+1}, also
$$v_{n+1} \perp \mathcal{P}_n.$$

Das $(n+1)$. orthogonale Polynom v_{n+1} erfüllt also gerade unsere Bedingung aus 3.1. Unsere Vermutung ist: Wir müssen die Stützstellen als Nullstellen dieses Polynoms wählen. Dazu zeigen wir zunächst, dass es ausreichend viele davon gibt.

Lemma 3.22 *(***Nullstellen orthogonaler Polynome***)*
v_n *hat in* (a, b) *genau* n *einfache Nullstellen.*

Beweis: Seien $x_0, \dots, x_{m-1}$ die Argumente, an denen v_n in (a, b) sein Vorzeichen ändert (insbesondere sind dann die x_j Nullstellen von v_n). Da v_n höchstens n Nullstellen haben kann, gilt
$$m \leq n.$$
Angenommen, $m < n$. Sei
$$q(x) := \prod_{j=0}^{m-1} (x - x_j),\ q \in \mathcal{P}_m.$$
Dann wechselt q sein Vorzeichen an denselben Stellen wie v_n, also ändert qv_n sein Vorzeichen nicht. Da $m < n$ ist $q \in P_{n-1}$, und es gilt
$$0 = (q, v_n) = \int_a^b w(x)q(x)v_n(x)\text{dx}.$$
Da $q(x)v_n(x)$ sein Vorzeichen nicht ändert und w positiv ist, folgt
$$q(x)v_n(x) = 0\, \forall x \in (a, b).$$
Also ist $q = 0$ im Widerspruch zur Definition von q. Also war die Annahme falsch, und es gilt
$$m = n.$$

□

Damit haben wir alles zusammen.

Satz 3.23 *(**Formel von Gauss, Gauss–Quadraturformel**)*
Seien $x_0, \ldots, x_n$ die nach 3.22 paarweise verschiedenen Nullstellen von v_{n+1}. Dann ist die Quadraturformel I_n mit den Stützstellen $x_0, \ldots, x_n$, die exakt ist von der Ordnung n, sogar exakt von der Ordnung $2n+1$.

Beweis: Sei $p \in \mathcal{P}_{2n+1}$. Dann ist mit Polynomdivision

$$p = qv_{n+1} + r,\ r \in \mathcal{P}_n,\ q \in \mathcal{P}_n.$$

Da I_n an den Nullstellen von v_{n+1} auswertet, gilt

$$0 = I_n(v_{n+1}) = I_n(qv_{n+1})$$

und

$$\begin{aligned}
\int_a^b wp\,\mathrm{dx} &= \underbrace{\int_a^b wqv_{n+1}\,\mathrm{dx}}_{=0,\ q \perp v_{n+1}} + \underbrace{\int_a^b wr\,\mathrm{dx}}_{=I_n(r),\ I_n \text{ exakt}} \\
&= 0 + I_n(r) \\
&= I_n(qv_{n+1}) + I_n(r) \\
&= I_n(qv_{n+1} + r) \\
&= I_n(p).
\end{aligned}$$

□

Satz 3.24 *(**Gewichte der Gaussformeln**)*
Die Gewichte A_j der Gaussformeln sind positiv.

Beweis: Es gilt

$$A_j = \int_a^b w(x) \prod_{i \neq j} \left(\frac{x - x_i}{x_j - x_i} \right)^2 \mathrm{dx}$$

(Übungen). □

Satz 3.25 *(**Optimalität der Gaussformeln**)*
Keine Quadraturformel mit $(n+1)$ Auswertungen an den Stützstellen $x_0, \ldots, x_n$ ist exakt von der Ordnung $2n+2$.

Beweis: Setze

$$p(x) := \prod_{k=0}^{n} (x - x_k)^2 \in \mathcal{P}_{2n+2}.$$

Dann ist

$$I_n(p) = 0,$$

denn I_n wertet an den Nullstellen von p aus. Es gilt

$$p(x) \geq 0 \, \forall x \in [a, b],$$

aber p ist nicht das Nullpolynom, also

$$I(p) > 0 = I_n(p).$$

I_n ist also nicht exakt für dieses $p \in P_{2n+2}$. □

Beispiel 3.26 *Gauss–Quadratur*
Wir betrachten $w = 1$ auf dem Intervall $[-1, 1]$. Die zugehörigen orthogonalen Polynome sind die Legendre–Polynome. Wir erhalten

n	$P(n,x)$	*Nullstellen*		
0	1			
1	x	0		
2	$\frac{1}{2}(3x^2-1)$	$\frac{\sqrt{3}}{3}$	$-\frac{\sqrt{3}}{3}$	
3	$\frac{1}{2}x(5x^2-3)$	$-\frac{1}{5}\sqrt{15}$	0	$\frac{1}{5}\sqrt{15}$
4	$\frac{35}{8}x^4 - \frac{15}{4}x^2 + 3/8$	$\pm 1/35\sqrt{525 \pm 70\sqrt{30}}$		

3.6 Vergleich der Quadraturformeln und Stabilität

In den folgenden Abbildungen 3.1 bis 3.3 werden die behandelten Quadraturformeln anhand typischer Beispiele verglichen. Wir halten jeweils die Anzahl N der Auswertungen fest und vergleichen die Genauigkeit, die die Formeln mit diesen Auswertungen liefern. Dargestellt ist jeweils der Fehler der jeweiligen Formel als $\log$–$\log$–Plot. Wegen der Beziehung

$$\log(h^k) = k \log(h)$$

wird dabei eine polynomiale Abhängigkeit des Fehlers von der Auflösung $h = 1/N$ als Gerade dargestellt. Benutzt wurden die Formeln der Länge 2^n, $n = 0 \ldots 7$.

Die Exponentialfunktion ist ihre eigene Ableitung, wir erwarten also keinen Anstieg des Fehlers bei der Polynominterpolation. Alle Formeln, die auf Polynominterpolation mit hohem Grad basieren, sollten hier sehr gut funktionieren. Das Beispiel von

Runge hat stark wachsende Ableitungen. Wir vermuten, dass alle Quadraturformeln hohen Grades problematisch sind. Für periodische Funktionen sollte die Trapezregel bereits optimal sein.

Dies wird in den Beispielen deutlich.

Zur Exponentialfunktion: Zusammengesetzte Trapez– und Simpsonregel liefern genau einen Fehler der Ordnung 2 bzw. 4, erkennbar an der Geraden im logarithmischen Plot. Die Gauss–Regel liefert das beste Ergebnis. Das Romberg–Verfahren liegt dazwischen. Die Newton–Cotes–Formeln sind bis $N = 16$ konkurrenzfähig, aber dann bricht ihre Genauigkeit plötzlich stark ein, der Fehler bei $N = 128$ beträgt 10^{20}. Dass die Genauigkeit einbricht, ist nach unserer Analyse überraschend. Nach Satz 3.11 müsste der Fehler mit n^{-n} gegen 0 gehen, weil die Ableitungen der Exponentialfunktion mit dem Grad der Ableitung nicht anwachsen. Die Erklärung dafür werden wir gleich suchen.

Zum Beispiel von Runge: Alle Formeln haben große Probleme, Gauss schlägt sich noch am Besten. Die auf einer Unterteilung des Intervalls basierenden zusammengesetzten Formeln sind nicht viel schlechter als im Exponentialfall. Auch das Romberg–Verfahren liefert hier schlechte Ergebnisse. Dies ist zu erwarten, denn auch das Romberg–Verfahren basiert auf der Entwicklung in höhere Ableitungen.

Zur periodischen Funktion: Wie erwartet, sind die zusammengesetzten Formeln optimal.

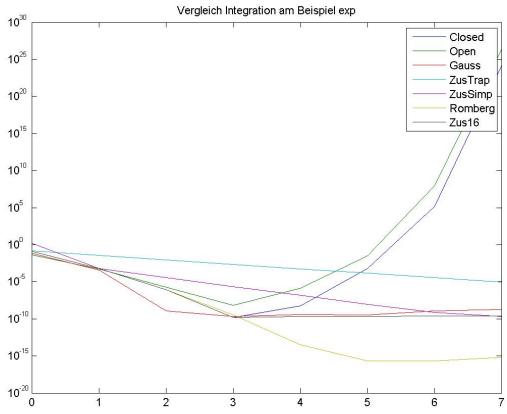


Abbildung 3.1: Vergleich von Quadraturformeln für die Exponentialfunktion

Klick für Bild integexp
Klick für Matlab Figure integexp

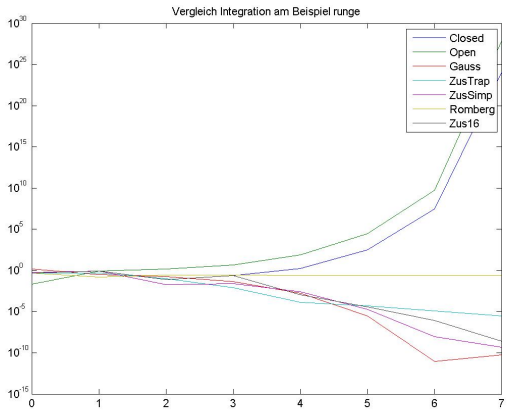


Abbildung 3.2: Vergleich von Quadraturformeln für das Rungebeispiel

Klick für Bild integrunge
Klick für Matlab Figure integrunge

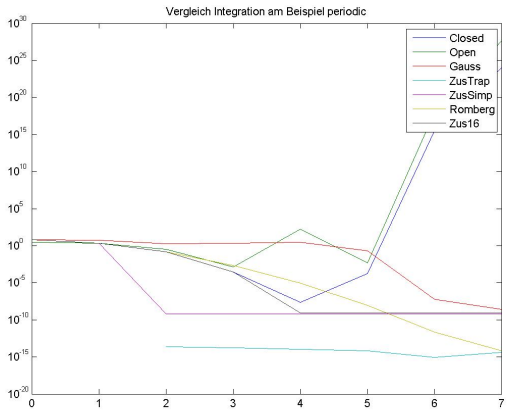


Abbildung 3.3: Vergleich von Quadraturformeln für eine periodische Funktion

Klick für Bild integperiodic
Klick für Matlab Figure integperiodic

```
function plotcompare
doit(@exp,0,1,exp(1)-1,'exp');
%waitforbuttonpress;
doit(@runge,-1,1,2/5*atan(5),'runge');
%waitforbuttonpress;
doit(@periodic,0,2*pi,2*pi,'periodic');
```

Listing 3.5: Vergleich von Quadraturformeln (integration/plotcompare.m)

Klicken für den Quellcode von integration/plotcompare.m

Wir müssen noch erklären, warum die Newton–Cotes–Algorithmen für $\exp$ und große Polynomgrade sagenhafte Fehler liefern, obwohl die Polynominterpolation vernünftig ist. Hierzu benötigen wir den Begriff der Stabilität.

Definition 3.27 (Stabilität von numerischen Algorithmen)
Ein Algorithmus heißt stabil, wenn bei fehlerhaften Eingangsdaten der Fehler des Algorithmus in der Größenordnung des Fehlers bei analytischer Auswertung (unvermeidlicher Fehler) liegt.

Satz 3.28 (Stabilität der Quadratur)
Quadraturformeln mit positiven Gewichten sind stabil.

Beweis: Falls statt f nur eine Näherung $\tilde{f}$ bekannt ist mit

$$|f(x) - \tilde{f}(x)| \leq \epsilon |f(x)| \, \forall x \in [a, b],$$

so gilt für das Integral

$$|I(f) - I(\tilde{f})| \leq \int_a^b |(f - \tilde{f})(x)| \mathsf{dx} \leq \epsilon I(|f|)$$

und die Fehlerschranke ist optimal in dem Sinne, dass es zu jedem f ein $\tilde{f}$ gibt mit relativem Fehler ϵ, das diese Fehlerschranke annimmt. Die rechte Seite ist der unvermeidbare Fehler.
Sei nun I_n eine Quadraturformel. Es gilt

$$|I(f) - I_n(\tilde{f})| \leq |I(f) - I_n(f)| + |I_n(f) - I_n(\widetilde{f})| \leq |I(f) - I_n(f)| + \epsilon \sum_k |A_k||f(x_k)|.$$

Der Fehler, den wir insgesamt machen, setzt sich also zusammen aus dem Fehler für die Integrationsformel (der klein wird, wie wir bewiesen haben), und dem Fehler bei der Auswertung unserer Formel. Für positive Gewichte gilt

$$\epsilon \sum_k |A_k||f(x_k)| = \epsilon I_n(|f|) \sim \epsilon I(|f|).$$

Wenn der Integrationsfehler klein ist, liegt also der auftretende Gesamtfehler in der Größenordnung der analytischen Fehlerschranke.
Für negative Gewichte kann die rechte Seite aber sehr viel größer sein. Tatsächlich

werden ab $n = 8$ die Gewichte negativ (die Gewichte liegen um $\pm 10^4$ für $n = 16$!), und das sorgt für den großen Fehler. □

Bemerkung: Die Stabilität eines Algorithmus hat nichts damit zu tun, ob der Fehler im Resultat klein ist, er sagt nur aus, in welcher Relation er zum unvermeidlichen Fehler steht. Für die Integration etwa gilt, falls f sein Vorzeichen im Intervall $[a, b]$ ändert:

$$|I(f) - I(\tilde{f})|/|I(f)| >> |I(f) - I(\tilde{f})|/I(|f|),$$

d.h., für den relativen Fehler im Ergebnis können wir in diesem Fall keine Abschätzung angeben. Trotzdem ist ein Algorithmus mit positiven Gewichten stabil.

Bemerkung: Stabilität ist einer der wichtigsten Begriffe der Numerik (und insbesondere dieser Vorlesung). Sie sagt aus, dass ein Algorithmus, von dem wir analytisch gezeigt haben, dass er sinnvoll ist, tatsächlich auch in der Anwendung einsetzbar ist. Dass dies keineswegs so sein muss, haben wir hier gesehen. Instabile Algorithmen produzieren typischerweise exorbitante Fehler, so dass sich die Frage nach der genauen Definition der Größenordnung in der Definition der Stabilität gar nicht stellt.

Insgesamt halten wir fest: Bei Funktionen, die ihr Vorzeichen nicht wechseln, liegt der unvermeidbare Fehler liegt in der Größenordnung des Eingangsfehlers. Die Integrationsaufgabe ist **gut gestellt** oder **gut konditioniert**. Die zusammengesetzten Integrationsregeln kleiner Ordnung liefern stabile Algorithmen. Gauss ist eine gute Alternative, die Schrittweite lässt sich aber nicht ändern, ohne alle Zwischenergebnisse zu verlieren. Newton–Cotes–Formln hoher Ordnung sind unbrauchbar, weil die Gewichte negativ werden. Richardson bzw. Romberg liefert effiziente Formeln.

Für die Differentiation sieht diese Betrachtung leider sehr viel schlechter aus.

3.7 Numerische Differentiation

Sei $f \in C^m$. Wir suchen nun eine Linearkombination von Auswertungen

$$f(hx_0), \ldots, f(hx_n),$$

die die Ableitung

$$f^{(l)}(0), l \leq m,$$

möglichst gut approximieren. Es gilt der Satz:

Satz 3.29 *(**Differentiationsformeln beliebiger Ordnung**)*
Sei $f \in C^{(n+1)}(\mathbb{R})$, $x_0, \ldots, x_n$ paarweise verschieden, $l \leq n$ fest. Dann gibt es Zahlen α_k, $k = 0 \ldots n$, so dass

$$\sum_{k=0}^{n} \alpha_k f(hx_k) = h^l f^{(l)}(0) + O(h^{n+1})$$

und damit

$$\frac{\sum_{k=0}^{n} \alpha_k f(hx_k)}{h^l} = f^{(l)}(0) + O(h^{n+1-l}).$$

Beweis: Entwickle die linke Seite in ihre Taylorentwicklung um 0. Dann gilt

$$\sum_{k=0}^{n} \alpha_k f(hx_k) = \sum_{j=0}^{n} \frac{f^{(j)}(0)}{j!} h^j \underbrace{\sum_{k=0}^{n} \alpha_k x_k^j}_{} + O(h^{n+1}).$$

Wähle die α_k so, dass

$$\sum_{k=0}^{n} x_k^j \alpha_k = \delta_{jl} \cdot l!.$$

Dies geht, denn die Koeffizienten bilden eine Vandermonde–Matrix. □

Beispiel 3.30 (Differentiationsformeln)

Vorwärts–Differenzenquotient ($f \in C^2$):

$$D_h^+(f)(x) = \frac{f(x+h) - f(x)}{h} = f'(x) + O(h)$$

Rückwärts–Differenzenquotient ($f \in C^2$):

$$D_h^-(f)(x) = \frac{f(x) - f(x-h)}{h} = f'(x) + O(h)$$

Zentraler Differenzenquotient ($f \in C^3$):

$$D_h(f)(x) = \frac{f(x+h/2) - f(x-h/2)}{h} = f'(x) + O(h^2)$$

Zentraler Differenzenquotient für die zweite Ableitung ($f \in C^4$):

$$D_h^2(f)(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + O(h^2)$$

Begründung z.B. für den letzten Fall:

$$\begin{aligned} D_h^2(f)(x) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \\ &= \frac{1}{h^2}\big(f(x) + hf'(x) + h^2 f''(x)/2 + h^3 f^{(3)}/6 + O(h^4) \\ &\quad - 2f(x) \\ &\quad + f(x) - hf'(x) + h^2 f''(x)/2 - h^3 f^{(3)}(x)/6 + O(h^4)\big) \end{aligned}$$

Das Zauberwort lautet also: Taylorentwicklung – wann immer wir wissen wollen, wie sich eine Formel für kleine h verhält, entwickeln wir meist alles in Taylorreihen und vergleichen die Terme. Dies wird die zentrale Idee bei der Behandlung gewöhnlicher Differentialgleichungen sein.

Natürlich lässt sich auch hier wieder die Richardson–Extrapolation einsetzen, wie wir in den Beispielen in 2.25 bereits gesehen haben.

Vorlesungsnotiz: 9.5.2011

Vorlesungsnotiz: 7.5.2013

3.8 Stabilität der numerischen Differentiation

Sei nun wieder statt der Funktion f nur eine Näherung $\tilde{f}$ bekannt mit

$$|f(x) - \tilde{f}(x)| \leq \epsilon |f(x)|.$$

Eine Stabilitätsbetrachtung wie bei der Integration ist hier gar nicht möglich: Egal, wie klein ϵ ist, $\tilde{f}$ wird mit dieser Bedingung im allgemeinen nicht einmal stetig, und erst recht nicht differenzierbar sein. Im Sinne unserer Definition oben ist der unvermeidbare Fehler sofort Unendlich.

Wir betrachten trotzdem die numerische Berechnung der ersten Ableitung an der Stelle 0, sei also

$$\Delta_h(f) = \frac{1}{h}\sum_{k=0}^{n} \alpha_k f(hx_k) = f'(0) + O(h^m)$$

eine Differentiationsformel der Ordnung m. Für $\Delta_h(\tilde{f})$ gilt

$$\begin{aligned} |\Delta_h(\tilde{f}) - f'(0)| &\leq |\Delta_h(\tilde{f}) - \Delta_h(f)| + |\Delta_h(f) - f'(0)| \\ &\leq \frac{\epsilon \sum_{k=0}^{n} |\alpha_k||f(hx_k)|}{h} + O(h^m) \end{aligned}$$

Wir erhalten damit: Ist h zu klein, so ist der erste Summand groß. Ist h zu groß, so ist der zweite Summand groß. Wir suchen also ein optimales h_0, so dass die Summe möglichst klein wird. Ein Ansatz könnte sein, die beiden Summanden gleich groß zu wählen. In diesem Fall wäre $h \sim \epsilon^{1/(m+1)}$, und der Gesamtfehler in der Ordnung $O(\epsilon^{1-1/(m+1)})$. Der Fehler ist dann zwar größer as ϵ, aber für hohe Ordnungen nicht viel größer. Insbesondere geht für $\epsilon \mapsto 0$ der Fehler tatsächlich gegen 0. Ein Algorithmus, der genau dies garantiert (falls die Datenfehler gegen 0 gehen, geht auch der Fehler im Ergebnis gegen 0), heißt Regularisierung des Problems.

Wir haben damit die paradoxe Situation, dass ein eigentlich analytisch gar nicht lösbares Problem (die Ableitungen von f und $\tilde{f}$ können sich beliebig voneinander unterscheiden, egal wie klein ϵ ist, im allgemeinen existiert $\tilde{f}'$ nicht einmal) eine vernünftige numerische Lösung besitzt. Dies führt in die Theorie der inversen und schlecht gestellten Probleme.

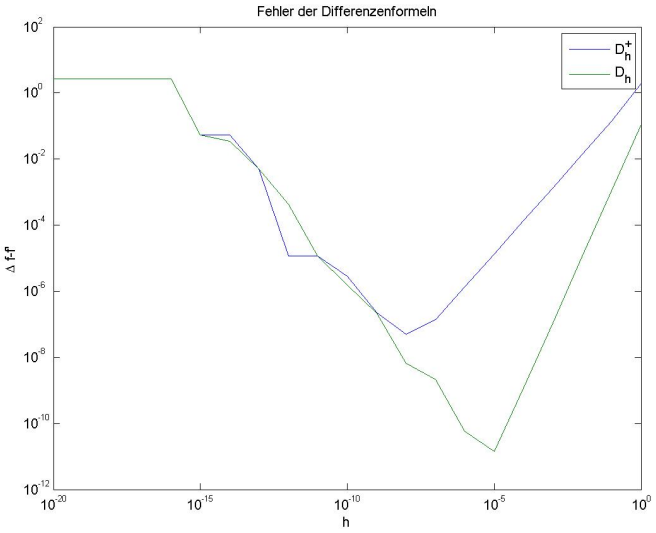


Abbildung 3.4: Fehler der Differenzenformeln gegen die Schrittweite, halblogarithmisch

Klick für Bild difffehler

```
function [ output_args ] = diffdemo( f,x,trueval )
%DIFFDEMO
close all;
if (nargin==0)
    f=@exp;
    x=1;
```

Listing 3.6: Fehler der Differenzenformeln (integration/diffdemo.m)

Klicken für den Quellcode von integration/diffdemo.m

Kapitel 4

Numerische Behandlung gewöhnlicher Differentialgleichungen

4.1 Einleitung und Beispiele

In diesem Kapitel suchen wir Funktionen

$$y : I \mapsto \mathbb{R}^n,\ I \subset \mathbb{R},$$

bei denen eine vorgegebene Gleichung

$$F(t, y(t), y'(t), \ldots, y^{(m)}(t)) = 0 \,\forall\, t \in I$$

für y und seine Ableitungen erfüllt ist. Die Gleichung nennen wir Differentialgleichung. y nennen wir die Lösung der Differentialgleichung. Wir beschränken uns dabei auf Funktionen, die auf dem eindimensionalen Raum $\mathbb{R}$ (gewöhnliche Differentialgleichungen, GDGL) definiert sind. Für den Zielraum lassen wir ausdrücklich zu, dass er mehrdimensional ist, die Ableitungen sind dann entsprechend Vektoren. I ist im Allgemeinen ein Intervall.

Ist I Teilmenge des $\mathbb{R}^N$, $N > 1$, so sind die Ableitungen partiell, und wir sprechen von partiellen Differentialgleichungen. Die Theorie und Numerik der partiellen Differentialgleichungen ist um Größenordnungen komplexer und Thema eigener Vorlesungen.

Insbesondere ist die Lösung einer Differentialgleichung immer eine Funktion. Häufig beschreibt die Funktion das Verhalten einer Anwendungsgröße in der Zeit, deshalb verwenden wir meist t als Parameter dieser Funktion. Formal halten wir zunächst fest:

Definition 4.1 (Gewöhnliche Differentialgleichungen)

Sei $m \geq 1$, $I \subset \mathbb{R}$ ein Intervall,

$$F : I \times (\mathbb{R}^n)^{m+1} \mapsto \mathbb{R}^n$$

fest. Dann heißt die Gleichung

$$F(t, y(t), y'(t), \ldots, y^{(m)}(t)) = 0 \, \forall \, t \in I \tag{4.1}$$

implizite gewöhnliche Differentialgleichung für die Funktion

$$y : I \mapsto \mathbb{R}^n, \, y \in C^{(m)}(I),$$

also eine m–mal stetig differenzierbare Funktion y auf dem Intervall I. Falls die Differentialgleichung nach der höchsten Ableitung aufgelöst ist, gilt also

$$y^{(m)}(t) = f(t, y(t), y'(t), \ldots, y^{(m-1)}(t)) \tag{4.2}$$

für eine Funktion

$$f : I \times (\mathbb{R}^n)^m \mapsto \mathbb{R}^n,$$

so nennen wir die gewöhnliche Differentialgeichung explizit.

Falls $n = 1$, so nennen wir die Gleichung skalar.

Falls $n > 1$, so heißen 4.1 bzw. 4.2 Systeme von gewöhnlichen Differentialgleichungen.

m heißt Ordnung der Differentialgleichung.

Bemerkung:

1. Nach dem Satz über implizite Funktionen kann man eine implizite gewöhnliche Differentialgleichung lokal in eine explizite umwandeln, wenn F stetig differenzierbar ist und die Ableitung von F nach $y^{(m)}$ invertierbar ist (im Fall von Systemen ist die Ableitung die Jakobimatrix). Wir betrachten daher immer gleich explizite Differentialgleichungen.
2. Wir werden sehen, dass sich Gleichungen höherer Ordnung in Systeme von Gleichungen der Ordnung 1 umwandeln lassen (4.4 und 4.7). Wir beschränken uns daher grundsätzlich auf die Betrachtung von Differentialgleichungen der Form
$$y'(t) = f(t, y(t)) \, \forall \, t \in I.$$
3. f und F können auch auf Teilmengen der zugrundeliegenden Räume definiert sein.

Gewöhnliche und partielle Differentialgleichungen sind **das** Handwerkszeug des Mathematikers zur Modellierung von Anwendungen. Dabei beschreibt die gesuchte Funktion y das gesuchte Verhalten etwa einer physikalischen Größe oder die Ausbreitungsrate von Epidemien. Beliebig viele Beispiele finden sich in der Literatur, z.B. im Buch Hanke-Bourgeois [2006] (Kapitel 11) oder Walter [2000], und natürlich in der Vorlesung Modellierung. Eine bekannte nicht–physikalische Anwendung sind die Black–Scholes–Gleichungen Black and Scholes [1973], die den Wert einer Wertpapieroption über die Zeit modellieren. Eine schöne Übersicht über biologische Anwendungen geben Prüß et al. [2008] und Murray [2002].

Gewöhnliche Differentialgleichungen sind auch ein wichtiges Werkzeug zur Lösung von Differentialgleichungen in mehreren Variablen (partielle Differentialgleichungen), siehe die Beispiele in den Übungen zur Trennung der Variablen und zu Charakteristiken.

In Anwendungsfällen können die Differentialgleichungen meist nicht analytisch gelöst werden, man ist also auf die numerische Lösung angewiesen, deshalb spielt die Lösung gewöhnlicher Differentialgleichungen in der Numerik eine sehr große Rolle.

Unsere Referenz für die theoretischen Grundlagen ist das klassische Buch Walter [2000]. Tatsächlich gibt es eine sehr große Auswahl an Lehrbüchern und Skripten zum Thema Differentialgleichungen, auf die wir zum Hintergrund verweisen.

Wir beginnen zunächst mit einigen (trivialen) Beispielen und wiederholen wenige theoretische Grundbegriffe.

Beispiel 4.2 *(Nuklearer Zerfall –* **Anfangswertaufgabe***)*

Sei $N(t)$ die Anzahl der Teilchen eines radioaktiven Stoffes, die zum Zeitpunkt t existieren. Es sei bekannt, dass zum Zeitpunkt $t = 0$ die Anzahl der Teilchen N_0 beträgt, also

$$N(0) = N_0.$$

Zu bestimmen ist die Funktion $N(t)$.

Die Anzahl der Teilchen, die in einem infinitesimal kleinen Zeitintervall dt zum Zeitpunkt t zerfallen, ist

$$dN = -qN\,dt \tag{4.3}$$

mit einem Proportionalitätsfaktor q, der von der Halbwertszeit abhängt. Dies gilt natürlich nur für unendlich kleine Intervalle. Wollen wir die zwischen den Zeitpunk-

ten 0 und $t > 0$ zerfallenen Teilchen schätzen, so müssen wir $[0, t]$ zu Zeitpunkten t_k aufteilen und erhalten

$$N(t) - N(0) \sim \sum_{k=1}^{N} (t_k - t_{k-1})(-qN(t_k)).$$

Im Limes unendlich feiner Zerlegungen wird diese Abschätzung korrekt, andererseits geht die rechte Seite gegen das Riemann–Integral von qN, es gilt also

$$N(t) - N(0) = -\int_0^t qN(s)ds.$$

Unter der Annahme, dass N stetig ist, können wir die rechte Seite differenzieren, und wir erhalten für $t \geq 0$ das (lineare) Anfangswertproblem

$$N'(t) = -qN(t),\ N(0) = N_0$$

mit der Lösung

$$N(t) = N_0 \exp(-qt).$$

Wir halten fest:

1. *Entscheidend für die Lösung war, dass wir die Anzahl der Teilchen zu einem Zeitpunkt kennen müssen. Daann können wir mit Hilfe der Differentialgleichung angeben, wie sich die Anzahl zu jedem Zeitpunkt ändert und daraus die Funktion berechnen. Aufgaben dieser Form heißen Anfangswertaufgaben.*
2. *Gewöhnliche Differentialgleichungen besitzen eine äquivalente Form als Integralgleichung für stetige Funktionen.*
3. *Rein formal kann man die Differentialgleichung auch direkt aus 4.3 gewinnen durch Teilen durch dt:*
$$N' = \frac{dN}{dt} = -qN$$
. Die nachfolgende Rechnung zeigt, dass dieser Ansatz korrekt ist.

Physikalisch darf man hier natürlich nicht zu genau hinschauen, die tatsächliche Modellierung ist umfangreicher.

Beispiel 4.3 *(Räuber–Beute–Modelle — **System von Differentialgleichungen**)*

Seien $F(t)$ und $H(t)$ die Anzahl der Füchse bzw. Hasen in einer Population zum Zeitpunkt t. Ohne Füchse entwickeln sich die Hasen ungestört, die Anzahl der Nachkommen ist proportional zur Zahl der Hasen. Andererseits verringert die Zahl der

Füchse auch die Zahl der Hasen proportional. Entsprechendes gilt für die Zahl der Füchse.

Insgesamt ergibt sich zum Zeitpunkt t eine Dynamik–Gleichung der Form

$$\begin{aligned} dH &= +a\,dt\,H - b\,dt\,F \cdot H \\ dF &= -c\,dt\,F + d\,dt\,F \cdot H \end{aligned}$$

oder

$$\begin{aligned} H'(t) &= +aH(t) - bF(t)H(t) \\ F'(t) &= -cF(t) + dF(t)H(t) \end{aligned}$$

mit Anfangswerten

$$H(0) = H_0,\ F(0) = F_0$$

für die Populationszahlen zum Zeitpunkt $t_0 = 0$ (Lotka–Volterra–Modell). Für eine exaktere Behandlung siehe z.B. Murray [2002].

Führen wir die Funktion

$$y : \mathbb{R}^+ \mapsto \mathbb{R}^2, y(t) = \begin{pmatrix} H(t) \\ F(t) \end{pmatrix}$$

ein, so gilt

$$y'(t) = f(t, y(t)),\ y(0) = \begin{pmatrix} H_0 \\ F_0 \end{pmatrix}$$

mit der Funktion

$$f : \mathbb{R}^+ \times \mathbb{R} \mapsto \mathbb{R}^2,\ f(t, y) := \begin{pmatrix} ay_1 - by_1y_2 \\ -cy_2 + dy_1y_2 \end{pmatrix}.$$

Wir erhalten also wieder eine gewöhnliche Differentialgleichung der Form

$$y'(t) = f(t, y(t)),$$

diesmal mit einer Funktion von $\mathbb{R}^+$ nach $\mathbb{R}^2$ mit einem Anfangswert bei $t = 0$. Dies ist eine Anfangswertaufgabe für ein System von gewöhnlichen Differentialgleichungen.

Beispiel 4.4 *(Beschleunigung —* **Differentialgleichung höherer Ordnung***)*

Ein Fahrzeug wird mit der Beschleunigung $a(t)$ angetrieben. Sei $x(t)$ die zurückgelegte Strecke. Dann gilt mit der Geschwindigkeit $v(t)$ zum Zeitpunkt t

$$\begin{aligned} dx &= dt\,v \\ dv &= dt\,a \end{aligned}$$

und wir erhalten das System

$$\begin{aligned} x'(t) &= v(t) \\ v'(t) &= a(t) \\ v(0) &= v_0 \\ x(0) &= x_0. \end{aligned}$$

Die Lösung ist gegeben durch

$$v(t) = \int_0^t a(s)ds + v_0,\ x(t) = \int_0^t v(r)dr + x_0.$$

Das System können wir durch Einsetzen von v auch schreiben als

$$x''(t) = a(t),\ x(0) = x_0,\ x'(0) = v_0,$$

also als Differentialgleichung zweiter Ordnung. Die Einführung der Geschwindigkeit als zusätzliche Funktion verwandelt also diese Gleichung zweiter Ordnung in ein System von Differentialgleichungen erster Ordnung.

Diese Beispiele wollen wir noch rasch formalisieren:

Definition 4.5 *(**Anfangswertprobleme für gewöhnliche Differentialgleichungen)***

Sei $I = [a, b]$ mit $a < b$,

$$f : I \times \mathbb{R}^n \mapsto \mathbb{R}^n$$

wie in 4.1 für die Ordnung 1. Weiter seien $t_0 \in [a, b]$, $y_0 \in \mathbb{R}^n$. Dann heißt die Aufgabe:

Bestimme eine stetig differenzierbare Funktion

$$y : I \mapsto \mathbb{R}^n$$

mit

$$y'(t) = f(t, y(t))\ \forall t \in [a, b],\ y(t_0) = y_0$$

Anfangswertaufgabe erster Ordnung für die Differentialgleichung

$$y'(t) = f(t, y(t)).$$

y heißt Lösung der Anfangswertaufgabe.

Im Allgemeinen gilt $t_0 = a$ (und deshalb heißt es *Anfangs*wertaufgabe).

Lemma 4.6 *Sei f stetig. Eine stetige Funktion*

$$y : I \mapsto \mathbb{R}^n$$

ist genau dann Lösung des Anfangswertproblems 4.5, wenn

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds. \tag{4.4}$$

Beweis: Mit dem Hauptsatz der Differential– und Integralrechnung. □

Lemma 4.7 *(***Umwandlung von Gleichungen höherer Ordnung***)*

Differentialgleichungen höherer Ordnung können in Systeme von Differentialgleichungen niedrigerer Ordnung umgewandelt werden.

Beweis: Zum Beispiel für eine skalare Differentialgleichung $n.$ Ordnung, $n > 1$,

$$y^{(n)}(t) = f(t, y(t), y'(t), \ldots, y^{n-1}(t)).$$

Sei

$$\widetilde{y} : I \mapsto \mathbb{R}^n,$$

und

$$\widetilde{f} : I \times \mathbb{R}^n,\ \widetilde{f}_k(t, y) := \begin{cases} y_{k+1}, & k < n \\ f(t, y_1, \ldots, y_n), & \text{sonst} \end{cases}.$$

Dann ist $\widetilde{y}_1(t)$ genau dann Lösung der Differentialgleichung $n.$ Ordnung, wenn $\widetilde{y}(t)$ das System

$$\widetilde{y}'(t) = \widetilde{f}(t, \widetilde{y}(t))$$

erfüllt.

Beweis: Es gilt

$$\widetilde{y}_k = \widetilde{y}_1^{(k-1)}.$$

□

Beispiel 4.8 (Existenz und Eindeutigkeit)

Selbst für sehr einfache Beispiele sind Existenz und Eindeutigkeit der Lösung von Anfangswertaufgaben nicht notwendig gesichert. Wir beginnen direkt mit zwei warnenden Beispielen.

1.

$$y'(t) = 1 + y(t)^2, y(0) = 0.$$

Lösung ist

$$y(t) = \tan t.$$

Obwohl f harmlos (ein Polynom) ist, ist y nicht auf ganz $\mathbb{R}$ stetig fortsetzbar. Die Anfangswertaufgabe hat nur eine Lösung in einem Intervall um den Anfangspunkt.

2.

$$y'(t) = y(t)^{1/3}, y(0) = 0.$$

Lösungen sind $y(t) = (\frac{2}{3}t)^{3/2}$ und $y = 0$, die Lösung ist also nicht eindeutig.

Bemerkung: $y^{1/3}$ ist nicht differenzierbar bei 0.

4.2 Einige klassische Typen von Differentialgleichungen

Beispiel 4.9 (Elementare Differentialgleichung)

Differentialgleichungen der Form

$$y'(t) = f(t)$$

heißen Elementare Differentialgleichungen. Die Lösung der zugehörigen Anfangswertaufgabe mit

$$y(a) = y_0$$

ist

$$y(t) = y_0 + \int_a^t y'(s)ds = y_0 + \int_a^t f(s)ds.$$

Beispiel 4.10 (autonome Differentialgleichung)

Skalare Differentialgleichungen der Form

$$y'(t) = f(y(t))$$

heißen autonome Differentialgleichungen. Der folgende Ansatz führt häufig zum Ziel: Sei z die Umkehrfunktion von y, also $z(y(t)) = t$. Dann ist

$$1 = z'(y(t))y'(t) = z'(y(t))f(y(t)) \,\forall\, t \in I.$$

Es gilt also

$$z'(y) = \frac{1}{f(y)} \,\forall y \text{ im Bild von } y(t).$$

Dies ist eine elementare Differentialgleichung für z, die durch Integration gelöst werden kann. y ist die Umkehrfunktion von z.

Zum Beispiel für

$$y'(t) = 1 + y(t)^2 :$$

$$z' = \frac{1}{1+y^2} \Rightarrow z(y) = \arctan(y) + C$$

und damit

$$t = z(y(t)) = \arctan(y(t)) + C \Rightarrow y(t) = \tan(t - C).$$

Beispiel 4.11 (lineare Differentialgleichung in einer Variablen)

Sei

$$y'(t) = a(t)y(t),\ y(0) = C.$$

Dann ist

$$a(t) = y'(t)/y(t) = \log(y(t))'.$$

Dies ist wieder eine elementare Differentialgleichung, und wir erhalten

$$y(t) = C \exp(\int_0^t a(s)ds).$$

Die affin–lineare Gleichung

$$y'(t) = a(t)y(t) + b(t)$$

löst man durch Variation der Konstanten (Übungen).

4.3 Grafische Lösung

Die meisten klassische Differentialgleichungen besitzen keine elementare Lösung, z.b. die Besselsche Differentialgleichung

$$t^2y''(t) + ty'(t) + (t^2 - n^2)y(t) = 0$$

oder die Riccati–Differentialgleichung

$$y'(t) = y(t)^2 + 1 - t^2.$$

Es kommen daher nur näherungsweise Lösungsmethoden in Frage. Eine einfache Möglichkeit ist die grafische Lösung für eindimensionale Anfangswertaufgaben.

Durch die Differentialgleichung ist in jedem Punkt (t, y) des $\mathbb{R}^2$ die Steigung der Tangente der Lösung der Differentialgleichung durch diesen Punkt festgelegt. Die Steigung wird für einige ausgewählte Punkte als Vektorfeld in ein Koordinatensystem eingetragen. Anhand der Vektoren wird die Lösung der Differentialgleichung, ausgehend von der Anfangsbedingung, im Koordinatensystem geschätzt. Dies ist bereits auch die Grundidee der einfachen numerischen Verfahren.

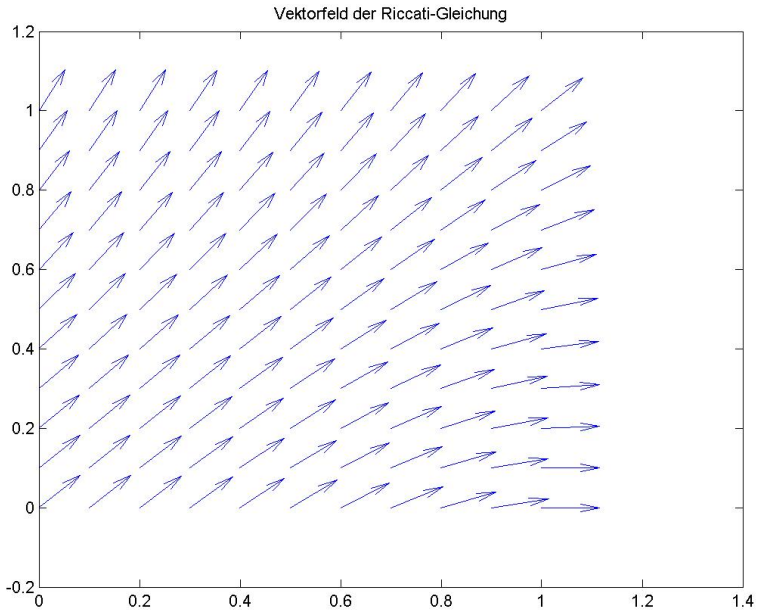


Abbildung 4.1: Vektorfeld der Riccati–Gleichung $y' = 1 + y^2 - t^2$

Klick für Bild vectorfieldriccatti
Klick für Matlab Figure vectorfieldriccatti

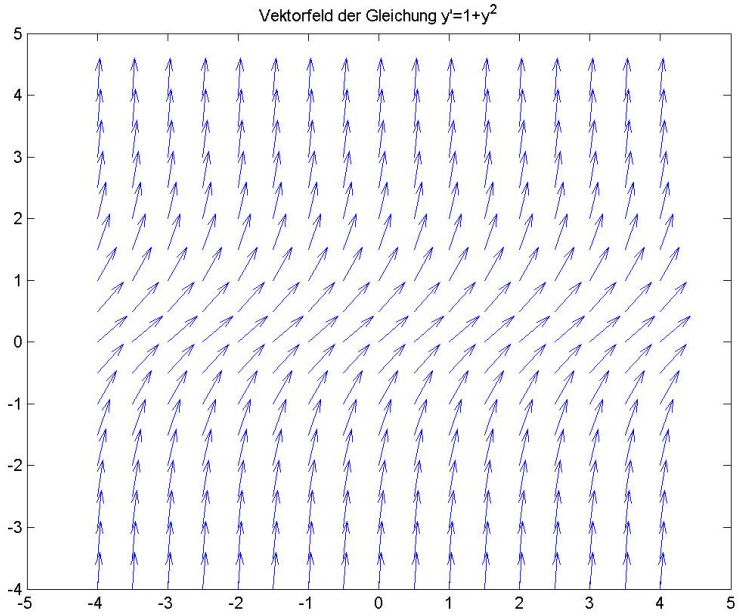


Abbildung 4.2: Vektorfeld der autonomen Gleichung $y' = 1 + y^2$

Klick für Bild vectorfieldtan
Klick für Matlab Figure vectorfieldtan

```
function [ output_args ] = vectorfield( input_args )
%VECTORFIELD
[x,y]=meshgrid(0:0.1:1,0:0.1:1);
z=y.*y+1-x.*x;
l=sqrt(1+z.*z);
B=ones(size(z))./l;
```

Listing 4.1: Vektorfelder von GDGL (gdgl/vectorfield.m)

Klicken für den Quellcode von gdgl/vectorfield.m

Wir werden dieses Verfahren gleich noch formalisieren und so das Eulersche Polygonzugverfahren erhalten. **Vorlesungsnotiz:** 10.5.2013

4.4 Wiederholung (?): Analysis gewöhnlicher Differentialgleichungen

Wir erinnern kurz an einige Ergebnisse aus der Analysis II.

Satz 4.12 (Fixpunktsatz von Banach)

Sei X ein vollständiger normierter Vektorraum, $Y \subset X$ abgeschlossen, $g : Y \mapsto Y$. g sei kontrahierend, d.h.

$$\exists\, q < 1 : ||g(x) - g(y)|| \leq q||x - y|| \forall x, y \in Y.$$

Dann hat g einen eindeutigen Fixpunkt $\overline{x}$, und die Fixpunktiteration $x_{k+1} = g(x_k)$ konvergiert für alle $x_0 \in Y$ gegen $\overline{x}$.

Beweis: In der Vorlesung Numerische Lineare Algebra .

Beweisidee: Es gilt für $m < n$

$$||x_n - x_m|| \leq \sum_{k=m}^{n-1} ||x_{k+1} - x_k|| \leq \sum_{k=m}^{n-1} q^k ||x_1 - x_0|| \leq \frac{q^m}{1-q} ||x_1 - x_0||,$$

also ist x_n eine Cauchy–Folge, die gegen ein $\overline{x}$ konvergiert. g ist stetig, also gilt

$$\overline{x} = \lim x_{k+1} = \lim g(x_k) = g(\lim x_k) = g(\overline{x}).$$

□

Wir wollen nun zeigen, dass Anfangswertprobleme eindeutige Lösungen besitzen. Hierzu betrachten wir die Integralgleichung 4.6. y ist also genau dann Lösung des Anfangswertproblems, wenn

$$y(t) = y_0 + \int_a^t f(s, y(s))\mathrm{ds}$$

ist. Wenn wir die rechte Seite in dieser Gleichung als Operator auf den stetigen Funktionen mit Parameter y auffassen, so steht dort eine Fixpunktgleichung: y ist genau dann Lösung des Anfangswertproblems, wenn

$$y = Ty.$$

Wenn wir zeigen können, dass T die Voraussetzungen des Fixpunktsatzes von Banach erfüllt, der insbesondere auch für unendlichdimensionale Vektorräume gilt, hat diese Gleichung (und damit das Anfangswertproblem) eine eindeutige Lösung. Hierzu benötigen wir die Voraussetzung der Lipschitzstetigkeit von f im zweiten Argument.

Definition 4.13 (Lipschitzstetigkeit)

Eine Funktion $f : D \mapsto X$ (D, X normiert) heißt lipschitzstetig genau dann, wenn es eine Lipschitzkonstante $L > 0$ gibt mit

$$||f(x) - f(y)|| \leq L||x - y|| \, \forall \, x, y \in D.$$

Lemma 4.14 (Eigenschaften der Lipschitzstetigkeit)

1. *Falls f lipschitzstetig ist, so ist f stetig.*
2. *Falls $L < 1$, so ist f kontrahierend. Kontrahierende Funktionen sind lipschitzstetig.*
3. *Falls D zusammenhängend und kompakt und f stetig differenzierbar ist, so ist f lipschitzstetig.*

Beweis: *kurz zu 3.: Da D zusammenhängend ist, gilt*

$$f(x) - f(y) = f'(\xi)(x - y).$$

Da D kompakt ist, existiert das Maximum L von $||f'||$ und erfüllt die Bedingung an die Lipschitzkonstante. □

Satz 4.15 (Satz von Picard–Lindelöf, lokaler Existenz– und Eindeutigkeitssatz)

Sei $G \subset \mathbb{R}^n$ offen, $I = [a, b]$, $y_0 \in G$, und

$$f : I \times \mathbb{R}^n \mapsto \mathbb{R}^n$$

stetig. Sei zusätzlich f Lipschitz–stetig im zweiten Argument auf der offenen und zusammenhängenden Menge $G \subset \mathbb{R}^n$, d.h.

$$\exists L : ||f(t, y) - f(t, z)|| \leq L||y - z|| \forall t \in I, y, z \in G.$$

Dann besitzt die zugehörige Anfangswertaufgabe

$$y'(t) = f(t, y(t)), \; y(a) = y_0$$

eine (lokale) Lösung auf einem Intervall $[a, a + \epsilon]$. Es gibt also ein $\epsilon > 0$ und eine differenzierbare Funktion

$$y_\epsilon : I_\epsilon := [a, a + \epsilon] \mapsto G$$

mit

$$y_\epsilon(a) = y_0, \; y'_\epsilon(t) = f(t, y_\epsilon(t)) \, \forall t \in I_\epsilon.$$

Bei festem ϵ ist y_ϵ eindeutig bestimmt. Für $\epsilon < \epsilon_0$ gilt $y_\epsilon = y_{\epsilon_0}|_{I_\epsilon}$.

Der Satz sagt also: Es gibt ein Intervall $[a, a+\epsilon]$, auf dem die Anfangswertaufgabe eine eindeutige Lösung hat, falls f Lipschitz–stetig ist im zweiten Argument.

Für das zweite Beispiel in 4.8 ist die rechte Seite nicht differenzierbar in y und nicht lipschitzstetig, deshalb garantiert uns 4.15 keine eindeutige Lösung.

Beweis: Wir zeigen mit Hilfe des Banachschen Fixpunktsatzes, dass die zugehörige Integralgleichung eine eindeutige Lösung besitzt.

G ist offen, $y_0 \in G$, also gibt es eine Umgebung von y_0, die ganz in G liegt:

$$\exists \delta > 0 : B := K_\delta(y_0) \subset G,$$

wobei

$$K_\delta(y_0) := \{x \in \mathbb{R}^n : ||x - y_0|| \leq \delta\}$$

die abgeschlossene Kugel um y_0 mit Radius δ bezeichnet.

$I \times B$ ist abgeschlossene und beschränkte Teilmenge des $\mathbb{R}^{n+1}$, also kompakt. f ist stetig, also existiert

$$M := \max_{t \in I, y \in B} |f(t, y)|.$$

Sei nun $0 < q < 1$ und

$$0 < \epsilon \leq \min(\frac{q}{L}, b - a, \frac{\delta}{M}),\ I_\epsilon := [a, a + \epsilon] \subset I.$$

Der Vektorraum X der stetigen Funktionen auf dem Intervall I_ϵ, versehen mit der Maximumnorm,

$$X := (C^0(I_\epsilon, \mathbb{R}^n), || \cdot ||_\infty),$$

ist vollständig (Analysis 2), d.h. jede Cauchyfolge von Funktionen in X konvergiert gegen eine stetige Funktion. Also dürfen wir den Fixpunktsatz in diesem Vektorraum anwenden.

Sei Y die Menge aller stetigen Funktionen auf I_ϵ mit Werten in B, also

$$Y = C^0(I_\epsilon, B).$$

B ist abgeschlossen, damit ist Y abgeschlossene Teilmenge von X.

Die Funktion

$$T : Y \mapsto Y, (Ty)(t) := y_0 + \int_a^t f(s, y(s))\mathsf{ds}$$

ist wohldefiniert:

Sei $y \in Y$. y und f sind stetig, also ist Ty stetig. Noch zu zeigen ist: Ty hat nur Werte in B.

$$||(Ty)(t) - y_0|| \leq \int_a^{a+\epsilon} |f(s, y(s))|\mathsf{ds} \leq \epsilon M \leq \frac{\delta}{M} M = \delta \,\forall\, t \in I_\epsilon,$$

also ist

$$(Ty)(t) \in B \,\forall\, t \in I_\epsilon$$

und damit

$$Ty \in C^0(I_\epsilon, B) = Y,$$

T ist also eine Selbstabbildung von Y nach Y.

Weiter gilt für $u, v \in Y$

$$\begin{aligned} ||Tu - Tv||_\infty &= || \int_a^t f(s, u(s)) - f(s, v(s))\mathsf{ds}||_\infty \\ &\leq \int_a^{a+\epsilon} ||f(s, u(s)) - f(s, v(s))||\mathsf{ds} \\ &\leq L \int_a^{a+\epsilon} ||u(s) - v(s)||\mathsf{ds} \\ &\leq L\epsilon ||u - v||_\infty \\ &\leq q||u - v||_\infty. \end{aligned}$$

Damit ist T kontrahierend. Nun haben wir alles für den Banachschen Fixpunktsatz zusammen, also hat T einen eindeutigen Fixpunkt $\overline{y}$ mit

$$T\overline{y} = \overline{y}$$

oder

$$\overline{y}(t) = y_0 + \int_a^t f(s, \overline{y}(s))\mathsf{ds}.$$

$\overline{y}$ ist stetig, löst die Integralgleichung und ist damit Lösung der Anfangswertaufgabe nach 4.6. Für festes ϵ ist die Lösung eindeutig.

Beweis des Zusatzes: y_{ϵ_0} ist Lösung der Anfangswertaufgabe auch im Intervall I_ϵ, und die Lösung dort ist eindeutig. □

In dieser Form garantiert der Satz die Existenz einer Lösung im Intervall

$$[a, a + \epsilon],\ \epsilon < \min(\frac{\delta}{M},\ \frac{1}{L}, b - a).$$

Wählen wir die Norm im Satz etwas geschickter, so erhalten wir eine bessere Abschätzung, die nicht mehr von der Lipschitzkonstanten abhängt.

Satz 4.16 (Picard–Lindelöf, globale Form)

Seien alle Bezeichnungen wie oben, und es gelte $\delta/M > (b-a)$*. Dann gibt es eine eindeutige Lösung im Intervall* $[a,b]$.

Eine Folgerung dieses Satzes ist: Falls f Lipschitz–stetig ist auf ganz $I \times \mathbb{R}^n$ (also für $n=1$ auf einem Streifen), so hat die Anfangswertaufgabe eine Lösung auf I. Ist $I=[a,\infty]$, so besitzt die Aufgabe eine globale Lösung.

Beweis: Betrachte auf E die zu $||\cdot||_\infty$ äquivalente Norm

$$|||f||| := ||\exp(-2Lt)f(t)||_\infty.$$

Genauer Beweis in den Übungen. Wir zeigen, dass die zugehörige Abbildung kontrahierend ist mit Kontraktionskonstante $\frac{1}{2}$:

$$\begin{aligned}
|||Tu-Tv||| &= \sup_{t\in[a,b]} |\exp(-2Lt)\int_a^t f(s,u(s))-f(s,v(s))\mathsf{ds}| \\
&\leq \sup_{t\in[a,b]} \exp(-2Lt)\int_a^t L\exp(2Ls)\exp(-2Ls)|u(s)-v(s)|\mathsf{ds} \\
&\leq L\sup_{t\in[a,b]} \exp(-2Lt)\int_a^t \exp(2Ls)|||u-v|||\mathsf{ds} \\
&\leq L|||u-v|||\sup_{t\in[a,b]} \exp(-2Lt)\frac{1}{2L}(\exp(2Lt)-\exp(2La)) \\
&\leq \frac{1}{2}|||u-v|||.
\end{aligned}$$

□

Diese Abschätzung kann man nun noch einmal verbessern. Ist $|f|$ durch M beschränkt, so gilt wie im Beweis zu 4.15 für die Lösung des Anfangswertproblems

$$|y(a+t)-y(a)| = |\int_a^{a+t} f(s,y(s))\mathsf{ds}| \leq tM.$$

Es reicht also für die Existenz einer Lösung auf I, dass f auf dem Kegel

$$K_M(a,y_0) := \{(t,y) : t\in I,\ |y-y_0| \leq M(t-a)\}$$

lipschitzstetig ist. Für skalare Differentialgleichungen ist dies natürlich nur ein gleichschenkliges Dreieck mit der Spitze bei (a,y_0).

Da für

$$(a', y_0') \in K_M(a, y_0)$$

gilt

$$K_M(a', y_0') \subset K_M(a, y_0),$$

ist unter den Voraussetzungen des Satzes also auch die Anfangswertaufgabe mit Anfangswert in (a', y_0') eindeutig auf dem ganzen Intervall $[a', b]$ lösbar.

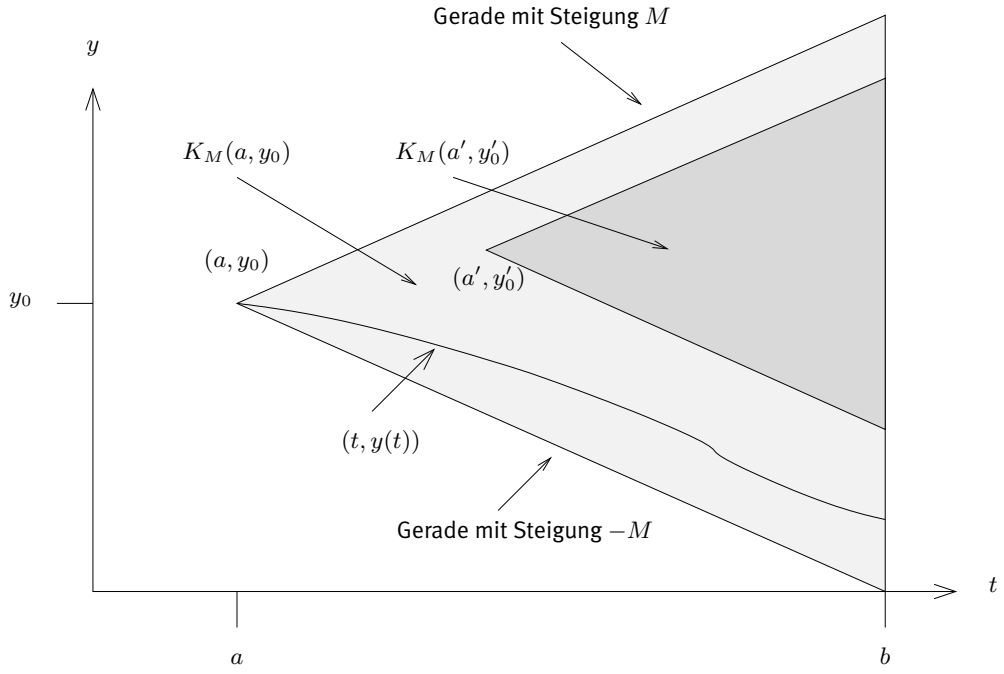


Abbildung 4.3: Kegel $K_M(a, y_0)$

Dieses Ergebnis ist überraschend. Die scheinbar so wichtige Lipschitzkonstante geht nun in diesen Satz gar nicht mehr ein. Zusätzlich hatten wir ja bereits das einführendes Beispiel 4.8, bei dem eine (wenn auch keine eindeutige) Lösung der Differentialgleichung existierte, obwohl f dort nur stetig und nicht lipschitzstetig war.

Tatsächlich benötigt man die Lipschitzstetigkeit nicht für die Existenz einer Lösung. Wir zitieren den alternativen Satz von Peano.

Satz 4.17 *(Satz von Peano)*

Falls f stetig ist auf dem Gebiet D, so besitzt jede Anfangswertaufgabe in D mindestens eine Lösung, die bis zum Rand von D geht.

Beweis: Neu ist hier, dass man die Lipschitzstetigkeit nicht benötigt, dafür bezahlt man mit dem Verlust der Eindeutigkeit. Hinter dem (im Gegensatz zu 4.15 nicht konstruktiven) Beweis steht der Satz von Arzela–Ascoli (Walter [2000]). □

Wir fragen uns nun, ob Differentialgleichungen überhaupt vernünftig lösbar sind. In 3.6 und 3.8 haben wir gesehen, dass die Aufgabe der Berechnung eines Integrals gut, die der Berechnung einer Ableitung schlecht gestellt ist. In den einleitenden Beispielen zu Differentialgleichungen haben wir gesehen, dass die Lösungsformeln sich normalerweise durch Integration ergeben, wir erwarten also auch hier einen Fehler in der Lösung von der Größenordnung des Fehlers in den Eingangsdaten. Dies ist richtig.

Eine Abschätzung für die Größenordnung von Lösungen gewöhnlicher Differentialgleichungen und des unvermeidlichen Fehlers beim Lösen von Gleichungen liefert das Lemma von Gronwall. Die Idee dabei ist leicht zu verstehen: Die Differentialgleichung

$$u'(t) = \beta u(t)$$

hat die Lösung

$$u(t) = u(a)\exp(\beta(t-a)).$$

Ersetzen wir in beiden Gleichungen $=$ durch $\leq$, so bleibt die Aussage richtig. Wir zitieren nur die skalare Form.

Satz 4.18 (Lemma von Gronwall)

Seien $I = [a, b]$ *und*

$$\alpha : I \mapsto \mathbb{R},\ \beta : I \mapsto \mathbb{R},\ u : I \mapsto \mathbb{R}$$

stetig.

1. *Differentielle Form: Falls* u *differenzierbar ist und*

$$u'(t) \leq \alpha(t) + \beta(t)u(t)\,\forall\, t \in I,$$

so gilt

$$u(t) \leq u(a)e^{\int_a^t \beta(s)ds} + \int_a^t \alpha(s)e^{\int_s^t \beta(\xi)d\xi}ds\,\forall\, t \in I.$$

2. *Integralform: Falls* $\beta \geq 0$ *und*

$$u(t) \leq \alpha(t) + \int_a^t \beta(s)u(s)ds\,\forall\, t \in I,$$

so gilt

$$u(t) \leq \alpha(t) + \int_a^t \alpha(s)\beta(s)e^{\int_s^t \beta(\xi)d\xi}ds\,\forall\, t \in I.$$

Die Vorbemerkung erhalten wir durch die Wahl $\alpha = 0$, $\beta(t) = \beta$ in der Differentialform.

Beweis: Wir zeigen einige einfache Lemmata.

1. Seien $u, v \in C^1(I, \mathbb{R})$ und

   $$u'(t) \leq v'(t),\ t \in I,\ u(a) \leq v(a).$$

   Dann gilt

   $$u(t) \leq v(t),\ t \in I.$$

   Beweis: $v - u$ ist monoton steigend.

2. Sei $u \in C^1(I, \mathbb{R})$, $v \in C^0(I, \mathbb{R})$, und

   $$u'(t) \leq v(t),\ t \in I.$$

   Dann gilt

   $$u(t) \leq u(a) + \int_a^t v(s)\mathsf{ds} =: w(t).$$

   Beweis: Es gilt

   $$u'(t) \leq v(t) = w'(t)$$

   und dann mit Lemma 1.

3. Sei v die Lösung von

   $$v'(t) = -\beta(t)v(t),\ v(a) = 1,$$

   also

   $$v(t) := e^{-\int_a^t \beta(s)\mathsf{ds}}.$$

   Dann gilt mit der Ungleichung aus der differentiellen Form

   $$(uv)' = u'v - \beta vu \leq \alpha v + \beta uv - \beta vu = \alpha v.$$

   Nach Lemma 2 gilt damit

   $$u(t)v(t) \leq u(a) + \int_a^t \alpha(s)v(s)\mathsf{ds},\ t \in I.$$

   Multiplikation mit $1/v(t)$ ergibt die Behauptung.

4. Sei nun

$$v(t) := \int_a^t \beta(s)u(s)\mathsf{ds}.$$

Mit den Voraussetzungen aus der Integralform gilt dann

$$v'(t) = \beta(t)u(t) \leq \beta(t)\alpha(t) + \beta(t)v(t),\ v(a) = 0.$$

Anwendung der Differentialform liefert das Gewünschte.

□

Satz 4.19 (Stetigkeit des Anfangswertproblems für Anfangswertaufgaben)

Sei

$$y'(t) = f(t, y(t)),\ y(a) = y_0$$

ein Anfangswertproblem, das die Voraussetzungen von 4.15 erfüllt, insbesondere sei f Lipschitz–stetig im zweiten Argument mit der Lipschitz–Konstanten L. Statt f und y_0 seien nur Näherungen $\widetilde{f}$ und $\widetilde{y}_0$ bekannt mit

$$||f - \widetilde{f}||_\infty \leq \epsilon,\ ||y_0 - \widetilde{y}_0|| \leq \widetilde{\epsilon}.$$

Falls die ungestörte Gleichung und die gestörte Gleichung

$$\widetilde{y}'(t) = \widetilde{f}(t, \widetilde{y}(t)),\ \widetilde{y}(a) = \widetilde{y}_0$$

Lösungen y bzw. $\widetilde{y}$ in einem Intervall $[a, b]$ besitzen, so gilt

$$||\widetilde{y}(t) - y(t)|| \leq (\widetilde{\epsilon} + \epsilon(t - a)) \exp(L(t - a))\forall t \in [a, b].$$

Beweis: Mit $u(t) := ||\widetilde{y}(t) - y(t)||$ gilt

$$\begin{aligned}
u(t) &= ||\widetilde{y}_0 - y_0 + \int_a^t \widetilde{f}(s, \widetilde{y}(s)) - f(s, y(s))\mathsf{ds}|| \\
&\leq ||\widetilde{y}_0 - y_0|| + \int_a^t (||\widetilde{f}(s, \widetilde{y}(s)) - f(s, \widetilde{y}(s))|| + ||f(s, \widetilde{y}(s)) - f(s, y(s)))||\mathsf{ds} \\
&\leq \widetilde{\epsilon} + \int_a^t (\epsilon + L||\widetilde{y}(s) - y(s)||)\mathsf{ds} \\
&\leq \underbrace{\widetilde{\epsilon} + \epsilon(t - a)}_{\alpha(t)} + \underbrace{L}_{\beta} \int_a^t u(s)\mathsf{ds}.
\end{aligned}$$

Anwendung von 4.18 liefert das Gewünschte:

$$\begin{aligned} u(t) &\leq (\widetilde{\epsilon} + \epsilon(t-a)) + L \int_a^t \underbrace{(\widetilde{\epsilon} + \epsilon(s-a))}_{\leq \widetilde{\epsilon} + \epsilon(t-a)} \exp(L(t-s))\mathsf{ds} \\ &\leq (\widetilde{\epsilon} + \epsilon(t-a))(1 - [\exp(L(t-s))]_a^t) \\ &= (\widetilde{\epsilon} + \epsilon(t-a)) \exp(L(t-a)). \end{aligned}$$

□

Der Satz zeigt: Die Lösung einer gewöhnlichen Differentialgleichung hängt stetig von den Parametern ab. Der unvermeidbare Fehler hängt linear von den Fehlern der Parameter und exponentiell von der Länge des Intervalls ab. Insbesondere ist die Aufgabe der Lösung einer Differentialgleichung, bei der f die Voraussetzungen des Satzes von Picard–Lindelöf erfüllt, gut gestellt. Wir erwarten also von den numerischen Verfahren, das auch sie bei gestörtem y_0 und f nur linear gestörte Ergebnisse liefern.

4.5 Diskrete Verfahren zur Lösung von Anfangswertaufgaben

Wir haben gesehen, dass Anfangswertaufgaben im Allgemeinen keine explizit angebbaren Lösungen besitzen. Der Fixpunktsatz von Banach liefert zwar konstruktiv Approximationen an die Lösung über die Fixpunktiteration, diese spielt aber numerisch praktisch keine Rolle. Wir betrachten rein diskrete Approximationen für die Lösung.

Definition 4.20 (Allgemeine Anfangswertaufgabe)

Sei nach den Vorbemerkungen nun immer

$$y'(t) = f(t, y(t)),\ y(a) = y_0$$

eine Anfangswertaufgabe für eine skalare Differentialgleichung oder ein System von Differentialgleichungen auf dem Intervall $I = [a, b]$. Dabei werden wir immer voraussetzen:

- *Die Funktion*

  $$f : D \mapsto \mathbb{R}^n,\ D \subset \mathbb{R} \times \mathbb{R}^n$$

  ist stetig.

- $||f||$ *ist durch* M *nach oben beschränkt.*
- f *ist lipschitzstetig im zweiten Argument mit Lipschitzkonstante* L*.*
- *Der Kegel* $K_M(a, y_0)$ *ist ganz im Definitionsgebiet* D *von* f *enthalten (Kegelbedingung).*
- *Es gebe ein* $\delta > 0$*, so dass für jeden Punkt* (a', y_0') *aus* $K_M(a, y_0)$ *seine* δ*–Umgebung bzgl.* y *ganz in* D *enthalten ist, also*

  $$B_\delta(a', y_0') = \{(a', y) : ||y - y_0'|| \leq \delta\} \subset D \, \forall (a', y_0') \in K_M(a, y_0).$$

  Damit ist 4.15 nach links und rechts anwendbar, jede Anfangswertaufgabe mit Startwert im Kegel besitzt eine lokale Lösung.
- *Nach 4.16 besitzt die Anfangswertaufgabe sogar eine Lösung* y *auf dem ganzen Intervall* $[a, b]$*. Jede Anfangswertaufgabe für die Differentialgleichung mit Anfangswert* $(a', y_0') \in K_M(a, y_0)$ *besitzt eine Lösung auf dem Intervall* $[a', b]$*.*

Im Abschnitt über grafische Lösungen haben wir bereits ein einfaches Lösungsverfahren kennengelernt. Wir wollen dies nun formalisieren. Dazu wählen wir eine Unterteilung von $[a, b]$ (ein Gitter)

$$I_h := (t_0 = a, t_1, t_2, \ldots, t_{N-1}, t_N = b)$$

und bestimmen einen Polygonzug durch Punkte (t_k, y_k) wie folgt:

- Starte am Punkt $P := (t_0, y_0)$,
- Bestimme die Steigung m der Tangente an die Lösungskurve im Punkt P mit der Differentialgleichung, also

  $$m = y'(t_0) = f(t_0, y_0).$$

- Bestimme den Schnittpunkt

  $$S = (t_1, y_0 + m(t_1 - t_0))$$

  der Tangente mit der Geraden $t = t_1$.
- Zeichne die Gerade PS ein.
- Wiederhole die letzten beiden Schritte, ausgehend vom Startpunkt $P := S$ zur nächsten t-Koordinate des Gitters, bis der Randpunkt b erreicht ist.

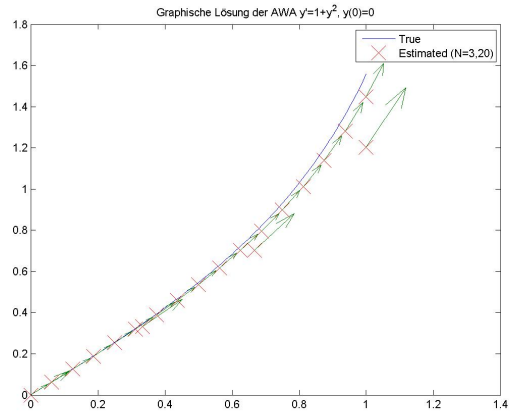


Abbildung 4.4: Graphische Lösung von AWA

Klick für Bild grafdgl
Klick für Matlab Figure grafdgl

```
function [ output_args ] = graphisch( input_args )
%GRAPHISCH grafische Loesung einer GDGL

close all;
x=(0:100)/100;
plot(x,truesol(x));
```

Listing 4.2: Graphische Lösung von AWA (gdgl/graphisch.m)

Klicken für den Quellcode von gdgl/graphisch.m

Für die Eckpunkte (t_k, y_k) dieses Polygonzugs gilt

$$y_{k+1} = y_k + h_k f(t_k, y_k), h_k = t_{k+1} - t_k.$$

y_k ist eine diskrete Approximation an $y(t_k)$. Die (h_k) könnten zu Beginn fest gewählt werden (z.B. wie hier konstant, äquidistantes Gitter) oder erst im Verlauf des Verfahrens bestimmt werden (Schrittweitensteuerung). Dass letzteres Sinn macht, sieht man bereits an unserem Beispiel: Gegen Ende des Intervalls wird die Ableitung groß, man würde dort die Abschnitte auf der t–Achse kleiner wählen.

Bemerkung: Das Verfahren ist durchführbar, denn

$$||y_{k+1} - y_0|| = ||\sum_{j=0}^{k}(t_{j+1} - t_j)f(t_j, y_j)|| \leq (t_{k+1} - a)M,$$

also liegt y_{k+1} in $K_M(a, y_0)$, und damit ist $f(t_{k+1}, y_{k+1})$ wohldefiniert mit der Kegelbedingung.

Definition 4.21 (zulässige) Gitter und numerische Verfahren

Sei

$$I_h := \{t_0 = a, t_1, \ldots, t_{N-1}, t_N = b\}$$
$$h_k := t_{k+1} - t_k > 0,\ k = 0 \ldots N-1.$$

Dann heißt I_h ein (zulässiges) Gitter zur Lösung der Anfangswertaufgabe 4.20.

$$h := \max_k h_k$$

heißt Feinheit des Gitters. Numerische Verfahren bestimmen zu vorgegebenem h ein Gitter I_h mit Feinheit $\leq h$ und eine diskrete Näherung für die Lösung y von 4.20

$$y_h : I_h \mapsto \mathbb{R}^n,\ y_h(t_k) \sim y(t_k).$$

y_h heißt Gitterfunktion. Statt $y_h(t_k)$ benutzen wir häufig, wenn sie eindeutig ist, die verkürzte Schreibweise

$$y_k := y_h(t_k).$$

Die Gitterfunktionen sind ausdrücklich nur auf dem Gitter definiert. Dies ist anders als bei den graphischen Verfahren, dort bekommen wir durch lineare Interpolation eine Approximation auf dem ganzen Intervall I.

Als erstes interessieren wir uns für den Fehler dieser Approximation.

Definition 4.22 (globaler Diskretisierungsfehler)

Sei y_h diskrete Näherung für die Lösung y der Anfangswertaufgabe 4.20 auf dem Gitter I_h. Dann heißt

$$e_h : I_h \mapsto \mathbb{R}^n,\ e_h := y|_{I_h} - y_h$$

die Fehlerfunktion der Näherung.

$$||e_h||_\infty = \max_{t \in I_h} ||e_h(t)||$$

heißt globaler Diskretisierungsfehler.

Definition 4.23 (Konvergenz von numerischen Verfahren)

Gegeben sei ein numerisches Verfahren für 4.20, das einem h das Gitter I_h mit Feinheit $\leq h$ und die diskrete Näherung y_h zuordnet, mit Fehlerfunktion e_h. Das Verfahren heißt konvergent, falls der globale Diskretisierungsfehler mit h gegen 0 geht. Also:

$$||e_h||_\infty \to_{h \to 0} 0.$$

Das Verfahren heißt konvergent von der Ordnung p, falls

$$||e_h||_\infty = O(h^p).$$

Ausdrücklich: Natürlich ist auch der globale Diskretisierungsfehler, wie die Näherung, nur auf den Gittern I_h definiert.

Wie schon bei der numerischen Integration bedeutet eine höhere Ordnung, dass für kleine h der Fehler um eine Größenordnung kleiner ist als bei niedrigerer Ordnung.

Häufig werden wir auf die Abhängigkeit von der Gitterwahl verzichten und zeigen, dass für alle zulässigen Gitter I_h mit Feinheit h gilt:

$$||e_h|| \leq e(h),\ e(h) \to 0 \text{ für } h \to 0.$$

Wir wollen numerische Verfahren für 4.20 entwickeln. Hierzu wollen wir wie beim grafischen Verfahren Gitterfunktionen bestimmen, die die Lösung approximieren, also Funktionen

$$y_h : I_h \mapsto \mathbb{R}^n,\ y_h(t_k) \sim y(t_k).$$

Nach 4.6 gilt

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\text{dt}. \tag{4.5}$$

Die Vorschrift zur Auswertung des Integrals wollen wir wie in 3.3 transformiert auf ein Standardintervall angeben. Es liegt also nahe, bei festem f Verfahren der Form

$$y_{k+1} = y_h(t_{k+1}) = y_k + h_k\varphi(t, y_h, I_h),\ k = 0 \ldots N-1$$

zu wählen.

Definition 4.24 *(Verfahrensfunktion)*

Für ein numerisches Verfahren zur Lösung von 4.20 gelte

$$y_{k+1} = y_h(t_{k+1}) = y_k + h_k\varphi(t_k, y_h, I_h),\ k = 0 \ldots N-1,$$

wobei

$$\varphi : \widetilde{D} \times \mathbb{R}^{N+1} \mapsto \mathbb{R}^n, \widetilde{D} = \{(t, y) : t \in I_h, y \in (\mathbb{R}^n)^{N+1} : (t, y_j) \in D\}.$$

Dann heißt φ Verfahrensfunktion.

Bemerkung: Diese Definition ist **sehr** formal. Sie ist angenehm, denn sie ermöglicht es, alle auftretenden Verfahren unter diesem gemeinsamen Dach zu behandeln. Tatsächlich werden wir immer nur die Einschränkungen aus 4.25 nutzen, deshalb reicht es, sich die Definitionen dort zu merken.

φ hängt ab vom gewählten Gitter und der Approximation y_h. Die Gleichung in der Definition ist also nicht unbedingt eine Zuweisung, sondern tatsächlich eine Gleichung, y_h kommt sowohl auf der linken als auch auf der rechten Seite vor.

Wegen Gleichung (4.5) sollte $h_k\varphi$ das Integral aus der analytischen Lösung approximieren. Hierzu haben wir bereits numerische Verfahren in Kapitel 3 kennengelernt. Wir dürfen erwarten, dass diese Verfahren auch direkt zu numerischen Verfahren zur Lösung gewöhnlicher Differentialgleichungen führen.

Wir unterscheiden zunächst vier Typen von Verfahrensfunktionen.

Definition 4.25 (Typen von numerischen Verfahren)

1. *Ein Verfahren heißt explizites Einschrittverfahren, falls*

$$y_{k+1} = y_k + h_k\varphi(t_k, y_k, h_k).$$

   Hier wird also, um y_{k+1} zu berechnen, nur die Näherung y_k benutzt. Unser graphisches Verfahren ist ein solches.

   Nachteil: Wir fangen in jedem Schritt des Verfahrens wieder an der Stelle (t_k, y_k) von vorn an und ignorieren, dass wir ja noch weitere Näherungen für y bereits ausgerechnet haben, die Informationen für y_{k+1} liefern könnten.

2. *Ein Verfahren heißt implizites Einschrittverfahren, falls*

$$y_{k+1} = y_k + h_j\varphi(t_k, y_k, y_{k+1}, h_k).$$

   Hier wird wieder nur y_k zur Berechnung von y_{k+1} benutzt, aber das y_{k+1} erscheint auf beiden Seiten der Gleichung. Um das Verfahren durchzuführen, muss in jedem Schritt ein (im allgemeinen nichtlineares) Gleichungssystem gelöst werden.

3. *Ein Verfahren heißt explizites Mehrschrittverfahren, falls*

$$y_{k+1} = y_k + h_k\varphi(t_0 \dots t_k, y_0 \dots y_k, h_0 \dots h_k).$$

   Diese Verfahren sind wieder explizit, denn auf der rechten Seite erscheinen nur die bereits berechneten Werte von y_h. Anders als bei den Einschrittverfahren lassen wir hier aber zu, dass **alle** *bereits berechneten Werte von y_h in y_{k+1}*

eingehen. Wir erwarten, dass wir durch den Rückgriff auf bereits berechnete Werte eine höhere Fehlerordnung und damit bessere Verfahren gewinnen können.

4. *Ein Verfahren heißt implizites Mehrschrittverfahren, falls*

$$y_{k+1} = y_k + h_k \varphi(t_0 \dots t_{k+1}, y_0 \dots y_{k+1}, h_0 \dots h_k).$$

Hier steht wieder das y_{k+1} auf der linken und rechten Seite, wieder muss in jedem Schritt ein Gleichungssystem gelöst werden.

Hierbei lassen wir ausdrücklich zu, dass wir ein System von Differentialgleichungen behandeln. Im folgenden werden wir einige Sätze nur skalar beweisen oder formulieren, alle gelten aber genau so auch für Systeme.

Wir gehen zunächst auf die expliziten Einschrittverfahren ein. Bevor wir uns Beispiele anschauen, bemerken wir, dass die Konvergenz der Definition nach schlecht zu prüfen ist, weil man für den globalen Diskretisierungsfehler die exakte Lösung kennen muss. Einfach ist es dagegen häufig, den Fehler von y_1 abzuschätzen.

Dies kann man sich im Fall der graphischen Näherung leicht klarmachen: (a, y_0) ist ein Punkt der Lösungskurve. Der Fehler in y_1 entsteht also ausschließlich dadurch, dass wir die Lösungskurve durch eine Tangente ersetzen.

Diesen lokalen, an der aktuellen Stelle entstehenden Fehler können wir leicht abschätzen. Leider gilt dies nur für die erste Approximation, für $k > 1$ setzt sich der Fehler von y_{k+1} aus dem bisher schon aufgetretenen Fehler in y_k und dem an dieser Stelle noch zusätzlich durch die Approximation auftretenden lokalen Fehler zusammen.

Wir definieren als lokalen Diskretisierungsfehler den Fehler von y_{k+1} unter der Annahme, dass y_k noch korrekt war, es ist also der Fehler, der in einem Schritt neu entsteht.

Definition 4.26 (lokaler Diskretisierungsfehler und Konsistenz)

Sei φ eine Verfahrensfunktion, $\overline{t} \in [a, b]$, $\overline{y} \in \mathbb{R}$, und es gelte

$$(\overline{t}, \overline{y}) \in K_M(y_0, a).$$

Sei $\widetilde{y}$ die Lösung der Differentialgleichung mit $\widetilde{y}(\overline{t}) = \overline{y}$. $\widetilde{y}$ erfüllt nicht notwendig die Anfangsbedingung.
Dann heißt

$$\tau_h(\overline{t}, \overline{y})) := \frac{1}{h} \underbrace{(\widetilde{y}(\overline{t} + h) - (\overline{y} + h\varphi(\overline{t}, \widetilde{y}|_{I_h}, I_h)))}_{=:D}$$

lokaler Diskretisierungsfehler. D ist dabei die Differenz des Wertes der Lösung $\widetilde{y}$ an der Stelle $t + h$ und der Näherung, die das Verfahren liefert bei Einsetzen der korrekten Lösung.

Das durch φ definierte numerische Verfahren heißt konsistent, falls

$$||\tau_h||_\infty \to_{h\to 0} 0.$$

Das Verfahren heißt konsistent von der Ordnung p, falls

$$||\tau_h||_\infty = O(h^p).$$

Hier bemerken wir eine Ungenauigkeit: Rechts von $\overline{t}$ existiert die Lösung $\widetilde{y}$ sicher, aber die Fortsetzbarkeit von $\widetilde{y}$ auf das gesamte Intervall nach links ist nicht gesichert. Dies wird aber später keine Rolle spielen. Tatsächlich benötigen wir $\widetilde{y}$ in allen Verfahren nur in einer kleinen Umgebung von $(\overline{t}, \overline{y})$. Dort ist die Existenz einer Lösung durch den lokalen Satz von Picard–Lindelöf gesichert.

Wieder ist diese Definition sehr formal. In den Einschränkungen auf die für uns interessanten Verfahren aus 4.25 ist alles einfacher zu verstehen. Die Definition lässt sich am einfachsten so zusammenfassen: Der lokale Diskretisierungsfehler ist der Fehler, der entsteht, wenn man in φ statt der Diskretisierung y_h die echte Lösung $\widetilde{y}$ einsetzt. Wir beschränken uns zunächst auf explizite Einschrittverfahren.

4.6 Konsistenz und Konvergenz für explizite Einschrittverfahren

Beispiel 4.27 *Lokaler Diskretisierungsfehler und Konsistenz für explizite Einschrittverfahren. Sei im Folgenden immer y irgendeine Lösung der Differentialgleichung.*

1. **Eulersches Polygonzugverfahren:**

   Das Eulersche Polygonzugverfahren ist ein explizites Einschrittverfahren mit der Verfahrensfunktion $\varphi(t, y, h) = f(t, y)$, also

   $$y_{k+1} = y_k + h_k f(t_k, y_k).$$

   Eine weitere Motivation neben der graphischen ist, dass wir das Integral aus 4.5 durch die Intervalllänge, multipliziert mit der Auswertung am linken Rand, approximieren. Sei f stetig differenzierbar. Dann ist, wegen

   $$y'(t) = f(t, y(t)),$$

y zweimal stetig differenzierbar auf I. Es gilt mit Taylorentwicklung und der Differentialgleichung

$$\begin{aligned}\tau_h(t,y(t)) &= \frac{1}{h}(y(t+h)-(y(t)+h\varphi(t,y(t),h))) \\ &= \frac{y(t+h)-y(t)}{h} - f(t,y(t)) \\ &= \frac{y(t)+hy'(t)+\frac{h^2}{2}y''(\xi(h))-y(t)}{h} - y'(t) \\ &= \frac{h}{2}y''(\xi(h)) \\ &\leq \frac{||y''||_\infty}{2} h \to_{h\to 0} 0.\end{aligned}$$

Das Eulerverfahren ist also konsistent, und zwar von der Ordnung 1. Das Eulerverfahren benötigt eine Auswertung von f in jedem Schritt.

2. **Verbessertes Eulerverfahren:**

Ein verbessertes Verfahren ergibt sich, wenn wir zur Approximation des Integrals in 4.5 die Mittelpunktregel anwenden und Taylorentwicklung nutzen:

$$\begin{aligned}\int_{t_k}^{t_k+h} f(t,y(t))dt &\sim h(f(t_k+h/2, y(t_k+h/2))) \\ &\sim h(f(t_k+\frac{h}{2}, y(t_k)+\frac{h}{2}y'(t_k)) \\ &= h(f(t_k+\frac{h}{2}, y(t_k)+\frac{h}{2}f(t_k,y(t_k)))).\end{aligned}$$

Als Verfahrensfunktion wählen wir also

$$\varphi(t,y,h) = f(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)).$$

Die Konsistenzordnung weisen wir wieder durch Taylorentwicklung nach. Wir benötigen diesmal, dass f zweimal stetig differenzierbar ist. Damit existiert die dritte Ableitung von y auf I. Zusätzlich beachten wir, dass

$$\begin{aligned}(y')'(t) &= \frac{d}{dt}f(t,y(t)) \\ &= f_t(t,y(t)) + f_y(t,y(t))y'(t) \\ &= f_t + ff_y.\end{aligned}$$

Hierbei sind f_t und f_y die Ableitungen von f nach der ersten bzw. zweiten Variablen. Mit ein– bzw. zweidimensionaler Taylorentwicklung gilt

$$\begin{aligned}
\tau_h(t,y(t)) &= \frac{1}{h}(y(t+h)-(y(t)+h\varphi(t,y(t),h))) \\
&= \frac{y(t+h)-y(t)}{h} - f(t+\frac{h}{2}, y(t)+\frac{h}{2}f(t,y(t))) \\
&= \frac{y(t)+hy'(t)+\frac{h^2}{2}y''(t)+\frac{h^3}{6}y'''(\xi(h))-y(t)}{h} - \\
&\quad (f(t,y(t))+\frac{h}{2}f_t(t,y(t))+\frac{h}{2}f(t,y(t))f_y(t,y(t))+O(h^2)) \\
&= O(h^2).
\end{aligned}$$

Das Verfahren ist konsistent von zweiter Ordnung und benötigt zwei Auswertungen von f pro Schritt.

3. **Verfahren von Heun:**

Wir können auch mit der Trapezregel integrieren, hierdurch ergibt sich das Verfahren von Heun. Wir nehmen an, dass f zweimal stetig differenzierbar ist.

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} f(t,y(t))dt &\sim \frac{h}{2}(f(t,y(t))+f(t+h,y(t+h))) \\
&\sim \frac{h}{2}(f(t_k,y(t_k))+f(t_k+h,y(t_k)+hf(t_k,y(t_k)))).
\end{aligned}$$

Die Verfahrensfunktion ist

$$\varphi(t,y,h) = \frac{1}{2}(f(t,y)+f(t+h,y+hf(t,y))).$$

Wieder gilt mit Taylorentwicklung (für f in zwei Dimensionen)

$$\begin{aligned}
\tau_h(t,y(t)) &= \underbrace{\frac{y(t+h)-y(t)}{h}}_{y'+h/2y''+O(h^2)} - \frac{1}{2}\underbrace{(f(t,y(t))+f(t+h,y(t)+hf(t,y(t))))}_{y'+(y'+h(f_t+ff_y)+O(h^2))=2y'+hy''+O(h^2)} \\
&= O(h^2).
\end{aligned}$$

Das Verfahren ist also ebenfalls konsistent von der Ordnung 2 und benötigt ebenfalls zwei Auswertungen von f pro Schritt.

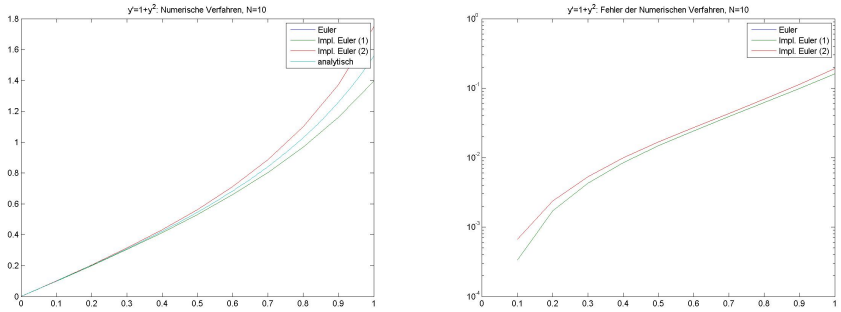


Abbildung 4.5: Einschrittverfahren auf $[0, 1]$ mit $y(0) = 1$.

Klick für Bild einschrittvef

Klick für Matlab Figure einschrittvef

Klick für Bild einschrittfehler

Klick für Matlab Figure einschrittfehler

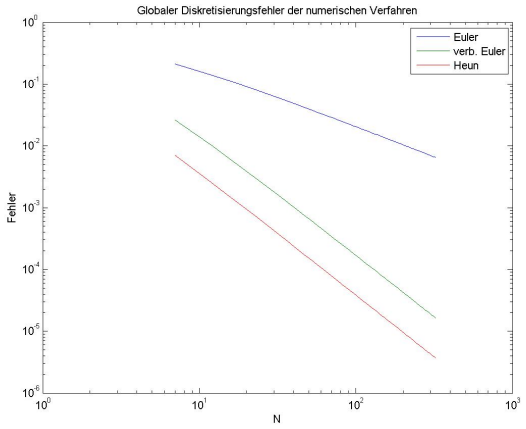


Abbildung 4.6: Asymptotische Entwicklung des Fehlers bei Einschrittverfahren

Klick für Bild gdgleinschrittasymp

Klick für Matlab Figure gdgleinschrittasymp

```
function out = euler( f,x,y,h,p )
%EULER
out=f(x,y);
end
```

Listing 4.3: Eulerverfahren (gdgl/euler.m)

Klicken für den Quellcode von gdgl/euler.m

```
function out = verbeuler( f,x,y,h,p )
%Verbesserter EULER
out=f(x+h/2,y+h/2*f(x,y));
end
```

Listing 4.4: Verbessertes Eulerverfahren (gdgl/verbeuler.m)

Klicken für den Quellcode von gdgl/verbeuler.m

```
function out = euler( f,x,y,h,p )
%Heun
y1=f(x,y);
out=1/2*(y1+f(x,y+h*y1));
end
```

Listing 4.5: Verfahren von Heun (gdgl/heun.m)

Klicken für den Quellcode von gdgl/heun.m

```
function [y,x,z] = einschritt( verf,f,a,b,yo,N,M,truesol,p )
%Einschrittverfahren
h=(b-a)/N;
y=zeros(1,N+1);
y(1)=yo;
x=a+(0:N)*h;
```

Listing 4.6: Einschrittverfahren (gdgl/einschritt.m)

Klicken für den Quellcode von gdgl/einschritt.m

```
function einschrittdemo
N=10;
a=0;
b=1;
M=2;
analfunc=@tan;
```

Listing 4.7: Demo Einschrittverf. (gdgl/einschrittdemo.m)

Klicken für den Quellcode von gdgl/einschrittdemo.m

Konsistenz ist also sehr einfach nachzuweisen durch Entwicklung aller auftretenden Terme in Taylorreihen und Ausnutzen der Differentialgleichung. Leider interessiert uns dies eigentlich gar nicht, wir wollen wissen, ob die Verfahren konvergent sind.

Bei den Einschrittverfahren haben wir den Glücksfall, dass aus aus Konsistenz bereits Konvergenz folgt. Es ist also zu zeigen, dass die in jedem Einzelschritt gemachten Fehler sich zu einem kleinen Gesamtfehler addieren. Hierzu benötigen wir eine diskrete Version des Lemmas von Gronwall.

Satz 4.28 (Diskretes Lemma von Gronwall)

Seien (α_k), (β_k), (e_k) *reelle nichtnegative Folgen und*

$$e_{k+1} \leq \alpha_k + (1+\beta_k)e_k,\ k \geq 0.$$

Dann gilt

$$e_k \leq (e_0 + \sum_{j=0}^{k-1} \alpha_j) \exp(\sum_{j=0}^{k-1} \beta_j).$$

Beweis: Durch vollständige Induktion:

$$\begin{aligned} e_{k+1} &\leq \alpha_k + (1+\beta_k)(e_0 + \sum_{j=0}^{k-1} \alpha_j) \exp(\sum_{j=0}^{k-1} \beta_j),\ \beta_k \geq 0 \\ &\leq \alpha_k \exp(\sum_{j=0}^{k} \beta_j) + (e_0 + \sum_{j=0}^{k-1} \alpha_j) \exp(\sum_{j=0}^{k} \beta_j),\ \text{denn } (\exp(\beta_k) \geq 1+\beta_k) \\ &= (e_0 + \sum_{j=0}^{k} \alpha_j) \exp(\sum_{j=0}^{k} \beta_j). \end{aligned}$$

□

Alternativ kann man die Abschätzung (mit verbesserter rechter Seite) auch direkt aus 4.18 gewinnen. Idee: Wir betrachten die Treppenfunktion $u(t)$ auf $[0, N]$, die durch die Punkte (k, e_k) geht. Dann bringen wir im diskreten Lemma das e_k auf die linke Seite, damit steht dort

$$e_{k+1} - e_k = (u(k+1) - u(k)).$$

Das erinnert bereits an eine Ableitung. Setzt man nun die rechte Seite formal in 4.18 ein, so erhält man bereits die gewünschte Abschätzung. Dies ist natürlich nicht korrekt, denn u ist gar nicht differenzierbar (wie in 4.18 vorausgesetzt). Man muss zeigen, dass die Argumentation dort auch für Treppenfunktionen gilt, deshalb haben wir hier den einfachen Weg über die Induktion gewählt.

Satz 4.29 (Konvergenz von expliziten Einschrittverfahren)

Ein explizites numerisches Einschrittverfahren zur Lösung der Anfangswertaufgabe 4.20 mit Verfahrensfunktion φ sei lipschitzstetig in der zweiten Variable y mit Lipschitzkonstanten L' und konsistent (von der Ordnung p). Dann ist das Verfahren auch konvergent (von der Ordnung p).

Beweis: Sei y Lösung der Anfangswertaufgabe 4.20. Sei $I_h = (t_k)$ das durch das Verfahren der Feinheit h zugeordnete zulässige Gitter mit zugehöriger numerischer Approximation y_h und globalem Diskretisierungsfehler e_h. Wie oben schon angedeutet schätzen wir den globalen Diskretisierungsfehler durch die Summe des lokalen Diskretisierungsfehler und des globalen Diskretisierungsfehlers aus dem letzten Schritt ab.

$$\begin{aligned}
||e_h(t_{k+1})|| &= ||y(t_{k+1}) - y_{k+1}|| \\
&= ||y(t_{k+1}) - (y_k + h_k\varphi(t_k, y_k, h_k))|| \\
&= ||y(t_{k+1}) - (y(t_k) + h_k\varphi(t_k, y(t_k), h_k)) + y(t_k) - y_k \\
&\quad + h_k(\varphi(t_k, y(t_k), h_k) - \varphi(t_k, y_k, h_k))|| \\
&\leq h_k|\tau_{h_k}(t_k, y(t_k))| + ||e_h(t_k)|| + h_kL'||y(t_k) - y_k|| \\
&= \underbrace{h_k|\tau_{h_k}(t_k, y(t_k))|}_{\alpha_k} + \underbrace{(1 + h_kL')}_{\beta_k} ||e_h(t_k)||.
\end{aligned}$$

Mit dem diskreten Lemma von Gronwall gilt also

$$\begin{aligned}
||e_h(t_k)|| &\leq \left(||e_h(t_0)|| + \sum_{j=0}^{k-1} h_j|\tau_{h_j}(t_j, y(t_j))| \right) \exp(L' \sum_{j=0}^{k-1} h_j) \\
&\leq (||e_h(t_0)|| + (t_k - a) \max_j |\tau_{h_j}(t_j, y(t_j)))| \exp\left(L'(t_k - a)\right) \\
&\leq (||e_h(t_0)|| + (t_k - a)||\tau_h||_\infty \exp\left(L'(t_k - a)\right) \\
&\to_{h\to 0} 0.
\end{aligned}$$

Falls ein Verfahren konsistent ist (von der Ordnung p), so ist es also auch konvergent (von der Ordnung p). Wir dürfen sogar noch zulassen, dass die Anfangswerte falsch sind (bis auf einen Fehler $O(h^p)$). □

Dies lässt sich (für lipschitzstetige Verfahrensfunktionen) in dem Merksatz zusammenfassen:

Für Einschrittverfahren gilt: Aus Konsistenz folgt Konvergenz.

Korollar 4.30 (Konvergenz der Referenzverfahren)

Das Eulerverfahren ist konvergent von der Ordnung 1. Das Verfahren von Heun und das verbesserte Eulerverfahren sind konvergent von der Ordnung 2.

Für lineare Einschrittverfahren, bei denen φ nach f differenzierbar ist, also φ lokal linear von f abhängt, gilt

Satz 4.31 (Numerische Stabilität von expliziten Einschrittverfahren)

Die linearen expliziten konvergenten Einschrittverfahren aus 4.27 sind numerisch stabil.

Beweis: Falls der Anfangswert y_0 falsch ist, so ist im Beweis zu 4.29 $e_h(t_0)$ nicht Null. Dieser Fehler geht dann linear in den Gesamtfehler ein.

Steht statt f nur eine Näherung $\widetilde{f}$ zur Verfügung, so geht dieser lokal linear in den lokalen Diskretisierungsfehler ein und dieser wiederum linear in den Gesamtfehler, insgesamt erhalten wir wieder eine lineare Abhängigkeit des Fehlers im Ergebnis von den Fehlern in den Daten.

Der auftretende Algorithmusfehler liegt also in der Größenordnung des unvermeidlichen Fehlers 4.19. Die Verfahren sind numerisch stabil. Tatsächlich sind sogar die Konstanten im Limes für $h \to 0$ dieselben wie in 4.19. □

4.7 Konvergenz und Konsistenz für implizite Einschrittverfahren

Wir werden sehen, dass implizite Verfahren nützlich sind. Es stellt sich aber die Frage, ob implizite Verfahren überhaupt wohldefiniert sind (d.h. ob die Gleichungen, die sie definieren, eindeutige Lösungen haben), und ob die so entstehenden Verfahren konvergent sind.

Satz 4.32 *(***Wohldefiniertheit für implizite Einschrittverfahren***)*

Sei $\varphi(t_k, y_k, y_{k+1}, h_k)$ die Schrittfunktion eines impliziten Einschrittverfahrens zur Lösung von 4.20. Sei φ stetig, und lipschitzstetig bzgl. y_k und y_{k+1} mit der Lipschitzkonstanten L. Dann gibt es ein h, so dass die Gleichung

$$y_{k+1} = y_k + h_k\varphi(t_k, y_k, y_{k+1}, h_k)$$

für $h_k \leq h$ für alle t_k und y_k lokal (in einer kleinen Umgebung von y_k) eindeutig nach y_{k+1} auflösbar ist. Es gibt also eine Funktion $v(t_k, y_k, h_k)$, so dass

$$y_{k+1} = v(t_k, y_k, h_k).$$

Insbesondere lassen sich implizite Einschrittverfahren als explizite Verfahren der Form

$$y_{k+1} = y_k + h_k\varphi(t_k, y_k, v(t_k, y_k, h_k), h_k)$$

formulieren.

v *ist lipschitzstetig in der zweiten Variablen.*

Beweis: Wir zeigen, dass die rechte Seite bei der Definition der impliziten Verfahren eine Selbstabbildung und kontrahierend ist, dann folgt die Wohldefiniertheit aus dem Banachschen Fixpunktsatz.

Sei δ wie in 4.20, und K'_M die abgeschlossene δ–Umgebung von $K_M(a, y_0)$ in y–Richtung, also mit dem B_δ aus 4.20

$$K'_M = \bigcup_{(a',y')\in K_M(a,y_0)} B_\delta(a', y').$$

Nach Definition 4.24 ist $\varphi(t, y, y', h)$ definiert falls $(t, y) \in K'_M$ und (t, y') in K'_M, $t \in [a, b]$. Die Menge K'_M ist kompakt, also nimmt $|\varphi|$ dort sein Maximum M' an. Sei nun h so klein, dass

$$q := Lh < 1 \text{ und } M'h \leq \delta,$$

also insbesondere unabhängig von y_k und t_k, und

$$h_k \leq h.$$

Seien t_k und y_k fest. Wir setzen

$$g : [y_k - \delta, y_k + \delta] \mapsto [y_k - \delta, y_k + \delta],\ g(y) := y_k + h_k\varphi(t_k, y_k, y, h_k).$$

Zu zeigen ist: g ist wohldefiniert und kontrahierend. Sei $y \in [y_k - \delta, y_k + \delta]$.

$$\begin{aligned} |g(y) - y_k| &= |h_k\varphi(t_k, y_k, y, h_k)| \\ &\leq M'h_k \leq \delta \end{aligned}$$

und damit $g(y) \in [y_k - \delta, y_k + \delta]$. Weiter gilt für $z,\ y \in [y_k - \delta, y_k + \delta]$ mit der Lipschitzkonstanten L

$$\begin{aligned} |g(z) - g(y)| &= |h_k(\varphi(t_k, y_k, z, h_k) - \varphi(t_k, y_k, y, h_k))| \\ &\leq h_k L|z - y| \leq q|z - y|. \end{aligned}$$

Also ist g auch kontrahierend und besitzt mit dem Banachschen Fixpunktsatz 4.12 einen eindeutigen Fixpunkt

$$y_{k+1} = v(t_k, y_k, h_k).$$

Die Fixpunktiteration für g konvergiert insbesondere für den Startwert y_k gegen y_{k+1}.

Zu zeigen ist noch: v ist lipschitzstetig in y_k.

Sei $u_1 = v(t_k, y_k, h_k)$, $u_2 = v(t_k, z_k, h_k)$. Dann gilt

$$\begin{aligned} |u_1 - u_2| &= |y_k + h_k\varphi(t_k, y_k, u_1, h_k) - (z_k + h_k\varphi(x, z_k, u_2, h_k))| \\ &\leq |y_k - z_k| + h_k|\varphi(t_k, y_k, u_1, h_k) - \varphi(t_k, z_k, u_1, h_k) \\ &\quad +\varphi(t_k, z_k, u_1, h_k) - \varphi(t_k, z_k, u_2, h_k)| \\ &\leq |y_k - z_k| + Lh_k|y_k - z_k| + Lh_k|u_1 - u_2|. \end{aligned}$$

also wegen $Lh_k < 1$

$$|v(t_k, y_k, h) - v(t_k, z_k, h)| = |u_1 - u_2| \leq \frac{1 + Lh_k}{1 - Lh_k}|y_k - z_k|.$$

Also ist v lipschitzstetig in der zweiten Variablen mit Lipschitzkonstante

$$L' = (1 + Lh_k)/(1 - Lh_k).$$

□

Implizite Einschrittverfahren sind also explizite Verfahren, nur etwas anders aufgeschrieben. Insbesondere gilt der Konvergenzsatz auch für implizite Verfahren:

Korollar 4.33 (Konvergenzsatz für implizite Einschrittverfahren)
Implizite konsistente Einschrittverfahren (von der Ordnung p) sind konvergent (von der Ordnung p).

Beweis: Implizite sind spezielle explizite Verfahren, das zugehörige φ ist lipschitzstetig, also folgt alles mit 4.29 und 4.32. □

Beispiel 4.34 (Beispiele für implizite Verfahren)

1. *Implizites Eulerverfahren:*

   $$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1})$$

   Wir berechnen wieder die Konsistenzordnung. Sei y wieder irgendeine Lösung der Differentialgleichung, und f differenzierbar.

   $$\begin{aligned} \tau_h(t, y(t)) &= \frac{1}{h}(y(t + h) - (y(t) + hf(t + h, y(t + h)))) \\ &= \frac{1}{h}(hy'(t) + O(h^2) - hy'(t + h)) \\ &= y'(t) - y'(t) + O(h) \end{aligned}$$

   und damit ist das Verfahren konsistent von der Ordnung 1, also auch konvergent von der Ordnung 1.

2. *Implizite Trapezregel:*

$$y_{k+1} = y_k + \frac{h_k}{2}(f(t_k, y_k) + f(t_{k+1}, y_{k+1})).$$

Sei f zweimal stetig differenzierbar.

$$\begin{aligned}\tau_h(t, y(t)) &= \frac{1}{h}(y(t+h) - (y(t) + \frac{h}{2}(f(t, y(t)) + f(t+h, y(t+h)))))\\ &= y'(t) + \frac{h}{2}y''(t) + O(h^2) - \frac{1}{2}(y'(t) + y'(t+h))\\ &= y'(t) - y'(t) + \frac{h}{2}y''(t) - \frac{h}{2}y''(t) + O(h^2)\end{aligned}$$

und damit ist das Verfahren von zweiter Ordnung.

Satz 4.32 gibt auch gleich eine Anleitung zur Durchführung der impliziten Verfahren. Glücklicherweise muss das v aus 4.32 nicht explizit ausgerechnet werden. Die Fixpunktiteration für das g aus dem Satz mit Startwert y_k konvergiert gegen y_{k+1}, also führt man in jedem Schritt des impliziten Einschrittverfahrens zur Lösung der impliziten Gleichung einige Schritte der Fixpunkiteration durch.

Beispiel 4.35 Fixpunktiteration für das implizite Eulerverfahren

Wir wenden das implizite Eulerverfahren an und lösen die Definitionsgleichung durch Fixpunktiterationen. Führen wir in jedem Schritt des Eulerverfahrens einen Schritt der Fixpunktiteration durch, so erhalten wir

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_k).$$

Dieses Verfahren hat die Ordnung 1 und benötigt eine Auswertung von f.

Führen wir zwei Schritte der Fixpunktiteration durch, so erhalten wir

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_k + h_k f(t_k, y_k)).$$

Hier machen wir ein schlechtes Geschäft: Wir benötigen zwei Auswertungen von f pro Schritt, aber das Verfahren ist trotzdem nur von der Ordnung 1.

Wir machen also in jedem Schritt jetzt zwei Fehler: Einmal den lokalen Diskretisierungsfehler, und zusätzlich den Abbruchfehler, der dadurch entsteht, dass wir die Fixpunktiteration nach dem $p.$ Schritt abbrechen.

Die Kontraktionskonstante im Fixpunktsatz von Banach war Lh. Wir gewinnen also in jedem Schritt der Fixpunktiteration einen Faktor $O(h)$, nach dem $p.$ Schritt ist unsere Approximation von der Ordnung $O(h^p)$. Der lokale Diskretisierungsfehler

erhöht sich also, wenn man die implizite Gleichung durch p Schritte der Fixpunktiteration ersetzt, um $O(h^p)$. Es liegt nahe, das p gerade so zu wählen, dass der lokale Diskretisierungsfehler asymptotisch ebenso groß ist wie der Fehler, der durch den Abbruch der Fixpunktiteration entsteht.

Satz 4.36 (Durchführung der impliziten Verfahren)

Gegeben sei ein implizites numerisches Einschrittverfahren der Konsistenzordnung p und seien alle Voraussetzungen des Konvergenzsatzes gegeben. Falls die Lösung der impliziten Gleichung ersetzt wird durch das p. Folgeglied der Fixpunktiteration mit Anfangspunkt y_k in jedem Schritt, so ist das entstehende Verfahren immer noch konvergent von der Ordnung p.

Beweis: Der lokale Diskretisierungsfehler des impliziten Verfahrens ist $O(h^p)$, er erhöht sich noch einmal um $O(h^p)$, also ergibt sich keine Änderung in der Konvergenzrate. □

Bemerkung: Hierdurch wird natürlich das implizite zu einem expliziten Verfahren. Macht man weniger als p Schritte der Fixpunktiteration, so dominiert der Fehler bei der Lösung der Gleichung den lokalen Diskretisierungsfehler, macht man mehr als p Schritte, so arbeitet man zu viel, ohne dass das Ergebnis substantiell besser wird. In Abbildung 4.7 wird dies deutlich: Wählt man im impliziten Eulerverfahren die Anzahl der Fixpunktiterationen $= 1$, so erhält man praktisch das Ergebnis aus dem expliziten Eulerverfahren. Wählt man zwei Fixpunktiterationen, so benötigt man zwei Auswertungen pro Schritt, die Geraden im $\log\log$–Plot werden aber nicht steiler, man bekommt keine höhere Konvergenzordnung.

Für die implizite Trapezregel: Führt man hier nur einen Schritt der Fixpunktiteration durch, so erzeugt man einen lokalen Diskretisierungsfehler der Ordnung 1, und das Verfahren ist insgesamt nur noch von der Ordnung 1. Bei zwei Fixpunktiterationen erhält man die Ordnung 2, bei drei Iterationen bleibt die Ordnung bei 2.

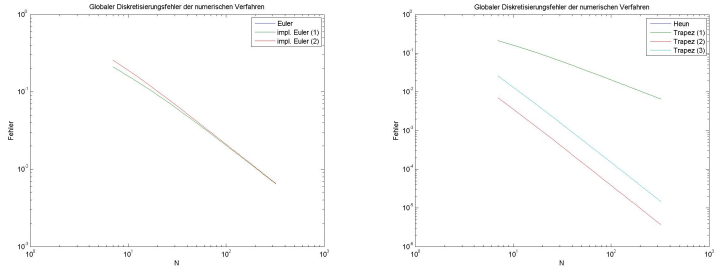


Abbildung 4.7: Fehler der impliziten Verfahren in Abhängigkeit von $h = 1/N$, in Klammern die Anzahl der Schritte der Fixpunktiteration.

Klick für Bild gdgleinschrittasympimpl
Klick für Matlab Figure gdgleinschrittasympimpl
Klick für Bild gdgltrapezasymp
Klick für Matlab Figure gdgltrapezasymp

```
function out = impliciteuler( f,x,y,h,p )
%IMPLICITEULER
y1=y;
for i=1:p
 y1=y+h*impgl(f,x,y,y1,h);
end
```

Listing 4.8: Implizite Trapezregel (gdgl/trapez.m)

Klicken für den Quellcode von gdgl/trapez.m

```
function out = impliciteuler( f,x,y,h,p )
%IMPLICITEULER
y1=y;
for i=1:p
 y1=y+h*impgl(f,x,y,y1,h);
end
```

Listing 4.9: Implizites Eulerverfahren (gdgl/impliciteuler.m)

Klicken für den Quellcode von gdgl/impliciteuler.m

4.8 Konstruktion von Einschrittverfahren

Wir wollen nun Verfahrensfunktionen φ für Einschrittverfahren konstruieren. Schon bei der Definition der Verfahrensfunktion in 4.24 hatten wir gesehen, dass

möglichst

$$h\varphi(t_k, y(t_k), h_k) = \int_{t_k}^{t_{k+1}} f(s, y(s))\mathsf{ds}$$

gelten sollte. Wir suchen also numerische Näherungsformeln für dieses Integral. Zunächst zeigen wir mit Hilfe der Taylorverfahren, dass es Einschrittverfahren beliebig hoher Ordnung gibt.

4.8.1 Taylor–Verfahren

Satz 4.37 (Taylor–Verfahren)
Sei $f \in C^p$, und alle Ableitungen seien lipschitzstetig im zweiten Argument mit der gemeinsamen Lipschitzkonstante L.

Wir definieren die Verfahrensfunktion φ für ein explizites Einschrittverfahren durch

$$\varphi_p(t_k, y_k, h_k) := \sum_{j=1}^{p} \frac{h_k^{j-1}}{j!} \widetilde{y}^{(j)}(t_k) = \sum_{j=1}^{p} \frac{h_k^{j-1}}{j!} \left(\frac{d}{dt}\right)^{j-1} f(t, \widetilde{y}(t))|_{(t_k, y_k)}$$

wobei $\widetilde{y}$ die Lösung der Differentialgleichung aus 4.20 mit Anfangswert (t_k, y_k) ist.

Dann ist das zugehörige Einschrittverfahren konvergent von der Ordnung p. Das Verfahren heißt Taylor–Verfahren der Ordnung p.

Beweis: φ ist lipschitzstetig in y_k nach Voraussetzung. Für den lokalen Diskretisierungsfehler gilt

$$\begin{aligned}
\tau_h(t, \widetilde{y}(t)) &= \frac{1}{h}(\widetilde{y}(t+h) - (\widetilde{y}(t) + h\varphi(t, \widetilde{y}(t), h)))) \\
&= \sum_{j=1}^{p} \frac{h^{j-1}}{j!} + O(h^p) - \sum_{j=1}^{p} \frac{h^{j-1}}{j!} h^{j-1} \\
&= O(h^p).
\end{aligned}$$

Also ist das Verfahren konsistent und konvergent von der Ordnung p nach Satz 4.29. □

Es sieht zunächst so aus, als sei das nicht sehr nützlich: Das in der Definition von φ_p auftretende $\widetilde{y}$ kennt man natürlich nicht. Tatsächlich braucht man es aber auch gar nicht. Der Wert von $\widetilde{y}$ wird nicht benutzt, die Ableitungen, die auftreten, lassen sich wegen $\widetilde{y}' = f$ alle durch Auswertungen von f und seinen Ableitungen an der Stelle (t_k, y_k) berechnen. Dies haben wir bereits beim verbesserten Eulerverfahren in 4.27 nachgerechnet.

Beispiel 4.38 *(Taylor–Verfahren, $p = 2$)*
Sei

$$f(t, y) = (1 + y^2)$$

und $\widetilde{y}$ eine Lösung der Differentialgleichung. Wegen

$$\begin{aligned}\widetilde{y}''(t) = \frac{d}{dt} f(t, \widetilde{y}(t)) &= f_t(t, \widetilde{y}(t)) + \widetilde{y}'(t, \widetilde{y}(t)) f_y(t, \widetilde{y}(t)) \\ &= f_t(t, \widetilde{y}(t)) + f(t, \widetilde{y}(t)) \cdot f_y(t, \widetilde{y}(t))\end{aligned}$$

gilt

$$\begin{aligned}\varphi(t_k, y_k, h_k) &= f(t_k, y_k) + \frac{h_k}{2}(f_t(t_k, y_k) + f(t_k, y_k) f_y(t_k, y_k)) \\ &= (1 + y_k^2) + \frac{h_k}{2}(1 + y_k^2)(2y_k) \\ &= (1 + y_k^2)(1 + h_k y_k).\end{aligned}$$

Mit Hilfe der Taylor–Entwicklung lassen sich also explizite Einschrittverfahren beliebig hoher Ordnung definieren.

Das Problem der Taylor–Verfahren ist, dass hohe Ableitungen von f explizit ausgerechnet werden müssen. Dies ist meist nicht möglich, etwa weil die analytische Form von f gar nicht bekannt ist. Zusätzlich gibt es keine feste, implementierbare Formel, für jedes f muss noch einmal gerechnet werden.

Andererseits liefert z.B. das Verfahren von Heun ebenfalls ein explizites Verfahren mit der gleichen Konvergenzordnung, aber es benutzt nur Auswertungen von f, nicht seiner Ableitungen. Es stellt sich die Frage: Kann man numerische Verfahren (beliebig hoher Ordnung) konstruieren, die nur Auswertungen von f benutzen? Dies liefert die Standard–Verfahren der Numerischen Mathematik, die Runge–Kutta–Verfahren, die in allen Bibliotheken zur numerischen Lösung gewöhnlicher Differentialgleichungen implementiert sind.

4.8.2 Runge–Kutta–Verfahren

Der Runge–Kutta–Ansatz liefert Verfahren beliebig hoher Ordnung, die nur Auswertungen von f benutzen. Wir betrachten zunächst noch einmal das Verfahren von Heun. Die Schrittfunktion φ war hier definiert durch

$$\varphi(t_k, y_k, h_k) = \frac{1}{2}(f(t_k, y_k) + f(t_{k+1}, y_k + h_k f(t_k, y_k))).$$

Dies schreiben wir in der Form:

$$\begin{array}{rcl} f_1 & = & f(t_k, y_k) \\ f_2 & = & f(t_k + h_k, y_k + hf_1) \\ \hline \varphi(t_k, y_k, h_k) & = & \frac{1}{2}f_1 + \frac{1}{2}f_2 \end{array}$$

Diese Schreibweise legt die folgende Definition nahe.

Definition 4.39 (Definition der Runge–Kutta–Verfahren)

1. *Seien α_j, γ_j, β_{jl} fest gewählt, $j = 1 \ldots m$, $l = 1 \ldots j-1$. Die Schrittfunktion φ sei definiert durch*

$$\varphi(t_k, y_k, h_k) = \gamma_1 f_1 + \gamma_2 f_2 + \ldots + \gamma_m f_m = \sum_{j=1}^{m} \gamma_j f_j$$

mit

$$\begin{array}{rcl} f_1 & = & f(t_k + \alpha_1 h_k, y_k) \\ f_2 & = & f(t_k + \alpha_2 h_k, y_k + h_k(\beta_{2,1} f_1)) \\ & \vdots & \\ f_m & = & f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m-1} \beta_{ml} f_l). \end{array}$$

Dann heißt das zugehörige numerische Verfahren m–stufiges explizites Runge–Kutta–Verfahren.

2. *Seien α_k, γ_k, β_{kl} fest gewählt, $k = 1 \ldots m$, $l = 1 \ldots m$. Die Schrittfunktion φ sei definiert durch*

$$\varphi(t_k, y_k, h_k) = \gamma_1 f_1 + \gamma_2 f_2 + \ldots \gamma_m f_m$$

mit

$$\begin{array}{rcl} f_1 & = & f(t_k + \alpha_1 h_k, y_k + h_k \sum_{l=1}^{m} \beta_{1,l} f_l) \\ f_2 & = & f(t_k + \alpha_2 h_k, y_k + h_k \sum_{l=1}^{m} \beta_{2,l} f_l) \\ & \vdots & \\ f_m & = & f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m} \beta_{m,l} f_l). \end{array}$$

Dann heißt das zugehörige numerische Verfahren m–stufiges implizites Runge–Kutta–Verfahren.

Natürlich sind die expliziten Verfahren implizite Verfahren, bei denen wir setzen

$$\beta_{jl} := 0 \text{ für } l \geq j.$$

Satz 4.40 (Ordnung der Runge–Kutta–Verfahren)

1. *Sei*

$$\sum_{j=1}^{m} \gamma_j = 1.$$

Dann ist das zugehörige Runge–Kutta–Verfahren mindestens konvergent von der Ordnung 1 für alle $f \in C^1$.

2. *Sei $f \in C^2$ und*

$$\sum_{j=1}^{m} \gamma_j = 1$$
$$\sum_{l} \beta_{jl} = \alpha_j$$
$$\sum_{j=1}^{n} \alpha_j \gamma_j = \frac{1}{2}.$$

Dann ist das zugehörige Runge–Kutta–Verfahren mindestens konvergent von der Ordnung 2.

Wegen dieses Satzes betrachten wir grundsätzlich nur Runge–Kutta–Verfahren, die die Normierungsbedingungen

$$\sum_{j=1}^{m} \gamma_j = 1, \sum_{l} \beta_{jl} = \alpha_j \forall j = 1 \dots m$$

erfüllen.

Beweis: Für Lipschitz–stetiges f ist auch φ Lipschitz–stetig für alle Runge–Kutta–Verfahren.

Zum Beweis der Konsistenz machen wir wieder die Taylorentwicklung. Sei also wieder y irgendeine Lösung der Differentialgleichung. Zunächst berechnen wir eine

Taylorentwicklung der auftretenden Zwischenstufen f_j. Wir setzen die Normierung für α gleich ein.

$$\begin{aligned} f_j &= f(t,y(t)) + \alpha_j h f_t(t,y(t)) + h\sum_{l=1}^{m} \beta_{m,l} f_l f_y(t,y(t)) + O(h^2) \\ &= f(t,y(t)) + \alpha_j h f_t(t,y(t)) + h\sum_{l=1}^{m} \beta_{m,l} (f(t,y(t)) + O(h)) f_y(t,y(t)) + O(h^2) \\ &= y'(t) + h\alpha_j (f_t(t,y(t)) + f(t,y(t)) f_y(t,y(t))) + O(h^2) \\ &= y'(t) + h\alpha_j y''(t) + O(h^2). \end{aligned}$$

Dabei haben wir die Formel aus 4.38 für die Berechung von y'' benutzt. Eingesetzt in die Definition des lokalen Diskretisierungsfehlers

$$\begin{aligned} \tau_h(t,y(t)) &= \frac{1}{h}(y(t+h) - y(t)) - \varphi(t,y(t),h) \\ &= y'(t) + \frac{h}{2} y''(t) - \sum_j \gamma_j f_j + O(h^2) \\ &= y'(t) - \sum_{j=1}^{m} \gamma_j y'(t) + h(\frac{1}{2} - \sum_{j=1}^{m} \gamma_j \alpha_j) y''(t) + O(h^2). \end{aligned}$$

□

Vorlesungsnotiz: 4.6.2013

Üblicherweise werden Runge–Kutta–Verfahren durch die Butcher–Diagramme repräsentiert in der Form (hier für explizite Verfahren)

$$\begin{array}{c|ccccc} \alpha_1 & & & & & \\ \alpha_2 & \beta_{2,1} & & & & \\ \alpha_3 & \beta_{3,1} & \beta_{3,2} & & & \\ \vdots & & & & & \\ \alpha_m & \beta_{m,1} & \beta_{m,2} & \cdots & \beta_{m,m-1} & \\ \hline & \gamma_1 & \gamma_2 & \cdots & \gamma_{m-1} & \gamma_m \end{array}$$

Für implizite Verfahren ist das Diagramm natürlich komplett ausgefüllt.

Beispiel 4.41 (Butcher–Diagramme)

1. *Euler*

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

2. *Implizites Eulerverfahren*

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

3. *Verbessertes Eulerverfahren*

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

4. *Verfahren von Heun*

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

5. *Standard–Runge–Kutta–Verfahren*

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Das Standard–Runge–Kutta–Verfahren hat die Ordnung 4.

Die Koeffizienten sollten natürlich so gewählt werden, dass die entstehende Ordnung möglichst hoch ist. Ein Trick dazu ist, sich zunächst auf autonome Gleichungen zu beschränken und nur dort die Verfahren zu entwickeln. Wir werden zeigen (4.46, dass diese Verfahren mit derselben Konsistenzordnung auf alle Gleichungssysteme anwendbar sind. Leider entstehen schon dabei nichtlineare (bilineare) Gleichungssysteme.

Schon für das Standard–Runge–Kutta–Verfahren ist der Aufwand, allein die Konsistenzordnung nachzurechnen, enorm. Das vereinfacht sich erheblich, wenn man sich auf autonome Gleichungen beschränkt. Wir untersuchen dies am Beispiel $m = 2$ und führen dieselbe Rechnung wie in 4.40 noch einmal für diesen Fall durch und zeigen, dass es keine zweistufigen Verfahren höherer Ordnung als 3 gibt. Wir setzen die dort gemachten Normierungsbedingungen voraus.

Die Anfangswertaufgabe 4.20 sei autonom, d.h.

$$y'(t) = f(y(t)).$$

Zunächst gilt

$$\frac{y(t+h) - y(t)}{h} = y'(t) + \frac{h}{2}y''(t) + \frac{h^2}{6}y'''(t) + O(h^3)$$

mit

$$\begin{aligned} y'(t) &= f(y(t)) \\ y''(t) &= f(y(t))f'(y(t)) \\ y'''(t) &= f'(y(t))^2 f(y(t)) + f(y(t))^2 f''(y(t)). \end{aligned}$$

Dann gilt

$$\begin{aligned} \varphi(t_k, y_k, h_k) &= \gamma_1 f_1 + \gamma_2 f_2 \\ &= \gamma_1 f(y_k) + \gamma_2 f(y_k + \beta_{2,1} h f(y_k)) \\ &= \gamma_1 f(y_k) + \gamma_2 (f(y_k) + \beta_{2,1} h_k f(y_k) f'(y_k) + \frac{\beta_{2,1}^2 h^2 f(y_k)^2}{2} f''(y_k) + O(h^3)) \end{aligned}$$

Durch Koeffizientenvergleich erhalten wir: Konsistente Verfahren der Ordnung 2 müssen die Gleichungen

$$\gamma_1 + \gamma_2 = 1,\ \gamma_2 \beta_{2,1} = \frac{1}{2}$$

erfüllen. Dies ist zum Beispiel durch die Wahl

$$\gamma_1 = 0,\ \gamma_2 = 1,\ \beta_{2,1} = \frac{1}{2}$$

(verbessertes Eulerverfahren) und

$$\gamma_1 = \frac{1}{2},\ \gamma_2 = \frac{1}{2},\ \beta_{2,1} = 1$$

(Verfahren von Heun) der Fall.

Die Gleichung

$$\gamma_2 \frac{\beta_{2,1}^2 f^2}{2} f'' = \frac{1}{6}(f'^2 f + f'' f^2)$$

ist nicht mehr allgemein zu erfüllen für alle $f \in C^3$. 2 ist also die maximale Konvergenzordnung eines expliziten 2–stufigen Runge–Kutta–Verfahrens.

Die maximale Konvergenzordnung eines expliziten p–stufigen Runge–Kutta–Verfahrens ist p. Leider wird diese Schranke nicht erreicht. Die folgende Tabelle gibt die höchste Konsistenzordnung expliziter Runge–Kutta–Verfahren an:

Stufenzahl	1	2	3	4	5	6	7	8	9	10	11
max. Ordnung	1	2	3	4	4	5	6	6	7	7	8

Für implizite Verfahren ist die Schranke $2p$, und sie wird auch angenommen. Ohne Beweis gilt der Satz:

Satz 4.42 (Konstruktion von impliziten Verfahren durch numerische Integration) *Gegeben sei ein numerisches Integrationsverfahren, das Polynome vom Grad p auf dem Intervall $[0,1]$ exakt integriert. Dann gibt es ein Runge–Kutta–Verfahren, das dieselben Auswertungspunkte (α_k) und Gewichte (γ_k) benutzt und die Ordnung $p+1$ hat. Umgekehrt: Hat ein Runge–Kutta–Verfahren die Ordnung p, so ist die zugehörige Integrationsformel von der Ordnung $p-1$.*

Beweis: Übungen, Hanke-Bourgeois [2006] Formel 76.3, 78.3.

Beweisidee: Es gilt

$$\begin{aligned}\frac{y(t+h)-y(t)}{h} - \varphi(t,y(t),h) &= \frac{1}{h}\int_t^{t+h} f(s,y(s))ds - \varphi(t,y(t),h)\\ &= \int_0^1 f(s,y(s))dx - \varphi(t,y(t),h)\end{aligned}$$

Nun wählen wir φ als Quadraturformel $I_h(f)$, die exakt ist bis zur Ordnung p. Entwickelt man nun $f(s,y(s))$ in seine Taylorreihe, so wird alles korrekt integriert bis zum Glied in s^{p+1}. □

Wählen wir als Integrationsregel die Gauss–Integration (Gauss–Verfahren), so gilt:

Korollar 4.43 (maximale Konsistenzordnung impliziter Verfahren)

Die maximale Konsistenzordnung eines impliziten Verfahren ist $2p$. Diese Grenze wird durch die Gauss–Verfahren erreicht.

Um später Extrapolationsverfahren anwenden zu können, bemerken wir:

Satz 4.44 (Taylorentwicklung des lokalen Diskretisierungsfehlers der Runge–Kutta–Verfahren)

Ein Runge–Kutta–Verfahren sei konsistent von der Ordnung p, und es sei $f \in C^q$,

$q > p$. Dann lässt sich der lokale Diskretisierungsfehler in eine Taylorreihe entwickeln, d.h. es gibt Funktionen F_k mit

$$\tau_h(t,y) = h^p F_p(t,y) + \ldots + h^{q-1} F_{q-1}(t,y) + O(h^q).$$

Beweis: Taylorentwicklung von τ_h. Die F_p sind einfach Summen von Produkten von Ableitungen von f. □

Satz 4.45 (Konvergenz der Runge–Kutta–Verfahren)

Runge–Kutta–Verfahren der Konsistenzordnung p sind konvergent von der Ordnung p.

Beweis: Die Schrittfunktion benötigt nur Auswertungen von f, und f ist lipschitzstetig in y. Also ist auch φ lipschitzstetig in y. □

Bemerkung: Wir haben uns hier stillschweigend auf die Analyse skalarer Runge–Kutta–Verfahren zurückgezogen. Tatsächlich sind Runge–Kutta–Verfahren der Ordnung p auch konsistent von der Ordnung p für Systeme von Differentialgleichungen.

Diese Bemerkung ermöglicht uns den Beweis des angekündigten Satzes über autonome Systeme.

Satz 4.46 (Umwandlung von Gleichungen in Systeme autonomer Gleichungen)
Jede skalare Differentialgleichung der Form $y'(t) = f(t, y(t))$ lässt sich schreiben als System autonomer Differentialgleichungen

$$(y_1'(t), y_2'(t)) = \tilde{f}(y_1(t), y_2(t)).$$

Beweis: Wähle

$$y_1'(t) = 1, y_1(a) = a$$

und

$$y_2'(t) = f(y_1(t), y_2(t)),\ y_2(0) = y_0.$$

Dies ist eine Anfangswertaufgabe für ein System autonomer Differentialgleichungen. Da aber offensichtlich

$$y_1(t) = t,$$

gilt

$$y_2'(t) = f(t, y_2(t)),\ y_2(0) = y_0$$

und damit ist y_2 Lösung der Anfangswertaufgabe. □

4.9 Extrapolation und Schrittweitensteuerung für Einschrittverfahren

Abschließend wollen wir, wie schon bei der Integration, mit Neville/Richardson/Romberg die Konsistenzordnung von Verfahren erhöhen, und Näherungen für den globalen Diskretisierungsfehler bestimmen. Wieder benötigen wir dazu zunächst einen Satz über die Taylorentwicklung des globalen Diskretisierungsfehlers.

Für den lokalen Diskretisierungsfehler bei Runge–Kutta–Verfahren haben wir dies bereits gezeigt (4.44). Die Vermutung ist, dass sich wieder mit Hilfe von Gronwall (4.28) der entsprechende Satz für den globalen Diskretisierungsfehler zeigen lässt.

Satz 4.47 (Taylorentwicklung des globalen Diskretisierungsfehlers)
Sei φ die Verfahrensfunktion eines expliziten Einschrittverfahrens mit der Konsistenzordnung p, und $f, \varphi \in C^{p+1}$, also

$$\tau_h(t, y(t)) = \tau(t)h^p + O(h^{p+1}).$$

Sei $\overline{y}(t)$ die Lösung von

$$\overline{y}'(t) = \rho(t)\overline{y}(t) - \tau(t),\ y'(a) = 0$$

mit

$$\rho(t) = f_y(t, y(t)).$$

Dann gilt

$$e_h(t_k) = y_h(t_k) - y(t_k) = \overline{y}(t_k)h^p + O(h^{p+1}).$$

Der globale Diskretisierungsfehler besitzt also eine Taylorentwicklung (und die Verfahren von Neville/Richardson/Romberg sind anwendbar).

Beweis: Sei ein Gitter I_h fest gewählt, y_h die numerische Näherung.

$$\begin{aligned} e_h(t_{k+1}) &= y_h(t_{k+1}) - y(t_{k+1}) \\ &= y_h(t_k) + h_k\varphi(t_k, y_h(t_k), h_k) \\ &\quad -(y(t_k) + h_k\varphi(t_k, y(t_k), h_k) + \tau(t_k)h_k^{p+1} + O(h_k^{p+2})) \\ &= e_h(t_k) + h_k e_h(t_k)\varphi_y(t_k, y(t_k), h_k) - \tau(t_k)h_k^{p+1} + O(|e_h(t_k)|^2 h_k + h_k^{p+2}). \end{aligned}$$

Hierbei haben wir im letzten Schritt das $\varphi(\ldots, y_h(t_k), \ldots)$ um $\varphi(\ldots, y(t_k), \ldots)$ entwickelt. Mit $\varphi \in C^2$ gilt

$$\begin{aligned} \varphi_y(t_k, y(t_k), h_k) &= \varphi_y(t_k, y(t_k), 0) + O(h_k) \\ &= f_y(t_k, y(t_k)) + O(h_k) \\ &= \rho(t_k) + O(h_k) \end{aligned}$$

und

$$e_h(t_{k+1}) \quad = \quad e_h(t_k) + h_k e_h(t_k)\rho(t_k) - h_k^{p+1}\tau(t_k) + O(h_k|e_h(t_k)|^2 + h_k^2|e_h(t_k)| + h_k^{p+2}).$$

Sei nun $e_h(t_k) = \overline{e}_h(t_k)h_k^p$. Eingesetzt in die Gleichung und direkt mit h_k^{-p} multipliziert erhalten wir

$$\overline{e_h(t_{k+1})} = \overline{e_h(t_k)} + h_k(\rho(t_k)\overline{e_h(t_k)} - \tau(t_k)) + O(h_k^2 + h_k^{p+1}|\overline{e}_h(t_k)|^2 + |\overline{e}_h(t_k)|h_k^2).$$

Dies ist bis auf einen Fehler $O(h^2)$, der bei der Berechnung des lokalen Diskretisierungsfehlers keine Rolle spielt, das Eulerverfahren für die oben angegebene Differentialgleichung. Also konvergieren die Gitterfunktionen e_h gegen die Lösung der Gleichung, und es gilt

$$\overline{e}_h(t_k) = \overline{y}(t_k) + O(h_k) \Rightarrow y(t_k) - y_h(t_k) = e_h(t_k) = h_k^p\overline{e}_h(t_k) = h_k^p\overline{y}(t_k) + O(h_k^{p+1}).$$

□

Dieses Argument lässt sich iterieren, auch die höheren Terme sind in ihre Taylorreihe entwickelbar. Alle Koeffizienten der Entwicklung können mit Hilfe der Differentialgleichung angegeben werden.

Die überraschende Aussage hier ist: Der erste Summand im Fehlerterm hängt überhaupt nicht vom Verfahren ab, sondern nur von der Differentialgleichung!

Korollar 4.48 (Extrapolation und Fehlerabschätzung für Einschrittverfahren)

1. *Seien $y_h(b)$, $y_{h/2}(b)$ die mit der Schrittweite h bzw. $h/2$ gewonnenen Näherungen an $y(b)$ für ein Verfahren der Ordnung p, und seien f, $\varphi \in C^2$. Dann gilt*

$$\frac{2^p y_{h/2}(b) - y_h(b)}{2^p - 1} = y(b) + O(h^{p+1}).$$

2. *Eine Abschätzung für den Fehler von $y_{h/2}(b)$ erhalten wir durch*

$$y(b) - y_{h/2}(b) = (y_{h/2}(b) - y_h(b))/(2^p - 1) + O(h^{p+1}).$$

Beweis: Durch Einsetzen. □

Da wir jetzt den globalen Diskretisierungsfehler abschätzen können, wählen wir die Schrittweite in jedem Schritt angepasst so, dass eine vorgegebene Fehlerschranke nicht unterschritten wird. Diese wrid am Anfang strenger sein als am Ende (wenn man eine konstante Fehlerbeschränkung einhalten will), denn die Verfahrensfehler am Anfang werden exponentiell verstärkt.

Leider ist dieser Algorithmus nicht so wertvoll wie der von Romberg für die Integralberechnung: Die Entwicklung des Fehlers geht in h, nicht in h^2, und die Näherung für $h/2$ liefert, ebenfalls anders als bei Romberg, nicht automatisch die Näherung für h ohne weitere Auswertungen von f.

Man wendet daher Paare von Runge–Kutta–Verfahren mit gleichen Auswertungen und unterschiedlichen Konvergenzordnungen an, z.B. das Verfahren von Dormand–Prince. Dies ist das Standardverfahren zur Lösung gewöhnlicher Differentialgleichungen in Matlab unter dem Namen ode45.

0							
$1/5$	$1/5$						
$3/10$	$3/40$	$9/40$					
$4/5$	$44/45$	$-56/15$	$32/9$				
$8/9$	$19372/6561$	$-25360/2187$	$64448/6561$	$-212/729$			
1	$9017/3168$	$-355/33$	$46732/5247$	$49/176$	$-5103/18656$		
1	$35/384$	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	
	$5179/57600$	0	$7571/16695$	$393/640$	$-92097/339200$	$187/2100$	$1/40$
	$35/384$	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	0

Die obere Zeile der γ_k liefert ein Verfahren fünfter Ordnung, die untere ein Verfahren vierter Ordnung.

In jedem Schritt berechnen wir nun zuerst die gemeinsamen Zwischenstufen f_j der Runge–Kutta–Verfahren. Anschließend berechnen wir für ein h_k zwei Näherungen y'_{k+1} und y''_{k+1} mit den beiden Verfahren, die sich nur durch die Koeffizienten γ_j unterscheiden. Der Aufwand dafür ist also gering. Mit dem folgenden Lemma schätzen wir dann den lokalen Diskretisierungsfehler in Abhängigkeit von h_k ab und wählen das h_{k+1} für den nächsten Schritt.

Satz 4.49 *(Bestimmung des Diskretisierungsfehlers aus Doppelverfahren)*
Es seien φ_1, φ_2 Verfahrensfunktionen der Konvergenzordnung $O(h^p)$ bzw. $O(h^q)$, $p > q$. Sei y die Lösung des Anfangswertproblems 4.20.

Es sei für ein $h_k > 0$

$$y'_{k+1} := y_k + \varphi_1(t_k, y_k, h_k)$$
$$y''_{k+1} = y_k + \varphi_2(t_k, y_k, h_k).$$

Weiter sei

$$y(t_k) - y'_k = C(t_k)h^p + O(h^{p+1})$$
$$y(t_k) - y''_k = C(t_k)h^q + O(h^{q+1}).$$

Der Entwicklungsterm ist dabei nicht abhängig von φ nach 4.47.

Dann gilt

$$C(t_k)h^p \sim \frac{y_k'' - y_k'}{h^p - h^q}h^p + O(h^{p+1}).$$

Beweis: Durch Einsetzen. □

Damit können wir also den Diskretisierungsfehler abschätzen. Wir wenden dies in einem Programmierbeispiel für unser Standardproblem $y'(t) = 1 + y(t)^2$ an..

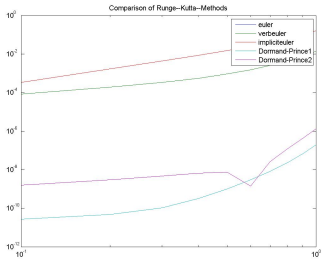


Abbildung 4.8: Vergleich einiger Runge–Kutta–Verfahren

Klick für Bild rungekuttacompare
Klick für Matlab Figure rungekuttacompare

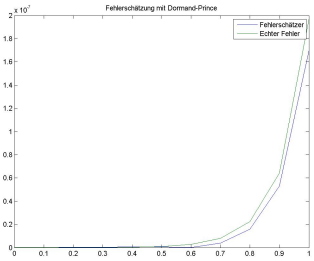


Abbildung 4.9: Fehlerschätzung mit Dormand–Prince

Klick für Bild fehlerschaetzung
Klick für Matlab Figure fehlerschaetzung

```
function y = rungekutta( alpha,beta,gamma,f,t,yo,h,M ,L)
%RUNGEKUTTA Allgemeiner Schritt eines Runge-Kutta-Verfahrens
%M=Ordnung des Verfahrens
%Laesst Systeme und implizite Verfahren zu.
```

```
%L=Anzahl der Schritte.
```

Listing 4.10: Runge–Kutta–Verfahren (gdgl/rungekutta.m)

Klicken für den Quellcode von gdgl/rungekutta.m

```
function [ alpha,beta,gamma,M ] = setup_rungekutta( name )
%SETUP_RUNGEKUTTA
if (strcmp(name,'euler'))
    alpha=0;
    beta=0;
    gamma=1;
```

Listing 4.11: Runge–Kutta–Setup (gdgl/setuprungekutta.m)

Klicken für den Quellcode von gdgl/setuprungekutta.m

```
function rungekuttatest(  )
%RUNGEKUTTATEST
N=10;
a=0;
b=1;
yo=0;
```

Listing 4.12: Runge–Kutta–Testprogramm (gdgl/rungekuttatest.m)

Klicken für den Quellcode von gdgl/rungekuttatest.m

Vorlesungsnotiz: Das Programm ist nur in Teilen durchkorrigiert (implizite Verfahren, Systeme).

4.10 Steifigkeit und A-stabile Verfahren

Zu guten numerischen Verfahren zur Lösung von gewöhnlichen Differentialgleichungen gehört noch etwas mehr als nur Stabilität und Konvergenz. Wir schauen zunächst auf ein Beispiel, in dem zwei Einschrittverfahren mit derselben Konsistenzordnung unterschiedliche physikalische Deutungen liefern.

Beispiel 4.50 (Dispersion)
Wir betrachten einen senkrechten Steinwurf mit Masse 1. Seien $v(t)$, $h(t)$ Geschwindigkeit und Höhe des Steins, g die Erdbeschleunigung. v und h erfüllen die Transportgleichung

$$v'(t) = -g,\ h'(t) = v(t).$$

Dahinter steht der Satz über die Energieerhaltung, die Summe aus potentieller und kinetischer Energie ist konstant. Sei

$$E(t) = gh(t) + \frac{v(t)^2}{2}.$$

Dann ist

$$E'(t) = gv - vg = 0.$$

Die Gesamtenergie ist also konstant. Ein gutes numerisches Verfahren sollte diese Größe auch erhalten. Für das Eulerverfahren gilt

$$\begin{aligned} v_{k+1} &= v_k - \Delta t g \\ h_{k+1} &= h_k + \Delta t v_k \\ E_{k+1} &= g h_{k+1} + \frac{v_{k+1}^2}{2} \\ &= g(h_k + \Delta t v_k) + \frac{(v_k - \Delta t g)^2}{2} \\ &= (g h_k + \frac{v_k^2}{2}) + \frac{\Delta t^2 g^2}{2} = E_k + \frac{\Delta t^2 g^2}{2} \end{aligned}$$

Die Energie bleibt also nicht konstant, es tritt ein kleiner Fehlerterm auf, die "Dispersion", eine Art künstliche Energiegewinnung. Sie geht zwar für Δt gegen 0 auch gegen 0, aber sie macht physikalisch hier keinen Sinn. Für das verbesserte Eulerverfahren gilt

$$\begin{aligned} y_{k+1} &= y_k + h f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f(t_k, y_k)) \\ v_{k+1} &= v_k - \Delta t g \\ h_{k+1} &= h_k + \Delta t (v_k - \frac{\Delta t}{2} g) \\ E_{k+1} &= g(h_k - \Delta t (v_k + \frac{\Delta t}{2} g)) + \frac{(v_k - \Delta t g)^2}{2} \\ &= g h_k + \frac{v_k^2}{2} = E_k \end{aligned}$$

Das Verfahren ist also energieerhaltend, unabhängig von der Schrittweite. Wir werden für diese konkrete Differentialgleichung also mit dem verbesserten Eulerverfahren (mal ganz abgesehen von der Konsistenzordnung) physikalisch sinnvollere Ergebnisse erwarten.

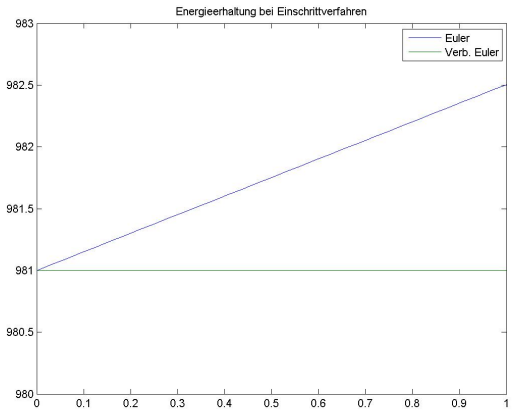


Abbildung 4.10: Energieerhaltung bei Einschrittverfahren

Klick für Bild Energieerhaltung
Klick für Matlab Figure Energieerhaltung

```
function [ output_args ] = energieerhaltung( input_args )
%ENERGIEERHALTUNG
function en=energie(y)
en=y(:,2)*g+y(:,1).*y(:,1)/2;
end
```

Listing 4.13: Energieerhaltung bei Einschrittverfahren (gdgl/energieerhaltung.m)

Klicken für den Quellcode von gdgl/energieerhaltung.m

Wir versuchen, die Verfahren so zu wählen, dass möglichst viele Eigenschaften der Lösungen der Differentialgleichung für die Lösungen der Differenzengleichungen erhalten bleiben. Wir betrachten dazu

Definition 4.51 (Modellproblem von Dahlquist)

$$y'(t) = \lambda y(t),\ y(0) = 1,\ \lambda \in \mathbb{C},\ Re\lambda < 0.$$

Die analytische Lösung in einer Dimension ist $y(t) = \exp(\lambda t)$. Offensichtlich geht die Lösung exponentiell gegen 0 für $t \to \infty$, und das erwarten wir auch von der numerischen Approximation, unabhängig von h.

Bemerkung: Lokal kann jede Differentialgleichung auf ein System solcher Modellgleichungen zurückgeführt werden (siehe Hanke-Bourgeois [2006], Abschnitt 77).

Definition 4.52 (A-Stabilität)
Ein Verfahren heißt A–stabil, wenn für das Dahlquist–Problem und äquidistante Gitter mit Schrittweite h die numerische Approximation im Betrag monoton fallend ist für alle h und alle λ mit Realteil kleiner 0, dass also gilt

$$|y_h(t_k)| \leq |y_h(t_{k+1})| \, \forall k.$$

Für Runge–Kutta–Verfahren lässt sich die A–Stabilität einfach überprüfen.

Satz 4.53 (Stabilitätsfunktion und Stabilitätsbereich von Runge–Kutta–Verfahren)

Für die durch Runge–Kutta–Verfahren gelieferten Näherungen für die Lösung des Dahlquist–Problems gilt

$$y_{k+1} = R(\lambda h) y_k$$

Für explizite Verfahren ist R ein Polynom, für implizite Verfahren eine rationale Funktion. Der Bereich von $\mathbb{C}$ mit $|R(z)| < 1$ heißt Stabilitätsbereich.

Es gilt

$$|y_h(t_k)| \leq |y_h(t_{k+1})| \, \forall k.$$

genau dann, wenn λh im Stabilitätsbereich liegt. Das Verfahren ist A–stabil genau dann, wenn der Stabilitätsbereich die komplette linke komplexe Halbebene umfasst.

Beweis: Übungen.

Beispiel 4.54 (Beispiele zur Stabilitätsfunktion)

1. *Euler explizit:*

   $$y_{k+1} = y_k + hf(t_k, y_k) = y_k + h\lambda y_k = (1 + h\lambda) y_k.$$

   Es ist also $R(z) = 1 + z$. Das Verfahren ist nicht A–stabil. Das Stabilitätsgebiet ist ein Kreis um -1 mit Radius 1.

2. *Euler implizit:*

   $$y_{k+1} = y_k + hf(t_k, y_{k+1}) = y_k + h\lambda y_{k+1} = \frac{1}{1 - h\lambda} y_k.$$

   Die Stabilitätsfunktion ist

   $$R(z) = \frac{1}{1 - z}.$$

   Es gilt $|R(z)| < 1$ außerhalb eines Kreises mit Radius 1 um die 1. Die linke komplexe Halbebene ist ganz im Stabilitätsbereich enthalten. Das Verfahren ist A–stabil.

Eine nette Visualisierung dieser Stabilität findet sich auf http://wwwisg.cs.uni-magdeburg.de/kontsim/applet/.

Satz 4.55 (A–Stabilität expliziter Runge–Kutta–Verfahren)

Kein explizites Runge–Kutta–Verfahren ist A–stabil. Wenn wir implizite Verfahren näherungsweise als explizite Verfahren lösen durch die Fixpunktiterationen, ist kein bisher betrachtetes Verfahren A–stabil.

Beweis: Für explizite Verfahren ist R ein Polynom p. Sei

$$p(x) = a_n x^n + \sum_{k=0}^{n-1} a_k x^k,\ a_n \neq 0.$$

Für

$$|x| \geq \max((\sum_{k=0}^{n-1} |a_k| + 2)/a_n, 1) =: R$$

gilt

$$\begin{aligned} |p(x)| &\geq a_n|x|^n - (\sum_{k=0}^{n-1} |a_k|)|x|^{n-1} \\ &= |x|^{n-1}(a_n|x| - \sum_{k=0}^{n-1} |a_k|) \\ &\geq |x| \\ &\geq 2. \end{aligned}$$

Also liegt das Stabilitätsgebiet ganz im Kreis um 0 mit Radius R. □

Satz 4.56 (Taylorentwicklung der Stabilitätsfunktion)

Sei $R(x)$ die Stabilitätsfunktion eines Runge–Kutta–Verfahrens der Ordnung q. Dann gilt

$$|R(h) - e^h| = O(|h|^{q+1}) \text{ für } h \to 0.$$

Die Taylorentwicklungen von R und e^x an der Stelle 0 stimmen also bis zum q. Glied überein.

Beweis: Wir wenden das Verfahren an auf das Modellproblem 4.51. Da es konsistent ist von der Ordnung q, gilt für $h > 0$

$$\begin{aligned} R(h) - e^h &= R(h) \cdot 1 - e^h \\ &= y_1 - y(h) \\ &= O(h^{q+1}). \end{aligned}$$

R ist rationales Polynom, also außerhalb der Nullstellen des Nenners holomorph, daraus folgt der Satz für alle komplexen h. □

Wir nutzen dies, um zu folgern:

Korollar 4.57 *Die 0 liegt immer auf dem Rand des Stabilitätsgebiets.*

Beweis: Nach 4.56 gilt

$$R(h) - 1 = h + O(h^2).$$

Insbesondere ist $R(0) = 1$, also liegt die 0 im Stabilitätsgebiet. $R(h) - 1$ wechselt für reelle h bei 0 sein Vorzeichen, links vom Nullpunkt ist das Verfahren stabil, rechts nicht. □

Die Bedingung der A–Stabilität ist besonders für große λ schwer zu erfüllen. Da das Stabilitätsgebiet beschränkt ist, sollten wir nach dem letzten Satz h dann klein wählen.

Für skalare Differentialgleichungen erster Ordnung ist das aber kein Problem: Wenn λ größer wird, so wird die Lösung steiler, also muss auch die Diskretisierung um den Faktor λ verfeinert werden, um Genauigkeitsanforderungen erfüllen zu können. Wir müssen also ohnehin ein kleines h wählen, die zusätzliche Stabilitätsanforderung fällt nicht ins Gewicht. Anders sieht es bei Systemen aus.

Wir betrachten nun ein Anfangswertproblem der Dimension 2 der Form

$$y'(t) = Ay(t)$$

mit

$$A = \begin{pmatrix} \lambda_1 + \lambda_2 & \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_2 & \lambda_1 + \lambda_2 \end{pmatrix}.$$

Die analytischen Lösungen sind von der Form

$$y(t) = \begin{pmatrix} ae^{\lambda_1 t} + be^{\lambda_2 t} \\ ae^{\lambda_1 t} - be^{\lambda_2 t} \end{pmatrix}$$

für a und b in $\mathbb{C}$ fest.

Wir nehmen an, dass die Realteile der λ_k negativ sind, und dass $|\mathrm{Re}\lambda_1| >> |\mathrm{Re}\lambda_2|$. In der Lösung fällt dann $ae^{\lambda_1 t}$ schnell, spielt also für die Lösung praktisch keine Rolle. $be^{\lambda_2 t}$ dominiert die Lösung, daher sollte die Genauigkeit an diesem Term ausgerichtet werden (und damit die Wahl der Schrittweite). Eine automatische Schrittweitensteuerung wird auch genau das versuchen und scheitern, denn:

Da der Betrag des Realteils von λ_1 sehr groß ist, muss bei nicht A–stabilen Verfahren die Schrittweite klein gewählt werden. Wir stehen also vor dem Dilemma: Wir könnten die Schrittweite eigentlich groß wählen für die Genauigkeit, müssen sie aber klein wählen, um in den Stabilitätsbereich zu gelangen. Differentialgleichungen mit dieser Eigenschaft nennt man **steife Differentialgleichungen**. Dies ist die Motivation für den Satz:

Satz 4.58 (allgemeine steife Differentialgleichungen)

Sei $y'(t) = f(t, y(t))$ ein System von gewöhnlichen Differentialgleichungen. Sei J die Jakobimatrix von f in einem Punkt. Falls J diagonalisierbar ist und nur Eigenwerte mit negativem Realteil besitzt, bei denen sich der Betrag des Realteils stark unterscheidet, so ist die Differentialgleichung steif.

Beweis: Linearisierung der Differentialgleichung. □

Man kann sich fragen, warum wir die Stabilität gerade an den Lösungen der Modellgleichung mit negativem Realteil von λ festgemacht haben. Könnten wir nicht auch eine Stabilität für positive Realteile definieren?

Ja, aber dies würde wenig Sinn machen. Da bei positivem Realteil die Lösung und die Genauigkeitsschranken durch das λ mit dem größten Realteil bestimmt werden, tritt das Problem der Steifigkeit nicht auf.

Kapitel 5

Lineare Mehrschrittverfahren

Bevor wir uns nun den Mehrschrittverfahren zuwenden, wiederholen wir noch einmal die zentralen Begriffe der Einschrittverfahren:

- Konsistenz: Das Verfahren ist Diskretisierung der Differentialgleichung. Der lokale Diskretisierungsfehler, bei dem man die echte Lösung y von 4.20 in die diskretisierte Gleichung eingeht, geht für kleine Schrittweiten h gegen 0.
- Konvergenz: Das Verfahren liefert für eine Familie von Gittern, deren Feinheit gegen 0 geht, die exakte Lösung (dies ist das eigentliche Ziel).
- Stabilität: Kleine Fehler im einzelnen Schritt führen zu kleinen Gesamtfehlern, begründet den Satz: Aus Konsistenz folgt Konvergenz (für Einschrittverfahren), und ist eine Folgerung der Gronwallschen Ungleichung.

5.1 Definition

Die Idee bei den Mehrschrittverfahren ist, die vergangenen Funktionsauswertungen bei der Berechnung der nächsten Approximation mitzunehmen. Wir erhoffen uns dadurch eine deutliche Verringerung des notwendigen Aufwands oder eine deutliche Erhöhung der möglichen Konsistenzordnung. In der Vorgehensweise ähneln die Einschrittverfahren den vielleicht aus der Stochastik bekannten gedächtnislosen Markovprozessen: Der nächste Wert hängt ausdrücklich nur vom aktuellen Zustand ab, nicht von einer Historie. Im Gegensatz dazu stehen die Mehrschrittverfahren.

Mehrschrittverfahren haben hohe Konsistenzordnungen bei wenigen Evaluationen (i.A. einer) von f. Auf der einen Seite sind sie nicht notwendig stabil, wie es die

Einschrittverfahren sind. Eine Schrittweitensteuerung ist schwierig. Daher sind sie häufig nicht die Standardsolver. In Matlab steht der Adams–Bashforth–Solver als Standard–Mehrschrittverfahren unter dem Namen ode113 zur Verfügung.

Wir werden in diesem Kapitel zunächst einige Verfahren herleiten, die Bezeichnungen der Einschrittverfahren auf Mehrschrittverfahren übertragen und überprüfen, welche Sätze erhalten bleiben. Zunächst schränken wir die komplette Betrachtung auf lineare Verfahren auf äquidistanten Gittern ein. Wir reduzieren die Definition aus 4.25 also auf

Definition 5.1 (lineare Mehrschrittverfahren)

Ein numerisches Verfahren, das auf einem **äquidistanten** *Gitter I_h mit Schrittweite h die Näherung $y_h(t_k) = y_k$ der Differentialgleichung berechnet mit*

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f(t_{k+j}, y_{k+j}),\ k = 0 \dots N - m$$

$(\alpha_m \neq 0)$ heißt lineares m–Schritt–Mehrschrittverfahren. Das Verfahren heißt explizit, falls $\beta_m = 0$, ansonsten implizit.

Wir dürfen also bei einem m–Mehrschrittverfahren zur Berechnung von y_{k+m} die letzen m Funktionsauswertungen y_k bis y_{k+m-1} noch mitbenutzen. Dies bedeutet natürlich auch, dass wir im ersten Schritt des Verfahrens das y_m berechnen und dafür y_0 bis y_{m-1} benötigen, wobei wir eigentlich nur y_0 haben. Die fehlenden Terme werden üblicherweise mit Einschrittverfahren berechnet, hat man alle zusammen, macht man ab hier mit den Mehrschrittverfahren weiter. Um diese Anlaufrechnung werden wir uns später kümmern.

Beispiel 5.2 (Mittelpunktregel)

$$y_{k+2} - y_k = 2hf(t_{k+1}, y_{k+1})$$

Die Definition der Konsistenz 4.26 ergibt für die Mehrschrittverfahren

Lemma 5.3 *Gegeben sei ein durch die Konstanten α_j und β_j beschriebenes lineares Mehrschrittverfahren. Sei y irgendeine Lösung der Differentialgleichung. Dann ist der lokale Diskretisierungsfehler gegeben durch*

$$\tau_h(t, y(t)) = \frac{1}{h} \sum_{j=0}^{m} \alpha_j y(t + jh) - \sum_{j=0}^{m} \beta_j f(t + jh, y(t + jh)).$$

Wieder bekommen wir den lokalen Diskretisierungsfehler, indem wir die Lösungen y der Differentialgleichung in die diskrete Gleichung einsetzen.

Korollar 5.4 (Konsistenzordnung der Mittelpunktregel)

Sei $f \in C^2$. Dann hat die Mittelpunktregel die Konsistenzordnung 2.

Beweis: Wir benutzen für

$$y(t+2h) - y(t)$$

jeweils eine Taylorentwicklung um $y(t+h)$. Dann fallen alle Terme in h^j mit geradem j weg, und es gilt

$$\begin{aligned}\tau_h(t, y(t)) &= \frac{1}{h}(y(t+2h) - y(t)) - 2f(t+h, y(t+h)) \\ &= 2y'(t+h) + O(h^2) - 2y'(t+h).\end{aligned}$$

Wir haben also sofort ein explizites Verfahren mit höherer Ordnung als bei den Einschrittverfahren hergeleitet, dort war die Grenze p für explizite Verfahren p. Stufe. □

Unsere Idee zur Konstruktion ist: Wir wandeln unsere Differentialgleichung wieder in die äquivalente Integralgleichung 4.6 um, und benutzen unsere Quadraturformeln. Diese haben wir durch Integration des Interpolationspolynoms hergeleitet.

Für äquidistante Gitter kann man das Interpolationspolynom sehr einfach angeben. In diesem Fall vereinfacht sich die Form der dividierten Differenzen aus 2.8 nämlich ganz erheblich.

Definition 5.5 (Vorgängerfunktion und Rückwärtsdifferenz)

Sei (y_k) eine Folge reeller Zahlen. Dann heißt $(Ty) = (0, y_0, y_1, \ldots)$ die Vorgängerfunktion und

$$(\nabla y) = (I - T)y = (y_0, y_1 - y_0, y_2 - y_1, \ldots)$$

die Rückwärtsdifferenz.

Beispiel 5.6 (Potenzen der Rückwärtsdifferenz)

Sei wieder $y = (y_k)$.

$$(\nabla^2 y) = \nabla(y_0, y_1 - y_0, y_2 - y_1, \ldots) = (y_0, y_1 - 2y_0, y_2 - 2y_1 + y_0, \ldots)$$

also

$$(\nabla^2 y)_k = y_k - 2y_{k-1} + y_{k-2}$$

mit $y_k = 0$ für $k < 0$.

Lemma 5.7 *(Rechenregeln für die Rückwärtsdifferenz)*

$$(\nabla^m y)_k = \sum_{j=0}^{m} (-1)^j \begin{pmatrix} m \\ j \end{pmatrix} y_{k-j}$$

$$y_{k-m} = \sum_{j=0}^{m} (-1)^j \begin{pmatrix} m \\ j \end{pmatrix} (\nabla^j y)_k$$

(jeweils für $k \geq m$).

Beweis: $\nabla^m = (I - T)^m$ bzw. $T^m = (I - \nabla)^m$, und dann mit den binomischen Formeln. □

Satz 5.8 (Interpolationspolynom für äquidistante Stützstellen)

Sei $t_k = a + kh$, y_k gegeben für $k \in \mathbb{N}$. Für $k \geq m$ definieren wir

$$p_{k-m}(t) := \sum_{j=0}^{m} (-1)^j \begin{pmatrix} s \\ j \end{pmatrix} (\nabla^j y)_k,\ s = \frac{t_k - t}{h}.$$

Hierbei ist der Binomialkoeffizient für reelle t wie üblich definiert durch

$$\begin{pmatrix} t \\ j \end{pmatrix} := \frac{t(t-1)\cdots(t-j+1)}{j!} \in \mathcal{P}_j.$$

Dann ist für $k \geq 0$ p_k das eindeutig bestimmte Interpolationspolynom in $\mathcal{P}_m$ mit

$$p_k(t_{k+j}) = y_{k+j},\ j = 0 \ldots m.$$

Beweis:
Sei $k \geq m$.

1. p_{k-m} ist Polynom vom Grad $\leq m$, denn alle Binomialkoeffizienten sind in $\mathcal{P}_m$.
2. Sei $l \leq m$. Wegen

   $$\begin{pmatrix} (t_k - t_{k-l})/h \\ j \end{pmatrix} = \begin{pmatrix} l \\ j \end{pmatrix}$$

   gilt

$$\begin{aligned} p_{k-m}(t_{k-l}) &= \sum_{j=0}^{m} (-1)^j \underbrace{\begin{pmatrix} l \\ j \end{pmatrix}}_{=0 \text{ für } j > l,\ l \text{ ganz}} (\nabla^j y)_k \\ &= \sum_{j=0}^{l} (-1)^j \begin{pmatrix} l \\ j \end{pmatrix} (\nabla^j y)_k \\ &= y_{k-l}. \end{aligned}$$

□

Wir rechnen das Beispiel für $m = 2$. Dann gilt

$$\begin{aligned} p_{k-2}(t) &= \nabla^0 y_k - \binom{s}{1}\nabla y_k + \binom{s}{2}\nabla^2 y_k \\ &= y_k - \frac{t_k - t}{h}(y_k - y_{k-1}) + \frac{(t_k - t)(t_{k-1} - t)}{2h^2}(y_k - 2y_{k-1} + y_{k-2}). \end{aligned}$$

Hier sieht man sofort, dass die Formel nichts anderes ist als die Newtonsche Form des Interpolationspolynoms 2.9 für äquidistante Stützstellen.

5.2 Konstruktion von Mehrschrittverfahren durch Integration

Für die Lösung unserer Anfangswertaufgabe 4.20 gilt

$$y(t_{k+m}) - y(t_{k+m-r}) = \int_{t_k+m-r}^{t_{k+m}} f(t, y(t))\mathrm{dt}.$$

Für die Approximation ersetzen wir die Funktion unter dem Integralzeichen durch sein Interpolationspolynom p_k mit den Stützstellen $t_k + jh$ und den Stützwerten

$$f_{k+j} = f(t_{k+j}, y_{k+j}).$$

Wir erhalten damit z.B. für ein explizites Verfahren, das die Stützwerte f_k bis f_{k+m-1} benutzt

$$\begin{aligned} y_{k+m} - y_{k+m-r} &= \int_{t_{k+m-r}}^{t_{k+m}} p_k(t)\mathrm{dt} \\ &= h\sum_{j=0}^{m-1} \underbrace{(-1)^j \int_0^r \binom{s}{j} \mathrm{ds}}_{\gamma_j} \nabla^j f_{k+m-1} \end{aligned}$$

Die festen Koeffizienten γ_j sind vertafelt.

Für das Interpolationspolynom haben wir dann noch die Wahl zwischen den Interpolationsstellen t_k bis t_{k+m} (implizit) und t_k bis t_{k+m-1} (explizit). Für $r = 1$ erhalten wir die Verfahren von Adams–Bashforth und Adams–Moulton, für $r = 2$ die Verfahren von Nyström und Milne–Simpson. Wir fassen das Ergebnis in folgender Tabelle zusammen:

Interpolation benutzt	$r=1$	$r=2$	
$f_k \dots f_{k+m-1}$	Adams–Bashforth	Nyström	explizit
$f_k \dots f_{k+m}$	Adams–Moulton	Milne–Simpson	implizit

Als Beispiel betrachten wir die Koeffizienten von Adams–Bashforth:

$$\begin{aligned}
\gamma_j &= (-1)^j \int_0^1 \binom{s}{j} \mathsf{ds} \\
\gamma_0 &= \int_0^1 1 \mathsf{ds} = 1 \\
\gamma_1 &= \int_0^1 (s) \mathsf{ds} = \frac{1}{2} \\
&\vdots
\end{aligned}$$

Satz 5.9 *(Konsistenzordnung der durch Integration hergeleiteten MSV)*
Ein numerisches Verfahren mit m Schritten sei durch Integration des Interpolationspolynoms an m (explizit) bzw. $(m+1)$ (implizit) Stützstellen hergeleitet worden. Dann hat es die Konsistenzordnung m (explizit) bzw. $(m+1)$ (implizit).

Beweis: Für explizite Verfahren: Sei y eine Lösung der Differentialgleichung. Weiter sei p das Integrationspolynom, das an den Stellen $t+jh$ den Wert $y(t+jh)$ annimmt, $j = 0 \dots m-1$. Dann gilt nach Konstruktion der Integrationsformeln

$$\begin{aligned}
\tau_h(t, y(t)) &= \frac{1}{h}(y(t+mh) - y(t+(m-r)h)) - \sum_{j=0}^{m-1} \beta_j f(t+jh, y(t+jh)) \\
&= \frac{1}{h} \int_{t+(m-r)h}^{t+mh} y(t) \mathsf{dt} - \frac{1}{h} \int_{t+(m-r)h}^{t+mh} p(t) \mathsf{dt} \\
&= O(h^m)
\end{aligned}$$

nach 3.8. □

Wir rechnen als Beispiel ein Milne–Simpson–Verfahren mit $r=2$ und $m=2$. Das zugehörige Interpolationspolynom p_k hatten wir gerade oben berechnet, es gilt also mit $f_k = f(t_k, y_k)$

$$
\begin{aligned}
y_{k+2} - y_k &= \int_{t_k}^{t_{k+2}} p_k(t)\mathrm{dt} \\
&= h(\int_0^2 1\, ds f_k - \int_0^2 (2-s)\, ds(f_k - f_{k-1}) \\
&\quad + \frac{1}{2}\int_0^2 (2-s)(1-s)\, ds(f_k - 2f_{k+1} + f_{k-2})) \\
&= h(2f_k - 2(f_k - f_{k-1}) + \frac{1}{3}(f_k - 2f_{k-1} + f_{k+1}))
\end{aligned}
$$

und damit

$$
y_{k+2} = y_k + \frac{h}{3}(f_k + 4f_{k-1} + f_{k-2}).
$$

Dies ist die Milne–Regel.

Eine schöne Übersicht über all diese Verfahren mit Rechnungen und Beispielen findet sich (mit leicht anderen Bezeichnungen) in Hairer et al. [1993], Kapitel III.1.

Bemerkung: Dies ist ein phantastisches Ergebnis: Mit nur einer zusätzlichen Auswertung von f können beliebig hohe Konsistenzordnungen erreicht werden!

Alternativ können Mehrschrittverfahren auch durch Differentiation des Interpolationspolynoms hergeleitet werden. Sei p wieder das Interpolationspolynom mit

$$
p(t_{k+j}) = y_{k+j},\ j = 0 \ldots m,\ p \in \mathcal{P}_m.
$$

Dann setzen wir

$$
f(t_{k+l}, y_{k+l}) = p'(t_{k+l})
$$

und erhalten wieder eine Rechenvorschrift. Diese Verfahren nennt man Backward–Difference–Formulas (BDF).

Wir betrachten als Beispiel das implizite Verfahren für $l = 1$ und $m = 2$: Das Interpolationspolynom an den Stützstellen (t_k, t_{k+1}, t_{k+2}) mit Stützwerten (y_k, y_{k+1}, y_{k+2}) ist

$$
h^2 p(t) = \frac{1}{2}(t - t_{k+1})((t - t_k)y_{k+2} + (t - t_{k+2})y_k) - (t - t_k)(t - t_{k+2})y_{k+1},
$$

zur Abwechslung mal nicht mit den Rückwärtsdifferenzen, sondern mit der Formel von Lagrange. Die Ableitung an der Stelle t_{k+1} liefert

$$
2h^2 p'(t_{k+1}) = hy_{k+2} + hy_k.
$$

Das implizite Verfahren lautet also

$$f(t_{k+1}, y_{k+1}) = p'(t_{k+1}) = \frac{1}{2h}(y_{k+2} - y_k)$$

oder

$$y_{k+2} = y_k + 2hf(t_{k+1}, y_{k+1}),$$

wir erhalten also wieder die Mittelpunktregel.

5.3 Stabilität von Mehrschrittverfahren

Beispiel 5.10 *Es werde ein Verfahren möglichst hoher Ordnung der Form*

$$y_{k+2} - (1+\alpha)y_{k+1} + \alpha y_k = h\left(\frac{3-\alpha}{2}f_{k+1} - \frac{1+\alpha}{2}f_k\right)$$

gesucht. Sei $f \in C^3$. Wir bestimmen α durch Taylorentwicklung:

$$\begin{aligned}
\tau_h(t) &= \frac{1}{h}(y(t+2h) - (1+\alpha)y(t+h) + \alpha y(t)) - \left(\frac{3-\alpha}{2}y'(t+h) - \frac{1+\alpha}{2}y'(t)\right) \\
&= y'(t)\underbrace{\left(2 - (1+\alpha) - \frac{3-\alpha}{2} + \frac{1+\alpha}{2}\right)}_{0} \\
&\quad + y''(t)\underbrace{\left(\frac{4h}{2} - (1+\alpha)\frac{h}{2} - \frac{3-\alpha}{2}h\right)}_{0} \\
&\quad + y'''(t)\underbrace{\left(\frac{8}{6}h^2 - (1+\alpha)\frac{h^2}{6} - \frac{3-\alpha}{2}\frac{h^2}{2}\right)}_{\frac{h^2}{12}(5+\alpha)} \\
&\quad + O(h^3)
\end{aligned}$$

Also insgesamt eine Konsistenzordnung 3 für $\alpha = -5$ und 2 sonst. Nach unseren Erfahrungen mit den Einschrittverfahren erwarten wir eine entsprechende Konvergenzordnung. Dies wollen wir noch kurz mit Matlab erproben.

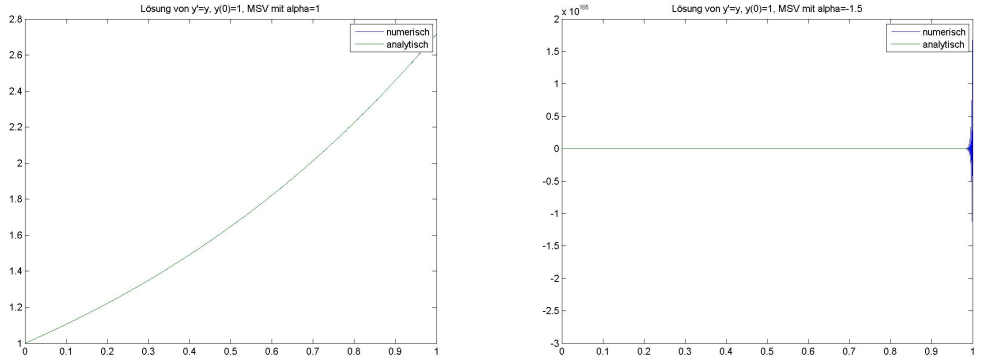


Abbildung 5.1: (In)stabilität von Mehrschrittverfahren
Klick für Bild msvalpha1
Klick für Matlab Figure msvalpha1
Klick für Bild msvalpha-1-5
Klick für Matlab Figure msvalpha-1-5

```
function [ output_args ] = msvdemo( input_args )
%MSVDEMO
    function compute(alpha)
        for k=1:N-1
            y(k+2)=(1+alpha)*y(k+1)-alpha*y(k)+h*((3-alpha)/2 *ef(k+1)-(1+alph
            ef(k+2)=f(x(k+2),y(k+2));
```

Listing 5.1: Instabilität von MSV (gdgl/msvdemo.m)

Klicken für den Quellcode von gdgl/msvdemo.m

Wir sehen: Das geht gewaltig schief. Für $\alpha > 1$ und $\alpha < -1.5$ ist der Algorithmus nicht einmal konvergent. Wir müssen vermuten: Der Algorithmus ist dort zwar konsistent, aber wegen mangelnder Stabilität ist er nicht mehr konvergent. Dies versteht man durch einen kleinen Ausflug in die Theorie der linearen Differenzengleichungen.

5.3.1 Lineare Differenzengleichungen

Definition 5.11 (Lineare Differenzengleichungen)

Seien $a_0, \ldots, a_m$, $m \geq 1$, $a_m \neq 0$, und die Folge (b_k) fest gewählt. Sei S der Operator

aus der Menge der komplexen Folgen in die Menge der komplexen Folgen,

$$(Sy)_k := \sum_{l=0}^{m} a_l y_{k+l}.$$

$y = (y_k)$ *heißt Lösung der linearen Differenzengleichung zu* $a_0, \ldots, a_m$ *und* (b_k)*, falls*

$$(Sy)_k = a_0 y_{k+0} + a_1 y_{k+1} + \ldots + a_m y_{k+m} = b_k,\ k \geq 0.$$

Falls $b_k = 0$ *für alle* k*, so heißt die Gleichung homogen, sonst inhomogen.*

$$\rho(x) = \sum_{j=0}^{m} a_j x^j$$

heißt charakteristisches Polynom der Gleichung.

Beispiel 5.12 (Fibonaccifolge)

$$-y_k - y_{k+1} + y_{k+2} = 0,\ \rho(x) = x^2 - x - 1.$$

Bemerkung: Seien $y_0, \ldots, y_{m-1}$ gegeben. Dann ist die Folge bei gegebenen a_k, b_k eindeutig bestimmt.

Bemerkung: Die Menge aller Folgen, die eine vorgegebene **homogene** Differenzengleichung erfüllen, ist ein m–dimensionaler Teilraum der Menge aller Folgen.

Begründung: Alle Linearkombinationen von Lösungen sind Lösungen, die Lösungen bilden also einen linearen Teilraum. Sei $y^{(j)}$ die Folge im Lösungsraum mit

$$(y^{(j)})_k = \delta_{kj},\ j = 0 \ldots m-1,\ k = 0 \ldots m-1.$$

Diese Folgen bilden eine kanonische Basis des homogenen Lösungsraums.

Bemerkung: Die Differenzengleichung kann wegen $a_m \neq 0$ immer durch a_m geteilt werden, dadurch ändert sich der Lösungsraum nicht. **Wir nehmen also ohne Einschränkung immer an, dass** $a_m = 1$**.**

Wir untersuchen nun, wie die Lösungen der homogenen und der inhomogenen Differenzengleichungen zusammenhängen.

Lemma 5.13 (Basislösungen für inhomogene Differenzengleichungen)
Sei $y^{(m-1)}$ *die Basislösung der Differenzengleichung, für die gilt*

$$(y^{(m-1)})_k = \begin{cases} 0, & k < m-1 \\ 1, & k = m-1 \end{cases}.$$

Weiter sei z^l die Folge, die entsteht, wenn wir in diese Folge vorn noch $l+1$ Nullen einschieben, also

$$z^{(l)} = T^{l+1} y^{(m-1)},$$

also

$$z_k^{(l)} = \begin{cases} y_{k-l-1}^{(m-1)}, & l \leq k-1 \\ 0, & \textit{sonst}\,. \end{cases}$$

Dann ist für jedes $l \geq 0$ und $k \geq 0$

$$(Sz^{(l)})_k = \sum_{j=0}^{m} a_j z_{k+j}^{(l)} = \delta_{kl}.$$

Beweis: Die Folge $y^{(m-1)}$ hat vorn $(m-1)$ Nullen. Die ersten $(l+m)$ Folgeglieder von $z^{(l)}$ sind also Null. Für $k < l$ stehen in der Summe also nur Nullen.

Für $k \geq l$ gilt nach Definition

$$(Sz^{(l)})_k = \sum_{j=0}^{m} a_j z_{k+j}^{(l)} = \sum_{j=0}^{m} a_j (y^{(m-1)})_{k-l+j+1} = \begin{cases} 0, & k > l \\ a_m = 1, & k = l. \end{cases}$$

□

Da der Satz ein bisschen unübersichtlich ist, hier nochmal kurz der einfache Fall für $z^{(0)}$:

$y^{(m-1)}$ hat an den Stellen 0 bis $m-2$ nur Nullen, dann eine 1. $z^{(0)}$ ist dieselbe Folge, nur noch mit einer weiteren Null vorn,

$$z^{(0)} = (\underbrace{0, \ldots, 0}_{(z^{(0)})_0, \ldots (z^{(0)})_{m-1}}, \underbrace{1}_{(z^{(0)})_m}, \ldots)$$

Offensichtlich gilt

$$(Sz^{(0)})_0 = a_m = 1,$$

Für einen Index größer als 0 treten in der Summe nur Terme auf, die auch schon bei $Sy^{(m-1)}$ aufgetreten sind, und die sind alle Null, denn $y^{(m-1)}$ ist Lösung der homogenen Differenzengleichung. Es gilt also

$$(Sz^{(0)}) = (1, 0, 0, \ldots).$$

Bei $z^{(l)}$ werden in diese Folge noch l Nullen links eingeschoben, es gilt also

$$Sz^{(l)} = (0, \ldots, 0, \underbrace{1}_{(z^{(1)})_l}, 0, \ldots).$$

Wir betrachten nun die Linearkombination

$$z := b_0 z^{(0)} + b_1 z^{(1)} + b_2 z^{(2)} + \dots .$$

Dann gilt

$$Sz = b_0 S z^{(0)} + b_1 S z^{(1)} + b_2 S z^{(2)} + \ldots = (b_0, b_1, b_2, \ldots).$$

Wir können also eine Lösung des inhomogenen Systems mit Hilfe der Lösungen des homogenen Systems erhalten. Dies fassen wir zusammen in

Satz 5.14 (Lösungen der inhomogenen Differenzengleichungen)

Wir betrachten die Differnzengleichung zu $(a_0, \ldots, a_m)$ und (b_k). Seien $z^{(l)}$ die verschobenen Basislösungen aus 5.13. Die Folge z sei definiert durch

$$z := \sum_{l=0}^{\infty} b_l z^{(l)}.$$

Dann ist z Lösung der inhomogenen Differenzengleichung

$$\sum_{j=0}^{m} a_j z_{k+j} = b_k$$

Alle Lösungen z' der inhomogenen Differenzengleichung sind von der Form

$$z' = z + \sum_{l=0}^{m-1} z'_l y^{(l)}.$$

Beweis: Nach Definition der $z^{(l)}$ ist

$$(z^{(l)})_k = 0,\ l > k - m.$$

Also gilt

$$z_k = \sum_{l=0}^{\infty} b_l (z^{(l)})_k = \sum_{l=0}^{k-m} b_l (z^{(l)})_k.$$

Insbesondere konvergiert die Reihe. Nach dem Lemma ist

$$(Sz)_k = (S \sum_{l=0}^{\infty} b_l (z^{(l)}))_k = \sum_{l=0}^{\infty} b_l (S z^{(l)})_k = b_k.$$

Es gilt

$$S(z' - z) = Sz' - Sz = (b_0, b_1, b_2, \ldots) - (b_0, b_1, b_2, \ldots) = (0, 0, 0, \ldots).$$

Also erfüllt $z - z'$ die homogene Differenzengleichung und kann durch die kanonische Basis ausgedrückt werden. Da $z_k = 0$ für $k \leq m - 1$ nach Definition der $z^{(l)}$, ist

$$(z' - z)_k = (z')_k$$

und damit

$$(z' - z) = \sum_{l=0}^{m-1} (z')_l y^{(l)}.$$

□

Wir sehen also, dass die Lösungen der inhomogenen Differenzengleichung bereits durch die Lösungen der homogenen Gleichung bestimmt sind. Die Lösungen der inhomogenen Gleichung bekommen wir dann einfach mit dem Verschiebeoperator. Wir können uns also bei der Betrachtung des Verhaltens der Lösungen von Differenzenverfahren auf die homogenen Gleichungen beschränken.

Satz 5.15 *(Darstellung der Lösung von homogenen Differenzengleichungen)*
Seien $\lambda_0, \ldots, \lambda_r$ die Nullstellen des charakteristischen Polynoms ρ einer linearen, homogenen Differenzengleichung mit Vielfachheit m_j. Dann gilt:

Eine Folge $y = (y_k)$ erfüllt genau dann die Differenzengleichung, falls es Polynome $q_j(k) \in \mathcal{P}_{m_j - 1}$ gibt mit

$$y_k = \sum_{j=0}^{r} q_j(k) \lambda_j^k.$$

Falls das charakteristische Polynom m verschiedene Nullstellen hat (also $r = m-1$), so erfüllt die Folge (y_k) genau dann die Differenzengleichung, wenn es Koeffizienten $\alpha_j \in \mathbb{C}$ gibt mit

$$y_k = \sum_{j=0}^{m-1} \alpha_j \lambda_j^k.$$

Vor dem Beweis betrachten wir

Beispiel 5.16 (Nichtrekursive Formel für Fibonacci–Zahlen)

$$\rho(x) = x^2 - x - 1,\ \lambda_1 = \frac{1 + \sqrt{5}}{2},\ \lambda_2 = \frac{1 - \sqrt{5}}{2}.$$

Beide Nullstellen haben die Vielfachheit 1. Alle Fibonacci–Folgen sind also von der Form

$$y_k = \alpha \left(\frac{1+\sqrt{5}}{2} \right)^k + \beta \left(\frac{1-\sqrt{5}}{2} \right)^k .$$

Für die Standardfolge mit $y_0 = 0$ *und* $y_1 = 1$ *erhalten wir*

$$\alpha + \beta = 0,\ \alpha \frac{1+\sqrt{5}}{2} + \beta \frac{1-\sqrt{5}}{2} = 1,$$

also

$$y_k = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^k - \left(\frac{1-\sqrt{5}}{2} \right)^k \right).$$

Beweis: (des Satzes)
Sei zunächst

$$y^{(l)} = ((y^{(l)})_k) := (\lambda_l^k),\ l = 0 \dots r.$$

Wegen

$$(Sy^{(l)})_k = \sum_{j=0}^{m} a_j (y^{(l)})_{k+j} = \sum_{j=0}^{m} a_j \lambda_l^{k+j} = \lambda_l^k \rho(\lambda_l) = 0$$

ist $y^{(l)}$ eine Lösung der homogenen Differenzengleichung.

Das charakteristische Polynom ρ habe nur unterschiedliche Nullstellen, es gelte also $r = m - 1$ und $m_j = 1$.

Wie bei den Fibonacci–Folgen bekommen wir dann m linear unabhängige Lösungen $y^{(l)}$, $l = 0 \dots m-1$, der Differenzengleichung. Die Dimension des Lösungsraums ist m nach der Vorbemerkung, damit bilden die $y^{(l)}$ eine Basis. y ist genau dann eine Lösung, wenn es Koeffizienten $\alpha_l \in \mathbb{C}$ gibt mit

$$y = \sum_{l=0}^{m-1} \alpha_l y^{(l)}.$$

Dies ist die Zusatzbemerkung.

Falls Nullstellen höherer Ordnung auftreten, so müssen wir etwas mehr arbeiten.

Sei wieder $a_m = 1$ und $y = (y_k)$ eine Lösung der homogenen Differenzengleichung. Sei

$$Y_k \in \mathbb{C}^m,\ Y_k := \begin{pmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k+m-2} \\ y_{k+m-1} \end{pmatrix}$$

und damit

$$Y_{k+1} = \begin{pmatrix} y_{k+1} \\ y_k \\ \vdots \\ y_{k+m-1} \\ y_{k+m} \end{pmatrix} = \begin{pmatrix} y_{k+1} \\ y_k \\ \vdots \\ y_{k+m-1} \\ -\sum_{j=0}^{m-1} a_j y_j \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -a_0 & & \cdots & & -a_{m-1} \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k+m-2} \\ y_{k+m-1} \end{pmatrix}}_{=Y_k}.$$

A heißt Begleitmatrix des Polynoms ρ, ihr charakteristisches Polynom ist ρ (Übungen, daher der Name von ρ). Sei

$$A = XJX^{-1},\ J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_r \end{pmatrix}$$

mit den Jordankästchen

$$J_l = \begin{pmatrix} \lambda_l & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_l \end{pmatrix} = \underbrace{\begin{pmatrix} \lambda_l & & \\ & \ddots & \\ & & \lambda_l \end{pmatrix}}_{=\lambda_l I_l} + \underbrace{\begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ & & & 0 \end{pmatrix}}_{=:N_l} \in \mathbb{C}^{m_l \times m_l}$$

die Jordan–Normalform von A. Zu jedem Eigenwert λ_l von A gibt es nur einen linear unabhängigen Eigenvektor (Übungen), deshalb gehört zu jedem Eigenwert genau ein Jordankästchen der Große $m_l \times m_l$.

Dann ist

$$Y_k = A^k Y_0 = XJ^k X^{-1} Y_0.$$

N_l ist nilpotent, es gilt $N_l^{m_l} = 0$, also

$$J_l^k = (\lambda_l I_l + N_l)^k = \sum_{j=0}^{m_l - 1} \binom{k}{j} N_l^j \lambda_l^{k-j}.$$

J_l^k enthält also nur Matrixeinträge der Form $p(k)\lambda_j^k$ mit Polynomen $p \in \mathcal{P}_{m_l - 1}$.

Damit enthält $Y_k = XJ^k X^{-1} Y_0$ nur Linearkombinationen dieser Einträge.

Wegen

$$y_k = (Y_k)_0$$

gibt es also für jede Lösung y der homogenen Differenzengleichung Polynome $q_j(k)$, $q_j \in \mathcal{P}_{m_1-1}$, mit

$$y_k = \sum_{j=0}^{r} q_j(k)\lambda_j^k.$$

Der Teilraum der Folgen dieser Form hat die Dimension

$$\sum_j m_j = m$$

und damit dieselbe wie der Lösungsraum, damit sind die Unterräume gleich. □

5.3.2 Stabilitätssätze von Dahlquist

Zur Motivation betrachten wir zunächst kurz den Zusammenhang zwischen Differenzengleichungen und Mehrschrittverfahren. Angenommen, wir suchen die Lösung $y(t) = C$ der trivialen Differentialgleichung

$$y'(t) = 0,\ y(0) = C$$

mit einem linearen Mehrschrittverfahren mit Koeffizienten α_k, β_k auf einem äqudistanten Gitter der Schrittweite h (5.1).

Mit einem Einschrittverfahren verschaffen wir uns zunächst Näherungen y_0 bis y_{m-1}. Anschließend können wir mit dem Mehrschrittverfahren weiterrechnen, wegen $f(t, y) = 0$ in der Differentialgleichung gilt mit der Definition 5.1

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = 0.$$

(y_k) ist also Lösung der homogenen Differenzengleichung und damit Linearkombination der Folgen $y^{(j)}$ aus der kanonischen Basis,

$$y = (y_k) = \sum_{j=0}^{m-1} \gamma_j y^{(j)}.$$

Da $y(t)$ beschränkt ist, erwarten wir das auch von den y_k, unabhängig von h. Sei nun z.B. die Folge $y^{(0)}$ unbeschränkt. Dann ist auch die Folge y_k unbeschränkt (oder wird es für $\gamma_0 = 0$, wenn man bei der Berechnung der Anfangswerte einen beliebig kleinen Fehler macht). Selbst diese extrem einfache Differentialgleichung kann

also nicht korrekt gelöst werden, wenn die homogene Differenzengleichung unbeschränkte Lösungen hat: Die Auswirkungen von Fehlern in den Anfangsdaten oder Fehlern bei der Berechnung von f sind über alle Massen groß, der Algorithmus wird instabil.

Die Dahlquist–Bedingung gibt an, wann dies der Fall ist.

Hierbei haben wir einen Fehler gemacht: Die Folge der y_k ist für jedes h endlich. Aber: Für $h \to 0$ wird die Folge immer länger, und da $f = 0$, hängt die Folge nicht von h ab.

Satz 5.17 (Wurzelbedingung von Dahlquist, Dahlquist–Bedingung)
Alle Lösungen einer homogenen linearen Differenzengleichung mit charakteristischem Polynom ρ sind beschränkt genau dann, wenn für die Nullstellen λ_i von ρ gilt

$$|\lambda_i| \leq 1$$

und zusätzlich

$$|\lambda_i| = 1 \Rightarrow \lambda_i \text{ ist einfache Nullstelle.}$$

Beweis: Satz 5.15. □

Definition 5.18 (Charakteristisches Polynom von Mehrschrittverfahren)

Ein Mehrschrittverfahren sei durch die Konstanten a_j und b_j definiert. Dann heißen

$$\rho(x) = \sum_{j=0}^{m} a_j x^j \text{ und } \sigma(x) = \sum_{j=0}^{m} b_j x^j$$

das (erste) bzw. zweite charakteristische Polynom des Verfahrens.

Bemerkung: Das Verfahren ist für $f \in C^1$ konsistent genau dann, falls $\rho(1) = 0$ und $\rho'(1) = \sigma(1)$ (Übungen).

Wir wollen nun den Begriff der Stabilität für Mehrschrittverfahren erweitern. Hierbei tritt ein Problem auf, dass wir bei den Einschrittverfahren nicht hatten: Zusätzlich zu den Fehlern, die in jedem Schritt entstehen, haben auch die Anfangswerte Fehler (denn nur der Anfangswert $y(a)$ ist gegeben, die anderen Anfangswerte im Mehrschrittverfahren werden mit einem Einschrittverfahren approximiert). Wir nennen ein Verfahren (asymptotisch) stabil, wenn sich der Fehler durch das Maximum der Fehler in Anfangswerten und Rechenfehler in jedem Schritt abschätzen lässt. Also:

Definition 5.19 (Asymptotische Stabilität für Mehrschrittverfahren)

Ein Mehrschrittverfahren sei von der allgemeinen Form (5.1)

$$\sum_{j=0}^{m} \alpha_j y_{k+j} = h\varphi(t_k, y_k, \ldots, y_{k+m}, h) = h \sum_{j=0}^{m} \beta_j f(t_{k+j}, y_{k+j}).$$

Seien $u_0 \ldots, u_{m-1}$ vorgegeben, und sei (u_k) die durch das Mehrschrittverfahren definierte Näherung.

Sei (v_k) eine weitere Näherung, bei der in den Anfangswerten und bei der Berechnung Fehler $(F_h(v))$ gemacht wurden, also

$$\begin{aligned} F_h(v_h)_k &= v_k - u_k,\ k = 0 \ldots m-1. \\ F_h(v_h)_{k+m} &= \frac{1}{h} \sum_{j=0}^{m} \alpha_j v_{k+j} - \varphi(t_k, v_k, \ldots, v_{k+m}, h),\ k \geq 0. \end{aligned}$$

Sei wie üblich $u_h(t_k) = u_k$ usw.

Das Verfahren heißt **(asymptotisch) stabil** *genau dann, wenn es ein $K > 0$ gibt, so dass für alle $u_k \in \mathbb{C}$, $k = 0 \ldots m-1$, $v_h, w_h : I_h \mapsto \mathbb{C}$, $h < h_0$, gilt*

$$||v_h - w_h||_\infty \leq K||F_h(v_h) - F_h(w_h)||_\infty.$$

Bemerkung: Natürlich ist $F_h(u_h) = 0$. Dann gilt für stabile Verfahren

$$||u_h - v_h||_\infty \leq K||F_h(v_h)||_\infty.$$

Der maximale Fehler im Ergebnis lässt sich gegen den maximalen Fehler in der Rechnung abschätzen, was unserer alten Definition von numerischer Stabilität entspricht. Also: (Nur) wenn das Differenzenverfahren stabil ist, lassen sich seine Lösungen überhaupt numerisch berechnen. Von der Differentialgleichung reden wir dabei zunächst mal gar nicht.

Der Satz von Dahlquist begründet wieder den Satz: Konsistenz plus Stabilität gleich Konvergenz.

Satz 5.20 (Satz von Dahlquist, Konvergenzsatz für Mehrschrittverfahren)
Die Anfangswerte $u_k = u_h(t_k)$, $k = 0 \ldots m-1$, heißen konsistent von der Ordnung p, falls

$$u_h(t_k) - y(t_k) = O(h^p).$$

1. *Ein Mehrschrittverfahren sei konvergent für die Anfangswertaufgabe*

   $$y'(t) = 0,\ y(a) = 0$$

   und alle konsistenten Anfangswerte. Dann ist die Dahlquist–Bedingung erfüllt (und es ist stabil nach 5.22).

2. *Ein Mehrschrittverfahren sei stabil und konsistent von der Ordnung p. Dann ist es konvergent von der Ordnung p für alle konsistenten Anfangswerte von der Ordnung p.*

Beweis: Zu 1.

Sei λ Nullstelle von ρ mit Vielfachheit r. Sei

$$y = (y_k) := (k^{r-1}\lambda^k h).$$

y ist die vom Mehrschrittverfahren gelieferte Näherung für die Anfangswerte $y_0, \ldots, y_{m-1}$, denn die Folge ist Lösung der homogenen Differenzengleichung. Die Anfangswerte sind konsistent, denn die Lösung der Gleichung ist $y(0) = 0$, und

$$(y_k - y(kh)) = (k^{r-1}\lambda^k)h - 0 = O(h),\ k = 0 \ldots m-1.$$

Sei $h = (b-a)/N$, also y_N Näherung für $y(b)$. Aus der Konvergenz von y_k gegen die Lösung $y = 0$ folgt

$$N^{r-1}\lambda^k \frac{(b-a)}{N} \mapsto 0$$

und das geht nur, wenn

$$|\lambda| < 1$$

oder

$$|\lambda| = 1,\ r = 1,$$

wenn also die Dahlquist–Bedingung erfüllt ist.
Zu 2.

Sei u_h die exakte Mehrschrittlösung, $y_k = y(t_k)$. Dann ist $(F_h(y))_k$ für $k \geq m$ der Konsistenzfehler. Für $k < m$ ist $F_h(y)_k$ der Fehler der Anfangswerte. Wie oben gilt $F_h(u_h) = 0$ und damit

$$||y - u_h||_\infty \leq K||F_h(y) - F_h(u_h)||_\infty = ||F_h(y)||_\infty = O(h^p).$$

□

Vorlesungsnotiz: 29.6.2013

Wir starten die Umkehrung dieses Satzes wieder mal mit einer kleinen Abwandlung der diskreten Ungleichung von Gronwall.

Lemma 5.21

Sei ϵ_k eine Folge nichtnegativer Zahlen, $\epsilon_0 \leq a$, und seien a, b positiv. Sei

$$\epsilon_k \leq a + b \sum_{l=0}^{k-1} \epsilon_l.$$

Dann gilt

$$\epsilon_k \leq a e^{kb}.$$

Beweis: Mit Gronwall oder direkt per Induktion:

$$\begin{aligned} \epsilon_{k+1} &\leq a + b \sum_{l=0}^{k} \epsilon_l \\ &\leq a + ab \frac{e^{(k+1)b} - 1}{e^b - 1} \\ &\leq a e^{(k+1)b}, \text{ wegen } e^b > 1 + b. \end{aligned}$$

□

Nach Satz 5.20 gilt bereits für konsistente Mehrschrittverfahren bei konsistenten Anfangswerten:

$$\text{stabil} \Rightarrow \text{konvergent} \Rightarrow \text{Wurzelbedingung für } \rho \text{ erfüllt}$$

Wir zeigen nun noch die fehlende Implikation, und damit sind alle drei Eigenschaften äquivalent.

Satz 5.22 (Stabilität von Mehrschrittverfahren)
Sei f lipschitzstetig in der zweiten Variablen. Gegeben sei ein lineares Mehrschrittverfahren mit charakteristischem Polynom ρ. Dann gilt:

Falls ρ die Wurzelbedingung erfüllt, so ist das Verfahren stabil.

Beweis: ρ erfülle die Wurzelbedingung.

u_0 bis u_{m-1} seien fest gewählt (sie tauchen im Satz nicht auf). Es seien

$$v_h = (v_k),\ w_h = (w_k)$$

zwei Gitterfunktionen und

$$\delta_h = (\delta_k) = v_h - w_h,\ d_h = (d_k) = F_h(v_h) - F_h(w_h).$$

Beweisidee: Zum Nachweis der Stabilität müssen wir zeigen, dass δ_h sich durch d_h abschätzen lässt. Wir zeigen zunächst, dass δ_h einer inhomogenen Differenzengleichung genügt. Dann können wir δ_h explizit angeben mit Hilfe der Lösungen der homogenen Gleichung, und diese sind genau dann beschränkt, wenn ρ die Wurzelbedingung erfüllt.

Nach Definition von F_h gilt

$$\delta_k = d_k,\ k = 0 \dots m-1.$$

Wir versuchen, δ_h durch d_h abzuschätzen. Es gilt

$$\begin{aligned} d_{k+m} &= (F_h(v_h) - F_h(w_h))_{k+m} \\ &= \frac{1}{h}\sum_{j=0}^{m} \alpha_j \delta_{k+j} - (\varphi(t_{k+j}, v_k, \dots, v_{k+m}, h) - \varphi(t_k, w_k, \dots, w_{k+m}, h)) \end{aligned}$$

und damit erfüllt δ_h die inhomogene Differenzengleichung

$$\sum_{j=0}^{m} \alpha_j \delta_{k+j} = h \underbrace{(d_{k+m} + \varphi(t_k, v_k, \dots, v_{k+m}, h) - \varphi(t_k, w_k, \dots, w_{k+m}, h))}_{=:\eta_k}.$$

Mit f ist auch φ lipschitzstetig in den y_k mit einer Lipschitzkonstanten L, also ist

$$|\eta_k| \le |d_{m+k}| + Lm \underbrace{\max_{j=0\dots m} |\delta_{k+j}|}_{=:\epsilon_k}.$$

Nach 5.14 hat δ_h die Darstellung

$$\delta_h = \sum_{j=0}^{m-1} d_j y^{(j)} + h \sum_{l=0}^{\infty} \eta_l z^{(l)} \qquad (*)$$

mit den kanonischen Basen $y^{(j)}$ und $z^{(l)}$ aus 5.14. Da ρ die Wurzelbedingung erfüllt, gibt es ein M mit

$$||y^{(j)}||_\infty,\ ||z^{(l)}||_\infty \le M.$$

Sei nun

$$d := ||d_h||_\infty = ||F_h(v_h) - F_h(w_h)||_\infty.$$

Nach Definition der $z^{(l)}$ gilt

$$(z^{(l)})_k = 0, l > k - m.$$

Die Reihe in $(*)$ ist also nur eine Summe, und wir können alle Einzelterme abschätzen:

$$\begin{aligned} |\delta_k| &\leq Mmd + Mh \sum_{l=0}^{k-m} (d + Lm\epsilon_l) \\ &= Md(m + (k - m + 1)h) + MhLm \sum_{l=0}^{k-m} \epsilon_l. \end{aligned}$$

Damit gilt

$$\begin{aligned} \epsilon_k &= \max_{j=0\ldots m} |\delta_{k+j}| \\ &\leq \max_{j=0\ldots m} Md(m + (k + j - m + 1)h) + MhLm \sum_{l=0}^{k-m+j} \epsilon_l \\ &= Md(m + (k + 1)h) + MhLm \sum_{l=0}^{k} \epsilon_l \end{aligned}$$

oder

$$\epsilon_k(1 - MhLm) \leq Md(m + (k + 1)h) + MhLm \sum_{l=0}^{k-1} \epsilon_l.$$

Sei nun

$$h < h_0 := \frac{1}{2MLm} \Rightarrow 1 - MhLm \geq \frac{1}{2}.$$

Dann gilt

$$\epsilon_k \leq \underbrace{2M(m + (b - a))}_{=:c} d + \underbrace{2MLm}_{=:D} h \sum_{l=0}^{k-1} \epsilon_l$$

und mit Lemma 5.21

$$|\delta_k| \leq \epsilon_k \leq ce^{Dkh}d.$$

Da

$$kh \leq (b - a),$$

gilt somit

$$||v_h - w_h||_\infty = ||\delta_h||_\infty \leq ce^{D(b-a)}||d_h||_\infty = ce^{D(b-a)}||F_h(v_h) - F_h(w_h)||_\infty$$

für $h < h_0$. Also ist das Verfahren stabil und konvergent von der Ordnung p. □

Bemerkung: Der Satz ist für allgemeinere Mehrschrittverfahren gültig im Sinne von 4.25. Wir haben die linearen Verfahren hier nur zur Vereinfachung der Schreibweise benutzt.

Bemerkung:

1. Der Fehler beim Mehrschrittverfahren wird abgeschätzt durch das Maximum der Fehler in den Anfangswerten und des lokalen Diskretisierungsfehlers. Man sollte zur Bestimmung der Lösung also immer passende Verfahren wählen, d.h. das Einschrittverfahren zur Bestimmung der Anfangswerte sollte dieselbe Konsistenzordnung haben wie das Mehrschrittverfahren.
2. Einschrittverfahren sind stabil, denn für ihr charakteristisches Polynom gilt

$$\rho(x) = x - 1.$$

3. Aus Integration gewonnene Mehrschrittverfahren sind stabil, denn für ihr charakteristisches Polynom gilt

$$\rho(x) = x^m - 1.$$

4. Es gibt konsistente Verfahren, die nicht von der Differentialgleichung abhängen (Übungen). Die sind natürlich nicht konvergent! Also müssen sie instabil sein.
5. In der Konstanten taucht wieder der Term $e^{L(b-a)}$ auf, im wesentlichen erhalten wir also ein ähnliches Ergebnis wie schon bei in 4.19 für den analytischen Fehler.

   Stabile Mehrschrittverfahren sind also auch in diesem Sinne stabil: Der tatsächliche Fehler liegt in der Größenordnung des unvermeidbaren Fehlers.

Mit diesen Sätzen können wir nun auch das katastrophale Beispiel 5.10 aufklären. Das charakteristische Polynom des Mehrschrittverfahrens ist

$$\rho(x) := x^2 - (1 + \alpha)x + \alpha$$

Die Nullstellen des charakteristischen Polynoms sind 1 und α. Das Verfahren ist also instabil für $|\alpha| > 1$ und $\alpha = 1$. Im zweiten Fall hat die Nullstelle 1 die Vielfachheit 2, also ist das Verfahren instabil. Für $\alpha = -1$ haben die Nullstellen vom Betrag 1 die Vielfachheit 1, also ist das Verfahren stabil.

5.3.3 Stabilität und Konsistenzordnung

Wir betrachten nun noch kurz den Zusammenhang von Stabilität und Konsistenzordnung. Zunächst geben wir eine einfache Vorschrift zur Berechnung der Konsistenzordnung eines Verfahrens an:

Satz 5.23 (Ordnung von Mehrschrittverfahren)

Seien ρ und σ die charakteristischen Polynome eines m–Schritt–Mehrschrittverfahrens. Sei

$$\varphi(\lambda) = \frac{\rho(\lambda)}{\log \lambda} - \sigma(\lambda).$$

Das Mehrschrittverfahren ist genau dann konsistent von der Ordnung p (für ausreichend glattes f), wenn φ bei $\lambda = 1$ eine Nullstelle der Ordnung p hat. Das Verfahren ist konsistent, falls $\varphi(1) = 0$.

Beweis: Sei $f \in C^p$, also $y \in C^{p+1}$ und t fest. Dann gilt

$$\begin{aligned} y(t+jh) &= y(t) + jhy'(t) + \ldots + \frac{(jh)^p}{p!} y^{(p)}(t) + O(h^{p+1}) \\ y'(t+jh) &= y'(t) + jhy''(t) + \ldots + \frac{(jh)^{(p-1)}}{(p-1)!} y^{(p)}(t) + O(h^p) \end{aligned}$$

Dies ergibt für den lokalen Diskretisierungsfehler

$$\begin{aligned} \tau_h(t) &= \frac{1}{h} \sum_{j=0}^{m} \alpha_j y(t+jh) - \sum_{j=0}^{m} \beta_j y'(t+jh) \\ &= \underbrace{\frac{1}{h} \sum_{j=0}^{m} \alpha_j}_{C_0} y(t) \\ &\quad + \underbrace{\left(\sum_{j=0}^{m} l\alpha_j - \sum_{j=0}^{m} \beta_j \right)}_{C_1} y'(t) \\ &\quad \vdots \\ &\quad + h^{p-1} \underbrace{\left(\frac{1}{p!} \sum_{j=0}^{m} j^p \alpha_j - \frac{1}{(p-1)!} \sum_{j=0}^{m} j^{p-1} \beta_j \right)}_{C_p} y^{(p)}(t) + O(h^p). \end{aligned}$$

Sei

$$\chi(z) = \varphi(e^z) = \frac{1}{z}\sum_{j=0}^{m} \alpha_j e^{jz} - \sum_{j=0}^{m} \beta_j e^{jz}.$$

Dann ist die Potenzreihe von χ um $z = 0$

$$\begin{aligned} \chi(z) &= \frac{1}{z}\sum_{j=0}^{m} \alpha_j \sum_{k=0}^{p} \frac{jz^k}{k!} - \sum_{j=0}^{m} \beta_j \sum_{k=0}^{p-1} \frac{(jz)^k}{k!} + O(z^p) \\ &= \frac{1}{z}C_0 + C_1 + \ldots + z^{p-1}C_p + O(z^p). \end{aligned}$$

Also ist äquivalent:

1. φ hat p–fache Nullstelle bei $\lambda = 1$.
2. χ hat p–fache Nullstelle bei $z = 0$.
3. $C_0 = C_1 = \ldots = C_p = 0$.
4. $\tau_h(t) = O(h^p)$.

□

Da Mehrschrittverfahren der Länge m insgesamt $2m + 1$ Variable haben (beachte $\alpha_m = 1$), kann man erwarten, damit $2m + 1$ Bedingungen zu erfüllen, also den Konsistenzgrad $2m$ zu erreichen. Leider sind die so erzeugten Verfahren alle instabil, es gilt der Satz von Henrici:

Satz 5.24 (Maximale Konsistenzordnung stabiler Verfahren)
Stabile m–Schritt–Verfahren haben die maximale Konsistenzordnung $m + 1$ für m ungerade und $m + 2$ für m gerade. Diese Schranke ist streng, d.h. es gibt für jedes m Verfahren, die sie erreichen.

Bemerkung: Adams–Moulton und Milne–Simpson sind optimal für m ungerade.

Beweis: Henrici [1962], Part II, Chapter 5, pp 226, mit 5.23 und ein bisschen Funktionentheorie. □

Von Henrici stammt auch eine Einführung in die Numerische Mathematik (Henrici [1964]), die auf Einsteigerniveau intensiver auf Differenzengleichungen und die Beziehungen zu Differentialgleichungen eingeht.

5.4 Extrapolation

Nach den Erfahrungen der anderen Kapitel erwarten wir, dass sich die Methode der Extrapolation, die wir zur Konvergenzverbesserung oder zur Abschätzung des globalen Diskretisierungsfehlers nutzen können, auch auf Mehrschrittverfahren anwenden lässt. Für Einschrittverfahren war dies richtig, für Mehrschrittverfahren ist es im Allgemeinen falsch.

Satz 5.25 *(Extrapolation für Mehrschrittverfahren)*
Sei $y(t_0, h)$ die Näherung, die ein Verfahren der Ordnung p für die Lösung $y(t_0)$ für $y'(t) = f(t, y(t))$ liefert. Dann ist bei Einschrittverfahren der globale Diskretisierungsfehler in eine Potenzreihe entwickelbar, bei Mehrschrittverfahren im allgemeinen nicht.

Beweis: In Stoer and Bulirsch [2005], 7.2.3., Beispiel in den Übungen. □

In den Übungen zeigen wir, dass schon für ein einfaches Beispiel der Fehler der Mittelpunktsregel nicht in eine Potenzreihe entwickelbar ist. Es taucht ein oszillierender Term auf, der die Konvergenz der Potenzreihe verhindert. Dies kann man einfach durch eine nachgeschaltete Glättung korrigieren. Wir erhalten das Verfahren von Gragg:

Satz 5.26 *(Verfahren von Gragg)*
Sei

$$\begin{aligned} z_0 &= y(a) \\ z_1 &= z_0 + hf(a, z_0) \\ z_{k+2} &= z_k + 2hf(x_{k+1}, z_{k+1}) \\ y_n &= \frac{1}{4}z_{n+1} + \frac{1}{2}z_n + \frac{1}{4}z_{n-1} \\ y_0 &= z_0 \\ y_N &= z_N \end{aligned}$$

das Verfahren von Gragg. Dann ist der globale Diskretisierungsfehler in eine Potenzreihen in h^2 entwickelbar.

Zum Beweis siehe wieder Stoer and Bulirsch [2005] und die Bemerkungen in den Übungen.

Vorlesungsnotiz: 20.6.2013

Kapitel 6

Randwertprobleme

Abschließend wollen wir uns noch der Behandlung linearer Randwertaufgaben zuwenden. Bisher haben wir ausschließlich Anfangswertaufgaben behandelt, d.h. wir suchten Funktionen $y(t)$ mit

$$y'(t) = f(t, y(t)),\ y(a) = y_0, \text{ auf } [a, b].$$

Für eine lineare Differentialgleichung zweiter Ordnung lautet das Anfangswertproblem

$$y''(t) + p(t)y'(t) + q(t)y(t) = f(t),\ y(a) = y_0,\ y'(a) = y'_0.$$

Wir müssen also im Punkt a Werte für y und seine Ableitung vorgeben. Diesen Fall haben wir sehr gut untersucht, die Lösbarkeitsbedingungen sind einfach analytisch angebbar, und wir haben konvergente numerische Methoden für diesen Fall angegeben.

Will man aber etwa die Auslenkung eines an zwei Seiten festgebundenen Seils beschreiben, so kann man das zwar durch eine lineare Differentialgleichung tun. Randbedingung ist aber offensichtlich, dass die Auslenkung am linken und rechten Rand verschwindet, d.h.

$$y(a) = y(b) = 0.$$

Gleiches passiert, wenn man die Temperaturverteilung eines Stabs berechnen möchte, der an beiden Enden erhitzt wird.

Wir landen also bei Randwertproblemen, bei denen die festen Bedingungen nicht nur am Randpunkt a, sondern auch am Randpunkt b vorgegeben sind.

Definition 6.1 (Lineare) Randwertprobleme
Ein Randwertproblem sucht Funktionen

$$y(t) \in C^1([a,b] \mapsto \mathbb{R}^n)$$

mit

$$y'(t) = f(t, y(t)),\ g(y(a), y'(a), y(b), y'(b)) = 0$$

für eine Funktion

$$g : \mathbb{R}^{4n} \mapsto \mathbb{R}^n.$$

Typischerweise haben wir Dirichlet–Randbedingungen (bei denen der Wert der Funktion vorgeschrieben ist), Neumann–Randbedingungen (bei denen der Wert der Ableitung vorgeschrieben ist) oder gemischte Bedingungen (bei denen eine Kombination aus Ableitung und Wert der Funktion vorgeschrieben ist, jeweils in einem Randpunkt). Sind g und f (affin) linear, so nennen wir das Randwertproblem linear.

Für $n = 2$ ergibt sich damit

1. $y(a) = z_0$, $y(b) = z_1$*: Dirichlet–Randbedingungen.*
   Falls $z_0 = z_1 = 0$: Homogene Dirichlet–Randbedingungen.
2. $y'(a) = z_0$, $y'(b) = z_1$*: Neumann–Randbedingungen.*
   Falls $z_0 = z_1 = 0$: Homogene Neumann–Randbedingungen.
3. $g_1(y(a), y'(a)) = 0$, $g_2(y(b), y'(b)) = 0$*: Gemischte Randbedingungen.*
4. $y(a) = y(b)$, $y'(a) = y'(b)$*: Periodische Randbedingungen.*

Die Lösbarkeit dieses Systems ist leider erheblich schwieriger zu zeigen als bei Anfangswertaufgaben. Wir betrachten als Beispiel eine lineare Differentialgleichung zweiter Ordnung mit konstanten Koeffizienten und Dirichlet–Randbedingungen:

$$-y''(t) + \alpha^2 y(t) = 0,\ \alpha \in \mathbb{R},\ y(a) = z_0,\ y(b) = z_1,\ b > a.$$

Alle Lösungen der Differentialgleichung sind von der Form

$$y(t) = c_1 y_1(t) + c_2 y_2(t),\ y_1(t) = \exp(\alpha t),\ y_2(t) = \exp(-\alpha t)$$

und zur Erfüllung der Randwerte erhalten wir die Gleichungen

$$c_1 y_1(a) + c_2 y_2(a) = z_0,\ c_1 y_1(b) + c_2 y_2(b) = z_1.$$

Das System ist also lösbar, wenn die Matrix

$$W = \begin{pmatrix} y_1(a) & y_1(b) \\ y_2(a) & y_2(b) \end{pmatrix}$$

invertierbar ist bzw. wenn ihre Determinante nicht verschwindet. Setzen wir die Lösungen ein, so gilt

$$W = \begin{pmatrix} \exp(\alpha a) & \exp(\alpha b)) \\ \exp(-\alpha a) & \exp(-\alpha b) \end{pmatrix}.$$

Die Determinante dieser Matrix ist gerade

$$\det(W) = \exp(\alpha(a-b)) - \exp(-\alpha(a-b))$$

und da die Exponentialfunktion für reelle Argumente monoton ist, verschwindet dieser Ausdruck nicht. Damit ist die Matrix invertierbar und die Dirichlet–Aufgabe eindeutig lösbar. Tatsächlich sind lineare Dirichlet–Randwertaufgaben zweiter Ordnung der Form

$$-y''(t) + q(t)y(t) = f(t),\ y(a) = y_0,\ y(b) = y_1$$

für stetiges f und $q(t) \geq 0$ immer eindeutig lösbar. Wir werden einen Beweis mit Hilfe der Variationsmethoden angeben.

Anders ist es im Fall $q(t) < 0$. Wir betrachten das fast gleiche Randwertproblem für die Helmholtzgleichung

$$-y''(t) - \alpha^2 y(t) = 0,\ y(0) = 0,\ y(\pi) = 0,\ \alpha \in \mathbb{R}.$$

Wir können die Analyse übertragen mit den Grundlösungen

$$y_1(t) = \cos(\alpha t),\ y_2(t) = \sin(\alpha t).$$

Die trigonometrischen Funktionen haben natürlich ein fundamental anderes Verhalten als die Exponentialfunktion. Insbesondere ist die Matrix

$$W = \begin{pmatrix} \sin(0) & \sin(\alpha\pi) \\ \cos(0) & \cos(\alpha\pi) \end{pmatrix}$$

genau für $\alpha \in \mathbb{Z}$ nicht invertierbar. Wir erhalten also: Die Randwertaufgabe ist eindeutig lösbar, falls $\alpha \notin \mathbb{Z}$. Eine allgemeine Lösbarkeit mit einem griffigen Kriterium (Lipschitz–Stetigkeit wie bei den Anfangswertaufgaben) ist nicht gegeben, es muss jede Gleichung einzeln untersucht werden.

An dem einfachen Beispiel der linearen Differentialgleichungen zweiter Ordnung mit homogenen Randbedingungen lasen sich bereits alle Schwierigkeiten studieren. Wir betrachten im Folgenden das **Sturm–Liouville–Problem**

$$-(p(t)y'(t))' + q(t)y(t) = f(t)$$

mit linearen Randbedingungen, differenzierbarer Funktion p und stetiger Funktion q. Wenn nicht ausdrücklich anders vermerkt, werden wir uns dabei auf den Fall $p = 1$ und zunächst homogene Dirichlet–Randbedingungen zurückziehen, also

$$-y''(t) + q(t)y(t) = f(t),\ y(a) = y(b) = 0.$$

Wir werden vier verschiedene Ansätze zur analytischen und numerischen Lösung dieses Problems untersuchen.

Bemerkung: Wir betrachten in diesem Abschnitt nur gewöhnliche Differentialgleichungen. Tatsächlich lassen sich, im Gegensatz zu Anfangswertaufgaben, alle vorgestellten Methoden und Sätze in höhere Dimensionen, also für partielle Differentialgleichungen, übertragen.

In einer Dimension sind sie aber viel anschaulicher, oft trivial. Zum Verständnis der Verhältnisse bei partiellen Differentialgleichungen ist es also sehr nützlich, die Interpretation als gewöhnliche Differentialgleichung im Kopf zu behalten. Wir werden hier immer einige Beziehungen zu den partiellen Differentialgleichungen angeben, auch wenn Sie diese bisher nicht gehört haben oder hören werden.

6.1 Analytische Lösung mit Greenschen Funktionen

Ein Verfahren zur Lösung der homogenen Differentialgleichung mit $f = 0$ haben wir oben bereits angegeben: Wir bestimmen zwei Grundlösungen, die Lösung des Sturm–Liouville–Problems bekommen wir dann durch Lösung eines Gleichungssystems. Diese Idee lässt sich für jede rechte Seite erweitern.

Satz 6.2 (Greensche Funktion)

Sei y_1 Lösung der Anfangswertaufgabe

$$-y_1''(t) + q(t)y_1(t) = 0,\ y_1(a) = 0,\ y_1'(a) = 1$$

und y_2 Lösung der Anfangswertaufgabe

$$-y_2''(t) + q(t)y_2(t) = 0,\ y_2(b) = 0,\ y_2'(b) = 1.$$

$$W(t) = \det \begin{pmatrix} y_1(t) & y_2(t) \\ y_1'(t) & y_2'(t) \end{pmatrix}$$

heißt Wronski–Determinante des Sturm–Liouville–Problems und ist konstant.

Falls $y = 0$ einzige Lösung des homogenen Sturm–Liouville–Problems ($f = 0$) ist, so ist das Sturm–Liouville–Problem für alle stetigen f eindeutig lösbar. Die Wronski–Determinante verschwindet dann nicht.

Die Funktion

$$G(t,s) = -\frac{1}{W(t)} \begin{cases} y_1(t)y_2(s), & t \le s \\ y_2(t)y_1(s), & t \ge s \end{cases}$$

heißt dann Greensche Funktion des Sturm–Liouville–Problems.

Die eindeutige Lösung des Sturm–Liouville–Problems ist dann gegeben durch

$$y(t) = \int_a^b G(t,s)f(s)ds.$$

Beweis: Nach Definition ist

$$W(t) = y_1(t)y_2'(t) - y_2(t)y_1'(t),$$

also

$$\begin{aligned} W'(t) &= y_1'(t)y_2'(t) + y_1(t) \underbrace{y_2''(t)}_{=q(t)y_2(t)} - y_2'(t)y_1'(t) - y_2(t) \underbrace{y_1''(t)}_{=q(t)y_1(t)} \\ &= q(t)y_1(t)y_2(t) - q(t)y_1(t)y_2(t) \\ &= 0 \end{aligned}$$

und damit $W(t) = c$ konstant.

Sei nun die Lösung des homogenen Problems eindeutig. Wäre $y_1(b) = 0$, so wäre y_1 nichttriviale Lösung des homogenen Problems. Nach Voraussetzung gilt also

$$c = W(b) = y_1(b) \neq 0.$$

Es ist

$$\begin{aligned} -cy(t) &= -c \int_a^b G(t,s) f(s) ds \\ &= \int_a^t y_2(t) y_1(s) f(s) ds + \int_t^b y_1(t) y_2(s) f(s) ds \\ &= y_2(t) \int_a^t y_1(s) f(s) ds + y_1(t) \int_t^b y_2(s) f(s) ds \end{aligned}$$

und insbesondere erfüllt y die homogenen Dirichlet–Randbedingungen.
Wir zeigen, dass y zweimal stetig differenzierbar ist. Zunächst

$$\begin{aligned} -cy'(t) &= y_2'(t) \int_a^t y_1(s) f(s) ds + y_2(t) y_1(t) f(t) \\ &\quad + y_1'(t) \int_t^b y_2(s) f(s) ds - y_1(t) y_2(t) f(t) \\ &= y_2'(t) \int_a^t y_1(s) f(s) ds + y_1'(t) \int_t^b y_2(s) f(s) ds. \end{aligned}$$

und

$$\begin{aligned} -cy''(t) = &\underbrace{y_2''(t)}_{=q(t)y_2(t)} \int_a^t y_1(s) f(s) ds + y_2'(t) y_1(t) f(t) \\ &+ \underbrace{y_1''(t)}_{=q(t)y_1(t)} \int_a^t y_2(s) f(s) ds - y_1'(t) y_2(t) f(t) \\ &= cf(t) - cq(t)y(t). \end{aligned}$$

Also ist y zweimal stetig differenzierbar, es gilt

$$-y''(t) + q(t)y(t) = f(t),\ y(a) = y(b) = 0$$

und damit ist y Lösung des Sturm–Liouville–Problems.

Seien y und z zwei Lösungen des Sturm–Liouville–Problems. Dann ist $y - z$ Lösung des homogenen Problems, also gilt $y = z$ und die Lösung ist eindeutig. □

Die Funktion G selbst ist natürlich nur für $t \neq s$ überhaupt differenzierbar in t. Für $t \neq s$ erfüllt sie die homogene Differentialgleichung. Dies werden wir später noch genauer untersuchen.

Damit haben wir die erste (analytische) Methode zur Lösung von Sturm–Liouville–Problemen gefunden.

1. Bestimme die Lösungen y_0, y_1 der Anfangswertprobleme.
2. Berechne die Greensche Funktion.
3. Berechne
$$y(t) = \int_a^b f(s)G(t,s)\mathrm{d}s.$$

6.2 Schießverfahren

Der Hintergrund dieser Verfahren ist einfach zu erraten. Eine Kanone, deren Abschusswinkel α variabel ist, stehe in einer Ebene am Punkt a. Das Zielobjekt stehe am Punkt b. Sei $y_\alpha(t)$ die Höhe der Kugel auf dem Weg von a nach b an einem Punkt t. Dann ist unser Ziel, durch Versuch und Irrtum $\alpha = \widetilde{\alpha}$ so zu wählen, dass $y_{\widetilde{\alpha}}(b) = 0$. y_α genügt einer gewöhnlichen Differentialgleichung, das gesuchte $y_{\widetilde{\alpha}}$ genügt den homogenen Dirichlet–Randbedingungen $y_{\widetilde{\alpha}}(a) = y_{\widetilde{\alpha}}(b) = 0$.

Dies interpretieren wir noch etwas um. Gesucht sei die Lösung y des Sturm–Liouville–Problems. Für jedes α besitzt die Anfangswertaufgabe mit der vorgegebenen Differentialgleichung und $y(a) = 0$, $y'(a) = \alpha$ eine Lösung und ist leicht numerisch berechenbar mit den Algorithmen des letzten Kapitels. Wir suchen nun ein α mit $F(\alpha) := y_\alpha(b) = 0$. F ist eine nichtlineare, eindimensionale, berechenbare Funktion. Wir suchen eine Nullstelle von F. Verfahren dazu lernen Sie in der Vorlesung Numerische Lineare Algebra kennen, etwa das Newton–Verfahren oder das Sekanten–Verfahren (Regula falsi). Voraussetzung zur Anwendung dieser Verfahren ist, dass die Funktion mindestens einmal stetig differenzierbar ist. Tatsächlich gilt der Satz:

Satz 6.3 (Differenzierbarkeit der Lösungen von Anfangswertaufgaben nach ihren Anfangswerten)
Sei $y_\alpha(t)$ die Lösung der Anfangswertaufgabe für die Differentialgleichung des

Sturm–Liouville–Problems für die Anfangswerte

$$y'_\alpha(a) = \alpha,\ y_\alpha(a) = 0.$$

Sei

$$F(\alpha) := y_\alpha(b).$$

Dann ist F differenzierbar.

Beweis: Übungen. □

Dies liefert uns ein numerisches Verfahren zur Lösung von Randwertaufgaben.

1. Implementiere die Funktion $F(\alpha)$, die eine Approximation für $y_\alpha(b)$ berechnet mit den numerischen Verfahren des letzten Kapitels.
2. Nutze ein numerisches Verfahren zur Bestimmung der Nullstelle von F, zum Beispiel das Newton–Verfahren oder das Sekanten–Verfahren.

6.3 Diskretisierungsverfahren

Wir können auch die Diskretisierungsidee aus dem letzten Kapitel zur Lösung einsetzen. Sei wieder

$$I_h = (t_0, \ldots, t_N)$$

ein zulässiges Gitter auf $[a, b]$, ohne Einschränkung wählen wir es in diesem Kapitel immer äquidistant. Sei y die Lösung des Sturm–Liouville–Problems. Wir suchen Gitterfunktionen

$$y_h = (y_k),\ y(t_k) \sim y_h(t_k).$$

In den inneren Punkten des Gitters diskretisieren wir die Differentialgleichung konsistent, gibt $N - 1$ Gleichungen, plus zwei Randbedingungen, macht insgesamt $N + 1$ Gleichungen für $N + 1$ Unbekannte.

Lemma 6.4 (Vorbemerkung zu Matrixnormen)
Sei $(V, ||\cdot||)$ ein normierter Vektorraum. Auf dem Vektorraum

$$L(V, V)$$

der linearen Abbildungen von V in sich selbst ist durch

$$||A|| = \sup_{x \in V,\, ||x||=1} ||Ax|| = \sup_{x \in V, x \neq 0} \frac{||Ax||}{||x||}$$

eine Norm definiert. Es gilt

$$||Ax|| \leq ||A|| \cdot ||x||.$$

Für $V = \mathbb{R}^n$ ist $L(V,V)$ der Raum der reellen $(n \times n)$–Matrizen. Es gilt dann für die euklidische Norm

$$||A||_2^2 = \rho(A^t A),$$

wobei $\rho(A^t A)$ der Betrag des betragsgrößten Eigenwerts von $A^t A$ ist. Insbesondere ist für symmetrisch positiv semidefinite Matrizen $||A||_2$ der größte Eigenwert von A.

Für die Maximumnorm gilt

$$||A||_\infty = \max_i \sum_k |A_{ik}|.$$

Falls A nur Einträge mit gleichem Vorzeichen hat, so ist also

$$||A||_\infty = ||A\mathbf{1}||_\infty$$

wobei $\mathbf{1}$ der Vektor im $\mathbb{R}^n$ ist, der nur aus Einsen besteht.

Beweis: In der Vorlesung Numerische Lineare Algebra . □

Wir betrachten als Modellproblem die eindimensionale **Poisson–Gleichung**

$$-y''(t) = f(t)$$

für t in $[0,1]$ auf einem äquidistanten Gitter mit homogenen Dirichlet–Randbedingungen, $h = 1/N$. Zur Diskretisierung nutzen wir die Formeln aus 3.30. Wir erhalten die Gleichungen

$$\begin{aligned} -\frac{y_0 - 2y_1 + y_2}{h^2} &= f_1 = f(t_1) \\ &\vdots \\ -\frac{y_{N-2} - 2y_{N-1} + y_N}{h^2} &= f_{N-1} = f(t_{N-1}) \end{aligned}$$

Setzen wir die Randbedingung $y_0 = y_N = 0$ ein, so erhalten wir für die Unbekannten $y_1, \ldots, y_{N-1}$ das lineare Gleichungssystem

$$\underbrace{\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}}_{=:L_h} \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix}}_{=:y_h} = \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{pmatrix}}_{=:f_h}.$$

Die Diskretisierung ist konsistent nach 3.30. Ist das Gesamtverfahren auch konvergent, d.h. gilt

$$||y|_{I_h} - y_h|| \mapsto 0$$

? Wir untersuchen das zunächst für die 2–Norm.

Die Eigenvektoren von L_h lassen sich leicht angeben: Es sind die Vektoren

$$y^k \in \mathbb{R}^{N-1},\ (y^k)_j = \sin(k\pi jh),\ k, j = 1, \ldots, N-1,$$

zu den Eigenwerten

$$\lambda_k = \frac{4}{h^2}\sin^2(kh\pi/2),\ k = 1, \ldots, N-1.$$

Mit den Additionstheoremen gilt nämlich

$$\begin{aligned}
&-\sin(x-y) + 2\sin(x) - \sin(x+y)\\
&= -\sin(x)\cos(y) + \cos(x)\sin(y) + 2\sin(x) - \sin(x)\cos(y) - \cos(x)\sin(y)\\
&= 2(1-\cos(y))\sin(x)\\
&= 2(1-\cos(\frac{y}{2}+\frac{y}{2}))\sin(x)\\
&= 2(1-\cos^2\frac{y}{2}+\sin^2\frac{y}{2})\sin(x)\\
&= (4\sin^2\frac{y}{2})\sin(x)
\end{aligned}$$

und dann durch Einsetzen von $x = k\pi jh$ und $y = k\pi h$.

Es gilt

$$\lambda_k \geq \lambda_1 \geq \frac{4}{h^2}\left(\frac{2}{\pi}\right)^2 \frac{h^2\pi^2}{4} = 4$$

unter Benutzung der Abschätzung

$$\sin x \geq \frac{2}{\pi}x,\ x \in [0, \pi/2].$$

L_h hat keinen Eigenwert 0. Insbesondere ist L_h invertierbar, das oben angegebene Gleichungssystem für y_h ist eindeutig lösbar, und sogar sehr effizient lösbar, denn L_h ist eine Tridiagonalmatrix.

L_h ist symmetrisch positiv semidefinit. Die Eigenwerte von L_h^{-1} sind die Kehrwerte der Eigenwerte von L_h, damit ist nach den Vorbemerkungen

$$||L_h^{-1}||_2 \leq 1/4.$$

Die Konvergenz definieren wir natürlich wieder wie in 4.23. Es gilt

$$\begin{aligned} ||y|_{I_h} - y_h|| &= ||L_h^{-1}(L_h y|_{I_h} - f_h)|| \\ &\leq \underbrace{||L_h^{-1}||}_{\text{Stabilitätsfaktor}} \cdot \underbrace{||L_h(y|_{I_h}) - f_h||}_{\text{Konsistenzfehler}} \end{aligned}$$

Für unser Verfahren gilt mit der euklidischen Norm: Der Stabilitätsfaktor ist beschränkt, die Konsistenz ist ein $O(h^2)$, also ist das Verfahren konvergent von der Ordnung 2. Dies legt nahe

Definition 6.5 (Stabilität für Diskretisierungen von Randwertproblemen)
Die Diskretisierung eines Randwertproblems heißt stabil, falls $||L_h^{-1}||$ unabhängig von h nach oben beschränkt ist.

Damit gilt sofort wieder unser Satz:

Satz 6.6 (Konvergenz für Diskretisierungen von Randwertproblemen)
Aus Stabilität und Konsistenz (der Ordnung p) folgt Konvergenz (der Ordnung p).

Leider steht hier zunächst nur die euklidische Norm. Dies ist ungünstig: Es garantiert uns keine punktweise Konvergenz und insbesondere keine punktweisen Fehlerabschätzungen.

Bisher haben wir Konvergenz immer bezüglich der Maximumnorm betrachtet. Das wollen wir auch beibehalten. Zur Untersuchung dieser Konvergenz benötigen wir einige spezielle Eigenschaften der oben angegebenen Matrix L_h.

Definition 6.7 (M-Matrizen)
Sei A eine reelle, invertierbare $N \times N$–Matrix. A heißt M–Matrix genau dann, wenn

$$A_{ii} > 0 \, \forall i, \; A_{ik} \leq 0 \, \forall i \neq k, \; A_{ik}^{-1} \geq 0 \, \forall i, k.$$

Die wesentliche Eigenschaft von Matrizen mit positiven Einträgen ist ihre Monotonie.

Lemma 6.8 (Monotonie für M–Matrizen)
Sei $L_h \in \mathbb{R}^{N \times N}$ eine M–Matrix. Seien

$$u, \, v \in \mathbb{R}^n, \, u \leq v.$$

Dann gilt

$$L_h^{-1} u \leq L_h^{-1} v.$$

Hier und im Folgenden sind Vektorungleichungen immer elementweise gemeint.

Beweis: Da alle Einträge von L_h^{-1} nichtnegativ sind, gilt

$$L_h^{-1}(v-u) \geq 0.$$

□

Definition 6.9 (schwach diagonaldominant)

Eine Matrix A heißt schwach diagonaldominant, falls in jeder Zeile die Summe der Beträge der Außerdiagonalelemente nicht größer ist als der Betrag des Hauptdiagonalelements, also

$$A_{ii} \geq \sum_{i \neq k} |A_{ik}| \, \forall i = 1, \ldots, N-1.$$

Satz 6.10 (L_h ist M–Matrix)
Sei A eine reelle, invertierbare, schwach diagonaldominante $N \times N$–Matrix mit positiven Hauptdiagonaleelementen und nicht–positiven Außerdiagonalelementen. Dann ist A eine M–Matrix. Insbesondere ist L_h eine M–Matrix.

Beweis: Sei $b < 0$, $Ax = b$. Angenommen,

$$x_j = \max\{x_1, \ldots, x_N\} > 0.$$

Dann gilt

$$\begin{aligned} A_{jj}x_j &= b_j - \sum_{k \neq j} A_{jk}x_k \\ &< -\sum_{k \neq j} A_{jk}x_j \\ &\leq A_{jj}x_j. \end{aligned}$$

Dies ist ein Widerspruch, also gilt

$$b < 0 \Longrightarrow A^{-1}b < 0.$$

Sei $b \leq 0$. Dann gilt

$$b^\epsilon = b - \epsilon < 0 \Longrightarrow 0 > A^{-1}b^\epsilon = A^{-1}b - A^{-1}\epsilon \Longrightarrow A^{-1}b \leq 0.$$

Mit der Wahl $b = -e_k$ folgt $A^{-1} \geq 0$, also ist A eine M–Matrix. □

Satz 6.11 (Stabilität des Standardverfahrens)
Das Standard–Differenzenschema zur Berechnung der Lösung der eindimensionalen Poisson–Gleichung ist stabil bezüglich der Maximumnorm.

Beweis: Zu zeigen ist: $||L_h^{-1}||_\infty$ ist unabhängig von h beschränkt. Wir betrachten das Problem auf dem Intervall $[0,1]$. Sei

$$w(t) = \frac{1}{2}t(1-t).$$

Dann ist

$$-w'' = 1,\ w(0) = 0,\ w(1) = 0,\ w \geq 0.$$

Sei $\mathbf{1}$ der Vektor, der nur aus Einsen besteht, und

$$w_h = w|_{I_h}.$$

L_h ist konsistente Diskretisierung der zweiten Ableitung, d.h.

$$||L_h(w_h) - \mathbf{1}||_\infty \leq Ch^2$$

und damit

$$L_h(w_h) \geq -Ch^2 + \mathbf{1}$$

und insbesondere, falls h klein genug ist,

$$L_h(w_h) \geq \frac{1}{2}\mathbf{1}.$$

Diese Vektorungleichungen sind wieder alle jeweils elementweise zu interpretieren. L_h ist M–Matrix. Nach 6.8 gilt

$$2w_h = 2L_h^{-1}L_h(w_h) \geq L_h^{-1}\mathbf{1}.$$

Dies setzen wir nun noch zusammen:

$$||L_h^{-1}||_\infty = ||L_h^{-1}\mathbf{1}||_\infty \leq 2||w_h||_\infty \leq 2||w||_\infty.$$

□

Bemerkung: Für den einfachen Fall der Poissongleichung, den wir hier betrachten, ist die Diskretisierung sogar exakt, d.h. $L_h w_h = 1$. Damit ist für diesen Fall sogar

$$||L_h^{-1}||_\infty \leq ||w||_\infty = \frac{1}{8}.$$

Bemerkung: Man könnte sich fragen, warum wir uns überhaupt die analytische Lösbarkeit der Sturm–Liouville–Probleme angeschaut haben bzw. noch anschauen: Am Ende wird ja doch nur diskretisiert und ein lineares Gleichungssystem gelöst. Gehen wir aber zum Sturm–Liouville–Problem mit $p = 1$ zurück, so sehen wir, dass unsere numerische Analyse mit leichten Änderungen bestehen bleibt, solange $q \geq 0$ ist — denn dann ist L_h weiterhin eine M–Matrix.

Sobald $q < 0$, also dort, wo das Finden analytischer Lösungen schwieriger war, bricht diese Argumentation aber zusammen. Unsere analytischen Schwierigkeiten für $q < 0$ übersetzen sich in numerische Schwierigkeiten.

Korollar 6.12 (Konvergenz des Standardverfahrens)
Das Standardverfahren zur Berechnung der Lösung der eindimensionalen Poisson–Gleichung ist konvergent von der Ordnung 2 (bezüglich der Maximumnorm).

Leider ist dieses Ergebnis unbefriedigend. Anders als bei den Anfangswertproblemen ist der Beweis recht uneinsichtig und nur schwer auf andere Differentialgleichungen oder Diskretisierungen übertragbar. Wir werden daher ein neues Hilfsmittel, die Variationsrechnung, kennenlernen.

6.4 Variationsmethoden

Bei der Behandlung des Steinwurfs in 4.50 hatten wir bereits bemerkt: Differentialgleichungen in der Physik entstehen häufig aus Energiebetrachtungen. Dabei wird meist ein durch Integrale definiertes Energiefunktional minimiert. Dies führt auf eine Integralgleichung für die Lösung.

Unter der Annahme, dass die Lösungen differenzierbar sind, können wir diese dann in eine Differentialgleichung umschreiben. Das ist allerdings eine Einschränkung und führt dazu, dass viele Probleme der Physik als Differentialgleichung keine Lösung besitzen. Es liegt daher nahe, statt der Differentialgleichungen die zugehörige Integralgleichung bzw. das Minimierungsproblem zu untersuchen.

Wir betrachten zunächst den Zusammenhang zwischen Integral– und Differentialgleichungen. Dies tun wir wieder an einem Spezialfall des Sturm–Liouville–Randwertproblems, diesmal mit **natürlichen Randbedingungen**

$$-(p(t)y'(t))' + q(t)y(t) = f(t),\ y(a) = 0,\ y'(b) = 0$$

mit p stetig differenzierbar, q stetig, und

$$p(t) \geq p_0 > 0, q(t) \geq 0.$$

Diese Differentialgleichung lässt sich, falls alle Ableitungen existieren, einfach zu einer Integralgleichung umschreiben. Sei y Lösung der Differentialgleichun. Sei

$$\varphi \in C^1,\ \varphi(a) = \varphi(b) = 0.$$

Dann gilt nach Multiplikation mit φ und Integration

$$\int_a^b (-(p(t)y'(t))' + q(t)y(t))\varphi(t)\mathrm{d}t = \int_a^b f(t)\varphi(t)\mathrm{d}t$$

und damit mit partieller Integration

$$\int_a^b p(t)y'(t)\varphi'(t) + q(t)y(t)\varphi(t)\mathrm{d}t = \int_a^b f(t)\varphi(t)\mathrm{d}t.$$

Der auftretende Randterm verschwindet, weil φ auf dem Rand verschwindet.

Umgekehrt: Sei $y \in C^2$. Falls die Integralgleichung für alle

$$\varphi \in C^\infty,\ \varphi(a) = \varphi(b) = 0$$

gilt, so ist y Lösung der Differentialgleichung (denn C^∞ liegt dicht in L^2). Die Differentialgleichung und die Integralgleichung sind also in diesem Fall äquivalent.

Bemerkenswert daran ist, dass die Integralgleichung einen erweiterten Lösungsbegriff hat: Sie macht bereits Sinn für Funktionen y in C^1, die Differentialgleichung nur für Funktionen in C^2. Damit die Integrale Sinn machen, benötigen wir lediglich, dass y und φ mit ihren Ableitungen in L^2 liegen.

Wir erinnern an die Ungleichung von Cauchy–Schwartz

$$|(f,g)| \leq ||f|| \cdot ||g||$$

und die übliche Definition des Skalarprodukts auf den quadratisch integrierbaren Funktionen

$$(f,g) := \int_a^b f(t)g(t)\mathrm{dt},\ ||f||_2^2 = (f,f).$$

Wir betrachten nun den Zusammenhang der Minimierung von Energiefunktionalen und linearen Gleichungen. Sätze dieser Art sind aus den Vorlesungen Numerische Lineare Algebra und der linearen Algebra wohlbekannt. Aufgaben dieser Form heißen **Variationsprobleme**.

Lemma 6.13 (Lösbarkeit von linearen Gleichungen und Variationsproblemen)
Sei A ein stetiger, positiv semidefiniter, symmetrischer, linearer Operator des euklidischen Vektorraums $(X, (\cdot,\cdot))$ X in sich selbst, d.h.

$$(Ax, x) \geq 0 \, \forall x \in X, \, (Ax, y) = (x, Ay) \, \forall x, y \in X.$$

Sei $b \in X$, und sei

$$I(x) := \frac{1}{2}(Ax, x) - (x, b).$$

Dann gilt

$$I(y) = \inf_{x \in X} I(x) \Leftrightarrow Ay = b.$$

Beweis: Nehme zunächst I sein Infimum in y an. Sei $v \in X$, $h \in \mathbb{R}$.

$$\begin{aligned} g(h) &:= I(y + hv) \\ &= \frac{1}{2}(Ay, y) + \frac{h}{2}(Ay, v) + \frac{h}{2}(Av, y) + \frac{1}{2}(Av, v) - (Ay, b) - h(Av, b) \\ &= I(y) + h(Ay - b, v) + \frac{1}{2}h^2(Av, v) \end{aligned}$$

hat dann ein Minimum bei $h = 0$. Also gilt

$$0 = g'(0) = (Ay - b, v),$$

also insbesondere für $v = Ay - b$

$$||Ay - b||_2^2 = 0$$

und damit $Ay = b$.

Sei nun $Ay = b$. Dann gilt wie oben

$$I(y + hv) = I(y) + \frac{1}{2}h^2(Av, v) \geq I(y)$$

und damit nimmt I sein Minimum in y an. Falls A positiv definit ist, so ist das Minimum eindeutig. □

Wir wollen nun Existenz und Eindeutigkeit der Lösung des Sturm–Liouville–Problems mit natürlichen Randbedingungen beweisen. Wir definieren zunächst unsere Grundräume.

Definition 6.14 (Grundräume zur Lösung des Variationsproblems)
Wir betrachten alle Funktionen immer auf dem ganzen Intervall $[a, b]$.

1. *L^2 ist der Raum der Funktionen auf $[a,b]$, die quadratisch integrierbar sind, d.h.*
$$\int_a^b f(t)^2 dt < \infty.$$
Mit dem Skalarprodukt
$$(f,g) = \int_a^b f(t)g(t)dt,\ ||f||^2 := (f,f)\,\forall f,g \in L^2$$
ist L^2 ein Hilbertraum, insbesondere vollständig. Es gilt die Cauchy–Schwarz–Ungleichung
$$|(f,g)| \leq ||f|| \cdot ||g||.$$

2. *C^k ist der Raum der k–mal stetig differenzierbaren Funktionen auf $[a,b]$. C^0 ist vollständig bzgl. der Unendlichnorm, aber nicht bezüglich der L^2–Norm (Analysis II).*

3. *C^∞ ist der Raum der unendlich oft differenzierbaren Funktionen.*

4. *C_0^∞ ist der Raum der unendlich of differenzierbaren Funktionen auf $[a,b]$ mit verschwindenden Randwerten, also*
$$C_0^\infty = \{\varphi \in C^\infty(a,b) : \varphi^{(k)}(a) = \varphi^{(k)}(b) = 0\,\forall k \geq 0\}.$$

In 2.47 haben wir bereits eine nichttriviale Funktion aus C_0^∞ kennengelernt.

Beispiel 6.15 (unendlich oft differenzierbare Funktion mit kompaktem Träger)
$$w_\epsilon(x) = \frac{C}{\epsilon}\begin{cases} e^{1/(x^2/\epsilon^2-1)}, & |x| < \epsilon \\ 0, & \textit{sonst} \end{cases}$$
ist eine nichttriviale C_0^∞–Funktion. C sei so gewählt, dass ihr Integral über $\mathbb{R}$ 1 ist.

Die Vorbemerkungen fassen wir formal zusammen.

Satz 6.16 (Variationsproblem und Differentialgleichung)
Sei
$$X = \left\{u \in C^1([a,b]) : u(a) = 0\right\}.$$
Es sei
$$B(v,w) = \int_a^b p(t)v'(t)w'(t) + q(t)v(t)w(t)dt\,\forall v,w\ \in X$$
und
$$F(v) = \int_a^b f(t)v(t)dt.$$

Weiter sei

$$I(v) = \frac{1}{2}B(v,v) - F(v).$$

1. *$I(v)$ nimmt genau dann sein Infimum an für $v = y$, falls*

$$B(y,\varphi) = F(\varphi)\,\forall \varphi \in X.$$

2. *Für $y \in C^2$ ist zusätzlich zu den beiden Aussagen äquivalent: y erfüllt die Differentialgleichung und $y(a) = y'(b) = 0$.*

Beweis:
Zu 1.: Setze wieder $v = y + hw$ wie in Lemma 6.13.

Zu 2.: Falls y die Randwertaufgabe erfüllt, so ist dies die Vorbemerkung mit partieller Integration. Sei nun die Variationsgleichung erfüllt. Wähle zunächst $\varphi \in X$ mit $\varphi(b) = 0$. Dann gilt

$$\begin{aligned}\int_a^b f(t)\varphi(t)\mathrm{d}t &= F(\varphi)\\ &= B(y,\varphi)\\ &= \int_a^b p(t)y'(t)\varphi'(t) + q(t)y(t)\varphi(t)\mathrm{d}t.\end{aligned}$$

Also ist wieder nach der Vorbemerkung y Lösung der Differentialgleichung.

Sei nun $\varphi \in X$ beliebig mit $\varphi(b) \neq 0$. Es gilt

$$\begin{aligned}0 &= B(y,\varphi) - F(\varphi)\\ &= \int_a^b p(t)y'(t)\varphi'(t) + q(t)y(t)\varphi(t)\mathrm{d}t - \int_a^b f(t)\varphi(t)\mathrm{d}t\\ &= -\int_a^b ((p(t)y'(t))' + q(t)y(t) - f(t))\varphi(t) + [py'\varphi]_a^b.\end{aligned}$$

Da y die Differentialgleichung erfüllt, verschwindet das Integral und wegen $\varphi(a) = 0$, $\varphi(b) \neq 0$ und $p(b) \geq p_0 > 0$ gilt $y'(b) = 0$. □

Vorlesungsnotiz: 28. Juni 2013

Damit sind also alle drei Aufgaben (fast) gleichwertig. Wollen wir etwa die Randwertaufgabe numerisch oder analytisch lösen, so können wir ebenso eine der anderen beiden wählen. Dies ist die Grundidee der Variationsrechnung. Es kann allerdings

der Fall auftreten, dass die Lösung y der Variationsaufgabe gar nicht zweimal differenzierbar und damit keine klassische Lösung ist. In diesem Fall sprechen wir von einer **schwachen Lösung der Randwertaufgabe**.

Wir haben also die Existenz einer Lösung des Randwertproblems auf die Existenz eines Minimierers von I zurückgeführt. Diese wollen wir beweisen. Üblicherweise geht man dabei so vor: Man zeigt, dass das Infimum I_0 von I endlich ist, und dass jede Minimalfolge, also eine Folge y_n mit

$$I(y_n) \to I_0,$$

eine Cauchyfolge ist. Aus der Vollständigkeit des Grundraums folgt dann die Existenz eines Minimierers. Leider ist unser Raum nicht einmal vollständig.

Dies gehen wir zunächst an. Wir wollen C^1 in L^2 vervollständigen, indem wir seinen Abschluss mit hinzunehmen. Bisher haben wir als einzige Eigenschaften der differenzierbaren Funktionen die partielle Integration genutzt. Es liegt also nahe, eine Erweiterung von C^1 als den Teilraum von L^2 zu definieren, in dem die partielle Integration erlaubt ist.

Definition 6.17 (schwache Differenzierbarkeit)
$v' \in L^2$ heißt schwache Ableitung von $v \in L^2$, falls $\forall \varphi \in C_0^\infty$

$$\int_a^b v(t)\varphi'(t)dt = -\int_a^b v'(t)\varphi(t)dt$$

und entsprechend für höhere Ableitungen.

Damit gilt natürlich insbesondere: Falls eine Funktion differenzierbar ist, so ist sie auch schwach differenzierbar und die Ableitungen sind gleich.

Beispiel 6.18 (schwache Differenzierbarkeit der Betragsfunktion)

$v(x) = |x|$ ist schwach differenzierbar auf $[-1, 1]$:

$$\begin{aligned}\int_{-1}^1 |t|\varphi'(t)dt &= \int_{-1}^0 (-t)\varphi'(t)dt + \int_0^1 t\varphi'(t)dt \\ &= \int_{-1}^0 \varphi(t)dt + \int_0^1 -\varphi(t)dt + [t\varphi]_0^1 - [t\varphi]_{-1}^0 \\ &= -\int_{-1}^1 \varphi(t)\,\mathrm{sgn}(t)dt\end{aligned}$$

Die (schwache) Ableitung der Betragsfunktion ist also die Signumfunktion. Für die Signumfunktion gilt

$$\int_{-1}^{1} \operatorname{sgn}(t)\varphi'(t)dt = -\int_{-1}^{0} \varphi'(t)dt + \int_{0}^{1} \varphi'(t)dt$$
$$= -2\varphi'(0).$$

Dies lässt sich nicht als Integral schreiben, also ist die Signumfunktion nicht schwach differenzierbar, obwohl sie eine L^2–Funktion ist. Es gilt also

$$H^1 \neq L^2.$$

Bemerkung: Offensichtlich ist die Signumfunktion im distributionellen Sinne differenzierbar, ihre Ableitung ist 2δ mit der Delta–Distribution δ, das betrachten wir in dieser Vorlesung nicht.

Definition 6.19 (Sobolev–Räume)

Der Raum H^1 ist der Raum der schwach differenzierbaren Funktionen und heißt Sobolev–Raum. Auf H^1 definieren wir das Skalarprodukt

$$(f,g) = (f,g)_{L^2} + (f',g')_{L^2} = \int_a^b f(x)g(x)dx + \int_a^b f'(x)g'(x)dx$$

mit der zugehörigen Norm $||f||_{H^1}^2 = (f,f)$.

Satz 6.20 (Vollständigkeit der Sobolev–Räume)
H^1 ist vollständig.

Beweis: Sei (f_n) eine Cauchyfolge bzgl. $||\cdot||_{H^1}$. Dann sind (f_n) und (f_n') Cauchyfolgen bzgl. L^2. L^2 ist vollständig, also gilt

$$f_n \to u,\ f_n' \to v,\ u,v \in L^2.$$

Dann gilt für $\varphi \in C_0^\infty$

$$\begin{aligned}\int_a^b u(t)\varphi'(t)\mathrm{d}t &= \lim_{k\to\infty}\int_a^b f_k(t)\varphi'(t)\mathrm{d}t \\ &= \lim_{k\to\infty} -\int_a^b f_k'(t)\varphi(t)\mathrm{d}t \\ &= -\int_a^b v(t)\varphi(t)\mathrm{d}t\end{aligned}$$

und damit ist v schwache Ableitung von u.

$$||f_n - u||_{H^1}^2 = ||f_n - u||_{L^2}^2 + ||f_n' - v||_{L^2}^2 \to 0$$

also konvergiert f_n gegen die schwach differenzierbare Funktion u bzgl. $||\cdot||_{H^1}$, also ist H^1 vollständig.

Tatsächlich ist H^1 der kleinste Raum mit dieser Eigenschaft, der C^1 enthält, also die Vervollständigung von C^1 bzgl. $||\cdot||_{H^1}$. □

Zwei der wichtigsten Sätze über Sobolevräume sind die Sobolevsche Ungleichung und der Sobolevsche Einbettungssatz. In einer Dimension sind sie trivial.

Satz 6.21 (Sobolevsche Ungleichung)
Sei $f \in C^1$. Dann gibt es ein $C' > 0$ mit

$$||f||_\infty \leq C'||f||_{H^1}.$$

Beweis: Wir betrachten das Problem auf $[-1, 1]$. Sei zunächst $s \leq 0$. Dann gilt

$$\begin{aligned} f(s) &= \int_0^1 ((t-1)f(t+s))' \mathrm{d}t \\ &= \int_0^1 f(t+s) + (t-1)f'(t+s) \mathrm{d}t \end{aligned}$$

und damit nach Cauchy–Schwarz

$$\begin{aligned} |f(s)| &\leq (\int_0^1 1 \mathrm{d}t)^{1/2} (\int_{-1}^1 f(t)^2 \mathrm{d}t)^{1/2} + (\int_0^1 (t-1)^2 \mathrm{d}t)^{1/2} (\int_{-1}^1 f'(t)^2 \mathrm{d}t)^{1/2} \\ &\leq C'||f||_{H^1}. \end{aligned}$$

Für $s > 0$ betrachtet man $-t + s$ statt $t + s$ und bekommt dieselbe Ungleichung. □

Satz 6.22 (H^1-Funktionen sind stetig, Sobolevscher Einbettungssatz)
Sei $f \in H^1$. Dann gibt es eine stetige Funktion g mit $f = g$ f.ü., und

$$||g||_\infty \leq C||f||_{H^1}.$$

Beweis: Sei $f \in H^1$. Die Funktionen

$$f_n(s) = \int_a^b w_{1/n}(s-t) f(t) \mathrm{d}t$$

liegen in C^∞ (Differentiation unter dem Integralzeichen). Sie konvergieren gegen f bzgl. H^1, also sind sie insbesondere eine Cauchyfolge in H^1. Dies sieht man anschaulich ein – ein einfacher, aber längerer Beweis für alle Dimensionen findet sich

in Evans [2010], Anhang C.4.

Es gilt

$$||f_n - f_m||_\infty \leq ||f_n - f_m||_{H^1} \to 0,$$

sie sind also auch eine Cauchyfolge in C^0 bzgl. $||\cdot||_\infty$. C^0 ist vollständig bezüglich $||\cdot||_\infty$, also gilt

$$f_n \to g \text{ bzgl. } ||\cdot||_\infty,\ g \in C^0.$$

Damit konvergiert f_n aber auch gegen g bzgl. L^2. f_n konvergiert also gegen g und f bezüglich L^2, also gilt $f = g$ f.ü. Weiter ist

$$||g||_\infty = \lim_{n\to\infty} ||f_n||_\infty \leq C \lim_{n\to\infty} ||f_n||_{H^1} = C||f||_{H^1}.$$

□

Der Sobolevsche Einbettungssatz hat eine wichtige Folgerung. Wir haben die H^1–Funktionen als L^2–Funktionen definiert. Damit besitzen sie keine Punktauswertung: L^2–Funktionen dürfen auf Nullmengen umdefiniert werden, ohne dass sie sich ändern. Da jetzt aber ohne Einschränkung jede H^1–Funktion stetig ist, können wir sie an Punkten auswerten.

Wir kommen zu einer der wichtigsten Ungleichungen für die Variationsrechnung.

Satz 6.23 (Poincaré-Ungleichung)

Sei

$$X = \{v \in H^1 : v(a) = 0\}.$$

Dann gibt es eine Konstante $C > 0$ mit

$$||v||_{L^2}^2 \leq C^2||v'||_{L^2}^2 \,\forall v \in X.$$

Beweis: Mit Cauchy–Schwarz:

$$\begin{aligned}(v(t))^2 &= \left(\int_a^t v'(s)\mathrm{d}s\right)^2 \\ &\leq \int_a^t 1\mathrm{d}s \cdot \int_a^t v'(s)^2\mathrm{d}s\end{aligned}$$

und damit

$$\begin{aligned}\int_a^b v(t)^2\mathrm{d}t &\leq \int_a^b (t-a)\int_a^t v'(s)^2\mathrm{d}s\mathrm{d}t \\ &\leq \frac{1}{2}(b-a)^2\int_a^b v'(s)^2\mathrm{d}s.\end{aligned}$$

□

Satz 6.24 (Lösbarkeit des Sturm–Liouville–Problems)
Sei

$$X := \{y \in H^1 : y(a) = 0\}$$

versehen mit der Norm

$$||\cdot||_X := ||\cdot||_{H^1}.$$

Sei wieder wie in 6.16

$$B(v,w) = \int_a^b p(t)v'(t)w'(t) + q(t)v(t)w(t)dt \, \forall v, w \in X$$

und

$$F(v) = \int_a^b f(t)v(t)dt$$

sowie

$$I(v) = \frac{1}{2}B(v,v) - F(v).$$

Dann hat I in X einen eindeutigen Minimierer, und damit das Randwertproblem eine (schwache) Lösung.

Mit dem Darstellungssatz von Riesz aus der Funktionalanalysis zeigt man leicht das Lemma von Lax–Milgram (z.B. in Alt [2007]), und hieraus folgt mit den gezeigten Sätzen die eindeutige Lösbarkeit des Variationsproblems. Wir zeigen den Satz zu Fuß.

Beweis: Sei

$$||v||_B^2 := B(v,v).$$

Nach Voraussetzung an p und q gilt

$$||v||_B^2 \geq p_0||v'||^2$$

und mit Poincaré

$$\begin{aligned} ||v||_X^2 &= ||v||_{L^2}^2 + ||v'||_{L^2}^2 \\ &\leq (C^2+1)||v'||_{L^2}^2 \\ &\leq \frac{C^2+1}{p_0}||v||_B^2. \end{aligned}$$

Damit ist $||\cdot||_B$ eine zu $||\cdot||_{H^1}$ äquivalente Norm, denn es gilt auch

$$||v||_B^2 \leq ||p||_\infty||v'||_{L^2}^2 + ||q||_\infty||v||_{L^2}^2,$$

und $B(\cdot,\cdot)$ ist ein Skalarprodukt.

Mit Cauchy-Schwarz und Poincaré gilt

$$\begin{aligned} I(v) &\geq \frac{p_0}{2}||v'||_{L^2}^2 - ||f||_{L^2} \cdot ||v||_{L^2} \\ &\geq \frac{p_0}{2}||v'||_{L^2}^2 - C||f||_{L^2} \cdot ||v'||_{L^2}. \end{aligned}$$

Die rechte Seite ist eine quadratische Funktion in $||v'||_{L^2}$, $p_0 > 0$, also ist I nach unten beschränkt und hat ein Minimum I_0.

Vorlesungsnotiz: 1. Juli 2013

Noch zu zeigen: Das Minimum wird angenommen.

Sei (v_n) eine Minimalfolge, also

$$I(v_n) \to I_0.$$

Wir wollen zeigen, dass v_n eine Cauchyfolge ist. Dazu gibt es einen Standardtrick aus der Linearen Algebra mit Hilfe der Parallelogrammidentität (siehe z.B. Alt [2007], S. 97)

$$||u+v||_B^2 + ||u-v||_B^2 = 2||u||_B^2 + 2||v||_B^2 :$$

$$\begin{aligned} \frac{p_0}{1+C^2}||v_n - v_m||_X &\leq ||v_n - v_m||_B^2 \\ &= 2||v_n||_B^2 + 2||v_m||_B^2 - ||v_n + v_m||_B^2 \\ &= 2||v_n||_B^2 + 4F(v_n) + 2||v_m||_B^2 + 4F(v_m) - 4\left|\left|\frac{v_n+v_m}{2}\right|\right|_B^2 - 8F\left(\frac{v_n+v_m}{2}\right) \\ &= 4I(v_n) + 4I(v_m) - 8I\left(\frac{v_n+v_m}{2}\right) \\ &\leq 4I(v_n) + 4I(v_m) - 8I_0 \\ &\to_{n,m\to\infty} 0. \end{aligned}$$

Also ist v_n eine Cauchyfolge und konvergiert gegen ein v, denn X ist vollständig.

Noch zu zeigen:

$$I(v) = I_0.$$

B und F sind stetig, denn mit Cauchy–Schwarz gilt

$$|B(u,v)| \leq ||p||_\infty ||u'||_{L^2}||v'||_{L^2} + ||q||_\infty ||u||_{L_2}||v||_{L_2}$$

und

$$|F(v)| \leq ||q||_{L^2}||f||^2_{L^2}||v||^2_{L^2}.$$

Also gilt

$$\begin{aligned} I(v) - I(v_n) &= \frac{1}{2}(B(v,v) - B(v_n,v_n)) + F(v - v_n) \\ &= \frac{1}{2}((B(v,v) - B(v_n,v)) + (B(v_n,v) - B(v_n,v_n)) + F(v - v_n) \\ &\to_{n\to\infty} 0 \end{aligned}$$

und damit $I(v) = I_0$. □

Nun können wir ein numerisches Verfahren zur Lösung von Randwertproblemen für gewöhnliche Differentialgleichungen mit Variationsmethoden definieren.

Definition 6.25 Ritz–Galerkin–Verfahren
Sei X_h ein endlichdimensionaler Teilraum von X. $y_h \in X_h$ heißt Galerkin–Lösung, falls y_h I auf X_h minimiert.

Mit Hilfe der Sätze können wir die Konvergenz des Galerkin–Verfahrens nachweisen.

Satz 6.26 (Konvergenz des Galerkin–Verfahrens und Fehlerabschätzung)
Sei X_h eine Folge von Räumen mit

$$\min_{x\in X_h} ||y - x||_{H^1} = d(y, X_h) \to 0 \,\forall y \in X.$$

Dann konvergiert y_h gegen y bezüglich der Maximumnorm.

Es gibt eine Konstante C, die nicht von h abhängt, mit

$$||y - y_h||_\infty \leq C d(y, X_h).$$

Beweis: Sei y die echte Lösung, y_h die approximative. Wegen der Minimierungseigenschaften gilt

$$B(y_h, v) = F(v) = B(y, v) \Rightarrow B(y_h - y, v) = 0 \,\forall v \in X_h.$$

B ist ein Skalarprodukt, also gilt wieder mit Cauchy–Schwarz

$$\begin{aligned} ||y - y_h||^2_B &= B(y - y_h, y - y_h) \\ &= B(y - y_h, y) && y_h \in X_h \\ &= B(y - y_h, y - v) \\ &\leq ||y - y_h||_B \cdot ||y - v||_B. \end{aligned}$$

Insbesondere gilt also

$$||y - y_h||_B \leq ||y - v||_B.$$

Da die B–Norm und die H^1–Norm äquivalent sind, gilt mit einer Konstanten C dann auch

$$||y - y_h||_{H^1} \leq C||y - v||_{H^1}.$$

Da $v \in X_h$ beliebig war, folgt schon mal

$$||y - y_h||_{H^1} \leq C \inf_{v \in X_h} ||y - v||_{H^1} = Cd(y, X_h)$$

und damit

$$y_h \to_{H^1} y.$$

Dies ist noch nicht ganz das Gewünschte: Das ist nur eine Konvergenz bezüglich der H^1–Norm.

Aber mit den bewiesenen Sätzen gilt

$$\begin{array}{rcll} ||y - y_h||_\infty & \leq & C'||y - y_h||_{H^1} & \text{(Sobolev 6.21)} \\ & = & C\,C'\,d(y, X_h). & \end{array}$$

□

Überzeugend an dieser Vorgehensweise ist, dass dieser Beweis deutlich eleganter ist als die Beweise für die Diskretisierung etwa bei den M–Matrizen: Die Konvergenz gilt unabhängig von der Art der Diskretisierung für alle Sturm–Liouville–Probleme, die die Voraussetzungen erfüllen.

Der Satz sagt: Die Qualität der Näherungen y_h hängt davon ab, wie gut die Teilräume X_h den Raum X bezüglich der H^1–Norm approximieren. Wir haben die Aufgabe der Lösung der Differentialgleichung auf die Aufgabe, die Approximationsgüte von linearen Unterräumen zu bestimmen, zurückgeführt und damit von der Differentialgleichung (bis auf eine Konstante) abgekoppelt.

Im Kapitel über Interpolation haben wir unter anderem die folgenden Möglichkeiten kennengelernt, Funktionen in X zu approximieren:

1. Polynome: Wir wählen als $X_{1/n}$ den Polynomraum $\mathcal{P}_n$ mit $p(a) = 0$.
2. Trigonometrische Polynome (Fouriertransformation): Hier wählen wir als Ansatzfunktionen $\sin(kt)$ und $\cos(kt)$.
3. Splines: Diese hatten die deutlich besten Approximationseigenschaften. Sie sind das Standardwerkzeug zur Lösung von Variationsproblemen. In diesem Zusammenhang bezeichnet man sie als Finite Elemente.

Die allgemeine Form von Finite–Elemente–Aproximationen lässt sich leicht erklären.

Sei X ein Funktionenraum über dem zusammenhängenden Gebiet Ω (dabei ist uns die Dimension von Ω egal, für gewöhnliche Differentialgleichungen der Raumdimension 1 ist Ω ein Intervall). Wir suchen die Lösung $y \in X$ des Variationsproblems

$$B(y, \varphi) = F(\varphi) \, \forall \varphi \in X.$$

Zunächst teilen wir Ω in Teilgebiete Ω_k auf, typischerweise in einer Dimension in Intervalle, in zwei Dimensionen in Dreiecke, in drei Dimensionen in Tetraeder oder Rechtecke.

Auf jedem Teilgebiet Ω_k definieren wir nun einen linearen Funktionenraum V_k, etwa den der Polynome vom Grad $\leq m$. Häufig wird dabei zunächst der Funktionenraum V_0 auf einem Referenzgebiet Ω_0 (Referenzelement) definiert. Sei F eine Abbildung, die Ω auf Ω_0 abbildet. Dann definieren wir

$$V_k := \{v \circ F : v \in V_0\}.$$

Unseren Approximationsraum X_h könnten wir nun definieren als

$$\{v : \Omega \mapsto \mathbb{R}, \, v|_{\Omega_k} \in V_k\}.$$

In diesem Fall wäre aber für die Sturm–Liouville–Probleme nicht einmal $X_h \subset X$, denn die Funktionen werden über die Ränder zwischen zwei Teilgebieten nicht stetig sein, also auch nicht in H^1 liegen. Wie bei den Splines fügen wir daher eine zusätzliche Voraussetzung an die Glattheit ein und definieren etwa

$$V_h := \{v : \Omega \mapsto \mathbb{R}, \, v|_{\Omega_k} \in V_k, \, v \text{ stetig}\}.$$

Die Funktionen in diesem Funktionenraum verhalten sich wie die Betragsfunktion (stetig, in einem Punkt nicht differenzierbar), sie sind wieder schwach differenzierbar wie in der Vorbemerkung, also in H^1.

Zur Bestimmung der Lösung y_h verschaffen wir uns zunächst eine Basis $v_1, \ldots, v_M$ von V_h. Da $y_h \in V_h$, gibt es Entwicklungskoeffizienten α_k mit

$$y_h = \sum_{k=1}^{M} \alpha_k v_k.$$

Dann gilt sicherlich

$$F(v_j) = B(y_h, v_j) = \sum_{k=1}^{M} B(v_k, v_j) \alpha_k, \; j = 1. \ldots, M.$$

Mit der **Steifigkeitsmatrix**

$$B \in \mathbb{R}^{M\times M},\ B_{k,j} := B(v_k, v_j)$$

und dem **Lastvektor**

$$F \in \mathbb{R}^{M},\ F_j := F(v_j)$$

gilt für den Vektor der Koeffizienten $\alpha = (\alpha_k)$

$$B\alpha = F.$$

Da die Bilinearform B symmetrisch ist, ist B eine symmetrische Matrix. die Basis wird dabei möglichst so gewählt, dass B möglichst viele Nullen enthält (dünn besetzt ist), damit das zugehörige lineare Gleichungssystem einfach gelöst werden kann.

Dies schauen wir uns nun an für gewöhnliche Differentialgleichungen und den Fall der linearen Splines der Ordnung 2. Sei also wieder

$$I_h = (t_0, \ldots, t_N)$$

ein zulässiges Gitter. In 2.44 hatten wir bereits eine Basis $\{v_0, \ldots, v_{N-1}\}$ des Splineraums angegeben, mit

$$v_i(t) = \begin{cases} \frac{t-t_i}{t_{i+1}-t_i}, & t_i \le x < t_{i+1} \\ \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}, & t_{i+1} \le t < t_{i+2} \\ 0, & \text{sonst.} \end{cases}.$$

$B_{k,j}$ ist höchstens dann ungleich 0, wenn die Träger von v_k und v_j überlappen. Also ist B eine Tridiagonalmatrix, das lineare Gleichungssystem ist sehr leicht lösbar. All dies können wir, anders als bei den Diskretisierungen, problemlos auf nicht–äquidistanten Gittern durchführen.

Wir testen nun, ob die Splineräume die Voraussetzung aus Satz 6.26 erfüllen. Wir nehmen hierzu an, dass $y \in H^2$, tatsächlich ist für die Lösungen unseres Sturm–Liouville–Problems dies immer erfüllt.

Wir benötigen eine Abschätzung für den Minimalabstand von y und dem Splineraum, gemessen in der H^1–Norm. Da bietet es sich an, die Differenz zwischen y und der Spline–Interpolation von y abzuschätzen.

Wir betrachten das Intervall $[0, 1]$ und die lineare Interpolation p von y an den Stellen 0 und 1 und $w = y - p$. Es gilt

$$p(t) = y(0) + (y(1) - y(0))t,\ p'(t) = y(1) - y(0)$$

und damit, wieder mal mit Cauchy–Schwarz,

$$||p'(t)||_{L^2}^2 = (y(1) - y(0))^2 = (\int_0^1 1 \cdot y'(t)\mathsf{dt})^2 \leq 1 \cdot \int_0^1 y'(t)^2\mathsf{dt} = ||y'||_{L^2}^2.$$

Da $w(0) = 0$, gilt mit Poincaré

$$\begin{aligned} ||w||_{L^2}^2 &\leq C^2||w'||_{L^2}^2 \\ &= C^2(||y'||_{L^2}^2 + (y(1) - y(0)) \int_0^1 y'(t)\mathsf{dt} + (y(1) - y(0))^2) \\ &= C^2||y'||_{L^2}^2. \end{aligned}$$

Da $w(0) = w(1) = 0$, ist

$$\int_0^1 w'(t)\mathsf{dt} = w(1) - w(0) = 0.$$

Mit Poincaré ist wegen $y \in H^2$ und $p'' = 0$

$$\begin{aligned} ||w'||_{L^2}^2 &\leq ||w'||_{L^2}^2 - 2w'(0) \int_0^1 w'(t)\mathsf{dt} + w'(0)^2 \\ &= ||w'(t) - w'(0)||_{L^2}^2 \\ &\leq C^2||w''||_{L^2}^2 \\ &= C^2||y''||_{L^2}^2. \end{aligned}$$

Der Satz von Poincaré gilt also auch, wenn

$$\int_0^1 w(t)\mathsf{dt} = 0.$$

Damit gilt aber auch

$$||w||_{L^2}^2 \leq C^2||y'||_{L^2}^2 \leq C^4||y''||_{L^2}^2.$$

Nun skalieren wir diese Ungleichungen, d.h. statt des Intervalls $[0, 1]$ betrachten wir $[0, h]$. Sei $F(t) = f(ht)$. Dann gilt

$$\begin{aligned} \int_0^h f(t)^2\mathsf{dt} &= h \int_0^1 F(t)^2\mathsf{dt} \\ &\leq hC^2 \int_0^1 (F'(t)^2)\mathsf{dt} \\ &\leq hh^2C^2 \int_0^1 (f'(ht)^2)\mathsf{dt} \\ &\leq h^2C^2 \int_0^h f'(t)^2\mathsf{dt}. \end{aligned}$$

Bei jeder Anwendung von Poincaré gewinnen wir also einen Faktor h^2. Für die obigen Formeln ergibt sich

$$||w'||_{L^2}^2 \leq C^2 h^2 ||y''||_{L^2}^2,\ ||w||_{L^2}^2 \leq C^4 h^4 ||y''||_{L^2}^2$$

und natürlich gelten diese Ungleichungen für jedes Intervall der Länge h.

Sei nun $I_h = (t_0 \ldots, t_N)$ ein zulässiges Gitter mit Feinheit h. Sei p der lineare Spline, der y an den Punkten t_k, $k = 0, \ldots, N$, interpoliert. Wir wenden die Ungleichung in jedem Teilintervall $[t_i, t_{i+1}]$ an und erhalten für $w = y - p$

$$\begin{aligned} ||w||_{L^2(a,b)}^2 &= \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} w(t)^2 \mathrm{dt} \\ &\leq C^4 h^4 ||y''||_{L^2(a,b)}^2 \end{aligned}$$

und

$$\begin{aligned} ||w'||_{L^2(a,b)}^2 &= \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} w'(t)^2 \mathrm{dt} \\ &\leq C^2 h^2 ||y''||_{L^2(a,b)}^2. \end{aligned}$$

Damit haben wir alles zusammen.

Satz 6.27 (Approximationssatz für Finite–Elemente–Näherungen mit Splines der Ordnung 2)

Sei y_h die Finite–Elemente–Näherung für ein Sturm–Liouville–Problem für lineare Splines der Ordnung 2 auf einem zulässigen Gitter I_h mit Feinheit h, und $y \in H^2$. Dann gibt es eine von h unabhängige Konstante c mit

$$||y - y_h||_{H^1} \leq ch||f||_{L^2}.$$

Beweis: Für $\varphi = y'$ gilt

$$\begin{aligned} p_0 ||y'||_{L^2}^2 &\leq B(y, y) \\ &= F(y) \\ &\leq ||f||_{L^2}^2 \cdot ||y||_{L^2}^2 \\ &\leq ||f||_{L^2}^2 C^2 ||y'||_{L^2}^2 \end{aligned}$$

und damit

$$||y'||_{L^2} \leq C^2 p_0 ||f||_{L^2}.$$

Weiter gilt

$$\int_a^b f(t)\varphi(t)\mathrm{dt} = \int_a^b (-p(t)y'(t))'\varphi(t)\mathrm{dt} = -\int_a^b (p'(t)y'(t) + p(t)y''(t))\varphi(t)\mathrm{dt}.$$

Für $\varphi = y''$ gilt somit

$$\begin{aligned} p_0||y''||_{L^2}^2 &\leq \int_a^b p(t)y''(t)^2 dt \\ &= -\int_a^b p'(t)y'(t)y''(t)\mathrm{dt} + \int_a^b q(t)y(t)y''(t)\mathrm{dt} - \int_a^b f(t)y''(t)\mathrm{dt} \\ &\leq ||p'||_\infty ||y'||_{L^2}||y''||_{L^2} + ||q||_\infty ||y||_{L^2}||y''||_{L^2} + ||f||_{L^2} \cdot ||y''||_{L^2} \end{aligned}$$

und damit

$$||y''|| \leq C'||f||_{L^2}.$$

Hierbei haben wir einige Randterme vernachlässigt $(y''(a) \neq 0)$, die sich aber genauso abschätzen lassen.

Sei wieder p die Interpolation von y an den Punkten t_k mit linearen Splines, und $w = y - p$. Dann gilt

$$\begin{aligned} ||y - y_h||_{H^1}^2 &\leq C'' \inf_{v \in X_h} ||y - v||_{H^1}^2 \\ &\leq C''||w||_{H^1}^2 \\ &= C''(||w||_{L^2}^2 + ||w'||_{L^2}^2) \\ &\leq C'''h^2||y''||_{L^2}^2 \\ &\leq C''''h^2||f||_{L^2}^2. \end{aligned}$$

Poincaré liefert wieder die Konvergenz in der Supremumnorm. □

Abschließend schauen wir nun noch kurz, was dies für die Poissongleichung und äquidistante Gitter bedeutet, die wir schon mit diskreten Methoden behandelt hatten. Wir wählen ein äquidistantes Gitter der Schrittweite h. Es gilt $p = 1$ und $q = 0$. Die Steifigkeitsmatrix ist eine Tridiagonalmatrix. Die Ableitungen der Basisfunktionen sind $\frac{1}{h}$ bzw. $-\frac{1}{h}$. Auf der Hauptdiagonalen stehen die Einträge

$$\int_{t_i}^{t_{i+1}} \frac{1}{h^2}\mathrm{dt} + \int_{t_{i+1}}^{t_{i+2}} \frac{1}{h^2}\mathrm{dt} = \frac{2}{h}.$$

Auf den Nebendiagonalen stehen die Einträge

$$\frac{1}{h^2}\int_0^h 1 \cdot (-1)\mathrm{dt} = -\frac{1}{h}.$$

Die Steifigkeitsmatrix ist bis auf einen Faktor h genau dieselbe wie bei der Diskretisierung. Dies ist sehr häufig der Fall: Für äquidistante Gitter erhält man die Diskretisierungsmethode zurück.

Literaturverzeichnis

Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables (is an Outgrowth of a Conference on Mathematical Tables Held at Cambridge, Mass., on 1954)*. Applied mathematics series. Dover Publ., 1965. ISBN 9780486612720. URL http://people.math.sfu.ca/~cbm/aands/abramowitz_and_stegun.pdf.

H.W. Alt. *Lineare Funktionalanalysis:*. Springer London, Limited, 2007. ISBN 9783540341871. URL http://books.google.de/books?id=TzeMiSx8_l4C.

Thomas Beth. *Verfahren der schnellen Fourier-Transformation: die allgemeine diskrete Fourier-Transformation–ihre algebraische Beschreibung, Komplexität und Implementierung*. Teubner-Studienbücher. Philologie. Teubner, 1984. ISBN 9783519023630. URL http://books.google.de/books?id=hk7vAAAAMAAJ.

Fischer Black and Myron S Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–54, May-June 1973. URL http://www.jstor.org/stable/1831029.

C. Sidney Burrus. *Fast Fourier Transforms*. Rice University. URL http://cnx.org/content/col10550/latest/.

Ingrid Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

Carl De Boor. Total positivity of the spline collocation matrix. *Indiana Univ. Math. J*, 25(6):541–551, 1976. URL http://www.math.tamu.edu/~rdevore/publications/47.pdf.

Carl de Boor. *A Practical Guide to Splines*. Number Bd. 27 in Applied Mathematical Sciences. Springer, 2001. ISBN 9780387953663. URL http://books.google.de/books?id=m0QDJvBI_ecC.

Ronald A. DeVore and Amos Ron. Developing a Computation-Friendly Mathematical Foundation for Spline Functions. *SIAM News*, 38(4):1–2, May 2005. URL http://www.siam.org/pdf/news/56.pdf.

L.C. Evans. *Partial Differential Equations*. Graduate Studies in Mathematics. American Mathematical Society, 2010. ISBN 9780821849743. URL http://books.google.de/books?id=Xnu0o_EJrCQC.

R.W. Freund and R.H.W. Hoppe. *Stoer/Bulirsch: Numerische Mathematik 1*. Springer-Lehrbuch. Springer London, Limited, 2007. ISBN 9783540453901. URL http://link.springer.com/book/10.1007/978-3-540-45390-1/page/1.

E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Solving Ordinary Differential Equations. Springer, 1993. ISBN 9783540566700. URL http://books.google.de/books?id=F93u7VcSRyYC.

M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Mathematische Leitfäden. Teubner, 2006. ISBN 9783835100909. URL http://books.google.de/books?id=tKrhTUmYNEoC.

P. Henrici. *Discrete variable methods in ordinary differential equations*. Wiley, 1962. URL http://books.google.de/books?id=j-5QAAAAMAAJ.

P. Henrici. *Elements of numerical analysis*. Wiley international edition. Wiley, 1964. URL http://ia700700.us.archive.org/23/items/ElementsOfNumericalAnalysis/Henrici-ElementsOfNumericalAnalysis.pdf.

Alfred K. Louis, Peter Maaß, and Andreas Rieder. *Wavelets:*. Teubner Studienbücher: Mathematik. Vieweg+Teubner Verlag, 1998. ISBN 9783519120940. URL http://books.google.de/books?id=QN2fg4sM5oIC.

Alfred Marshall Mayer. *Researches in Acoustics*, volume 151. 1896. URL http://www.ajsonline.org/content/s4-1/2/81.full.pdf+html?sid=9ff838fa-4010-4f5c-ad78-88212b258bf0.

J.D. Murray. *Mathematical Biology: I. An Introduction*. Interdisciplinary Applied Mathematics. Springer, 2002. ISBN 9780387952239. URL http://books.google.de/books?id=4WbpP90Gk1YC.

Frank Natterer. *Vorlesungsskript zur Vorlesung Effiziente Algorithmen im WS 94/95*. Institut für Numerische und Angewandte Mathematik der Universität Münster, 1994. URL http://wwwmath.uni-muenster.de/num/Vorlesungen/EffAlg_WS94/.

J.W. Prüß, R. Schnaubelt, and R. Zacher. *Mathematische Modelle in der Biologie: Deterministische homogene Systeme*. Mathematik Kompakt. Birkhäuser Basel, 2008. ISBN 9783764384364. URL http://books.google.de/books?id=IRH_k7vukdsC.

Carl Runge and Hermann König. *Vorlesungen über numerisches Rechnen*. Springer Göttingen, 1925. URL http://resolver.sub.uni-goettingen.de/purl?PPN373207646.

Isaac Jacob Schoenberg and Anne Whitney. On polya frequency function. iii. the positivity of translation determinants with an application to the interpolation problem by spline curves. *Transactions of the American Mathematical Society*, 74 (2):pp. 246–259, 1953. ISSN 00029947. URL http://www.jstor.org/stable/1990881.

J. Stoer and R. Bulirsch. *Numerische Mathematik 2: Eine Einführung - unter Berücksichtigung von Vorlesungen von F.L.Bauer*. Numerische Mathematik: eine Einführung - unter Berücksichtigung von Vorlesungen von F. L. Bauer. Springer, 2005. ISBN 9783540237778. URL http://books.google.de/books?id=_TPRZ9pabGcC.

Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Software, Environments and Tools Series. Cambridge University Press, 2000. ISBN 9780898714654. URL http://books.google.de/books?id=pB4xiZKZ4ecC.

W. Walter. *Gewöhnliche Differentialgleichungen: Eine Einführung*. Springer-Lehrbuch Series. Springer Singapore Pte. Limited, 2000. ISBN 9783540676423. URL http://books.google.de/books?id=tyAdMH69NRYC.

Shmuel Winograd. On computing the discrete fourier transform. *Mathematics of Computation*, 32(141):pp. 175–199, 1978. ISSN 00255718. URL http://www.jstor.org/stable/2006266.

Abbildungsverzeichnis

Listings