Numerische Analysis im SS 2011

Frank Wübbeling

29. Juni 2011

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der vorliegende Text entstand als Begleitmaterial zur Vorlesung Numerische Analysis im Sommersemester 2011. Die Vorlesung richtet sich an Studierende des Bachelorstudiengangs Mathematik im vierten Semester sowie Studierende in den Lehramtsstudiengängen Mathematik. Für die Korrektheit des Textes wird keinerlei Garantie übernommen, vermutlich sind noch reichlich Schreibfehler enthalten. Für Bemerkungen und Korrekturen bin ich dankbar.
Macht es Sinn, der großen, bereits existierenden Zahl von Skripten zu Einführungsveranstaltungen der Numerischen Mathematik noch ein weiteres hinzuzufügen? Die Antwort ist wohl ja, denn zumindest die Auswahl der Themen und vor allem Schwerpunkte im breiten Spektrum geschieht subjektiv durch den Dozenten.
Da der Großteil der Studierenden heute kaum noch einen physikalischen Hintergrund hat, habe ich auf die Darstellung der Beziehungen zwischen Angewandter Mathematik und Physik, wie sie in den klassischen Lehrbüchern und Vorlesungen üblich war, größtenteils verzichtet. Übungen zu den einzelnen Kapiteln finden sich im Netz, ebenso eine ausführliche (subjektive) Literaturliste.
Ich habe mich bemüht, zu allen vorgestellten Algorithmen eine Beispiel–Implementation in Matlab zu liefern. Einige Programme nutzen dabei die Imaging–Toolbox oder die SymbolicMath–Toolbox. Die zugehörigen Dateien sind in der PDF–Datei enthalten. Klick auf die jeweilige Textstelle öffnet die Beispielimplementation in Matlab. Ebenso sind alle Bilder beigefügt, Klick liefert jeweils das zugehörige Bild. Dies funktioniert in Acrobat (Reader) und in einigen anderen PDF–Readern, in vielen PublicDomain–Readern aber nicht.

Billerbeck, im Frühjahr 2011

Frank Wübbeling

Kapitel 2

Interpolation

Mit 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 $s : [0, N] \mapsto \mathbb{R}^2$: $s(i) = x_i$, $i = 0 \ldots n$.

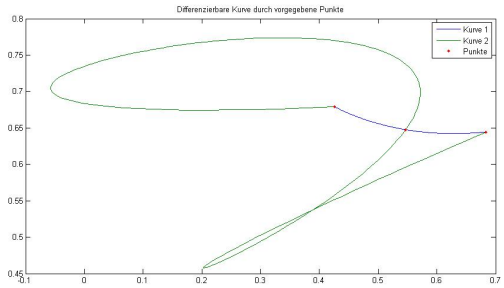


Abbildung 2.1: Interpolationsfunktionen

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 zur 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

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

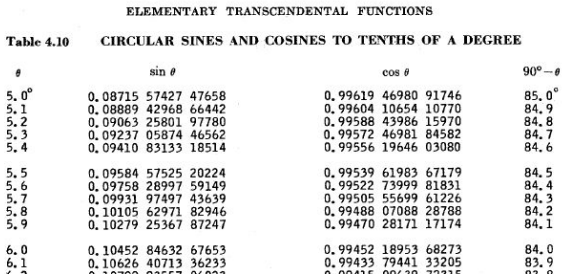
für alle Funktionen $g : [a,b] \mapsto \mathbb{R}$ mit $g(x_i) = y_i$, $i = 0 \dots N$, $g \in C^2([a,b])$.

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

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.

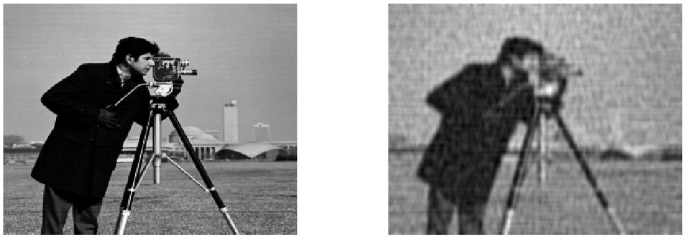


Abbildung 2.4: Original, zu stark komprimiertes Bild

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');
```

Listing 2.1: Bildkompression (comprimg.m)

Klicken für den Quellcode von 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 lautet 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) *Sei $N \in \mathbb{N}$.*

1. *Mit $\mathcal{P}_N$ bezeichnen wir den Raum der Polynome vom Grad kleiner oder gleich N.*
2. *Seien $x_0, \ldots, x_N$ paarweise verschieden, $y_0, \ldots, y_N$ gegeben. Dann lautet 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

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

   Dann ist $w_j(x) \in \mathcal{P}_N$, und $w_j(x_k) = \delta_{kj}$ für $k, j = 0..N$. Hier ist

$$\delta_{kj} = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$$

   das Kronecker-Delta. Sei

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

   Dann ist $p \in P_N$, und es gilt $p(x_i) = y_i$ für alle $i = 0 \ldots N$.

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 $i = 0 \ldots N$. 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) *Die Matrix $V \in \mathbb{C}^{n \times n}$ heißt Vandermondematrix zu $x_0, \ldots, x_N$, falls $V_{ik} = (x_i)^k$ mit paarweise verschiedenen Zahlen x_i, $i = 0..N$, $k = 0..N$.*

Also:

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

Satz 2.5 (Invertierbarkeit der Vandermondematrizen)

1. *Sei $p(x) = \sum_{k=0}^{N} a_k x^k$. Sei $y = (y_0, \ldots, y_N)^t$, $a = (a_0, \ldots, a_N)^t$, $V = V(x_0, \ldots, x_N)$ Vandermonde–Matrix. p ist genau dann Lösung des Polynominterpolationsproblems, falls $Va = y$.*
2. *Vandermondematrizen sind invertierbar.*

Beweis:

1. Klar wegen $(Va)_j = p(x_j)$.
2. Jede Aufgabe 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 $p_{i\ldots k} \in \mathcal{P}_{k-i}$ die Lösung der Aufgabe für die Stützstellen $x_i, \ldots, x_k$ und Stützwerte $y_i, \ldots, y_k$. Dann ist $p = p_{0\ldots N}$ die Lösung der vollen Aufgabe, und es gilt*

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

Beweis: Sei q das Polynom auf der rechten Seite. Dann ist $q \in \mathcal{P}_{k+1-i}$, und offensichtlich ist $q(x_j) = y_j$ 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):

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

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

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

Klicken für den Quellcode von 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)
```

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

Klicken für den Quellcode von nevilleeval.m

Eine zweite Möglichkeit zur rekursiven Berechnung des Interpolationspolynoms gibt die Form von Newton. Mit den Bezeichnungen aus der Formel von Neville gilt die

Satz 2.7 (Formel von Newton)

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

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
Der Koeffizient $[y_i, \dots, y_k] = \frac{(y_k - p_{i\dots k-1}(x))}{(x_k - x_i)\cdots(x_k - x_{k-1})}$ *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*

$$p(x) = \sum_{j=0}^{N} [y_0, \ldots, y_j](x - x_0) \cdots (x - x_{j-1}).$$

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:

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

```
function out = divdiff( x,y)
%divdiff
%compute divided differences of x and y
if (nargin<1)
```

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

Klicken für den Quellcode von 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

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

Neville, ausgewertet für $x = 1$*:*

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

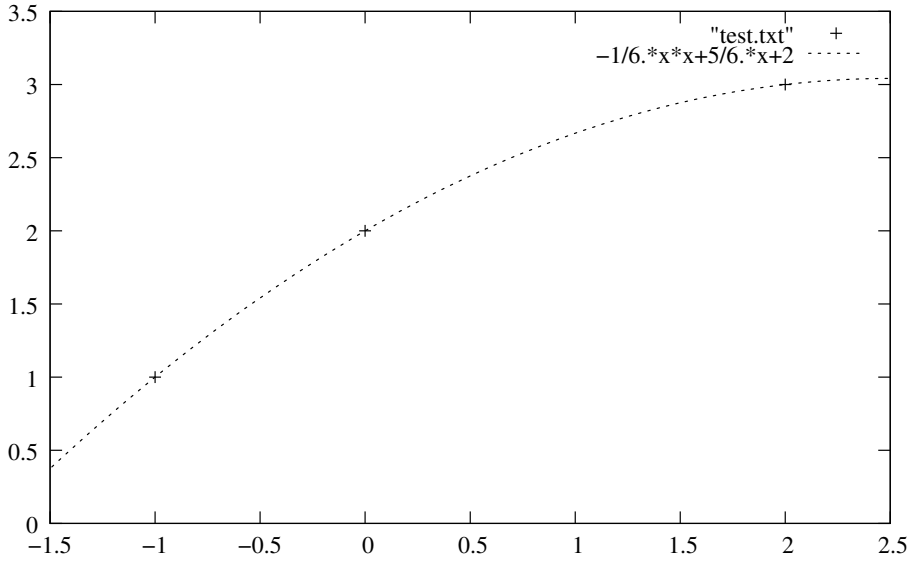
Newton

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

und damit

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

Natürlich erzeugen alle Rechenvorschriften dasselbe Interpolationspolynom. Graphisch:



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.
```

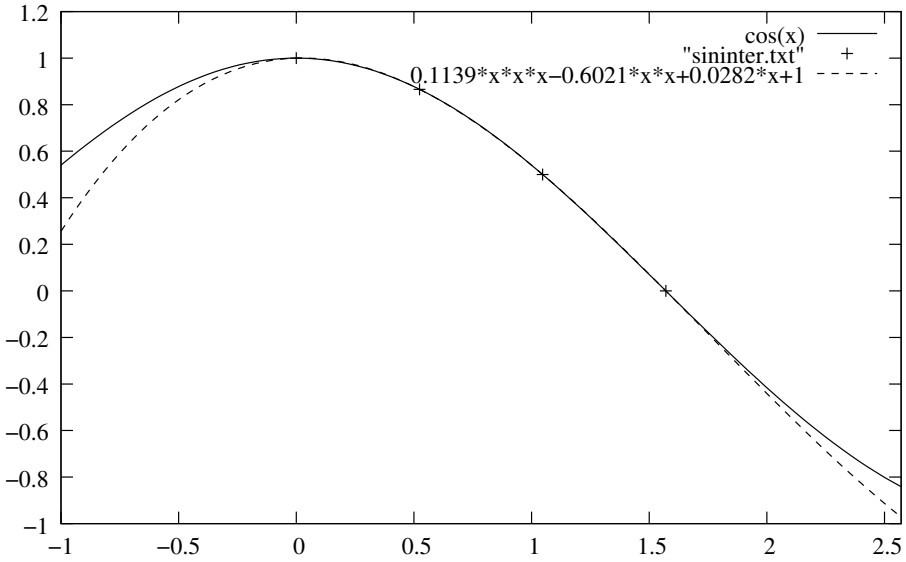
Listing 2.5: Polynominterpolation (interpolate.m)

Klicken für den Quellcode von interpolate.m

2.2 Interpolationsfehler bei Polynomen

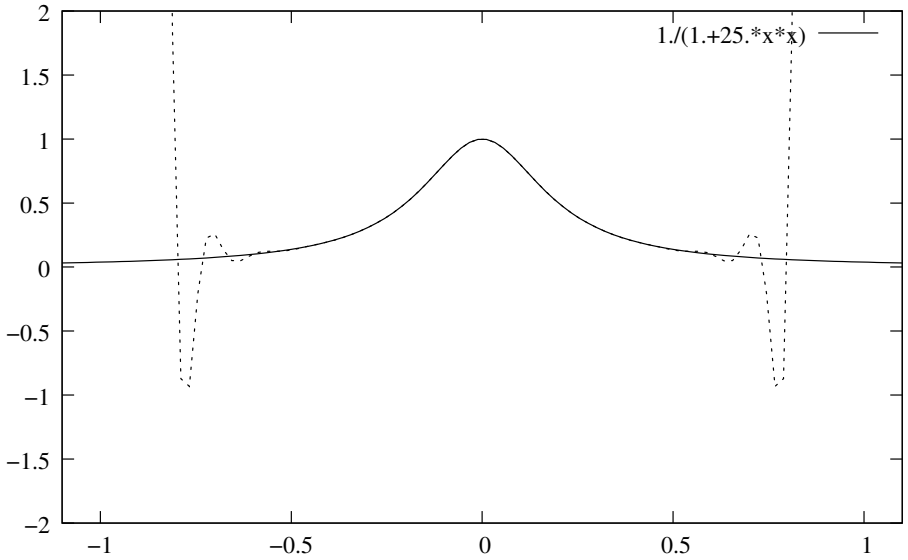
Wir können die Polynominterpolation nutzen, um Funktionen zu approximieren. Sei in diesem Abschnitt f eine Funktion auf dem Intervall $[a, b] \subset \mathbb{R}$ nach $\mathbb{R}$. Wir werten f an den paarweise verschiedenen Stellen x_i, $i = 0 \dots N$, aus und betrachten das Interpolationspolynom p_N 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 Interpolationsfehler klein? Konvergiert p_N für wachsendes N und im Intervall $[a, b]$ gleichverteilte Stützstellen $x_0, \dots, x_N$ punktweise gegen f? 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 $x_k = a + k * (b - a)/N$, $k = 0 \dots N$, 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) *Leider sind die Verhältnisse nicht immer so gut. Von Runge stammt das Beispiel der Funktion $f(x) = 1/(1 + 25x^2)$ 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.

```
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)
```

Listing 2.6: Polynomapproximation (polapprox.m)

Klicken für den Quellcode von polapprox.m

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

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

Klicken für den Quellcode von rungebeispiel.m

Der folgende Satz schätzt den maximal zu erwartenden Interpolationsfehler ab.

Satz 2.13 *Sei $f \in \mathbb{C}^{N+1}([a,b])$, $f : [a,b] \mapsto \mathbb{R}$. Seien x_i paarweise verschieden in $[a,b]$, $i = 1 \dots N$, 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 $f(\overline{x}) = p(\overline{x}) = w(\overline{x}) \Rightarrow$ Behauptung.
2. Sei $\overline{x} \neq x_i$ für alle $i = 0 \dots N$, also $w(\overline{x}) \neq 0$. Sei

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

   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 $[a,b]$.

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

   und damit

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

□

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 2.14 *1. Äquidistante Stützstellenwahl im Beispiel von Runge lieferte keine Konvergenz an den Rändern des Intervalls. Dies ist mit Hilfe des Satzes leicht verständlich: Die Maximumnorm von $f^{(N)}$ konvergiert für das Rungebeispiel schnell gegen ∞.*

Ohne Einschränkung sei $[a,b] = [0,1]$ und $h = 1/N$ der Abstand zwischen zwei Punkten. Dann ist

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

und diese Funktion hat ihr Maximum bzw. Minimum zwischen x_0 und x_1 bzw. zwischen x_{N-1} und x_N.

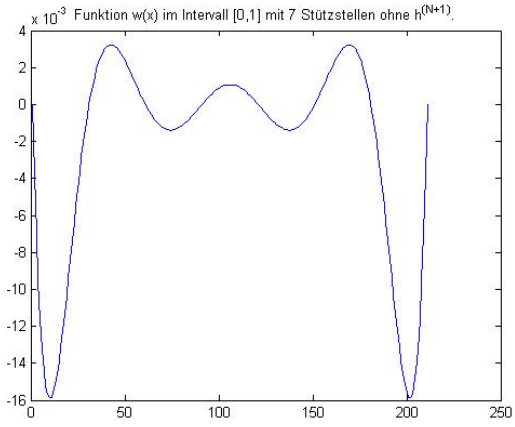


Abbildung 2.5: $w(x)$ ohne den Faktor h^{N+1} für $N = 7$

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.

Unabhängig davon bedeutet ein hoher Polynomgrad 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;
```

Listing 2.8: Approximationsfehler (approtest.m)

Klicken für den Quellcode von 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.

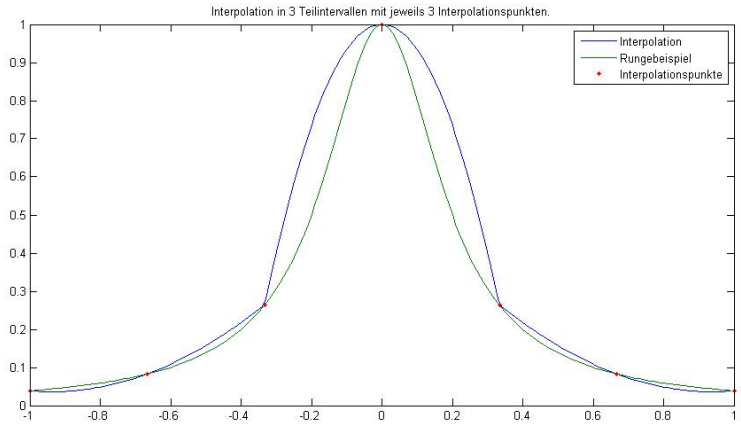


Abbildung 2.6: Interpolation in Teilintervallen

Klick für Bild partinter

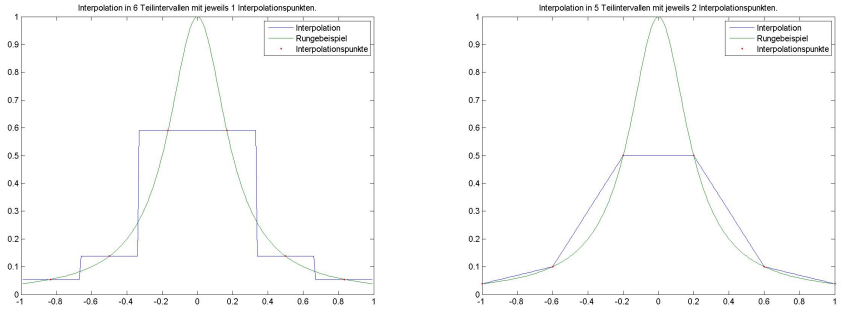


Abbildung 2.7: Auswertung und Polygonzug–Approximation

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;
```

Listing 2.9: Interpolation in Teilintervallen (partinter.m)

Klicken für den Quellcode von partinter.m

Korollar 2.15 *Im allgemeinen konvergiert die Polynominterpolation bei steigendem Polynomgrad und äquidistanter Stützstellenwahl nicht gegen die zu interpolierende Funktion.*

2.3 Optimale Wahl der Stützstellen, Tschebyscheff–Interpolation

Im Licht von Satz 2.13 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 2.16 $T_n : [-1, 1] \mapsto \mathbb{R}$, $T_n(x) = \cos(n \arccos x)$, $n \in \mathbb{N}$, *heißt Tschebyscheff–Polynom der Ordnung* n.

Satz 2.17 *Für die Tschebyscheff–Polynome* T_n *gilt:*

1. $T_n \in \mathcal{P}_n$.
2. *Die* T_n *bilden ein Orthogonalsystem bezüglich des Skalarprodukts*
$$(p, q) = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} p(x)q(x)dx.$$
3. *Die Nullstellen* x_k^n *von* T_{n+1} *sind* $\cos(\pi(2k + 1)/(2(n + 1))$, $k = 0 \dots n$.
4. $T_n(x)/2^{n-1}$ *hat den Höchstkoeffizient 1. 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, $k = 0 \dots n$, *so ist*
$$w(x) = \prod_{k=0}^{n} (x - x_k^n) = \frac{1}{2^{n-1}} T_n(x).$$

Beweis: Siehe Übungen. Beweisidee zu 4.: Sei $q(x) = T_n(x)/(2^{n-1})$. Angenommen, p sei ein Polynom in $\mathcal{P}_n$ mit Höchstkoeffizient 1 und $||p||_\infty < 1/2^{n-1} = ||q||_\infty$, $r = p - q \in \mathcal{P}_{n-1}$. q nimmt sein Betragsmaximum mit wechselnden Vorzeichen an den Stellen $z_k = cos(k\pi/n)$ an, $k = 0 \ldots n$. Wegen $|p(z_k)| \leq ||p||_\infty < ||q||_\infty = |q(z_k)|$ gilt $\mathrm{sgn}(r(x_k)) = \mathrm{sgn}(q(z_k))$. r wechselt also $n+1$-Mal sein Vorzeichen, hat also n Nullstellen, also gilt $r = 0$ und damit $p = q$ im Widerspruch zur Annahme. □

Für die Polynominterpolation an den Nullstellen der Tschebyscheff–Polynome erhalten wir daher die Abschätzung

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

Bemerkung: Durch die Abbildung $x \mapsto -1 + 2(x-a)/(b-a)$ wird das Intervall $[a, b]$ auf $[-1, 1]$ abgebildet, für ein allgemeines Intervall $[a, b]$ lauten die Tschebyscheff–Stützstellen also

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

Bemerkung: Für die Tschebyscheff–Interpolation konvergiert für $N \mapsto \infty$ das Interpolationspolynom für das Rungebeispiel gegen die zu interpolierende Funktion (siehe unten). Es lassen sich aber unendlich oft differenzierbare Funktionen f konstruieren, so dass auch die Tschebyscheff–Interpolation nicht punktweise konvergiert.

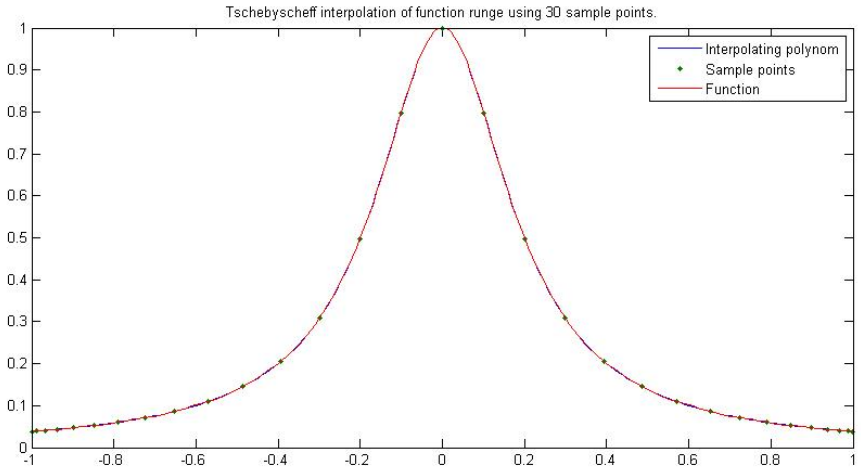


Abbildung 2.8: Tschebyscheff–Interpolation

Klick für Bild tschebyscheff

```
function tscheb( N )
%TSCHEB
a=-1;
b=1;
```

Listing 2.10: Tschebyscheff–Interpolation (tscheb.m)

Klicken für den Quellcode von tscheb.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 2.18 *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. □

Bemerkung: Für die Hermite–Interpolation gilt die Fehlerabschätzung 2.13 mit $w(x) = \prod_k (x - x_k)^{N_k}$, der Beweis verläuft wörtlich wie der Beweis zu 2.13.

Beispiel 2.19 *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.*

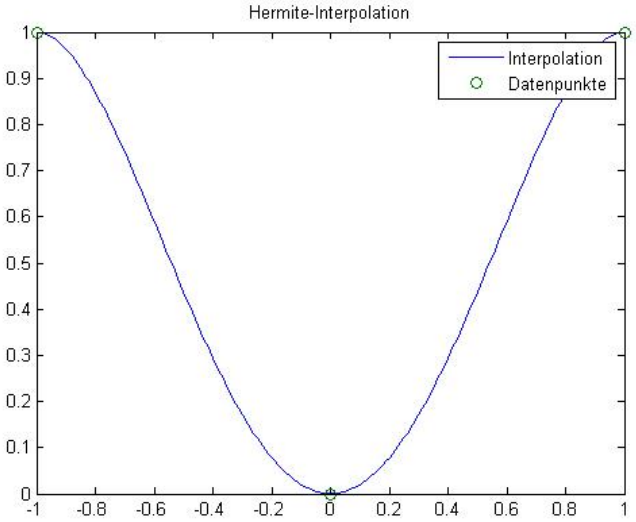


Abbildung 2.9: Beispiel zur Hermite–Interpolation

Klick für Bild hermite

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

Listing 2.11: Programm zur Hermite–Interpolation (hermitebeispiel.m)

Klicken für den Quellcode von hermitebeispiel.m

2.5 Richardson-Extrapolation

Es sei der Grenzwert a_0 einer Funktion f für $h \mapsto 0$ zu bestimmen. Zur Verfügung stehen Auswertungen der Funktion $f(h_k)$, $h_k = 1/(2^k), k = 0 \ldots \infty$. Es sei bekannt, dass f eine Taylorreihe bis zum Grad n in h^p besitzt, also

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

mit (unbekannten) Koeffizienten a_j und einer (unbekannten) beschränkten Funktion $C(h)$. Dies ist zum Beispiel 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?
Die einfache Idee ist, das Interpolationspolynom in $f(h^{1/p})$ vom Grad n durch $(n+1)$ Stützstellen zu legen und es mit der Form von Neville für $h = 0$ auszuwerten.

Beispiel 2.20 *Gesucht sei der Grenzwert der Funktion* $\mathrm{sinc}(x) = sinx/x$ *für* $x \mapsto 0$. *Der* sinc *ist gerade, hat also eine Taylorreihenentwicklung in* x^2. *Dann hat* $g(x) := \mathrm{sinc}(\sqrt{|x|})$ *eine Taylorentwicklung in* x. *Zur Verfügung stehen Auswertungen von* $\mathrm{sinc}(2^{-k})$, $k = 0 \ldots N$. *Wir werten aus für* $x = 0$ *mit dem Schema von Neville. Angegeben ist jeweils der Fehler, also die Differenz des berechneten Werts zum korrekten Ergebnis 1:*

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

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

Listing 2.12: Richardson–Extrapolation der Sinc–Funktion (richardson1.m)

Klicken für den Quellcode von 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 die $O()$–Notation ein.

Definition 2.21 *Seien* $f, g : X \mapsto \mathbb{R}$, $X = \mathbb{R}$ *oder* $\mathbb{C}$. f *heißt von der Ordnung* g, *falls:*

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 2.22

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–Tableau, gilt: Die erste Spalte hat eine Genauigkeit von $O(h)$, die zweite von $O(h^2)$ usw. Einfaches Beispiel:
Seien $f(h)$ und $f(h/2)$ bekannt, und f habe eine Taylorentwicklung bis zur Ordnung 1 in h, also $f(h) = a_0 + a_1 h + O(h^2)$. Tableau:

$$\begin{array}{cc} h & a_0 + a_1 h + O(h^2) \\ h/2 & a_0 + a_1 h/2 + O(h^2) \end{array} \quad \frac{1}{h/2-h}((h/2-0)f(h) + (0-h)(f(h/2)) = a_0 + O(h^2)$$

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. Genauer gilt:

Satz 2.23 *$f(h)$ besitze eine Taylorreihe bis zum Grad N in h^p. Sei $g(h) = f(h^{(1/p)})$. Sei N das Neville–Tableau für die Stützstellen $x_k = a^k h$, $|a| < 1$, und Stützwerte $y_k = g(x_k)$. Dann gilt für die Einträge $p_{i\ldots i+k-1}(0)$ in der k. Spalte des Tableaus*

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

mit Zahlen $a_{k,j}$.

Beweis: 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 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 exakt gleich für alle Zeilen durchgeführt werden, das gibt die Aussage des Satzes für $p = 1$.
Sei nun p beliebig. Dann besitzt nach Voraussetzung $g(h) = f(h/1/p)$ eine Taylorentwicklung in h bis zur Ordnung N, also gilt für $f(h) = g(h^p)$ die Aussage für h^p an Stelle von h, und der Satz ist bewiesen. □

Bemerkung: Die Rechnung lässt sich genau so auch für beliebige Auswertepunkte h_k durchführen.
Bemerkung: Ein Spezialfall der Richardson–Interpolation ist die Romberg–Integration **??**.

2.6 Interpolation mit allgemeinen Ansatzfunktionen

Bei der Polynominterpolation versucht man, eine Funktion f durch eine Linearkombination der Monome x^k, $k = 0..N$, 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 2.24 *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 x, so sollte zumindest eine der Ansatzfunktionen ebenfalls unstetig sein in x.
Das zugehörige Gleichungssystem für die a_j ergibt sich mit der Vandermonde–ähnlichen Matrix

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

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 Spezialfall die Interpolation mit trigonometrischen Funktionen.

2.7 Trigonometrische Interpolation: 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

$$x_k = e^{2\pi ik/n} = \omega_n^k,\ \omega_n = e^{2\pi i/n}.$$

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

Satz 2.25 *Seien x_k die $n.$ Einheitswurzeln, $k = 0..n-1$, $W = V(x_0, \ldots, x_{n-1})$ die zugehörige Vandermonde–Matrix. Dann gilt $\overline{W} = V(\overline{x}_0, \ldots, \overline{x}_{n-1})$ und*

$$W\overline{W} = nI \text{ und damit } W^{-1} = \frac{1}{n}\overline{W}.$$

Beweis: Durch Ausrechnen:

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

□

Bemerkung:

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

Bemerkung: W ist symmetrisch, aber nicht selbstadjungiert.
Mit Hilfe dieser Formel lassen sich die Koeffizienten des Interpolationspolynoms der trigonometrischen Interpolation sofort angeben.

Korollar 2.26 *Seien x_k die n. Einheitswurzeln, $y_k \in \mathbb{C}$, $k = 0 \dots n-1$. Dann ist das Interpolationspolynom $p \in P_{n-1}$ gegeben durch*

$$p(x) = \frac{1}{n} \sum_{k=0}^{n-1} \widehat{y}_k x^k,\ \widehat{y}_k = \sum_{j=0}^{n-1} y_j \overline{\omega}_n^{kj} = \sum_{j=0}^{n-1} y_j e^{-2\pi i k j/n},\ k = 0 \dots n-1. \qquad (2.1)$$

Umgekehrt gilt

$$y_j = p(x_j) = \frac{1}{n} \sum_{k=0}^{n-1} \widehat{y}_k x_j^k = \frac{1}{n} \sum_{k=0}^{n-1} \widehat{y}_k \omega_n^{kj} = \frac{1}{n} \sum_{k=0}^{n-1} \widehat{y}_k e^{2\pi i k j/n},\ j = 0 \dots n-1. \quad (2.2)$$

Definition 2.27 *2.1 heißt diskrete Fouriertransformation von $(y_0, \dots, y_{n-1})$. 2.2 heißt inverse diskrete Fouriertransformation von $(\widehat{y}_0, \dots, \widehat{y}_{n-1})$. Üblicherweise wird die Fouriertransformation eines Vektors mit $\widehat{y}$ bezeichnet, die inverse Fouriertransformation mit $\widetilde{y}$.*

Bemerkung: In der Literatur finden sich einige 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 letzteren Fall wird V unitär. Wir werden im folgenden bei der Berechnung der Fouriertransformierte den Faktor häufig weglassen.
Die diskrete Fouriertransformation wird genutzt zur Interpolation von periodischen Funktionen auf dem gesamten Intervall. Sei $f : \mathbb{R} \mapsto \mathbb{R}$, $f \in C^n(\mathbb{R})$, periodisch mit der Periode 2π. Seien $x_k = \pi/n$, $k = 0 \dots n-1$, gleichverteilte Stützstellen im Intervall $[0, 2\pi]$ (Achtung: Wegen $f(x_0) = f(0) = f(2\pi) = f(x_n)$ bringt die n. Stützstelle keine zusätzliche Information, und wir lassen sie weg). Als Ansatzfunktionen für die allgemeine Interpolationsaufgabe wählen wir die periodischen Funktionen

$f_k(x) = e^{ikx}$. Dann liefert die Fouriertransformation die zugehörigen Entwicklungskoeffizienten der Interpolationsfunktion.
Sei nun y reell und N ungerade, dann gilt

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

(denn $\cos(x_k) = \cos(x_{N-k})$ und $\sin(x_k) = -\sin(x_{N-k})$). Ist y gerade, also $y_k = y_{N-k}$, ist $\widehat{y}_k$ reell, ist y ungerade, also $y_k = -y_{N-k}$, so ist $\widehat{y}_k$ rein imaginär.
In diesen Fällen heißt die Fouriertransformierte die Cosinustransformation bzw. Sinustransformation von $y_0, \ldots, y_{(N-1)/2}$ (Inversionsformeln in den Übungen).

2.7.1 n-dimensionale 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 2.1 als Skalarprodukt interpretiert, d.h.

$$\widehat{y}_k = \sum_j y_k \overline{\omega}_n^{kj} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} y_{j_1,j_2} e^{-2\pi i(k_1 j_1/n_1 + k_2 j_2/n_2)},\ k_1 = 0 \ldots n_1 - 1,\ k_2 = 0 \ldots n_2 - 1$$

für $k = (k_1, k_2)$ und $j = (j_1, j_2)$. Das entspricht einer Fouriertransformation auf den Zeilen von y, gefolgt von einer Fouriertransformation auf den Spalten.
Die Formeln für die inverse Fouriertransformation, $\cos$– und $\sin$–Transformation gelten entsprechend.

2.7.2 Faltungssatz

Einer der wichtigsten Sätze über die Fouriertransformation ist der Faltungssatz.

Definition 2.28 *Seien $y \in \mathbb{C}^N$, $z \in \mathbb{C}^M$. Dann ist die diskrete Faltung $(y * z) \in \mathbb{C}^{N+M-1}$ von y und z definiert durch*

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

wobei $z_{k-j} = 0$ für $k - j < 0$ oder $k - j > M - 1$.
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

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

wobei für z der Index modulo N genommen wird, also $z_0 = z_N$ usw.

Wieder ist die höherdimensionale Faltung entsprechend definiert, indem man die Indizes als Tupel wählt. Die normale Faltung kann mit Hilfe der symmetrischen Faltung berechnet werden.
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;
```

Listing 2.13: Eindimensionale Faltung (faltung1D.m)

Klicken für den Quellcode von faltung1D.m

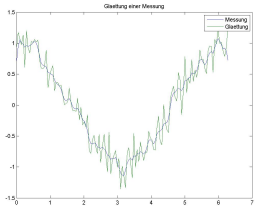


Abbildung 2.10: 1D–Faltung mit glattem Vektor, Glättung

Klick für Bild glatt1D

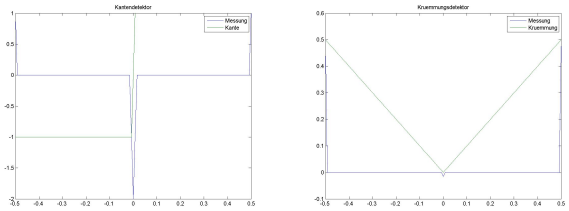


Abbildung 2.11: 1D–Faltung mit einem Kanten– bzw. Krümmungsdetektor

Klick für Bild kant1D
Klick für Bild kruem1d

```
function faltung2D
%FALTUNG2D
Y=imread('cameraman.tif');
Y=double(Y);
```

Listing 2.14: Zweidimensionale Faltung (faltung2D.m)

Klicken für den Quellcode von faltung2D.m

Alle diese Filter sind aus Bildbearbeitugsprogrammen 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 gibt der Faltungssatz:

Satz 2.29 *Seien $y,\ z \in \mathbb{C}^N$. Dann gilt für die symmetrische diskrete Faltung von y und z*

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

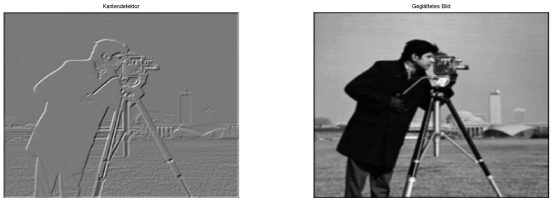


Abbildung 2.12: 2D Kantendetektor, Glättung

Klick für Bild kante2d
Klick für Bild glatt2d

Beweis:

$$\begin{aligned}
\widehat{y}_p\widehat{z}_p &= \left(\sum_{k=0}^{N-1} y_k\omega_N^{pk}\right)\left(\sum_{j=0}^{N-1} z_j\omega_N^{pj}\right) \\
&= \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} y_k z_j \omega^{p(k+j)} \quad l=k+j,\ j=l-k \\
&= \sum_{k=0}^{N-1}\sum_{l=k}^{N-1+k} y_k z_{l-k}\omega_N^{pl} \\
&= \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} y_k z_{l-k}\omega^{pl} \text{ z periodisch im Index, } \omega_N^N = 1 \\
&= \sum_{l=0}^{N-1}\omega_N^{pl}\sum_{k=0}^{N-1} y_k z_{l-k} \\
&= \widehat{(y*z)}_p.
\end{aligned}$$

□

Also: Falls nur $(y*z)$ und z bekannt sind, so gilt $\widehat{y} = (\widehat{(y*z)}/\widehat{z})$ und hieraus lässt sich y durch inverse Fouriertransformation berechnen. Dies ist gültig, solange die Fourierkoeffizienten von z nicht verschwinden. Theoretisch lassen sich mit dieser Formel unscharfe Bilder scharfrechnen, praktisch funktioniert das nur sehr eingeschränkt, da die Fourierkoeffizienten tatsächlich sehr schnell gegen o gehen (unscharf maskieren, s. Übungen).

2.7.3 Bedeutung der trigonometrischen Interpolation

Aus der Analysis 1 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

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

(analytische Fouriertransformation). Offensichtlich sind die trigonometrischen Transformationen Approximationen an diese Integrale, siehe Kapitel II.
Bei natürlichen Tönen (etwa von Musikinstrumenten) verschwindet typischerweise der Großteil der Koeffizienten, nur ein kleiner Teil der c_k ist wesentlich ungleich o. Zur Beschreibung des Tons reicht es daher häufig, relativ wenige der Fourierkoeffizienten anzugeben, dies ist eine der Ideen der MP3-Kompression. Natürlich ist in diesem Fall nicht die Funktion selbst bekannt, es stehen nur äquidistante Messwerte zu Zeitpunkten x_k zur Verfügung. Approximieren wir das Integral in der Formel etwa für die c_k durch $\frac{1}{N}\sum_{j=0}^{N-1} f(x_j)e^{-ikx_j}$, erhalten wir die Formel für die diskrete Fouriertransformation. Da alle Werte reell sind, kann man alternativ auch die diskrete Cosinustransformation nutzen.

Beispiel 2.30 *Sei $f(x) = |x|$ im Intervall $[-\pi, \pi]$, periodisch fortgesetzt auf $\mathbb{R}$. Die Abbildung zeigt die Approximation an f für verschiedene Interpolationsgrade.*

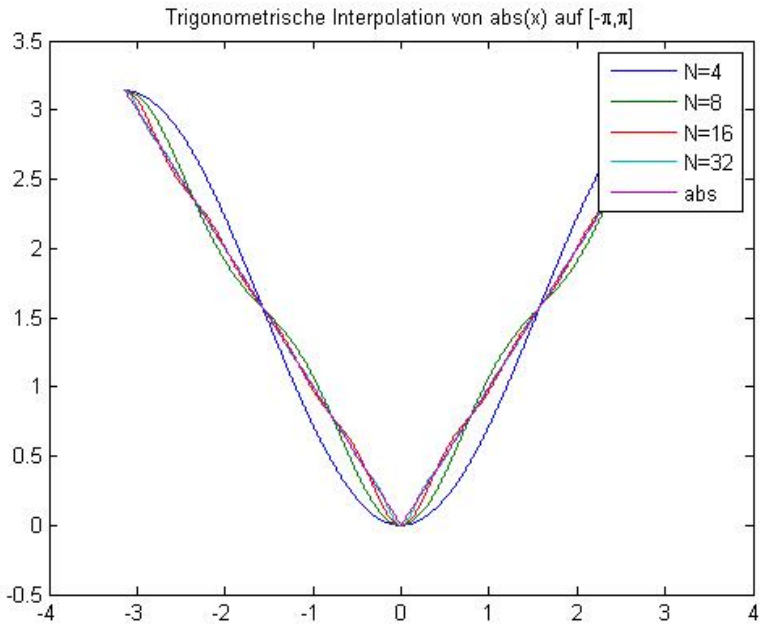


Abbildung 2.13: Trigonometrische Interpolation

Klick für Bild simplefour

```
function simplefour
%SIMPLEFOUR

    function y1=compute(N)
```

Listing 2.15: Trigonometrische Interpolation (simplefour.m)

Klicken für den Quellcode von simplefour.m

Beispiel 2.31 *Das gleiche Beispiel mit der Cosinus–Transformation. Das Ergebnis ist natürlich dasselbe, nur die Programme unterscheiden sich.*

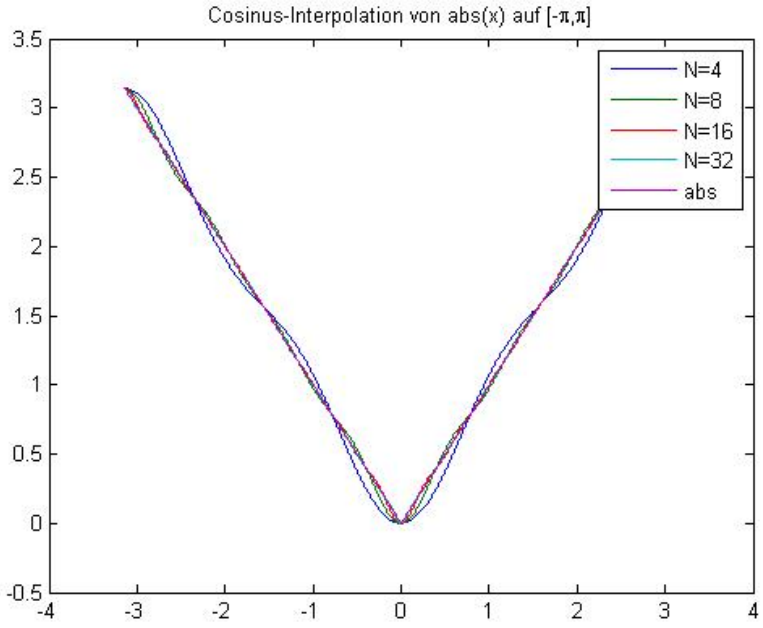


Abbildung 2.14: Interpolation mit der Cosinus-Transformation

Klick für Bild simplecos

```
function simplecostrans
%SIMPLECOSTRANS

    function y1=compute(N)
```

Listing 2.16: Cosinus-Transformation (simplecostrans.m)

Klicken für den Quellcode von simplecostrans.m

Technisch passiert also ungefähr das folgende:

1. Ein Signal wird aufgenommen.
2. Das Signal wird transformiert mit der Cosinus–Transformation.
3. Einige Koeffizienten der Cosinus-Transformation werden übermittelt.
4. 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 **??** und **??** sind dazu nicht geeignet, der Aufwand zur Transformation von N Werten ist N^2 Additionen und Multiplikationen. Wir werden deutlich einfachere Formeln herleiten, die mit $O(N \log N)$ Rechenoperationen auskommen.
Dieselbe Idee lässt sich auch für Bilder durchführen (s. Einleitung): Wir definieren die Fouriertransformation eines (zweidimensionalen) diskreten Bildes, indem wir die Fouriertransformation erst auf jeder Zeile, dann auf jeder Spalte der Matrix ausführen. Wieder wird unsere Funktion als Linearkombination von trigonometrischen Funktionen geschrieben, wieder ist die Idee, dass der Eindruck des Bildes durch relativ wenige Entwicklungskoeffizienten beschrieben wird. Dies ist die Grundlage der JPEG–Kompression (wieder mit der Cosinustransformation an Stelle der Fouriertransformation).

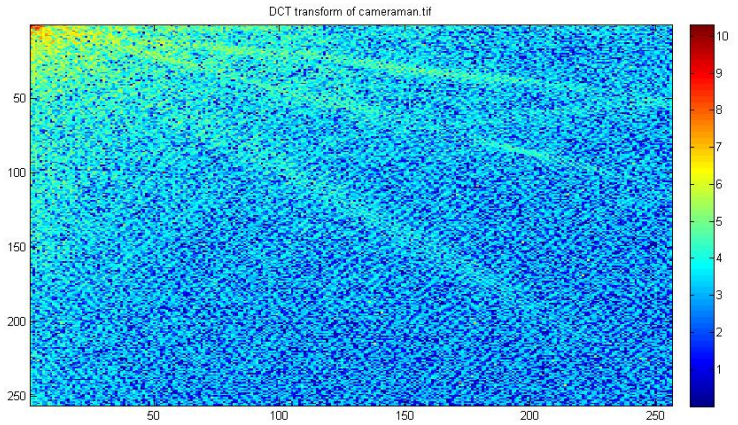


Abbildung 2.15: DCT des Kameramann-Bildes (abs val of log)

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).

2.7.4 FFT: Schnelle Fourier–Transformation

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

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

Für $N = 4$ ist $\omega_N = i$, also

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

Offensichtlich lässt sich die Anzahl der Rechenoperationen für $N = 4$ 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 (`http://www.siam.org/news/news.php?id=637`).

Satz 2.32 (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: Wir bemerken zunächst, dass für jede Zahl $n \in 0..N-1$ gilt

$$n = \lfloor n/p \rfloor\, p + (n \bmod p) = \lfloor n/q \rfloor\, q + (n \bmod q)$$

sowie

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

Es gilt für $k = 0 \ldots N-1$:

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

Diese Summe berechnen wir nach der folgenden Vorschrift:

1. Für jedes r von $0 \ldots p-1$ und k_2 von $0 \ldots q-1$ berechne

$$A^r_{k_2} = \sum_{s=0}^{q-1} \omega_q^{k_2 s} y_{ps+r}.$$

   Für festes r ist das jeweils eine FT der Länge q. Insgesamt sind das p Fouriertransformationen der Länge q, ausgeführt auf den Vektoren $y_{p\cdot+r}$.

2. 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.

3. Für jedes k_1 von $0 \ldots p-1$ und k_2 von $0 \ldots q-1$ 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 $B^{\cdot}_{k_2}$.

□

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 2.33 *Sei M_p die Anzahl der Multiplikationen, A_p die Anzahl der Additionen für eine Fouriertransformation der Länge $N = 2^p$. Dann gilt $M_p = N * p/2 = N/2 \log_2 N$, $A_p = Np = N \log_2 N$.*

Beweis: Nur für M_p. Zur Berechnung einer Fouriertransformation der Länge 2^{p+1} werden nach Satz **??** 2 Fouriertransformationen der Länge 2^p, 2^p Fouriertransformationen der Länge 2 und 2^{p+1} Multiplikatioen benötigt.
Es gilt $M_1 = 1$.

$$\begin{aligned} M_{p+1} &= 2M_p + N \\ &= Np + N \\ &= (2N)/2\,(p+1). \end{aligned}$$

□

Damit ist die Theorie der schnellen FT aber noch nicht beendet. Tatsächlich gibt es für alle Längen spezielle Algorithmen, die eine schnelle Ausführung ermöglichen (siehe etwa Beth, Verfahren der schnellen FT, Mehlhorn, Effiziente Algorithmen, oder diverse Artikel von S. Winograd). Die Schnelle Fouriertransformation ist beispielsweise in der FFTW (`www.fftw.org`) implementiert, sie gilt allgemein als die schnellste Implementation.

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

Listing 2.17: Einfache Implementierung der FFT (myfft.m)

Klicken für den Quellcode von myfft.m

2.8 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 das Buch von de Boor.

2.8.1 Splines

Im Kapitel über Interpolationsfehler haben wir gesehen, dass es von Vorteil ist, die Interpolation auf kleinen Intervallen durchzuführen. Leider geht dabei die Differenzierbarkeit verloren. Splines beheben diesen Mangel: Sie sind Polynome vom Grad $k-1$ auf den einzelnen Intervallen $[x_i, x_{i+1}]$ mit der zusätzlichen Forderung, dass an den Schnittstellen der Intervalle die Polynome von links und rechts bis zur $(k-2)$–ten Ableitung übereinstimmen. Dies ermöglichen wir, indem wir den Polynomgrad auf den Intervallen erhöhen.

Beispiel 2.34 *Gegeben seien die Stützstellen* $-1, 0, 1$ *und die Stützwerte* $0, 0, 1$. *Das interpolierende Polynom ist* $x(x+1)/2$. *Der Polygonzug ist nicht differenzierbar. Lassen wir auf den einzelnen Intervallen quadratische Polynome zu, so interpoliert die Funktion* $(\max(x, 0)^2$ *und ist stetig differenzierbar in o.*

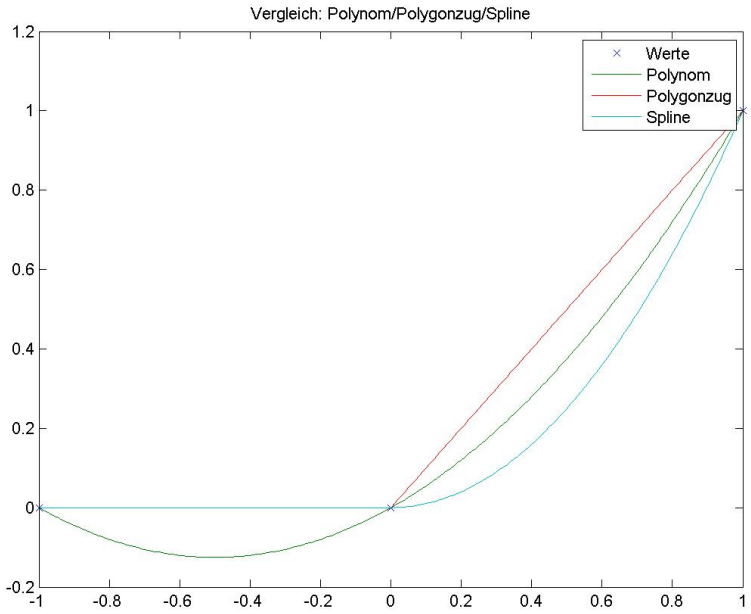


Abbildung 2.16: Vergleich Polynom/Polygon/Spline

Klick für Bild splinetest

Definition 2.35 *Seien $x_0 < x_1 < \ldots < x_n$ reelle Zahlen. Eine Funktion $s : [x_0, x_n] \mapsto \mathbb{R}$ heißt Spline der Ordnung k (zu den Knoten $x_0 \ldots x_n$), falls*

1. $s \in C^{(k-2)}([x_0, x_n])$ *für* $k > 1$.
2. $s|_{[x_i, x_i+1[} \in \mathcal{P}_{k-1}$, $i = 0 \ldots n-1$.

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

Beispiel 2.36

1. *Für $f : \mathbb{R} \mapsto \mathbb{R}$ sei*

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

   Dann ist $(x^{k-1})^+$ in $C^{(k-2)}$ (aber nicht $C^{(k-1)}$) und damit Spline der Ordnung k.

2. *Splines der Ordnung 1 sind genau die Funktionen, die konstant sind in jedem Intervall (siehe **??**).*

3. *Sei s stetig und linear in jedem Intervall, also ein Polygonzug. Genau dann ist s Spline der Ordnung 2.*

Wir werden nun eine Basis des Raums der Splines der Ordnung k angeben und diese als Ansatzfunktionen zur Interpolation nutzen. Hierzu bestimmen wir zunächst die Dimension des Raums: 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 x_1 bis x_{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. Der Ansatzraum hat damit die Dimension $nk-(n-1)(k-1) = n+k-1$.

Definition 2.37 *Seien $x_0 < x_1 < \ldots < x_n$ reelle Zahlen. Dann heißen*

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

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

B–Splines der Ordnung k.

Beispiel 2.38 *(rekursive Definition von B–Splines)*

1. *B-Splines der Ordnung 1 sind stückweise konstant, also Splines der Ordnung 1.*

2. *Sei nun $k = 2$. Für $x < x_i$ verschwinden $B_{i,1}$ und $B_{i+1,1}$, daher ist auch $B_{i,2}(x) = 0$. Gleiches gilt für $x > x_{i+1}$. Sei $x \in [x_i, x_{i+1})$. Dann ist $B_{i,1}(x) = 1$ und $B_{i,2}$ ist linear, $B_{i,2}(x_i) = 0$, $B_{i,2}(x_{i+1}) = 1$. Ebenso $B_{i,2}(x_{i+2}) = 0$. Also ist $B_{i,2}$ ein Polygonzug, der auf allen x_k verschwindet außer für $k = i+1$.*

Bemerkung: Diese Definition kann auch für mehrfache Knoten ($x_i = x_{i+1}$) erweitert werden, in diesem Fall ist B definiert als der Grenzwert für $x_{i+1} \mapsto x_i$.
Das Beispiel lässt sich fortsetzen, es gilt der Satz:

Satz 2.39 *(Eigenschaften der B–Splines)*

1. *$B_{i,k}$ ist Spline der Ordnung k.*

2. *Der Träger von $B_{i,k}$ ist (x_i, x_{i+k}) ($[x_i, x_{i+k})$ für $k = 1$).*

3. *Falls links von x_0 und rechts von x_n noch jeweils $(k-1)$ zusätzliche Punkte x_{-k+1} bis x_{-1} bzw. $x_{n+1} \dots x_{n+k-1}$ eingefügt werden, so sind die B–Splines $B_{j,k}$, $j = -k-1 \dots n-1$, Basis des Spline–Ansatzraums.*

Beweis: Durch einfaches Nachrechnen. Zum letzten Punkt: Zusammen mit den eingefügten Punkten gibt es gerade $(n+k-1)$ B–Splines $B_{i,k}$, $i = -k+1 \dots n-1$, mit Träger in (x_0, x_n). Wegen der Träger–Bedingung sind sie linear unabhängig. □
Bemerkung: Eine alternative Wahl für eine Basis des Spline–Ansatzraums der Ordnung k sind die bereits eingeführten Funktionen $(0^0 = 0)$

$$y_j(x) = (\max(0, x - x_j)^{k-1}), \ j = -(k-1) \dots n-1.$$

```
function y = bspline( i,k,xo,x )
%BSPLINE
    function y=Bint(i,k,xo)
        if (k==1)
```

Listing 2.18: Berechnung von B–Splines (bspline.m)

Klicken für den Quellcode von bspline.m

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

Listing 2.19: Berechnung der Ableitung (dbspline.m)

Klicken für den Quellcode von 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];
```

Listing 2.20: B–Splines (bsplinedemo2.m)

Klicken für den Quellcode von 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)
```

Listing 2.21: Berechnung der Koeffizienten von B–Splines (splinecoeff.m)

Klicken für den Quellcode von splinecoeff.m

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

Listing 2.22: Auswertung von B–Splines anhand der Koeffizienten (testsplinecoeff.m)

Klicken für den Quellcode von testsplinecoeff.m

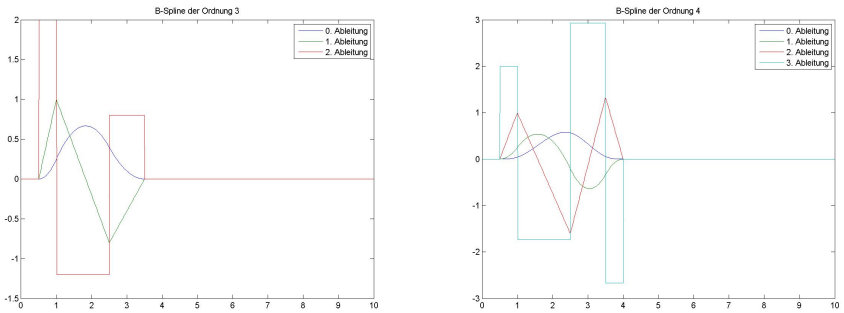


Abbildung 2.17: Splines der Ordnung 3 und 4

Klick für Bild spline3
Klick für Bild spline4

Damit haben wir eine Basis des Spline–Ansatzraums gefunden, und wir können eine Interpolation mit allgemeinen Ansatzfunktionen durchführen. Bleibt noch die Frage offen, ob das Interpolationsproblem eindeutig lösbar ist, d.h. ob die zugehörige Vandermonde–artige Matrix invertierbar ist. Es gilt der Satz:

Satz 2.40 *Seien $x_0 < x_1 < \ldots < x_n$ reelle Zahlen. Gegeben seien Stützstellen t_j mit Stützwerten y_j, $j = 0 \ldots n-k$, $t_j \in (x_j, x_{j+k})$. Dann gibt es genau einen Spline s der Ordnung k, $s(x) = \sum_{i=0}^{n-k} a_i B_{ik}(x)$, so dass $s(t_j) = y_j$, $j = 0 \ldots n-k$.*

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. □

Meist wählt man als Stützstellen gleich einige der Knoten, dies ist nach dem Satz jedoch nicht notwendig. Zur Bestimmung der Koeffizienten der B–Splines muss ein lineares Gleichungssystem gelöst werden. Da die Ansatzfunktionen einen kleinen Träger haben, hat die Matrix höchstens die Bandbreite k, das Gleichungssystem kann also sehr einfach gelöst werden. Zusätzlich fallen natürlich auch bei der Auswertung der Spline–Funktion in der Summe fast alle Summanden weg, d.h. auch hier ist der Aufwand zur Auswertung gering.
de Boor berechnet auch die Kondition der Matrizen, sie ist klein, falls die Interpolationspunkte einigermaßen gleichmäßig verteilt sind.
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.

Lemma 2.41 *Es gilt*

$$g_{\epsilon,x_0}(x) = \begin{cases} \exp(1/((x-x_0)^2 - \epsilon^2)) & |x - x_0| < \epsilon \\ 0 & \textit{sonst} \end{cases} \in C^\infty(\mathbb{R})$$

und $g_{\epsilon,x_0} > 0$ auf $(x_0 - \epsilon, x_0 + \epsilon)$.

Beweis: Durch Nachrechnen, an den Enden des Intervalls geht die Funktion mit allen ihren Ableitungen exponentiell gegen o. $g_{\epsilon,x0}$ ist also eine C^∞–Funktion mit kompaktem Träger. □

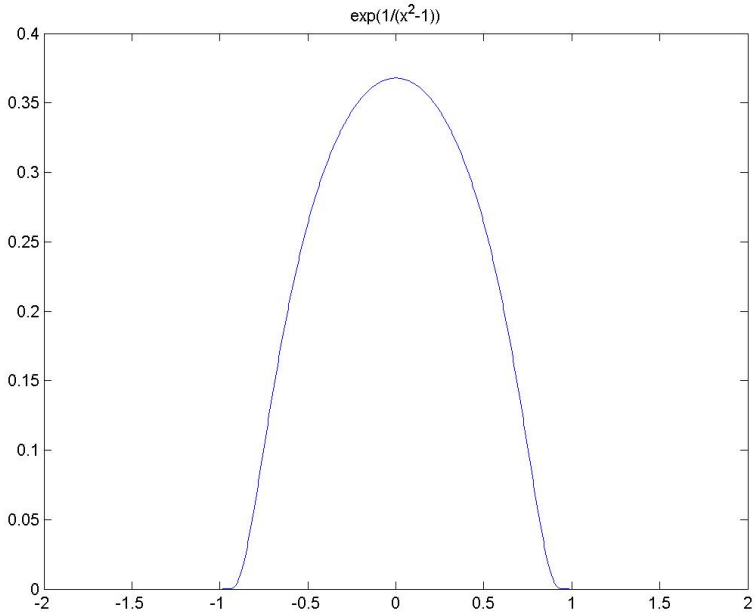


Abbildung 2.18: C^∞–Funktion mit kompaktem Träger

Klick für Bild expfrac

Satz 2.42 *(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*

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

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

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

und die Halbnorm

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

definiert. Dann gilt:

$$u = \arg\min_{v \in V} |v| \Rightarrow u \text{ ist Spline der Ordnung 4 zu } x_0 \dots x_n.$$

Im Sinne unseres Straklattenproblem ist damit der natürliche kubische Spline die Lösung der Aufgabe, eine Kurve mit minimaler Krümmung zu finden, die durch vorgegebene Punkte geht. Achtung: Die Bedingung, dass v linear fortgesetzt werden kann, bedeutet $v''(x_0) = v''(x_n) = 0$, dadurch haben wir zusammen mit der Interpolationsbedingung gerade $n + 1 + 2 = n + 3 = n + k - 1$ Bedingungen, d.h. die Funktion ist auch eindeutig definiert.
Beweis: Zu zeigen ist: u ist auf jedem Intervall ein Polynom vom Grad 3.
Sei $\alpha \in \mathbb{R}$, dann gilt

$$v \in V \Rightarrow u + \alpha(v - u) \in V.$$

$$\begin{aligned} g(\alpha) &:= |u + \alpha(v-u)|^2 \\ &= (u + \alpha(v-u), u + \alpha(v-u)) \\ &= |u|^2 + 2\alpha(u, v-u) + \alpha^2|v-u|^2. \end{aligned}$$

Da $|v|$ sein Minimum in u annimmt, muss $g(\alpha)$ ein Minimum bei 0 haben, also $g'(0) = 0$ und damit $(u, v - u) = 0$.
Sei nun i fest. Sei $v \in V$, $v = u$ außerhalb von $[x_i, x_{i+1}]$ und beliebig auf diesem Intervall. Da $u, v \in C^2$, gilt $u(x_i) = v(x_i)$, $u'(x_i) = v'(x_i)$ und $u''(x_i) = v''(x_i)$ und ebenso für x_{i+1}. Weiter gilt mit partieller Integration

$$\begin{aligned} 0 &= (u, v-u) \\ &= \int_{x_i}^{x_{i+1}} u''(v-u)''\mathrm{dx} \\ &= [u''(v-u)']_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} u'''(v-u)'\mathrm{dx} \\ &= -[u'''(v-u)]_{x_i}^{x_{i+1}} + \int_{x_i}^{x_{i+1}} u''''(v-u)\mathrm{dx} \\ &= \int_{x_i}^{x_{i+1}} u''''(v-u)\mathrm{dx} \end{aligned}$$

Angenommen, $\exists z \in (x_i, x_{i+1})$ mit $u^{(4)}(z) > 0$. Da $u^{(4)}$ stetig ist in z, ist $u^{(4)}$ positiv in einem ϵ–Intervall um z. Wähle ϵ so klein, dass $\epsilon < \min_j(|z - x_j|)$. Setze $v = u + g_{\epsilon,z}$. Dann ist $v \in V$, denn es ist C^∞ und erfüllt die Inerpolationsbedingung, da $g_{\epsilon,z}$ verschwindet auf allen Stützstellen und u die Interpolationsbedingung erfüllt. Auf $(z - \epsilon, z + \epsilon)$ ist $v - u = g_{\epsilon,z}$ positiv, $u^{(4)}$ auch, außerhalb ist es o, also kann das Integral nicht 0 sein, im Widerspruch zur Annahme. Ebenso für negative Werte, also ist $u^{(4)} = 0$ im Inneren jedes Intervalls. □

```
function b = splineinterpol4
```

```
%SPLINEINTERPOL natural cubic splines
K=4;  %Splinegrad
N=8; %Stuetzstellen
```

Listing 2.23: Interpolation mit B–Splines (splineinterpol4.m)

Klicken für den Quellcode von splineinterpol4.m

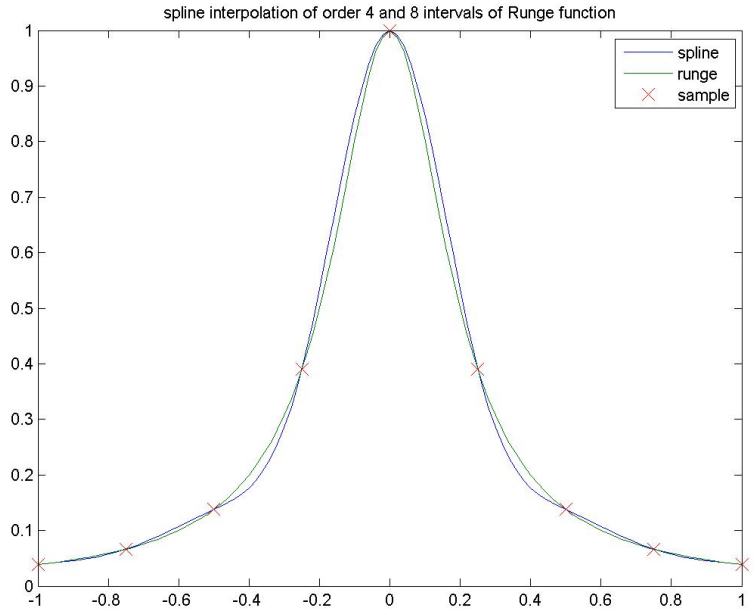


Abbildung 2.19: Spline–Interpolation der Runge–Funktion

Klick für Bild splineinterpol

2.8.2 Spline–Interpolation in höheren Dimensionen

Falls die Auswertepunkte in höheren Dimensionen auf einem rechteckigen Gitter liegen, kann man in höheren Dimensionen so vorgehen wie bei der Fouriertransformation: Für ein Bild etwa, 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 2.43 Lineare Dreiecke. *Sei $G = \bigcup_k \overline{G}_k$, G_k jeweils das Innere eines Dreiecks, und keine Ecke falle auf eine Seite. Als Ansatzraum auf jedem G_k wählen wir die linearen Funktionen. Sei f nun eine Funktion auf G, die linear ist auf jedem G_k. Sei P eine beliebige Ecke. Falls für jedes Dreieck, das P als Ecke hat, der Wert der Fortsetzung von $f|_{G_k}$ gleich ist, hat f insgesamt eine stetige Fortsetzung. Unter dieser Bedingung heißt f 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 $f(P_k) = g(P_k)$.

2.9 Alternative Interpolationsmethoden

Aus Zeitgründen gehen wir auf einige Interpolationsmethoden hier nur am Rande ein.

2.9.1 Rationale Interpolation

Seien $x_0, \ldots, x_N$, $y_0, \ldots, y_N$ Stützstellen und –werte. Wir interpolieren mit einer Funktion $w(x)$ der Form

$$w(x) = \frac{\sum_{k=0}^{P} a_k x^k}{\sum_{j=0}^{Q} b_j x^j}$$

mit $w(x_k) = y_k$ und $N = P + Q + 1$.

Rationale Interpolation liefert häufig bessere Interpolationseigenschaften als die normale Polyominterpolation (das Runge–Beispiel etwa kann sie natürlich exakt approximieren!).
Die Koeffizienten a_k und b_k lassen sich (nach Multiplikation mit dem Nenner) als Lösung eines linearen Gleichungssystems berechnen.

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

Listing 2.24: Rationale Interpolation (ratinterp.m)

Klicken für den Quellcode von ratinterp.m

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

Listing 2.25: Rationale Auswertung (rateval.m)

Klicken für den Quellcode von rateval.m

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

Listing 2.26: Beispiel zur rationalen Interpolation (ratdemo.m)

Klicken für den Quellcode von ratdemo.m

2.9.2 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;
```

Listing 2.27: Approximation durch Ausgleichspolynome (ausgleichspol.m)

Klicken für den Quellcode von ausgleichspol.m

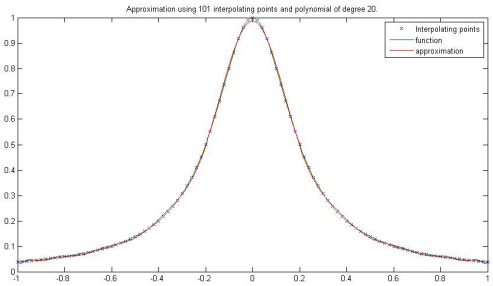


Abbildung 2.20: Approximation durch Ausgleichspolynome

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,
```

Listing 2.28: Bestapproximation in Maple (minimax.m)

Klicken für den Quellcode von minimax.m

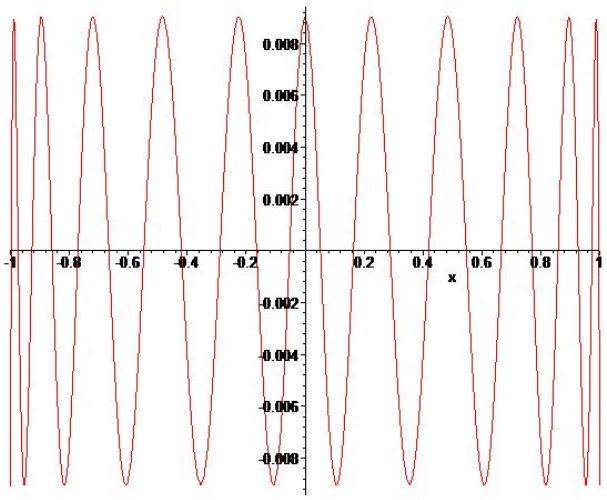


Abbildung 2.21: Fehler der Bestapproximation des Rungebeispiels, berechnet mit Maple

Klick für Bild minimax

```
function y1=matlabminmax(n)
%MATLABMINMAX
%Achtung: Ab Matlab R2009 benoetigt dieses
%Programm die maple-Toolbox
```

Listing 2.29: Bestapproximation in Matlab (matlabminmax.m)

Klicken für den Quellcode von matlabminmax.m

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 das zugehörige Interpolationspolynom.

3.1 Klassische Quadraturformeln durch Interpolation

Definition 3.1 *Sei $f \in C([a,b])$, $x_k \in [a,b]$ paarweise verschiedene Stützstellen, $k = 0 \ldots n$. 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.
$I_n(f)$ 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 $p \in \mathcal{P}_n$ mit $p(x_j) = f(x_j)$, $j = 0 \ldots n$ und berechnen p mit der Lagrange–Formel.

Satz 3.2 *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). Dann ist die zugehörige Quadraturformel

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

exakt von der Ordnung n. Die Gewichte sind eindeutig bestimmt.

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

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

Also gilt

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

Zur Eindeutigkeit setze $p = w_k$, damit gilt $A_k = I_n(w_k) = I(w_k)$. □

Definition 3.3 *(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.*

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.*

Bemerkung: Üblicherweise gibt man die Formeln auf einem Standardintervall an und transformiert linear auf $[a,b]$. Sei also etwa eine Formel auf $[0,1]$ gegeben, dann erhält man durch $x = a + t(b-a)$

$$\int_a^b f(x)\mathsf{dx} = (b-a)\int_0^1 f(a+t(b-a))\mathsf{dt} \sim (b-a)\sum_{k=0}^{n} A_k f(x_k)$$

eine Quadraturformel für $[a,b]$.

Beispiel 3.4 *Berechnung der Gewichte für geschlossene Newton–Cotes–Formeln und $w = 1$ durch Transformation auf $(0,n)$:*
Es gilt mit der Substitution $x = ht + a$:

$$\begin{aligned} A_j &= \int_a^b w(x)w_j(x)\mathsf{dx} \\ &= \int_a^b w(x)\prod_{l\neq j}\frac{x-x_l}{x_j-x_l}\mathsf{dx} \\ &= h\int_0^n \prod_{l\neq j} w(x)\frac{t-l}{k-l}\mathsf{dt} \\ &=: ha_j. \end{aligned}$$

Für die offenen Formeln geht das Integral bis $(n+2)$, und das Produkt von 1 bis $n+1$. Die a_j sind vom Integrationsintervall unabhängige feste Zahlen, insbesondere gilt

$$I_n(f) = h\sum_j a_j f(x_j).$$

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

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

$$x_0 = (a+b)/2,\ a_0 = \int_0^2 1\mathsf{dx} = 2,\ I_0(f) = f\left(\frac{a+b}{2}\right)(b-a).$$

   Dies ist die Mittelpunktsregel.

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

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

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

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

Dies ist die Trapezregel.

Die folgende Tabelle gibt die Gewichte a_j für die geschlossenen Newton–Cotes–Formeln 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	$\frac{2}{45}$	Milne–Regel

Für $n \geq 8$ werden die Gewichte teils negativ, was zu Auslöschung und Instabilität führt (s. Numerische LA).

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

Listing 3.1: Gewichte für Quadraturformeln (quadratur.m)

Klicken für den Quellcode von quadratur.m

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

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

Klicken für den Quellcode von newtoncotesclosed.m

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

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

Klicken für den Quellcode von newtoncotesopen.m

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

Beispiel 3.6 *(Geschlossene Newton–Cotes für die Exponentialfunktion)*

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

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 $f \in C^{n+1}([a,b])$, $W(x) = \prod(x - x_j)$. Dann gilt*

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

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. *Sei n gerade, $n = 2m$, $f \in C^{n+2}([a,b])$. Seien die Gewichtsfunktion w und die Stützpunkte $x_0, \ldots, x_n$ gerade bzgl. des Mittelpunkts $c = (a+b)/2$, also $x_k = x_{n-k} \forall k = 0 \ldots n$ und $w(a+x) = w(b-x)$, $x \in [0, b-a]$. Es gilt*

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

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 in $\mathcal{P}_n$ mit Stützstellen x_j und Stützwerten $f(x_j)$. Nach (2.13) gilt

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

Da $|x - x_j| \leq (b-a)$, gilt $W(x) \leq (b-a)^{n+1}$, daraus folgt Teil 1.
Seien nun die Voraussetzungen aus Teil b erfüllt. Dann gilt $a + x_m = b - x_m$, also $x_m = c$. Da die x_k symmetrisch zu c liegen, gilt $W(c+x) = -W(c-x)$. Begründung:

$$(c+x) - x_m = (c+x) - c = x = -((c-x) - x_m)$$

und

$$(c + x - x_{m+k})(c + x - x_{m-k}) = -(c - x - x_{m-k}) \cdot -(c - x - x_{m+k}).$$

Da w gerade ist bzgl. c, gilt insbesondere $\int_a^b w(x)W(x)\mathrm{dx} = 0$.
Da $f \in C^{n+2}$, können wir f^{n+1} in die Taylorreihe der Ordnung 1 um c entwickeln, es

gilt also

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

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

Die Abschätzung für W geht vom schlimmsten Fall aus, dass alle Auswertepunkte am linken oder rechten Rand verteilt sind. Natürlich bekommen wir für gleichmäßig verteilte Punkte (oder die Tschebyscheff–Interpolationspunkte) eine viel bessere Abschätzung. Hier für die geschlossenen Newton–Cotes–Formeln.

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 mit* $h = (b-a)/n$

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

mit $c_n = \frac{1}{(n+1)!}\int_0^n \prod_{k=0}^n |t-k|\mathsf{dt}||w||_\infty$.
Falls n *gerade ist,* w *gerade bezüglich* $(a+b)/2$ *und* $f \in C^{(n+2)}([a,b])$*, so gilt*

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

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

Beweis: Es gilt mit der Substitution $x = a + th$

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

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

Bemerkung: Die Konstanten hängen nicht mehr vom Intervall, sondern nur noch von n ab.
Bemerkung: Die Simpson–Regel liefert einen asymptotisch deutlich besseren Wert als die Trapezregel, aber die Newton–Regel ist nur unwesentlich besser als die Simpson–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 bekommt man tatsächlich für $n \mapsto \infty$ im allgemeinen keine Konvergenz von I_n gegen I, denn der Fehler hängt vom Verhalten der Ableitungen von f ab. 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 $\tilde{I}_n(f)$ 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)*

$$\begin{aligned}
\tilde{I}_1(f) &= h\left(((f(x_0)+f(x_1))+(f(x_1)+f(x_2))+\ldots+(f(x_{n-1})+f(x_n)))\right) \\
&= h(f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{n-1})+f(x_n)) \\
&= h\left(f(x_0)+2\sum_{k=1}^{n-1} f(x_k)+f(x_n)\right)
\end{aligned}$$

2. (Zusammengesetzte Simpson–Regel)

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

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

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

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:

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

□

Korollar 3.12 *(Fehler von 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 $f \in C^4$

$$|I(f)-\tilde{I}_2(f)| \leq \frac{b-a}{180}h^4||f^{(4)}||_\infty.$$

3.3 Romberg–Verfahren

Zu berechnen sei ein Integral im Intervall $[a, b]$. Wir approximieren es mit einer Folge von zusammengesetzten Newton–Cotes–Formel auf $p_m = 2^m$ Intervallen. Dann können wir für jedes p_m die Funktionsauswertungen aus p_{m-1} wiederverwerten. Wir erhalten dadurch Näherungen $\tilde{I}_n(h)$, $\tilde{I}_n(h/2)$ usw. Zur Verbesserung der Konvergenzordnung bietet sich natürlich die Richardson–Extrapolation an – dies ist das Romberg–Verfahren. Es ist umso effizienter, je größer der Exponent in der Taylorentwicklung des Fehlers ist. Für die zusammengesetzte Trapezregel $\tilde{I}_1(h)$ besitzt der Quadraturfehler eine Entwicklung in h^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*

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

mit

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

und den Bernoulli–Zahlen B_k.

Zum Beweis 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*

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

Bemerkung: Hieraus folgt 3.14 sofort durch die Substitution $g(x) = f(a + xh)$. Beachte das h, das durch die innere Ableitung an den Integralen entsteht.
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. □

9.Mai 2011
Bemerkung: 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;
```

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

Klicken für den Quellcode von symbolicromberg.m

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 3.16 *(Harmonische Analyse)*

1. *Sei $f \in C^{2m}(\mathbb{R})$ periodisch, $I(f)$ das Integral über die ganze Periode. 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 gegen 0 als jede Potenz von h.*

Beweis: Satz 3.14. □

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 bei der Polynominterpolation). Bei periodischen Funktionen gibt es keine Intervallenden, also ist eine gleichmäßige Verteilung der Auswertepunkte optimal.

3.5 Gauss–Formeln

Wir haben gesehen, dass man für gerades n und gerades w die Ordnung, und damit die Konvergenzordnung, durch die Wahl symmetrischer Stützstellen erhöhen kann. Es stellt sich die Frage, welche Ordnung man durch geschickte Wahl der Stützstellen maximal erreichen kann. Hierzu erinnern wir zunächst an die Definition der orthogonalen Polynome.

Definition 3.17 *Sei $(,)$ ein Skalarprodukt auf dem Vektorraum der Polynome. Dann liefert das Gram–Schmidtsche Orthogonalisierungsverfahren, angewandt auf die Folge der Monome x^k, eine Folge von Polynomen $v_k \in P_k$, die orthonormal sind bezüglich dieses Skalarprodukts. Sie heißen orthogonale Poynome.*

Sei ab jetzt immer $(p,q) = \int_a^b w(x)p(x)q(x)\mathrm{dx}$.

Beispiel 3.18 *(orthogonale Polynome)*

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

Bemerkung: grad $v_k = k$. $v_k \perp \mathcal{P}_{k-1}$, denn $v_0, \ldots, v_{k-1}$ sind eine ONB von $\mathcal{P}_{k-1}$ und stehen senkecht auf v_k.
Wir betrachten nun Skalarprodukte der Form $(p,q) = \int_a^b w(x)p(x)q(x)\mathrm{dx}$ mit positiver Gewichtsfunktion w und zugehörigen orthogonalen Polynomen v_k.

Lemma 3.19 *v_k hat in (a,b) genau k einfache Nullstellen.*

Beweis: Seien $x_1, \ldots, x_m$ die Argumente, an denen v_k in (a,b) sein Vorzeichen ändert. Angenommen, $m < k$. Sei $q(x) := \prod_{j=1}^m (x - x_j) \in \mathcal{P}_m$. Da $m < k$ ist

$$(q, v_k) = \int_a^b w(x)q(x)v_k(x)\mathrm{dx} = 0.$$

Da $q(x)v_k(x)$ sein Vorzeichen nicht ändert, folgt $q(x)v_k(x) = 0 \forall x \in (a,b)$, aber qv_k ist Polynom vom Grad größer als 0. Dies ist ein Widerspruch, also ist $m = k$. □

Satz 3.20 *(Formel von Gauss, Gauss–Quadraturformel)*
Seien $x_0, \ldots, x_n$ die (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

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

und

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

□

Korollar 3.21 *(Gewichte der Gaussformeln)*
Die Gewichte A_k der Gaussformeln sind positiv, insbesondere gilt

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

Beweis: Übungen.

Satz 3.22 *Keine Quadraturformel mit $(n+1)$ Auswertungen an den Stützstellen $x_0, \ldots, x_n$ ist exakt von der Ordnung $2n+2$.*

Beweis: Setze $p(x) = \prod_k (x - x_k)^2 \in \mathcal{P}_{2n+2}$. $p \geq 0$, also $I(p) > 0 = I_n(p)$. □

Beispiel 3.23 *Gauss–Quadratur*
Wir betrachten $w = 1$ auf dem Intervall $[-1, 1]$. Die zugehörigen orthogonalen Polynome sind die Legendre–Polynome. Wir erhalten (für nicht normierte Polynome $P(n, x)$):

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 der Auswertungen fest und vergleichen die Genauigkeit, die die Formeln mit diesen Auswertungen liefern. Dargestellt ist jeweils der Fehler der jeweiligen Formel. Auf der y–Achse ist der Fehler logarithmisch dargestellt, auf der x–Achse der $\log_2$ der Anzahl der Auswertungen. 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 ist die Trapezregel bereits optimal.

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}. Die Erklärung dafür werden wir anschließend sehen, ab hier sind die Gewichte der NC–Formeln negativ.
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.
Zur periodischen Funktion: Wie erwartet, sind die zusammengesetzten Formeln optimal. Gauss schlägt sich hier erstaunlich schlecht.

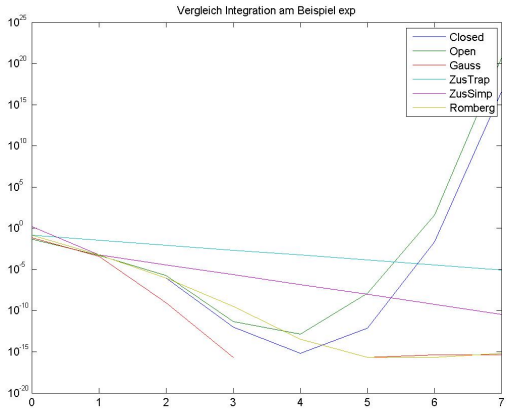


Abbildung 3.1: Vergleich von Quadraturformeln Exponentialfunktion

Klick für Bild integexp

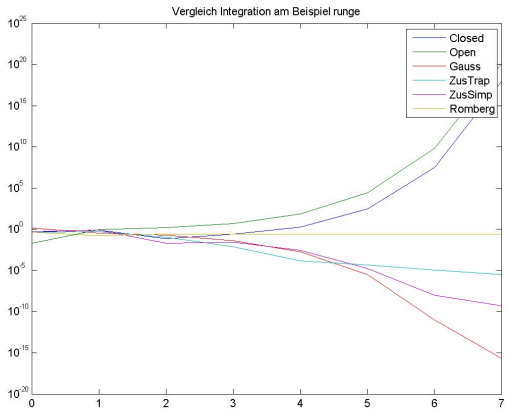


Abbildung 3.2: Vergleich von Quadraturformeln Rungebeispiel

Klick für Bild integrunge

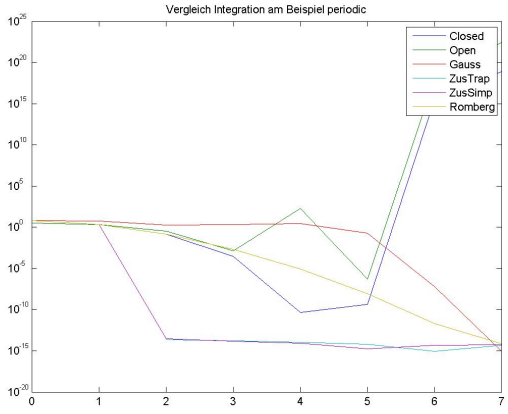


Abbildung 3.3: Vergleich von Quadraturformeln Periodische Funktion

Klick für Bild integperiodic

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

Listing 3.5: Vergleich von Quadraturformeln (plotcompare.m)

Klicken für den Quellcode von plotcompare.m

Wir müssen noch erklären, warum die NC–Algorithmen für große Polynomgrade sagenhafte Fehler liefern, obwohl die Polynominterpolation vernünftig ist (etwa für die $\exp$-Funktion). Hierzu benötigen wir den Begriff der Stabilität.

Definition 3.24 *(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. Hierbei betrachten wir alle Fehler immer relativ zum korrekten Ergebnis.

Satz 3.25 *Quadraturformeln mit positiven Gewichten sind stabil.*

Beweis: Falls statt f nur eine Näherung $\tilde{f}$ bekannt ist mit $|f(x) - \tilde{f}(x)| \leq \epsilon |f(x)|$, 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.
Sei nun I_n eine Quadraturformel. Es gilt

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

Für positive Gewichte ist die rechte Seite $\epsilon I_n(|f|)$, also eine Näherung für $\epsilon I(|f|)$, also in der Größenordnung der analytischen Fehlerschranke. Für negative Gewichte kann die rechte Seite sehr viel größer sein. Dies erklärt die Probleme der NC–Formeln für großes n in **??**. □
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 Fehler im Ergebnis können wir in diesem Fall keine Abschätzung angeben. **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 bei den NC–Formeln 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. Für die Differentiation sehen die Dinge leider sehr viel schlechter aus.

3.7 Numerische Differentiation

Sei $f \in C^m$. Wir suchen 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 3.26 *Sei $f \in C^{n+1}(\mathbb{R})$, $x_0, \ldots, x_n$ paarweise verschieden, $l \leq n$ Dann gibt es Zahlen α_k, $k = 0 \ldots n$, so dass*

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

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

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

Wähle die α_k so, dass $\sum_k \alpha_k x_k^j = \delta_{jl}$. Dies geht, denn die Koeffizienten bilden eine Vandermonde–Matrix. □

Beispiel 3.27 *(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 ($f \in C^4$):

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

9.5.2011

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 im allgemeinen nicht einmal stetig, und erst recht nicht differenzierbar sein.
Wir betrachten die numerische Berechnung der ersten Ableitung an der Stelle 0, sei also

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

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

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

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 (im WS 2011).

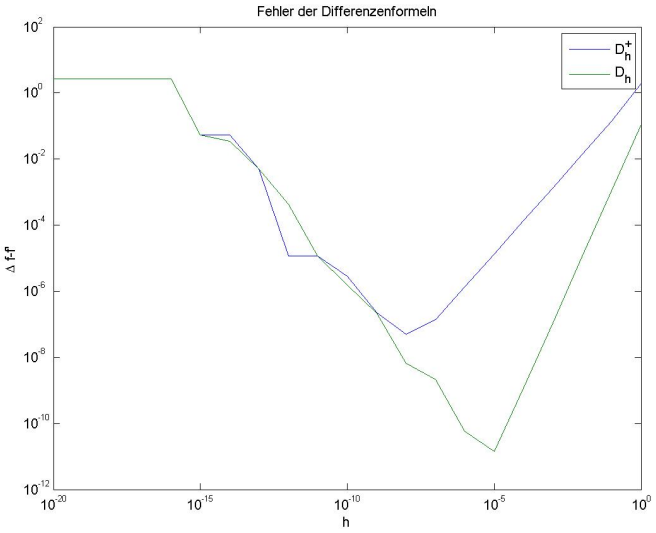


Abbildung 3.4: Fehler der Differenzenformeln gegen die Schrittweite, halblogarithmisch

Klick für Bild difffehler

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

Listing 3.6: Fehler der Differenzenformeln (diffdemo.m)

Klicken für den Quellcode von diffdemo.m

Kapitel 4

Numerische Behandlung gewöhnlicher Differentialgleichungen

4.1 Einleitung und Beispiele

Viele Anwendungsprobleme in der Mathematik lassen sich durch Differentialgleichungen modellieren. Im einfachsten Fall wird dabei das Verhalten einer physikalischen Größe oder etwa der Wert eines Wertpapiers in der Zeit beschrieben (Dynamik), insbesondere in einer Variablen. Einige genauere Beispiele finden sich z.B. im Buch von Martin Hanke (Kapitel 11) oder Walter. 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. Referenz für die theoretischen Grundlagen ist das klassische Buch von Wolfgang Walter, Gewöhnliche Differentialgleichungen – eine Einführung.

Beispiel 4.1 *(Nuklearer Zerfall – Anfangswertaufgaben)*
Sei $N(t)$ die Anzahl der Teilchen eines radioaktiven Stoffes, die zum Zeitpunkt t existieren. Es sei bekannt, dass zum Zeitpunkt $t = 0$ $N(0) = N_0$. Zu bestimmen ist N. Die Anzahl der Teilchen, die in einem kleinen Zeitintervall zerfallen, ist proportional zu $N(t)$ und zur Länge des Intervalls, es gilt also

$$N(t + \Delta t) = N(t) - q\,\Delta t\,N(t).$$

Genauer gilt eigentlich:

$$N(t + \Delta t) = N(t) - q \int_t^{t+\Delta t} N(s)ds,$$

wir benutzen zur Vereinfachung die obere approximative Gleichung für kleine Zeitintervalle.

Unter der Annahme, dass $N(t)$ differenzierbar ist, gilt damit

$$-q\,N(t) = \frac{N(t+\Delta t) - N(t)}{\Delta t} \to_{\Delta t \to 0} N'(t).$$

Wir erhalten also für $t > 0$ das (lineare) Anfangswertproblem

$$N'(t) = -qN(t),\ N(0) = N_0$$

mit der Lösung

$$N(t) = N_0 \exp(-qt).$$

Bemerkung: Physikalisch darf man hier natürlich nicht zu genau hinschauen - in der einfachen Modellierung war N ganzzahlig und damit sicherlich nicht differenzierbar, hier hilft aber der Übergang zu Dichten oder Aufenthaltswahrscheinlichkeiten.

Beispiel 4.2 *(Einfachstes Räuber–Beute–Modell — Systeme)*
Seien $F(t)$ und $H(t)$ die Anzahl der Füchse bzw. Hasen in einer Population. Ohne Füchse entwickeln sich die Hasen ungestört, die Anzahl der Nachkommen ist proportional zur Zahl der Hasen. Ohne Hasen ist die Anzahl der sterbenden Füchse proportional zur Anzahl der Füchse.
Insgesamt ergibt sich eine Dynamik–Gleichung der Form

$$\begin{aligned} H(t+\Delta t) &= H(t) + a\Delta t H(t) - b\Delta t F(t)H(t) \\ F(t+\Delta t) &= F(t) - c\Delta t F(t) + d\Delta t F(t)H(t) \end{aligned}$$

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 und F_0 für die Populationszahlen zum Zeitpunkt $t = 0$.
Nach wie vor ist der einzige Parameter t, wir erhalten also mit $y = (F, H)$ wieder eine gewöhnliche Differentialgleichung der Form $y'(t) = f(t, y(t))$, diesmal mit einer Funktion im $\mathbb{R}^2$. Dies ist ein (gekoppeltes) System von gewöhnlichen Differentialgleichungen. Die Lösungen sind i.a. nicht explizit angebbar. Sie sind in der Regel periodisch.

Beispiel 4.3 *(Beschleunigung — Differentialgleichungen höherer Ordnung)*
Ein Fahrzeug wird (ohne Reibung) mit der Kraft $K(t)$ beschleunigt. Sei $x(t)$ die zurückgelegte Strecke. Dann gilt mit der Geschwindigkeit $v(t)$ und der Masse M

$$\begin{aligned} x(t+\Delta t) &= x(t) + \Delta t v(t) \\ v(t+\Delta t) &= v(t) + \Delta t \frac{K(t)}{M} \end{aligned}$$

und damit das System

$$\begin{aligned} x'(t) &= v(t) \\ v'(t) &= \frac{K(t)}{M}. \end{aligned}$$

Die Lösung ist gegeben durch

$$v(t) = \int^t \frac{K(s)}{M} ds + v_0,\ x(t) = \int^t v(r) dr + x_0.$$

Vereinfacht können wir das auch schreiben als $x''(t) = K(t)/M$, also als Differentialgleichung höherer Ordnung. Die Einführung der Geschwindigkeit als zusätzliche Funktion verwandelt also diese Gleichung höherer Ordnung (mit höheren Ableitungen) in ein System von Differentialgleichungen erster Ordnung. Dies lässt sich immer durchführen, wir können uns also bei der Betrachtung von gewöhnlichen Differentialgleichungen auf Systeme mit ersten Ableitungen beschränken.

Definition 4.4 *(Anfangswertprobleme für Systeme von gewöhnlichen Differentialgleichungen)*

1. *Sei $I = [a, b]$ mit $a < b$. Sei $G \subset \mathbb{R}^n$ offen und zusammenhängend, $f : (I \times G) \mapsto \mathbb{R}^n$ stetig. Eine Gleichung der Form*

$$y'(t) = f(t, y(t)) \tag{4.1}$$

   heißt gewöhnliche Differentialgleichung. Falls für $y : I \mapsto G$, $y \in C^1$, für alle $x \in I$ die Gleichung 4.1 erfüllt ist, so heißt y Lösung der gewöhnlichen Differentialgleichung.

2. *Sei zusätzlich $y_0 \in G$ gegeben. Dann heißt $y : I \mapsto G$ Lösung der Anfangswertaufgabe*

$$y'(t) = f(t, y(t)),\ y(a) = y_0$$

   falls y die Differentialgleichung erfüllt und es gilt $y(a) = y_0$.

Bemerkung: Falls y Lösung des Anfangswertproblems ist, so gilt

$$y(t) = y_0 + \int_a^t f(x, y(x)) \mathrm{dx}. \tag{4.2}$$

Falls y stetig ist und 4.2 erfüllt, so ist sogar $y \in C^1$ und y erfüllt die Anfangswertaufgabe. Für stetige Funktionen ist die Lösung der Integralgleichung also äquivalent zur Lösung der Anfangswertaufgabe.

Beispiel 4.5 *(Beispiele für Anfangswertaufgaben, Existenz und Eindeutigkeit)*

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* $\mathbb{R}$ *definiert. Die Anfangswertaufgabe hat also nur eine Lösung in einem Intervall um den Anfangspunkt.*

2. $y'(t) = y^{1/3}$, $y(0) = 0$. *Lösungen sind* $y(t) = (2/3t)^{3/2}$ *und* $y = 0$, *die Lösung ist also nicht eindeutig. Bemerkung:* $y^{1/3}$ *ist nicht differenzierbar bei* 0.

4.2 Klassische Typen von Differentialgleichungen

Beispiel 4.6 *(Elementare Differentialgleichung)*
Differentialgleichungen der Form

$$y'(t) = f(t)$$

heißen Elementare Differentialgleichungen. Die Lösung der zugehörigen AWA ist

$$y(t) = y_0 \int_a^t y'(x) dx.$$

Beispiel 4.7 *(autonome Differentialgleichung)*
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*

$$1 = z'(y(t))y'(t) = z'(y(t))f(y(t)).$$

Es gilt also

$$z'(y) = \frac{1}{f(y)},$$

also eine elementare Differentialgleichung für z, *die durch Integration gelöst werden kann.* y *ist dann die Umkehrfunktion von* z. *Beispiel für* $y' = (1 + y^2)$*:*

$$z' = \frac{1}{1+y^2} \Rightarrow t = z(y(t)) = \arctan(y(t)) + C \Rightarrow y(t) = \tan(t - C).$$

Beispiel 4.8 *(lineare Differentialgleichung in einer Variablen)*
Sei

$$y'(t) = a(t)y(t).$$

Dann ist

$$a(t) = y'(t)/y(t) = \log(y(t))',$$

also

$$y(t) = C\exp(\int^t a(x)dx).$$

Die affin–lineare Gleichung

$$y'(t) = a(t)y(t) + b(t)$$

löst man durch Variation der Konstanten (Übungen).

4.3 Grafische Lösung

Viele klassische Differentialgleichungen haben keine elementare Lösung, z.b. die Besselsche Differentialgleichung

$$t^2y'' + ty' + (t^2 - n^2)y = 0$$

oder die Riccati–Differentialgleichung

$$y' = y^2 + 1 - t^2.$$

Es kommen daher häufig nur näherungsweise Lösungsmethoden in Frage. Eine einfache Möglichkeit ist die grafische Lösung für eindimensionale Gleichungen.
Durch die Differentialgleichung ist in jedem Punkt (t, y) des $\mathbb{R}^2$ die Steigung der Tangente einer Lösung der Differentialgleichung durch (t, y) festgelegt. Die Steigung wird für ausgewählte Punkte als Vektorfeld in ein Koordinatensystem eingetragen. Anhand der Vektoren wird die Lösung der Differentialgleichung, ausgehend von der Anfangsbedingung, in das Koordinatensystem eingetragen. Dies ist bereits auch die Grundidee der einfachen numerischen Verfahren.

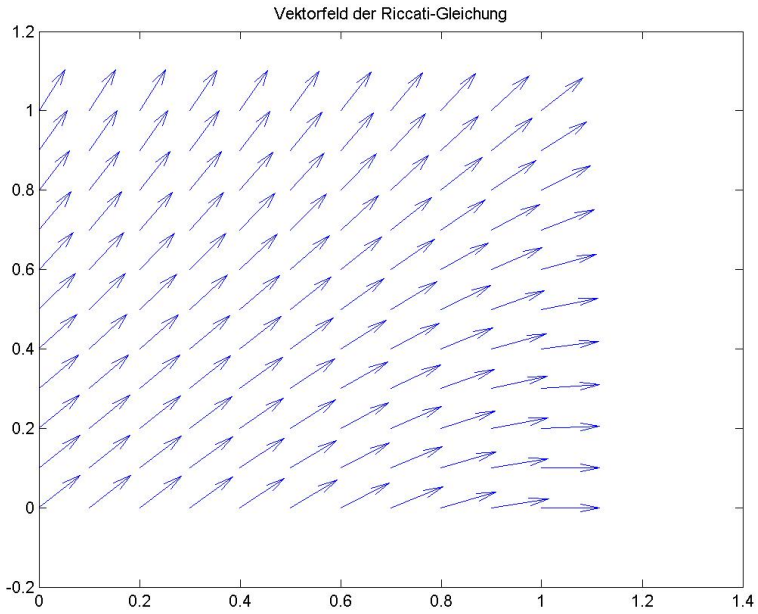


Abbildung 4.1: Vektorfeld der Riccati–Gleichung $y' = 1 + y^2 - t^2$

Klick für Bild vectorfieldriccatti
Klick für Matlab Figure vectorfieldriccatti

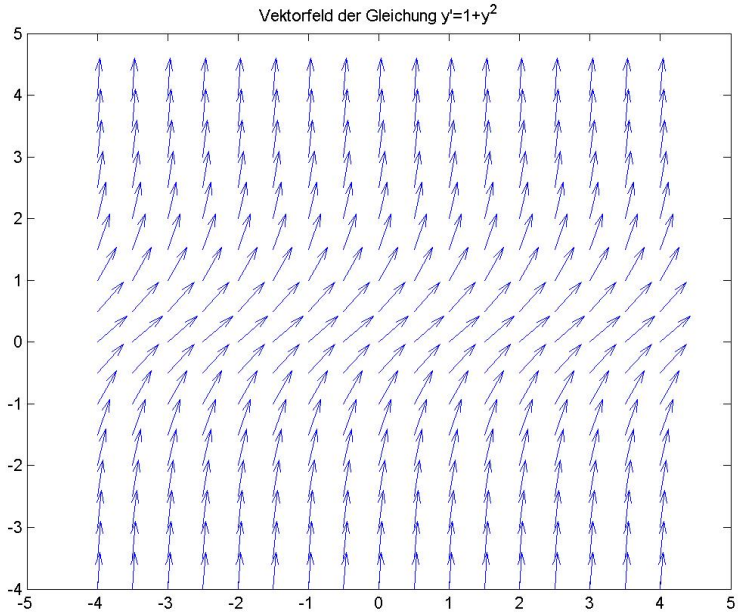


Abbildung 4.2: Vektorfeld der autonomen Gleichung $y' = 1 + y^2$

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;
```

Listing 4.1: Vektorfelder von GDGL (vectorfield.m)

Klicken für den Quellcode von vectorfield.m

4.4 Wiederholung: Analysis gewöhnlicher Differentialgleichungen

Wir erinnern kurz an einige Grundergebnisse aus der Analysis II.

Satz 4.9 *(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: Nur die Idee: 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 $g(\overline{x}) = \overline{x}$. □

Satz 4.10 *(Satz von Picard–Lindelöf, lokaler Existenzsatz)*
Sei alles wie in 4.4, insb. f stetig. Sei zusätzlich f Lipschitz–stetig im zweiten Argument, d.h.

$$\exists L : ||f(t, y) - f(t, z)|| \leq L||y - z|| \forall t \in I, y, z \in G.$$

Dann gibt es ein $\epsilon > 0$ und eine 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. f ist insbesondere Lipschitz–stetig, wenn es differenzierbar ist.
Beweis: Wir zeigen mit Hilfe des Banachschen Fixpunktsatzes, dass die zugehörige Integralgleichung eine eindeutige Lösung besitzt. Sei zunächst $0 < \epsilon < (b - a)0$. Dann ist $E := (C^0([a, a + \epsilon], \mathbb{R}^n), || \cdot ||_\infty)$ vollständig, denn

$$g_n \in C^0, \; ||g_n - g||_\infty \mapsto 0 \Rightarrow g \in C^0.$$

Wir können den Fixpunktsatz also in diesem Vektorraum anwenden.
G ist offen.

$$y_0 \in G \Rightarrow \exists B := K_\delta(y_0) \subset G,$$

wobei immer

$$K_\delta(y) = \{x : ||y - x|| \leq \delta\}.$$

$I \times B$ ist kompakt, f stetig, also existiert

$$M := \max_{x \in I, y \in B} |f(x, y)|.$$

Sei nun

$$0 < \epsilon < \min(b - a, \delta/M), \;\; I_\epsilon := [a, a + \epsilon] \subset I.$$

Sei

$$D = C^0(I_\epsilon, B).$$

D ist abgeschlossene Teilmenge von E. Die Funktion

$$T : D \mapsto D, (Ty)(t) := y_0 + \int_a^t f(x, y(x))\mathsf{dx}$$

ist wohldefiniert: Sei $y \in D$. y ist stetig, also ist Ty stetig (und sogar differenzierbar), und

$$||(Ty)(t) - y_0|| \leq \int_a^{a+\epsilon} |f(x, y(x))|\mathsf{dx} \leq \epsilon M \leq \frac{\delta}{M} M = \delta,$$

also ist $(Ty)(t) \in B$ und damit

$$Ty \in C^0(I_\epsilon, B) = D.$$

Weiter gilt für $u, v \in D$

$$\begin{aligned} ||Tu - Tv||_\infty &= || \int_a^t f(x, u(x)) - f(x, v(x))\mathsf{dx}||_\infty \\ &\leq \int_a^{a+\epsilon} |f(x, u(x)) - f(x, v(x))|\mathsf{dx} \\ &\leq L \int_a^{a+\epsilon} |u(x) - v(x)|\mathsf{dx} \\ &\leq L\epsilon ||u - v||_\infty. \end{aligned}$$

Wähle nun

$$\epsilon < \min(\frac{1}{L}, b - a, \frac{\delta}{M}).$$

Dann ist T kontrahierend. Also hat T einen eindeutigen Fixpunkt $\overline{y}$ mit

$$T\overline{y} = \overline{y}.$$

$\overline{y}$ ist stetig nach Voraussetzung, löst die Integralgleichung, also ist $y_\epsilon := \overline{y}$ eindeutige Lösung der Anfangswertaufgabe.
Beweis des Zusatzes: y_{ϵ_0} ist Lösung der AWA auch im Intervall I_ϵ, und die Lösung ist eindeutig. □
Bemerkung: Wegen $T\overline{y} = \overline{y}$ gilt insbesondere $|\overline{y}(t) - y_0| \leq M(t-a)$.

Satz 4.11 *Seien alle Bezeichnungen wie oben, und es gelte $\delta > M(b-a)$. Dann gibt es eine eindeutige Lösung im Intervall $[a,b]$.*

Beweis: Betrachte auf E die zu $||\cdot||_\infty$ äquivalente Norm $|||f||| := ||\exp(-2Lt)f(t)||_\infty$ (Übungen).

$$\begin{aligned}
|||Tu - Tv||| &= \sup_{x\in[a,b]} |\exp(-2Lx)\int_a^x f(t,u(t)) - f(t,v(t))\mathrm{dt}| \\
&\leq \sup_{x\in[a,b]} \exp(-2Lx)\int_a^x L\exp(2Lt)\exp(-2Lt)|u(t)-v(t)|\mathrm{dt} \\
&\leq L\sup_{x\in[a,b]} \exp(-2Lx)\int_a^x \exp(2Lt)|||u-v|||\mathrm{dt} \\
&\leq L|||u-v|||\sup_{x\in[a,b]} \exp(-2Lx)\frac{1}{2L}(\exp(2Lx)-\exp(2La)) \\
&\leq \frac{1}{2}|||u-v|||.
\end{aligned}$$

□
Eine wichtige Abschätzung für die Größenordnung von Lösungen gewöhnlicher Differentialgleichungen und insbesondere eine Aussage über den unvermeidlichen Fehler beim Lösen von Gleichungen (s.o. Stabilität) liefert das Lemma von Gronwall. Die Idee dabei ist leicht zu verstehen: Die Differentialgleichung $u'(t) = \beta u(t)$ hat die Lösung $u(t) = u(a)\exp(\beta(t-a))$. Vertauscht man $=$ durch $\leq$, so bleibt die Aussage richtig.

Satz 4.12 *(Lemma von Gronwall) (originale Form, s. Walter)*
Sei

$$u'(t) \leq \beta u(t)\, \forall t \in [a,b].$$

Dann gilt

$$u(t) \leq u(a)\exp(\beta(t-a)).$$

Beweis: Sei

$$v(t) := \exp(\beta(t-a)),$$

also

$$v'(t) = \beta v(t),\ v(t) > 0,\ v(a) = 1.$$

Dann gilt

$$\frac{d}{dt}\frac{u(t)}{v(t)} = \frac{u'(t)v(t) - v'(t)u(t)}{v(t)^2} \leq \frac{\beta u(t)v(t) - \beta v(t)u(t)}{v(t)^2} = 0$$

und damit

$$\frac{u(t)}{v(t)} \leq \frac{u(a)}{v(a)} = u(a).$$

□

Satz 4.13 *(Lemma von Gronwall, zweite Form)*
Sei $\beta > 0$ fest, $u, \alpha \in C^0([a,b])$. Falls

$$u(t) \leq \alpha(t) + \beta \int_a^t u(s)ds,$$

so gilt

$$u(t) \leq \alpha(t) + \beta \int_a^t \alpha(s)\exp(\beta(t-s))ds.$$

Falls $\beta = \beta(s) \in C^0([a,b] > 0$, so gilt für

$$u(t) \leq \alpha(t) + \int_a^t \beta(s)u(s)ds :$$

$$u(t) \leq \alpha(t) + \int_a^t \beta(s)\alpha(s)\exp\left(\int_s^t \beta(r)dr\right)ds.$$

Für $\alpha(t) = u(a)$ erhalten wir die erste Form zurück.
Beweis: Wir zeigen nur die erste Form (β konstant), die andere geht analog (Übungen). Sei

$$v(t) = \exp(\beta(a-t)) \int_a^t \beta u(s)\mathsf{ds}.$$

Dann gilt

$$\begin{aligned} v'(t) &= -\beta\exp(\beta(a-t)) \int_a^t \beta u(s)\mathsf{ds} + \exp(\beta(a-t))\beta u(t) \\ &= \beta\exp(\beta(a-t)) \underbrace{(u(t) - \beta\int_a^t u(s)\mathsf{ds})}_{\leq\alpha(t)} \\ &\leq \beta\alpha(t)\exp(\beta(a-t))),\ \beta > 0 \end{aligned}$$

und damit

$$v(t) = \underbrace{v(a)}_{0} + \int_a^t v'(s)\mathsf{ds} \leq \beta \int_a^t \alpha(s)\exp(\beta(a-s))\mathsf{ds}$$

und

$$\beta \int_a^t u(s)\mathsf{ds} = \exp(\beta(t-a))v(t) \leq \beta \int_a^t \alpha(s)\exp(\beta(t-s))\mathsf{ds}.$$

□

Bemerkung: Die Vorzeichen von u und α sind beliebig. Die Stetigkeit wird benutzt bei der Ableitung des Integrals.

Satz 4.14 *(Stetigkeit des Anfangswertproblems für GDGL)*
Seien die Voraussetzungen von 4.10 gegeben, 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

$$||f - \widetilde{f}||_\infty \leq \epsilon,\ ||y_0 - \widetilde{y}_0|| \leq \widetilde{\epsilon}.$$

Sei bekannt, dass beide Probleme Lösungen y bzw. $\widetilde{y}$ in einem Intervall $[a,b]$ besitzen. Dann gilt

$$||\widetilde{y}(t) - y(t)|| \leq (\tilde{\epsilon} + \epsilon(t-a))\exp(L(t-a))\forall t \in [a,b].$$

Beweis: Mit $u(t) := ||\widetilde{y}(t) - y(t)||$ gilt

$$\begin{aligned} u(t) &= ||\widetilde{y}_0 - y_0 + \int_a^t \widetilde{f}(s,\widetilde{y}(s)) - f(s,y(s))\mathsf{ds}|| \\ &\leq \widetilde{\epsilon} + \int_a^t (\epsilon + ||f(s,\widetilde{y}(s)) - f(s,y(s)))||\mathsf{ds} \\ &\leq \underbrace{\widetilde{\epsilon} + \epsilon(t-a)}_{\alpha(t)} + \underbrace{L}_{\beta} \int_a^t u(s)\mathsf{ds}. \end{aligned}$$

Anwendung von 4.13 liefert das Gewünschte:

$$\begin{aligned} u(t) &\leq (\widetilde{\epsilon} + \epsilon(t-a)) + L\int_a^t \underbrace{(\widetilde{\epsilon} + \epsilon(s-a))}_{\leq \widetilde{\epsilon} + \epsilon(t-a)} \exp(L(t-s))\mathsf{ds} \\ &\leq (\widetilde{\epsilon} + \epsilon(t-a))(1 + [\exp(L(t-s))]_x^a) \\ &\leq (\widetilde{\epsilon} + \epsilon(t-a))\exp(L(t-a)). \end{aligned}$$

□

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.
Abschließend machen wir noch zwei Bemerkungen zur Lage der Lösungen.

Lemma 4.15 *Sei wieder $|f|$ durch M nach oben beschränkt (nicht notwendig Lipschitz–stetig), y eine Lösung des Anfangswertproblems, und sei G ein Intervall. Weiter sei*

$$K_M(a, y_0) = \{(t, y) \in [a, b] \times \mathbb{R}^n : ||y - y_0||_\infty \leq M|t - a|\}.$$

Dann ist der Graph von y ganz in $K_M(a, y_0)$ enthalten.

Beweis: Mit dem Mittelwertsatz gilt

$$|y(t) - y_0| = |y'(\xi)(t - a)| \leq M|t - a|.$$

□

Die Lösung kann also insbesondere auf dem Intervall $[a, b]$ nicht gegen ∞ gehen, die Situation aus Abbildung 4.2 ist damit ausgeschlossen. Man kann also vermuten, dass die AWA auf dem ganzen Intervall $[a, b]$ eine Lösung besitzt, wenn K_M im Definitionsgebiet von f enthalten ist. Tatsächlich gilt

Satz 4.16 *(Satz von Peano)*
K_M liege in G. Dann hat die AWA eine Lösung auf $[a, b]$.

Beweis: Neu ist hier, dass man in diesem Fall die Lipschitzstetigkeit nicht benötigt. Hinter dem Beweis steht der Satz von Arzela–Ascoli (Walter).
Für lipschitzstetiges f ist das einfach nur der globale Picard–Lindelöf. □

4.5 Numerische Lösungen: Einschrittverfahren

Sei nach diesen Vorbemerkungen nun immer

$$y'(t) = f(t, y(t)),\ y(a) = y_0$$

eine Anfangswertaufgabe. f sei immer lipschitzstetig im zweiten Argument mit Lipschitzkonstante L, $|f|$ sei durch M nach oben beschränkt. $K_M(a, y_0)$ sei immer ganz im Definitionsgebiet von f enthalten.

Im Abschnitt über grafische Lösungen haben wir bereits ein einfaches Lösungsverfahren kennengelernt: Starte am Punkt $(t_0 = a, y_0)$. Verfolge die Lösungskurve $(t, y(t))$ in Richtung der Tangente mit dem Richtungsvektor $(1, y'(t_0)) = (1, f(t_0, y_0))$. Wir landen an einem Punkt

$$(t_1, y_1) = (t_0, y_0) + h_0(1, f(t_0, y_0)) = (t_0 + h, y_0 + hf(x_0, y_0)) =: (t_1, y_1).$$

Dies lässt sich iterieren, wir bekommen das Eulersche Polygonzugverfahren

$$y_{k+1} = y_k + h_k f(t_k, y_k), h_k = t_{k+1} - t_k$$

für eine Unterteilung (t_k) des Intervalls $[a, b]$. y_k ist eine diskrete Approximation an $y(t_k)$. Hierbei könnte (t_k) fest sein (z.B. äquidistant) oder im Verlauf des Verfahrens gewählt werden (Schrittweitensteuerung).
Bemerkung: Hier sieht man auch, wofür wir die Kegelbedingung benötigen: Sie stellt gerade sicher, dass f auf allen Iterierten definiert ist.

Definition 4.17 *Sei $I_h = \{t_0, \ldots, t_N\} \subset I$, $t_{k+1} > t_k$, $t_0 = a$, $t_N = b$, $h_k = t_{k+1} - t_k$, $k = 0 \ldots N - 1$. Dann heißt I_h ein (zulässiges) Gitter zur Lösung der AWA. $h = \max_k h_k$ heißt Feinheit des Gitters. Numerische Verfahren bestimmen geeignete Gitter und eine diskrete Näherung $y_h : I_h \mapsto \mathbb{R}$. Statt $y_h(t_k)$ benutzen wir häufig, wenn sie eindeutig ist, die verkürzte Schreibweise $y_k = y_h(x_k)$.*

Definition 4.18 *Sei y_h diskrete Näherung für eine AWA auf dem Gitter I_h, y die Lösung. Dann heißt*

$$e_h : I_H \mapsto \mathbb{R}^n,\ e_h(t) = y(t) - y_h(t)$$

die Fehlerfunktion der Näherung.

$$||e_h||_\infty = \max_{t \in I_h} ||e_h(t)||$$

heißt globaler Diskretisierungsfehler.

Definition 4.19 *Sei I_h eine Folge von Gittern mit Feinheit h, denen ein numerisches Verfahren die Näherungen u_h zuordnet. Das Verfahren heißt konvergent, falls der globale Diskretisierungsfehler mit h gegen 0 geht. Also:*

$$||e_h||_\infty \mapsto_{h \mapsto 0} 0.$$

Das Verfahren heißt konvergent von der Ordnung p, falls

$$||e_h||_\infty = O(h^p).$$

Wir wollen numerische Verfahren entwickeln. Wegen

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\mathrm{dt}$$

wählen wir Verfahren der Form

$$y_{k+1} = y_k + h_k\varphi(t, y, h)$$

wobei $t = (t_j)$, $y = (y_j)$, $h = (h_j)$. φ heißt Verfahrensfunktion. $h_k\varphi$ sollte das Integral aus der analytischen Lösung approximieren (es gibt also einen engen Zusammenhang zwischen numerischer Integration und numerischer Lösung von gewöhnlichen Differentialgleichungen).
Wir unterscheiden vier Typen von Verfahrensfunktionen:

Definition 4.20 *(Typen von numerischen Verfahren)*

1. *Ein Verfahren heißt explizites Einschrittverfahren, falls*

$$y_{j+1} = y_j + h_j\varphi(t_j, y_j, h_j).$$

2. *Ein Verfahren heißt implizites Einschrittverfahren, falls*

$$y_{j+1} = y_j + h_j\varphi(t_j, y_j, y_{j+1}, h_j).$$

3. *Ein Verfahren heißt explizites Mehrschrittverfahren, falls*

$$y_{j+1} = y_j + h_j\varphi(t_0 \dots t_j, y_0 \dots y_j, h_0 \dots h_j).$$

4. *Ein Verfahren heißt implizites Mehrschrittverfahren, falls*

$$y_{j+1} = y_j + h_j\varphi(t_0 \dots t_{j+1}, y_0 \dots y_{j+1}, h_0 \dots h_{j+1}).$$

Bemerkung: Die expliziten Verfahren sind direkte Zuweisungen. Bei den impliziten Verfahren steht das y_{j+1} auf der linken und rechten Seite einer Gleichung, mit einem geeigneten numerischen Verfahren muss y_{j+1} hier also noch approximiert werden. Die Einschrittverfahren greifen nur auf die aktuelle Näherung zurück, die Mehrschrittverfahren auf alle bisher berechneten Werte.
In diesem Abschnitt werden wir zunächst auf die expliziten Einschrittverfahren eingehen. 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.

Definition 4.21 *(lokaler Diskretisierungsfehler und Konsistenz)*
Sei φ eine Verfahrensfunktion, $t \in [a,b]$, y eine Lösung der Differentialgleichung. Dann heißt

$$\tau_h(t,y(t)) := \frac{1}{h}(y(t+h) - (y(t) + h\varphi(t,y(t),h))),\ (t,y(t)) \in K_M$$

lokaler Diskretisierungsfehler. Der Wert in der Klammer ist dabei die Näherung für $y(t+h)$, die das Verfahren für die Anfangswerte $(t,y(t))$ liefert, wenn man also in das numerische Verfahren die exakte Lösung einsetzt.
Das durch φ definierte numerische Verfahren heißt konsistent, falls $|||\tau_h||_\infty \mapsto_{h\mapsto 0} 0$.
Das Verfahren heißt konsistent von der Ordnung p, falls

$$||\tau_h||_\infty = O(h^p).$$

Beispiel 4.22

Euler:
Das Eulersche Polygonzugverfahren ist ein explizites Einschrittverfahren mit der Verfahrensfunktion $\varphi(t,y,h) = f(t,y)$. Sei f stetig differenzierbar, dann ist, wegen $y' = f$, y zweimal stetig differenzierbar. Es gilt mit Taylorentwicklung

$$\begin{aligned}\tau_h(t,y(t)) &= \frac{y(t+h)-y(t)}{h} - f(t,y(t)) \\ &= \frac{y(t)+hy'(t)+h^2/2y''(\xi(h))-y(t)}{h} - y'(t) \\ &= \frac{h}{2}y''(\xi) \mapsto 0.\end{aligned}$$

Das Eulerverfahren ist konsistent von der Ordnung 1. Das Eulerverfahren benötigt eine Auswertung von f in jedem Schritt.

2. *Verbessertes Eulerverfahren: Ein verbessertes Verfahren ergibt sich, wenn wir die Mittelpunktsregel anwenden:*

$$\begin{aligned}\int_{t_k}^{t_k+h} f(t,y(t))dt &\sim h(f(t_k+h/2, y(t_k+h/2))) \\ &\sim h(f(t_k+h/2, y(t_k)+h/2y'(t_k)) \\ &\sim h(f(t_k+h/2, y_k+h/2f(t_k,y_k))).\end{aligned}$$

Falls f zweimal stetig differenzierbar ist, so ist das Verfahren konsistent von der Ordnung 2. Das Verfahren benötigt zwei Auswertungen von f in jedem Schritt.

3. *Verfahren von Heun:*
Wir können auch mit der Trapezregel integrieren, hierdurch ergibt sich das Verfahren von Heun. Wir approximieren für f zweimal stetig differenzierbar:

$$\begin{aligned}\int_{t_k}^{t_{k+1}} f(t,y(t))dt &\sim \frac{h}{2}(f(t,y(t))+f(t+h,y(t+h)))\\ &\sim \frac{h}{2}(f(t_k,y_k)+f(t_k+h,y_k+hf(t_k,y_k)))\end{aligned}$$

Wir bemerken zunächst, dass

$$y''(t) = f(t,y(t))' = f_t(t,y(t))+y'(t)f_y(t,y(t)) = f_t(t,y(t))+f(t,y(t))f_y(t,y(t)).$$

Der Einfachheit halber lassen wir im folgenden die Argumente weg. Wieder gilt mit Taylorentwicklung (für f in zwei Dimensionen)

$$\tau_h(t,y(t)) = \underbrace{\frac{y(t+h)-y(t)}{h}}_{y'+h/2y''+O(h^2)} - \frac{1}{2}\underbrace{(f(t,y(t))+f(t+h,y(t)+hf(t,y(t))))}_{y'+(y'+h(f_t+ff_y)+O(h^2))=2y'+hy''}$$

Das Verfahren ist also ebenfalls konsistent von der Ordnung 2 und benötigt zwei Auswertungen von f pro Schritt.

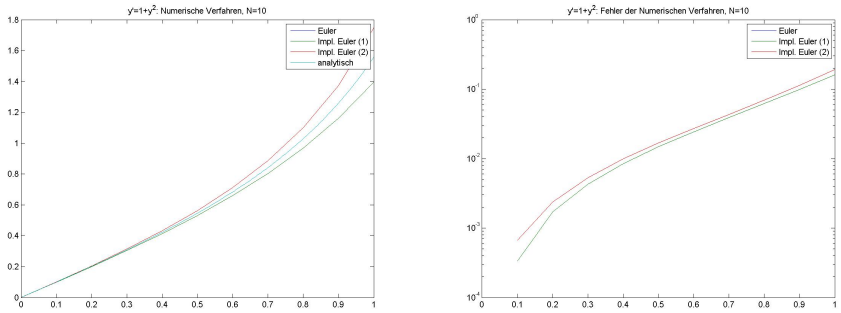


Abbildung 4.3: Einschrittverfahren. Der Kegel K_M ist in rot eingezeichnet.

Klick für Bild einschrittvef
Klick für Matlab Figure einschrittvef
Klick für Bild einschrittfehler
Klick für Matlab Figure einschrittfehler

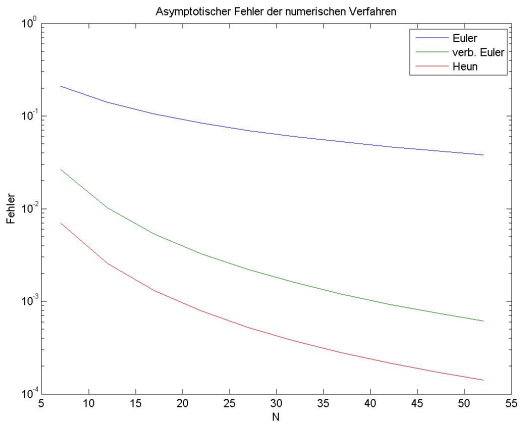


Abbildung 4.4: Asymptotische Entwicklung des Fehlers bei Einschrittverfahren

Klick für Bild gdgleinschrittasymp
Klick für Matlab Figure gdgleinschrittasymp

```
function out = euler( f,x,y,h,p )
%EULER
out=f(x,y);
end
```

Listing 4.2: Eulerverfahren (euler.m)

Klicken für den Quellcode von 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 4.3: Verb. Eulerverfahre (verbeuler.m)

Klicken für den Quellcode von verbeuler.m

```
function out = euler( f,x,y,h,p )
%Heun
y1=f(x,y);
out=1/2*(y1+f(x,y+h*y1));
```

Listing 4.4: Verfahren von Heun (heun.m)

Klicken für den Quellcode von 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);
```

Listing 4.5: Einschrittverfahren (einschritt.m)

Klicken für den Quellcode von einschritt.m

```
function einschrittdemo
N=10;
a=0;
b=1;
```

Listing 4.6: Demo Einschrittverf. (einschrittdemo.m)

Klicken für den Quellcode von einschrittdemo.m

Für Einschrittverfahren gilt der Satz: Aus Konsistenz folgt Konvergenz. Es ist 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 4.23 *(Diskretes Lemma von Gronwall)*
Seien (α_k), (β_k), $(e_k) \geq 0$ *reelle 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:

$$\begin{aligned} e_{k+1} &\leq \alpha_k + (1+\beta_k)(e_0 + \sum_{j=0}^{k-1} \alpha_j) \exp(\sum_{j=0}^{k-1} \beta_j),\ \beta_k \geq 0 \\ &\leq \alpha_k + (e_0 + \sum_{j=0}^{k-1} \alpha_j) \exp(\sum_{j=0}^{k} \beta_j),\ (e^{\beta_k} \geq 1+\beta_k) \\ &\leq (e_0 + \sum_{j=0}^{k} \alpha_j) \exp(\sum_{j=0}^{k} \beta_j),\ \beta_k \geq 0 \end{aligned}$$

□

Dies ist nicht die schärfste Form, aber die, die wir nutzen werden. Genauer gilt: Approximiert man im zweiten Lemma von Gronwall die Integrale durch Summen, kommt gerade die diskrete Form heraus (Übungen).

Satz 4.24 *(Konvergenz von expliziten Einschrittverfahren)*
Ein explizites numerisches Einschrittverfahren mit Verfahrensfunktion φ sei lipschitzstetig in der zweiten Variable y mit Lipschitzkonstante L und konsistent (von der Ordnung p). Dann ist das Verfahren auch konvergent (von der Ordnung p).

Beweis: Sei y Lösung der AWA. Sei $I_h = (t_k)$ ein zulässiges Gitter mit zugehöriger numerischer Approximation y_k. Dann gilt:

$$\begin{aligned}
|e_{k+1}| &= |y(t_{k+1}) - y_{k+1}| \\
&= |y(t_{k+1}) - (y(t_k) + h_k\varphi(t_k, y(t_k), h_k)) + y(t_k) - y_k \\
&\quad + h_k(\varphi(t_k, y(t_k), h_k) - \varphi(t_k, y_k, h_k))| \\
&\leq h_k|\tau_{h_k}(t_k, y(t_k))| + |e_k| + h_k L|e_k| \\
&= \underbrace{h_k|\tau_{h_k}(t_k, y(t_k))|}_{\alpha_k} + \underbrace{(1 + h_k L)}_{\beta_k}|e_k|.
\end{aligned}$$

Mit dem diskreten Lemma von Gronwall gilt also

$$\begin{aligned}
|e_k| &\leq \left(|e_0| + \sum_{j=0}^{k-1} h_j|\tau_{h_j}(t_j, y(t_j))|\right) \exp(L\sum_{j=0}^{k-1} h_j) \\
&= (|e_0| + (t_k - a)\max_j |\tau_{h_j}(t_j, y(t_j)))|) \exp\left(L(t_k - a)\right).
\end{aligned}$$

Also: Falls ein Verfahren konsistent ist (von der Ordnung p), so ist es konvergent (von der Ordnung p). Wir dürfen sogar noch zulassen, dass die Anfangswerte falsch sind (bis auf einen Fehler $O(h^p)$). □

Bemerkung: Die letzte Ungleichung zeigt, dass der Algorithmusfehler für Einschrittverfahren in der Größenordnung des unvermeidlichen Fehlers liegt, sie ist gerade eine Diskretisierung der Abschätzung 4.14. Einschrittverfahren sind also numerisch stabil.

Korollar 4.25 *Das Eulersche Polygonzugverfahren ist konvergent von der Ordnung 1. Das Verfahren von Heun und das verbesserte Eulerverfahren sind konvergent von der Ordnung 2.*

Wir werden sehen, dass implizite Verfahren sehr 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 4.26 *(Implizite Einschrittverfahren)*
Sei $\varphi(t_k, y_k, y_{k+1}, h_k)$ die Schrittfunktion eines impliziten Einschrittverfahrens. Sei φ lipschitzstetig bzgl. y_k und y_{k+1} mit der gemeinsamen Lipschitzkonstante L. Weiter sei φ stetig. Dann gibt es ein h_0, so dass die Gleichung

$$y = y_k + h\varphi(t_k, y_k, y, h)$$

für $h < h_0$ eindeutig auflösbar ist. Es gibt also eine Funktion $v(t_k, y_k, h_k)$, so dass

$$y_{k+1} = y_k + h\varphi(t_k, y_k, v(t_k, y_k, h), h).$$

v ist lipschitzstetig in der zweiten Variablen.

Beweis: Wir zeigen, dass die rechte Seite eine Selbstabbildung und kontrahierend ist, dann folgt die Wohldefiniertheit aus dem Banachschen Fixpunktsatz.
Der Kegel K_M liegt ganz in G, K_M ist kompakt, G ist offen, also ex. $\epsilon > 0$, so dass die ϵ–Umgebung von K_M bzgl. y ganz in $[a, b] \times G$ liegt. φ ist stetig, hat also auf dieser abgeschlossenen Umgebung für $h <= (b - y_k)$ ein Maximum P.
Sei h_0 so klein, dass

$$Lh_0 < 1 \text{ und } Ph_0 < \epsilon$$

und $h < h_0$. Wir setzen

$$g : [y_k - \epsilon, y_k + \epsilon] \mapsto [y_k - \epsilon, y_k + \epsilon],\ g(y) := y_k + h\varphi(t_k, y_k, y, h_k).$$

Zu zeigen ist: g ist wohldefiniert und kontrahierend. Sei $y \in [y_k - \epsilon, y_k + \epsilon]$.

$$\begin{aligned} |g(y) - y_k| &= h|\varphi(t_k, y_k, y, h)| \\ &\leq Ph < \epsilon. \end{aligned}$$

Weiter gilt für $z,\ y \in [y_k - \epsilon, y_k + \epsilon]$

$$|g(z) - g(y)| \leq Lh|z - y|.$$

Wegen $Lh < 1$ ist also g kontrahierend, die Fixpunktiteration für g konvergiert gegen den eindeutig bestimmten Fixpunkt.
Zu zeigen ist noch: v ist lipschitzstetig in y_k.
Sei $u_1 = v(t_k, y_k, h)$, $u_2 = v(t_k, z_k, h)$. Dann gilt

$$\begin{aligned} |u_1 - u_2| &= |y_k + h\varphi(t_k, y_k, u_1, h) - (z_k + h\varphi(x, z_k, u_2, h))| \\ &\leq |y_k - z_k| + h|\varphi(t_k, y_k, u_1, h) - \varphi(t_k, z_k, u_1, h) + \varphi(t_k, z_k, u_1, h) - \varphi(t_k, z_k, u_2, h)| \\ &\leq |y_k - z_k| + Lh|y_k - z_k| + Lh|u_1 - u_2|. \end{aligned}$$

also wegen $Lh < 1$

$$|v(t_k, y_k, h) - v(t_k, z_k, h)| = |u_1 - u_2| \leq \frac{1 + Lh}{1 - Lh}|y_k - z_k|.$$

Also ist v lipschitzstetig in der zweiten Variablen mit Lipschitzkonstante

$$L' = (1 + Lh)/(1 - Lh).$$

□

Implizite Einschrittverfahren sind also explizite Verfahren, nur etwas anders aufgeschrieben. Insbesondere gilt der Konvergenzsatz auch für implizite Verfahren:

Korollar 4.27 *Implizite konsistente Einschrittverfahren (von der Ordnung p) sind konvergent (von der Ordnung p).*

Beweis: Implizite sind spezielle explizite Verfahren, also gilt 4.24. □

Beispiel 4.28

1. *Implizites Eulerverfahren (konvergent von der Ordnung 1):*

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1})$$

2. *Trapezregel (konvergent von der Ordnung 2):*

$$y_{k+1} = y_k + h_k \frac{1}{2}(f(t_k, y_k) + f(t_{k+1}, y_{k+1})).$$

Insbesondere folgt aus dem Satz, dass die Fixpunktiteration gegen y_{k+1} konvergiert. Typischerweise berechnet man also bei impliziten Verfahren einfach mehrere Folgeglieder der Fixpunktiteration mit Startwert y_k.

Beispiel 4.29 *Fixpunktiteration für das implizite Eulerverfahren*
Wir wenden das implizite Eulerverfahren an und lösen die Definitionsgleichung durch Fixpunktiterationen. Führen wir nur einen Iterationsschritt durch, so erhalten wir

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_k),$$

also (fast) das Euler–Verfahren. Führen wir zwei Iterationsschritte 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 dann zwei Auswertungen von f pro Schritt, aber das Verfahren ist nur von der Ordnung 1.

Die Kontraktionskonstante im Fixpunktsatz von Banach war Lh. Wir gewinnen also in jedem Schritt der Fixpunktiteration einen Faktor Lh, 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 $O(h^p)$. Daher:

Satz 4.30 *(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 Konvergenz. □

Bemerkung: Hierdurch wird natürlich das implizite zu einem expliziten Verfahren (wie im Beweis zum Konvergenzsatz). 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 4.5 wird dies deutlich: Zusätzliche Iterationen verkleinern den Fehler, aber nicht asymptotisch.

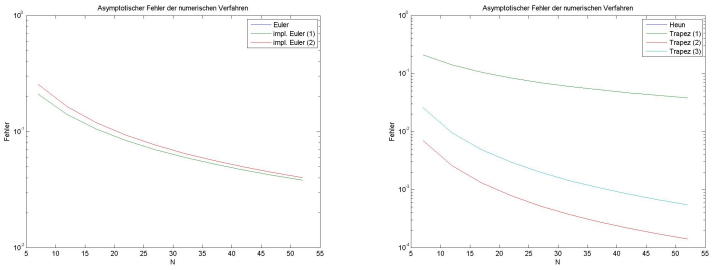


Abbildung 4.5: Fehler der impliziten Verfahren, p Schritte der Fixpunktiteration

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
```

Listing 4.7: Implizite Trapezregel (trapez.m)

Klicken für den Quellcode von trapez.m

```
function out = impliciteuler( f,x,y,h,p )
%IMPLICITEULER
y1=y;
for i=1:p
```

Listing 4.8: Implizites Eulerverfahren (impliciteuler.m)

Klicken für den Quellcode von impliciteuler.m

Bemerkung: In diesem Kapitel haben wir der Einfachheit halber häufig Bezeichnungen und Implementationen für $n = 1$ gewählt. Alle Beweise und Sätze gelten ohne Änderung auch für Systeme von GDGL.

4.6 Konstruktion von Einschrittverfahren

Wir konstruieren nun Schrittfunktionen φ. Schon bei der Definition der Schrittfunktion hatten wir gesehen, dass möglichst

$$h\varphi(t_k, y(t_k), h_k) = \int_{t_k}^{t_{k+1}} f(s, y(s))\mathrm{ds}$$

gelten sollte. Wir suchen also numerische Näherungsformeln für dieses Integral. Zunächst zeigen wir mit Hilfe der einfachen Taylorverfahren, dass es Einschrittverfahren beliebig hoher Ordnung gibt.

4.6.1 Taylor–Verfahren

Satz 4.31 *(Taylor–Verfahren)*
Sei $\widetilde{y}(x)$ die Lösung der AWA $\widetilde{y}'(t) = f(t, \widetilde{y}(t))$, $\widetilde{y}(t_k) = y_k$. Sei $f \in C^p$, und alle

Ableitungen seien lipschitzstetig im zweiten Argument mit der gemeinsamen Lipschitzkonstante L. Wir definieren die Schrittfunktion

$$\varphi_p(t_k, y_k, h) = \sum_{k=1}^{p} \frac{h^{k-1}}{k!} \widetilde{y}^{(k)}(t_k) = \sum_{k=1}^{p} \frac{h^{k-1}}{k!} \left(\frac{d}{dt}\right)^{k-1} f(t, \widetilde{y}(t))|_{(t_k, y_k)}.$$

Dann ist das zugehörige numerische Verfahren konvergent von der Ordnung p.

Beweis: Für $y_k = y(t_k)$ ist $\widetilde{y} = y$. Mit Taylorentwicklung gilt

$$\frac{y(t_k + h) - y(t_k)}{h} - \varphi_p(t_k, y(t_k), h) = O(h^p)$$

und φ_p ist lipschitzstetig nach Voraussetzung an die Ableitungen. □

Es sieht zunächst so aus, als sei das nicht sehr nützlich: Das in 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 mit Hilfe von f direkt berechnen.

Beispiel 4.32 *(Taylor–Verfahren, $p = 2$)*
Sei

$$f(t, y) = (1 + y^2).$$

Wegen

$$\frac{d}{dt} f(t, \widetilde{y}(t)) = f_t + \widetilde{y}' f_y = f_t + f \cdot f_y$$

gilt

$$\begin{aligned} \varphi(t_k, y_k, h) &= f(t_k, y_k) + \frac{h}{2}(f_t(t_k, y_k) + f(t_k, y_k) f_y(t_k, y_k)) \\ &= (1 + y_k^2) + \frac{h}{2}(1 + y_k^2)(2y_k) \\ &= (1 + y_k^2)(1 + hy_k). \end{aligned}$$

Das Problem der Taylor–Verfahren ist, dass hohe Ableitungen von f möglichst explizit ausgerechnet werden müssen. Dies ist häufig nicht möglich, etwa wenn f nicht exakt 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 zumindest optional implementiert sind.

4.6.2 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:

$$\begin{array}{rcl} f_1 & = & f(t_k, y_k) \\ f_2 & = & f(t_k + h_k, y_k + h f_1) \\ \hline \varphi(t_k, y_k, h_k) & = & \frac{1}{2} f_1 + \frac{1}{2} f_2 \end{array}$$

Diese Schreibweise legt die folgende Definition nahe.

Definition 4.33 *Runge–Kutta–Verfahren*

1. *Seien α_k, γ_k, β_{kl} fest gewählt, $k = 1 \ldots m$, $l = 1 \ldots k-1$. Die Schrittfunktion φ sei definiert durch*

   $$\varphi(t_k, y_k, h_k) = \gamma_1 f_1 + \gamma_2 f_2 + \ldots \gamma_m f_m$$

   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 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

$$\begin{aligned} f_1 &= f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m} \beta_{1,l} f_l) \\ f_2 &= f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m} \beta_{2,l} f_l) \\ &\vdots \\ f_m &= f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m} \beta_{m,l} f_l). \end{aligned}$$

Dann heißt das zugehörige numerische Verfahren implizites Runge–Kutta–Verfahren.

Satz 4.34 *(Ordnung der Runge–Kutta–Verfahren)*

1. *Sei*
$$\sum_{k=1}^{m} \gamma_k = 1.$$
Genau 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*
$$\alpha_k = \sum_l \beta_{kl} \text{ und } \sum_{k=1}^{n} \alpha_k \gamma_k = \frac{1}{2}.$$
Dann ist das zugehörige Runge–Kutta–Verfahren mindestens konvergent von der Ordnung 2.

Beweis: Taylorentwicklung. □

Üblicherweise werden Runge–Kutta–Verfahren durch die Butcher Diagramme repräsentiert in der Form (hier für explizite Verfahren)

$$\begin{array}{c|ccccc} \alpha_1 & & & & & \\ \alpha_2 & \beta_{2,1} & & & & \\ \alpha_3 & \beta_{3,1} & \beta_{3,2} & & & \\ \vdots & & & & & \\ \alpha_m & \beta_{m,1} & \beta_{m,2} & \dots & \beta_{m,m-1} & \\ \hline & \gamma_1 & \gamma_2 & \dots & \gamma_{m-1} & \gamma_m \end{array}$$

Für implizite Verfahren ist das Diagramm natürlich komplett ausgefüllt.

Beispiel 4.35 *(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. Man kann zeigen, dass diese Verfahren mit derselben Konsistenzordnung auf alle Gleichungssysteme anwendbar sind (siehe unten). Leider entstehen schon dabei nichtlineare (bilineare) Gleichungssysteme. Wir untersuchen dies am Beispiel $m = 2$.
Wir wollen ein Verfahren möglichst hoher Ordnung konstruieren und wählen deshalb die α_k gleich wie in 4.34. Sei f 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 (ohne die bekannten Argumente)

$$\begin{aligned} \varphi(t_k, y_k, h_k) &= \gamma_1 f_1 + \gamma_2 f_2 \\ &= \gamma_1 f(y_k) + \gamma_2 f(y_k + \beta_{2,1} h f(y_k)) \\ &= \gamma_1 f + \gamma_2 (f + \beta_{2,1} h_k f f' + \frac{\beta_{2,1}^2 h^2 f^2}{2} f'' + O(h^3)) \end{aligned}$$

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, d.h. es gibt kein explizites Runge–Kutta–Verfahren der Ordnung 3 für alle $f \in C^3$. 2 ist also die maximale Konvergenzordnung eines expliziten 2–Schritt Runge–Kutta–Verfahrens.
Die maximale Konvergenzordnung eines expliziten p–Schritt–Verfahrens ist p. Leider wird diese Schranke nicht erreicht. Die folgende Tabelle gibt die höchste Konsistenzordnung eines Runge–Kutta–Verfahrens 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 4.36 *(Konstruktion von impliziten Verfahren)*
Gegeben sei ein numerisches Integrationsverfahren, das Polynome vom Grad p korrekt integriert. Dann gibt es ein Runge–Kutta–Verfahren, das dieselben Auswertungspunkte (α_k) und Gewichte (γ_k) benutzt und die Ordnung $(2p)$ hat.

Bemerkung: Die Konstruktion ist leider nicht immer so einfach wie bei der Trapezregel.
Wählen wir als Integrationsregel die Gauss–Integration, so gilt sofort:

Korollar 4.37 *(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 4.38 *(Taylorentwicklung des Fehlers 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*

$$\tau_h(t, y) = h^p F_p(x, y) + \ldots + h^{q-1} F_{q-1}(x, y) + O(h^q).$$

Beweis: Taylorentwicklung von τ_h wie im Beispiel. Die F_p sind einfach Summen von Produkten von Ableitungen von f. □

Satz 4.39 *(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. □

Bemerkung: Falls die Voraussetzungen von Satz 4.34 erfüllt sind, so sind die skalaren Runge–Kutta–Verfahren der Ordnung p auch konsistent von der Ordnung p für Systeme von Differentialgleichungen.
Diese Bemerkung ermöglicht uns den Beweis der vorher gemachten Bemerkung über autonome Systeme.

Satz 4.40 *(Umwandlung von Gleichungen in Systeme von autonomen Gleichungen)*
Jede skalare Differentialgleichung der Form $y'(t) = f(t, y(t))$ *lässt sich schreiben als System von autonomen Differentialgleichungen*

$$(y_1'(t), y_2'(t)) = \tilde{f}(y_1(t), y_2(t)).$$

Beweis: Wähle $\tilde{f}_1(y_1(t), y_2(t)) = 1$, $\tilde{f}_2(y_1(t), y_2(t)) = f(y_1(t), y_2(t))$, $y_1(a) = a$, $y_2(a) = y_0$. Dann folgt $y_2(t) = y(t)$ aus $y_1(t) = t$. □

4.7 Extrapolation und Schrittweitensteuerung für Einschrittverfahren

Abschließend wollen wir, wie schon bei der Integration, mit Neville/Richardson/Romberg die Konsistenzordnung von Verfahren erhöhen und Näherungen den globalen Diskretisierungsfehler bestimmen. Wieder benötigen wir dazu zunächst einen Satz über die Taylorentwicklung des globalen Diskretisierungsfehlers. Für den lokalen Fehler haben wir das bereits gezeigt, die Vermutung ist, dass sich wieder mit Hilfe von Gronwall der entsprechende Satz für den globalen Diskretisierungsfehler zeigen lässt. Das ist auch tatsächlich so, allerdings nutzen wir zum Beweis den Konvergenzsatz für das Eulerverfahren.

Satz 4.41 *Gegeben sei ein Einschrittverfahren mit Verfahrensfunktion φ und 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.

$$\begin{aligned} e_h(t_{k+1}) &= y_h(t_{k+1}) - y(t_{k+1}) \\ &= y_h(t_k) + h_k\varphi(t_k, y_h(t_k), h_k) \\ &\quad -(y(t_k) + h_k\varphi(t_k, y(t_k), h_k) + \tau(t_k)h_k^{p+1} + O(h_k^{p+2})) \\ &= e_h(t_k) + h_k e_h(t_k)\varphi_y(t_k, y(t_k), h_k) - \tau(t_k)h_k^{p+1} + O(|e_h(t_k)|^2 h_k + h_k^{p+2}). \end{aligned}$$

Mit Konsistenz und $\varphi \in C^2$ gilt

$$\frac{y(t_k + h_k) - y(t_k)}{h_k} - \varphi(t_k, y(t_k), h_k) = f(t_k, y(t_k)) - \varphi(t_k, y(t_k), 0) + O(h_k) \mapsto 0$$

und damit

$$\begin{aligned}\varphi_y(t_k, y(t_k), h_k) &= \varphi_y(t_k, y(t_k), 0) + O(h_k) \\ &= f_y(t_k, y(t_k)) + O(h_k) \\ &= \rho(t_k) + O(h_k)\end{aligned}$$

und

$$e_h(t_{k+1}) = e_h(t_k) + h_k e_h(t_k)\rho(t_k) - h_k^{p+1}\tau(t_k) + O(h_k|e_h(t_k)|^2 + h_k^2|e_h(t_k)| + h_k^{p+2})$$

Sei nun $e_h(t_k) = \overline{e}_h(t_k)h_k^p$. Eingesetzt in die Gleichung und mit h_k^{-p} multipliziert

$$\overline{e_h(t_{k+1})} = \overline{e_h(t_k)} + h_k(\rho(t_k)\overline{e_h(t_k)} - \tau(t_k)) + O(h_k^2 + h_k^{p+1}|\overline{e}_h(t_k)|^2 + |\overline{e}_h(t_k)|h_k^2).$$

Dies ist (fast) das Eulerverfahren für die oben angegebene Differentialgleichung! Bis auf einen Fehler $O(h^2)$, der bei der Berechnung des lokalen Diskretisierungsfehlers aber keine Rolle spielt. Also ist das Verfahren konvergent 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.

Korollar 4.42 *(Extrapolation und Fehlerabschätzung)*

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. Die muss natürlich 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. Die folgende, heuristische Betrachtung ist daher häufig sehr erfolgreich:
Gegeben seien zwei explizite Runge–Kutta–Verfahren, die dieselben Funktionsauswertungen benutzen (d.h. dieselben α_k und β_k), mit den Ordnungen $p_1 > p_2$, die Näherungen y_1 und y_2 für $y(b)$ liefern. Dann ist $|y_2 - y_1|$ eine Schätzung für den Fehler von y_2. Entsprechend ist $h^{p_2-p_1}$ eine Schätzung für den Fehler von y_1.
Ein Beispiel für ein Paar von Runge–Kutta–Verfahren mit gleichen Auswertungen und unterschiedlichen Konvergenzordnungen ist 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.

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.
- Konvergenz: Das Verfahren liefert für Gittergüte gegen o 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 gelten sie als problematisch (warum, werden wir sehen), zusätzlich ist eine Schrittweitensteuerung schwierig. Daher sind sie häufig nicht die Standardsolver. In Matlab steht der Adams–Bashforth–Solver unter dem Namen ode113 zur Verfügung.

Wir werden in diesem Kapitel zunächst einige Verfahren herleiten, die Bezeichnungen auf Mehrschrittverfahren (MSV) übertragen und überprüfen, welche Sätze erhalten bleiben. Zunächst schränken wir die Betrachtung auf lineare Verfahren auf äquidistanten Gittern ein. Wir reduzieren die Definition aus 4.20 auf

Definition 5.1 *Ein numerisches Verfahren, dass 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 äquidistantes lineares m–Schritt–Mehrschrittverfahren. Das Verfahren heißt explizit, falls $\beta_m = 0$, ansonsten implizit.

Beispiel 5.2 *(Mittelpunktsregel)*

$$y_{k+2} - y_k = h f(x_{k+1}, y_{k+1})$$

Die Definition der Konsistenz können wir von den Einschrittverfahren sinngemäß übernehmen.

Definition 5.3 *Gegeben sei ein durch die Konstanten α_j und β_j beschriebenes lineares MSV. Dann heißt*

$$\tau_h(t) = \frac{1}{h} \sum_{j=0}^{m} \alpha_j y(t + jh) - \sum_{j=0}^{m} \beta_j f(t + jh, y(t + jh))$$

lokaler Diskretisierungsfehler.

Zur Herleitung der Standard–Formeln benutzen wir eine Kurzschreibweise, die eng mit den dividierten Differenzen zusammenhängt, und die uns erlaubt, Interpolationspolynome auf äquidistanten Gittern besonders einfach anzugeben. Unsere Idee wird (natürlich) sein, ein Interpolationspolynom durch die bereits berechneten Werte zu legen, und dadurch den nächsten Wert einfach angeben zu können.

Definition 5.4 *Sei (y_k) eine Folge reeller Zahlen. Dann heißt $(Ty) = (0, y_0, y_1, \dots)$ die Vorgängerfunktion und*

$$(\nabla y) = (I - T)y = (y_0, y_1 - y_0, y_2 - y_1, \dots)$$

die Rückwärts–Differenz.

Beispiel 5.5

$$(\nabla^2 y)_k = \nabla(y_0, y_1 - y_0, y_2 - y_1, \ldots) = (y_0, y_1 - 2y_0, y_2 - 2y_1 + y_0, \ldots)$$

also

$$(\nabla^2 y)_k = y_k - 2y_{k-1} + y_{k-2},\ k \geq 2.$$

Lemma 5.6 *(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. $T^m = (I - \nabla)^m$. □

Satz 5.7 *Sei $t_k = a + kh$, y_k gegeben für $k \in \mathbb{N}$. Für $k \geq m$ definieren wir*

$$p_k(t) = \sum_{j=0}^{m} (-1)^j \begin{pmatrix} -s \\ j \end{pmatrix} (\nabla^j y)_k,\ s = \frac{t - t_k}{h}.$$

Dann ist p_k das eindeutig bestimmte Interpolationspolynom in $\mathcal{P}_m$ mit

$$p_k(t_{k-j}) = y_{k-j},\ j = 0 \ldots m$$

mit der üblichen Definition

$$\begin{pmatrix} t \\ j \end{pmatrix} = \frac{t(t-1)\cdots(t-j+1)}{j!}.$$

Beweis:

1. p_k ist Polynom vom Grad $\leq m$, denn die Binomialkoeffizienten sind in $\mathcal{P}_j$.
2. Sei $l \leq m$. Wegen
$$\begin{pmatrix} (t_k - t_{k-l})/h \\ j \end{pmatrix} = \begin{pmatrix} l \\ j \end{pmatrix}$$

gilt

$$\begin{aligned} p_k(t_{k-l}) &= \sum_{j=0}^{m}(-1)^j \binom{l}{j} (\nabla^j y)_k \\ &= \sum_{j=0}^{l}(-1)^j \binom{l}{j} (\nabla^j y)_k \text{ denn der BK ist o für } j > l \\ &= y_{k-l}. \end{aligned}$$

□

5.2 Konstruktion von MSV durch Integration

Für die Lösung unserer GDGL gilt

$$y(t_{k+m}) - y(t_{k+m-r}) = \int_{t_k+m-r}^{t_{k+m}} f(t, y(t))\text{dt}.$$

Für die Approximation ersetzen wir die Funktion unter dem Integralzeichen durch sein Interpolationspolynom p_k mit den Stützwerten $f_k = f(t_k, y_k)$. Wir erhalten damit

$$\begin{aligned} y_{k+m} - y_{k+m-r} &= \int_{t_{k+m-r}}^{t_{k+m}} p_k(t)\text{dt} \\ &= h \sum_{j=0}^{m-1} \underbrace{(-1)^j \int_0^r \binom{-s}{j} \text{ds}}_{\gamma_j} \nabla^j f_{k+m-1} \end{aligned}$$

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 auf/r	$r = 1$	$r = 2$	
$x_k \dots x_{k+m-1}$	Adams–Bashforth	Nyström	explizit
$x_k \dots x_{k+m}$	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 5.8 *(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:

$$\tau_h(t_{k+m}) = \frac{1}{h} \int_{t_{k+m-r}}^{t_{k+m}} f(t) - p(t) \mathsf{dt}$$

und dann mit 3.2. □

Bemerkung: Dies ist ein phantastisches Ergebnis: Mit nur einer zusätzlichen Auswertung von f können beliebig hohe Konsistenzordnungen erreicht werden! **Bemerkung:** Alternativ können MSV auch durch Differentiation des Interpolationspolynoms hergeleitet werden (s. Übungen).
Bemerkung: Offensichtlich können die MSV nur zur Berechnung der y_k mit $k \geq m$ eingesetzt werden. Die Berechnung ist daher immer zweistufig: Zunächst werden mit einem Einschrittverfahren die ersten $m-1$ Werte berechnet, ab dort setzt das Mehrschrittverfahren ein.

5.3 Stabilität von Mehrschrittverfahren

Beispiel 5.9 *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.

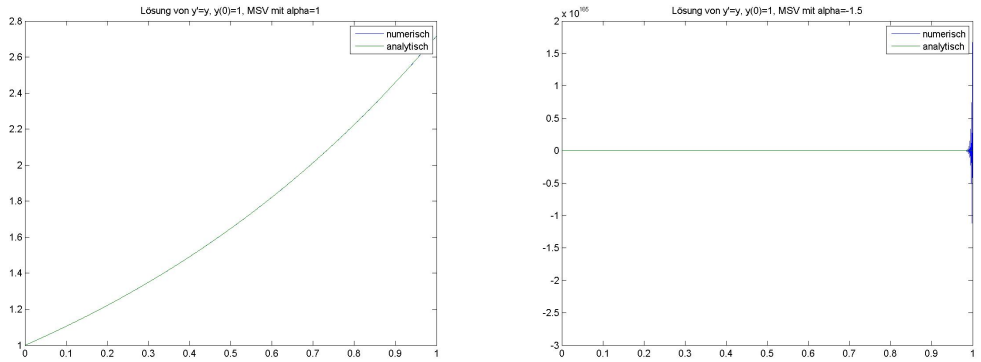


Abbildung 5.1: (In)stabilität von Mehrschrittverfahren

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
```

Listing 5.1: Instabilität von MSV (msvdemo.m)

Klicken für den Quellcode von 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 leicht durch einen kleinen Ausflug in die Theorie der linearen Differenzengleichungen.

5.3.1 Lineare Differenzengleichungen

Definition 5.10 *Sei $y = (y_0, y_1, \ldots)$, $y_k \in \mathbb{C}$. y heißt Lösung der linearen Differenzengleichung zu $a_0, \ldots, a_m$ und (b_k), falls*

$$a_0 y_{k+0} + a_1 y_{k+1} + \ldots + a_m y_{k+m} = b_k,\ k \geq 0$$

mit $a_0, \ldots, a_m \in \mathbb{C}$, $a_m \neq 0$, $b = (b_0, b_1, \ldots)$, $b_k \in \mathbb{C}$. Falls $b_k = 0$ für alle k, so heißt die Gleichung homogen, ansonsten inhomogen.

$$\rho(x) = \sum_{j=0}^{m} a_j x^j$$

heißt charakteristisches Polynom der Gleichung.

Beispiel 5.11 *(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 homogene vorgegebene 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 $y^{(k)}$ die Folge im Lösungsraum mit $y_j^{(k)} = \delta_{kj}$, $j = 0 \ldots m-1$. Diese Folgen bilden die kanonische Basis zu $a_0, \ldots, a_m$ des

Lösungsraums nach der ersten Bemerkung.
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 $a_m = 1$.

Lemma 5.12 *(Basislösungen für inhomogene Differenzengleichungen)*
Sei

$$z^{(l)} = T^{l-1} y^{(m-1)}, \text{ also } z_k^{(l)} = y_{k-l}^{(m-1)}.$$

Dabei ist $y^{(m-1)}$ Element der kanonischen Basis zu $a_0, \ldots, a_m$. Dann ist

$$\sum_{j=0}^{m} a_j z_{k+j}^{(l)} = \delta_{kl}$$

Beweis: Für $k < l$ stehen in in der Summe nur Nullen. Für $k > l$ erfüllt die Folge die Differenzengleichung. Für $k = l$ ist die Summe $a_m 1 = 1$ wegen $a_m = 1$. □

Satz 5.13 *Die Folge (z) sei definiert durch*

$$z = \sum_{l=0}^{\infty} f_l z^{(l)}.$$

Dann ist z Lösung der inhomogenen Differenzengleichung

$$\sum_{j=0}^{m} a_j z_{k+j} = f_k$$

Alle Lösungen z' der inhomogenen Differenzengleichung sind von der Form

$$z' = z + \sum_{l=0}^{m-1} z'_l y^{(l)}.$$

Beweis: Die Summe konvergiert, denn für das $k.$ Element der Folge bricht sie nach $k - m$ Summanden ab. Nach dem Lemma ist

$$\sum_{j=0}^{m} a_j z_{k+j} = \sum_{j=0}^{m} a_j f_k z_{k+j}^{(k)} = f_k.$$

Die Bemerkung zu z' folgt, weil $z - z'$ die homogene Differenzengleichung erfüllt und $z_0 \ldots z_{m-1} = 0$. □

Wir sehen also, dass die Lösungen der inhomogenen Differenzengleichung bereits durch die Lösungen der homogenen Gleichung bestimmt sind. Wir betrachten daher zunächst nur die homogenen Gleichungen.

Satz 5.14 *(Darstellung der Lösung von homogenen Differenzengleichungen)*
Seien $\lambda_0, \ldots, \lambda_r$ die Nullstellen des charakteristischen Polynoms ρ einer linearen, homogenen Differenzengleichung. Eine Folge (y_k) erfüllt genau dann die Differenzengleichung, falls es Polynome $q_j(k) \in \mathcal{P}_{m_j-1}$ gibt mit

$$y_k = \sum_{j=0}^{r} q_j(k)\lambda_j^k.$$

Beispiel 5.15 *(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

$$y_k = \frac{2}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k\right).$$

Beweis: Wir zeigen: Alle Lösung der Differenzengleichung haben die angegebene Form. Dann sind der Lösungsraum und der angegebene Raum aber gleich, denn sie haben dieselbe Dimension.
Sei zunächst $y_k = \lambda_l^k$. Wegen

$$\sum_{j=0}^{m} a_j y_{k+j} = \sum_{j=0}^{m} a_j \lambda_l^{k+j} = \lambda_l^k \rho(\lambda_l) = 0$$

ist damit y_k eine Lösung der Differenzengleichung. Im einfachen Fall, dass ρ nur unterschiedliche Nullstellen hat, sind wir damit bereits fertig. Andernfalls müssen wir etwas mehr arbeiten.
Sei wieder $a_m = 1$. Sei

$$Y_k \in \mathbb{C}^m,\ Y_k = \begin{pmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k+m-1} \end{pmatrix}$$

und damit

$$Y_{k+1} = \begin{pmatrix} y_{k+1} \\ y_k \\ \vdots \\ y_{k+m} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -a_0 & & \cdots & & -a_{m-1} \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k+m-1} \end{pmatrix}}_{Y_k}.$$

A heißt Begleitmatrix des Polynoms ρ, sein charakteristisches Polynom ist wieder ρ. Sei

$$A = XJX^{-1},\ J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_r \end{pmatrix}, J_l = \begin{pmatrix} \lambda_l & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_l \end{pmatrix}$$

die Jordan–Normalform von A mit den Jordan–Kästchen J_l. Dann ist

$$Y_k = A^k Y_0 = XJ^k X^{-1} Y_0.$$

Nachrechnen zeigt

$$J_l^k = \begin{pmatrix} \lambda_l^k & \alpha_{12} & \cdots & \alpha_{1m} \\ & \ddots & & \vdots \\ & & \ddots & \alpha_{m-1,m} \\ & & & \lambda_l^k \end{pmatrix},\ \alpha_{ij} = \underbrace{\left(\lambda_l^{j-i} \begin{pmatrix} k \\ j-i \end{pmatrix}\right)}_{\in \mathcal{P}_{j-i}} \lambda_l^k.$$

J^k enthält also nur Matrixeinträge der Form $q_j(k)\lambda_j^k$. Damit enthält $Y_k = XJ^kX^{-1}Y_0$ also nur Linearkombinationen dieser Einträge, also ist (y_k) von der angegebenen Form. Der von den Folgen dieser Form aufgespannte Teilraum hat die Dimension $\sum_j m_j = m$. □

Bemerkung: Alle Eigenräume der Begleitmatrix haben die Dimension 1 (Übungen).

5.3.2 Stabilitätssätze von Dahlquist

Satz 5.16 *(Wurzelbedingung von Dahlquist)*
Alle Lösungen einer homogenen linearen Differenzengleichung mit charakteristischem Polynom ρ sind beschränkt genau dann, wenn für die Nullstellen λ_i von ρ gilt

$$|\lambda_i| \leq 1,\ |\lambda_i| = 1 \Rightarrow \lambda_i \textit{ ist einfache Nullstelle.}$$

Beweis: Satz 5.14. □

Definition 5.17 *Ein Mehrschrittverfahren sei durch die Konstanten a_j und b_j definiert. Dann ist*

$$\rho(x) = \sum_{j=0}^{m} a_j x^j \text{ bzw } \sigma(x) = \sum_{j=0}^{m} b_j x^j$$

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 5.18 *Ein Mehrschrittverfahren sei von der allgemeinen Form*

$$\sum_{j=0}^{m} a_j y_{k+j} = h\varphi(t_k, y_k, \ldots, y_{k+m}, h) = h \sum_{j=0}^{m} b_j f(t_{k+j}, y_{k+j}).$$

Seien $u_0 \ldots, u_{m-1}$ berechnete Anfangswerte. Sei v eine Funktion auf I_h. Dann ist die Folge der Rechenfehler F_h definiert durch

$$\begin{aligned} F_h(v)_k &= v_k - u_k,\ k = 0 \ldots m-1. \\ F_h(v)_{k+m} &= \frac{1}{h} \sum_{j=0}^{m} a_j v_{k+j} - \varphi(t_k, v_k, \ldots, y_{v+m}, h),\ k \geq 0 \end{aligned}$$

(wie üblich mit $v_k = v(t_k)$ usw.).
Das Verfahren heißt (asymptotisch) stabil genau dann, wenn es ein $K > 0$ gibt, so dass für alle $v, w : I_h \mapsto \mathbb{C}$, $h < h_0$, gilt

$$||v - w||_\infty \leq K||F_h(v) - F_h(w)||_\infty$$

für alle Anfangswerte u_k, $k = 0 \ldots m-1$.

Bemerkung: Bei exakter Rechnung ist natürlich $F_h(u) = 0$. Dann gilt für stabile Verfahren

$$||u - v||_\infty \leq K||F(v)||_\infty,$$

der maximale Fehler im Ergebnis lässt sich also gegen den maximalen Fehler in der Rechnung abschätzen, was unserer alten Definition von numerischer Stabilität entspricht.
Der Satz von Dahlquist begründet wieder den Satz: Konsistenz plus Stabilität gleich Konvergenz.

Satz 5.19 *(Satz von Dahlquist)*
Die Anfangswerte $(u_h)_k$, $k = 0 \ldots m-1$, heißen konsistent von der Ordnung p, falls $(u_h)_k - y(a + kh) = O(h^p)$.

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 5.21).*

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 $y_k = k^{r-1}\lambda^k h$. y_k ist die vom Mehrschrittverfahren gelieferte Näherung für die Anfangswerte $y_0, \ldots, y_{p-1}$, denn die Folge ist Lösung der homogenen Differenzengleichung. 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 die Dahlquist–Bedingung erfüllt ist.
Zu 2. Sei u die Mehrschrittlösung, $y_k = y(t_k)$. Dann ist $F_h(y)$ der Konsistenzfehler, $F_h(u) = 0$ und damit

$$||y - u||_\infty \leq K||F_h(y) - F_h(u)||_\infty = ||F_h(y)||_\infty = O(h^p).$$

□

Lemma 5.20 *Sei ϵ_k eine Folge nichtnegativer Zahlen, $\epsilon_0 = 0$, und seien a, b positiv. Sei*

$$\epsilon_k \leq a + b\sum_{l=0}^{k-1} \epsilon_l.$$

Dann gilt

$$\epsilon_k \leq ae^{kb}.$$

Beweis: Mit Gronwall oder direkt per Induktion:

$$\begin{aligned}\epsilon_{k+1} &\leq a + b\sum_{l=0}^{k}\epsilon_l \\ &\leq a + ab\frac{e^{(k+1)b}-1}{e^b-1} \\ &\leq ae^{(k+1)b}, \text{ wegen } e^b > 1+b.\end{aligned}$$

□

Satz 5.21 *(Stabilität von Mehrschrittverfahren)*
Sei f lipschitzstetig in der zweiten Variablen. Gegeben sei ein lineares Mehrschrittverfahren mit charakteristischem Polynom ρ. Dann gilt:

1. *Falls das Verfahren stabil ist für $f = 0$, so erfüllt ρ die Wurzelbedingung.*
2. *Falls ρ die Wurzelbedingung erfüllt, so ist das Verfahren stabil.*

Beweis: Zu 1. Sei $f = 0$, also $\varphi = 0$. Sei u eine beliebige Lösung der homogenen Differenzengleichung

$$\sum_{j=0}^{m} a_j u_{k+j} = 0$$

Dann gilt wegen der Stabilität für $f = 0$ für die konstante Folge $\mathbf{0}_k = 0$

$$||u - \mathbf{0}||_\infty \leq K||F_h(u) - F_h(\mathbf{0})||_\infty = K \max_{i=0\ldots m-1}|u_i|$$

mit einer von h unabhängigen Konstanten K. Also sind alle Lösungen der homogenen Differenzengleichung beschränkt, und ρ erfüllt die Wurzelbedingung.
Zu 2. Erfülle nun ρ die Wurzelbedingung.
Beweisidee: Wir zeigen, dass $(v-w)$ einer inhomogenen Differenzengleichung genügt. Dann können wir $(v-w)$ explizit angeben mit Hilfe der Lösungen der homogenen Gleichung, und diese sind genau dann beschränkt, wenn ρ die Wurzelbedingung erfüllt.
Sei

$$\delta = v - w,\ d = F_h(v) - F_h(w).$$

Nach Definition gilt

$$\delta_k = d_k,\ k = 0\ldots m-1.$$

Wir versuchen, δ durch d abzuschätzen. Es gilt

$$\begin{aligned} d_{k+m} &= (F_h(v) - F_h(w))_k \\ &= \frac{1}{h}\sum_{j=0}^{m} a_j \delta_{k+j} - (\varphi(t_{k+j}, v_k, \ldots, v_{k+m}, h) - \varphi(t_k, w_k, \ldots, w_{k+m}, h)) \end{aligned}$$

und damit erfüllt δ die inhomogene Differenzengleichung

$$\sum_{j=0}^{m} a_j \delta_{k+j} = h \underbrace{d_{k+m} + (\varphi(t_{k+j}, v_k, \ldots, v_{k+m}, h) - \varphi(t_k, w_k, \ldots, w_{k+m}, h))}_{=:\eta_k}.$$

Nach 5.13 hat δ die Form

$$\delta = \sum_{j=0}^{m-1} d_j y_k^{(j)} + h \sum_{l=0}^{\infty} \eta_l z^{(l)}$$

mit den kanonischen Basen $y^{(j)}$ und $z^{(j)}$ aus 5.13. Da ρ die Wurzelbedingung erfüllt, gibt es ein M mit

$$||y^{(j)}||_\infty,\ ||z^{(j)}||_\infty \leq M$$

Sei L die Lipschitzkonstante von f. Dann ist φ lipschitzstetig in den y_k, also

$$|\eta_k| \leq |d_{m+k}| + \underbrace{Lm}_{=:\widetilde{L}} \underbrace{\max_{j=0\ldots m} |\delta_{k+j}|}_{=:\epsilon_k}$$

Sei nun

$$d := ||d_k||_\infty = ||F_h(v) - F_h(w)||_\infty.$$

Dann gilt

$$\begin{aligned} |\delta_k| &\leq Mmd + Mh \sum_{l=0}^{k-m} (d + \widetilde{L}\epsilon_l) \\ &= Md(m + (k-m+1)h) + Mh\widetilde{L} \sum_{l=0}^{k-m} \epsilon_j \end{aligned}$$

und damit nach Definition der ϵ_k

$$\epsilon_k \leq Md(m + (k+1)h) + Mh\widetilde{L} \sum_{j=0}^{k} \epsilon_j$$

oder

$$\epsilon_k(1 - Mh\widetilde{L}) \leq Md(m + (k+1)h) + Mh\widetilde{L}\sum_{j=0}^{k-1}\epsilon_j.$$

Sei nun

$$h < h_0 := \frac{1}{2M\widetilde{L}} \Rightarrow 1 - Mh\widetilde{L} \geq \frac{1}{2}.$$

Dann gilt

$$\epsilon_k \leq \underbrace{2M(m + (b-a))}_{=:c} d + \underbrace{2M\widetilde{L}}_{=:D} h \sum_{j=0}^{k-1}\epsilon_j$$

und mit dem Lemma

$$|\delta_k| \leq \epsilon_k \leq cde^{Dkh} \Rightarrow ||v - w||_\infty \leq ce^{D(b-a)}||F_h(v) - F_h(w)||_\infty$$

für $h < h_0$. □

Bemerkung: Der Satz ist für allgemeine Mehrschrittverfahren gültig, wir haben die Linearität hier nur zur Vereinfachung der Schreibweise benutzt.
Bemerkung:

1. Der Fehler beim Mehrschrittverfahren wird abgeschätzt durch das Maximum der Fehler in den Anfangswerten und den lokalen Diskretisierungsfehler. 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 das charakteristische Polynom ist $\rho(x) = x - 1$).

3. Aus Integration gewonnene Mehrschrittverfahren sind stabil (denn das charakteristische Polynom ist $\rho(x) = x^m - 1$).

4. Es gibt konsistente Verfahren, die nicht von der Differentialgleichung abhängen (Übungen). Die sind natürlich nicht konvergent! Also müssen sie instabil sein.

5.3.3 Stabilität und Konsistenz

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 5.22 *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

$$\begin{aligned} \tau_h(t) &= \frac{1}{h} \sum_{j=0}^{m} \alpha_j y(t+jh) - \sum_{j=0}^{m} \beta_j y'(t+jh) \\ &= \frac{1}{h} \underbrace{\sum_{j=0}^{m} \alpha_j}_{C_0} y(t) \\ &\quad + \underbrace{\left(\sum_{j=0}^{m} l\alpha_j - \sum_{j=0}^{m} \beta_j \right)}_{C_1} y'(t) \\ &\quad \vdots \\ &\quad + h^{p-1} \underbrace{\left(\frac{1}{p!} \sum_{j=0}^{m} j^p \alpha_j - \frac{1}{(p-1)!} \sum_{j=0}^{m} j^{p-1} \beta_j \right)}_{C_p} y^{(p)}(t) + O(h^p). \end{aligned}$$

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 5.23 *(Maximale Konvergenzordnung stabiler Verfahren)*
Stabile m–Schritt–Verfahren haben die maximale Konsistenzordnung $m + 1$ für m ungerade und $m + 2$ für m gerade. Die maximale Konvergenzordnung wird erreicht.

Bemerkung: Adams–Moulton und Milne–Simpson sind optimal für m ungerade.
Beweis: Henrici, Elements of Numerical Analysis 1964, mit 5.22 und ein bisschen Funktionentheorie. □

5.4 Spezialitäten

Abschließend wollen wir noch kurz zwei Themen anschneiden, die eine viel eingehendere Behandlung verdient hätten.

5.4.1 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 die Behandlung gewöhnlicher Differentialgleichungen anwenden lässt. Dies ist für Einschrittverfahren richtig, für Mehrschrittverfahren aber im Allgemeinen nicht.

Satz 5.24 *(Extrapolation)*

1. *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.*

2. *Bei Einschrittverfahren lässt sich der globale Diskretisierungsfehler durch Interpolation abschätzen, es gilt also*

$$y(t_0, h/2) - y(t) \sim \frac{1}{2^p - 1}(y(t_0, h) - y(t_0, h/2))$$

   für hinreichend kleine h.

3. *Bei Mehrschrittverfahren besitzt der globale Diskretisierungsfehler im allgemeinen keine Potenzreichenentwicklung.*

Beweis: (1), (2) in Stoer–Bulirsch II, 7.2.3. Beispiel für (3) in den Übungen.
In den Übungen zeigen wir, dass schon für ein einfaches Beispiel der Fehler der Mittelpunktsregel nicht entwickelbar ist. Es taucht an der Stelle t_j 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 5.25 *(Verfahren von Gragg)*
Sei

$$\begin{aligned} z_0 &= y(a) \\ z_1 &= z_0 + hf(a, z_0) \\ z_{k+2} &= z_k + 2hf(x_{k+1}, z_{k+1}) \\ y_n &= \frac{1}{4}y_{n+1} + \frac{1}{2}y_n + \frac{1}{4}y_{n-1} \\ y_0 &= z_0 \\ y_N &= z_N \end{aligned}$$

das Verfahren von Gragg. Dann ist der globale Diskretisierungsfehler in eine Potenzreihen in h^2 entwickelbar.

Zum Beweis siehe wieder Stoer–Bulirsch und die Bemerkungen in den Übungen.

5.4.2 Steifigkeit

Zu guten Differentialgleichungen gehört noch etwas mehr als nur Stabilität und Konvergenz. Bisher sind wir immer davon ausgegangen, dass h beliebig klein ist, und

haben uns etwa für die Stabilität damit zufriedengegeben, dass für ausreichend kleine h die Stabilitätsbedingung erfüllt ist. Ist ein Verfahren nicht für alle Schrittweiten stabil, so müssen wir aber folgendes erwarten: Bis zu einer gewissen Grenze liefert das Verfahren instabile Ergebnisse (mit exponentiellem Fehler), ab dieser Schranke liefert es sehr gute Ergebnisse (die unter Umständen viel besser sind, als wir sie benötigen). Die Schrittweite wird bei diesen Verfahren also unter Umständen nicht über die Genauigkeit, sondern über die Stabilität vorgegeben. Das gilt auch und insbesondere für Einschrittverfahren!
Ist eine Differentialgleichung so beschaffen, dass dieses Verhalten auftritt, so heißt die Differentialgleichung steif. Beispiele dafür werden wir gleich kennenlernen. Zunächst schauen wir uns aber ein Beispiel an, in dem zwei Einschrittverfahren mit derselben Konsistenzordnung unterschiedliche physikalische Deutungen liefern.

Beispiel 5.26 *(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

$$E(t) = gh(t) + v(t)^2/2.$$

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

$$\begin{aligned} v_{k+1} &= v_k - \Delta t g \\ h_{k+1} &= h_k + \Delta t v_k \\ E_{k+1} &= g h_{k+1} + v_{k+1}^2/2 \\ &= g(h_k + \Delta t v_k) + (v_k - \Delta t g)^2/2 \\ &= (\Delta t^2 g^2)/2 + E_k \end{aligned}$$

Die Energie bleibt also nicht konstant, es tritt ein kleiner Fehlerterm auf, die "Dispersion", eine Art künstliche Reibung. Sie geht zwar für Δt gegen 0 auch gegen 0, aber

sie macht physikalisch hier keinen Sinn. Für das verbesserte Eulerverfahren gilt

$$\begin{aligned} y_{k+1} &= y_k + hf(t_k + h/2, y_k + h/2 f(t_k, y_k)) \\ v_{k+1} &= v_k - \Delta t g \\ h_{k+1} &= h_k + \Delta t(v_k + \Delta t/2g) \\ E_{k+1} &= g(h_k - \Delta t(v_k + \Delta t/2g)) + (v_k + \Delta t g)^2/2 \\ &= E_k \end{aligned}$$

Das Verfahren ist also energieerhaltend, unabhängig von der Schrittweite! Wir werden für diese konkrete Differentialgleichung also mit dem verbesserten Eulerverfahren bessere Ergebnisse erwarten.

Wir betrachten nun das Modellproblem von Dahlquist

$$y'(t) = \lambda y(t),\ y(0) = t,\ \lambda \in \mathbb{C},\ \mathrm{Re}\lambda < 0.$$

Die analytische Lösung in einer Dimension ist $\exp(\lambda t)$. Offensichtlich geht die Lösung gegen 0, und das erwarten wir auch von der numerischen Approximation, unabhängig von h.

Definition 5.27 *(A-stabil)*
Ein Verfahren heißt A–stabil, wenn für das Dahlquist–Problem die numerische Approximation monoton fallend ist für alle h und alle λ mit Realteil kleiner o.

Für Ein– und Mehrschrittverfahren lässt sich die A–Stabilität einfach überprüfen.

Satz 5.28 *Für die durch Ein– und Mehrschrittverfahren gelieferten Näherungen für die Lösung des Dahlquist–Problems gilt*

$$y_{k+1} = R(\lambda h) y_k.$$

Für Einschrittverfahren ist R ein Polynom, für Mehrschrittverfahren eine rationale Funktion. Ein Verfahren ist A–stabil, falls $|R(z)| < 1$ für alle z mit Realteil kleiner 0. Der Bereich von $\mathbb{C}$ mit $|R(z)| < 1$ heißt Stabilitätsbereich.

Beweis: Übungen.

Beispiel 5.29 *(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 $R(z) = \frac{1}{1-z}$, das Verfahren ist A–stabil.

Satz 5.30 *Kein explizites Verfahren ist A–stabil. (Und da wir implizite Verfahren immer nur näherungsweise als explizite Verfahren lösen durch die Fixpunktiterationen, ist kein bisher betrachtetes Verfahren A–stabil.)*

Beweis: Für explizite Verfahren ist R ein Polynom. □

Die Bedingung der A–Stabilität ist besonders für große z, also große λ, schwer zu erfüllen. Für skalare Differentialgleichungen erster Ordnung ist das aber kein Problem: Wenn λ größer wird, so muss auch die Diskretisierung um den Faktor λ verfeinert werden, um die 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

$$y'(t) = Dy(t)$$

mit einer Diagonalmatrix D und den Einträgen λ_1, λ_2 auf der Hauptdiagonalen. Wir nehmen an, dass die Realteile der λ_k negativ sind, und dass $|\text{Re}\lambda_1| >> |\text{Re}\lambda_2|$. Das System besitzt die Lösungen $a\exp(\lambda_1 t) + b\exp(\lambda_2)t$. Der erste Term fällt schnell, spielt also für die Lösung praktisch keine Rolle. Der zweite Term dominiert die Lösung, daher sollte die Genauigkeit an ihm ausgerichtet werden (und damit die Wahl der Schrittweite).
Aber: Da der Betrag des Realteils von λ_1 sehr groß ist, muss bei nicht A–stabilen Verfahren die Schrittweite so gewählt werden, dass überhaupt erst einmal das Gesamtverfahren stabil ist. Wir stehen also vor dem Dilemma: Wir könnten die Schrittweite eigentlich groß wählen für die Genauigkeit, müssen sie aber klein wählen für die Stabilität. Das war gerade unsere Definition für steife Differentialgleichungen. Dies ist die Motivation für den Satz:

Satz 5.31 *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. □

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

$$y'(t) = t(t, y(t)),\ y(a) = y_0, \text{ auf } [a, b].$$

Gleichungen höherer Ordnung haben wir in Systeme umgewandelt, für eine lineare Differentialgleichung zweiter Ordnung lautet das Anfangswertproblem dann also

$$y''(t) + p(t)y'(t) + q(t)y(t) = f(t),\ y(a) = y_0,\ y'(a) = y'_0.$$

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. $y(a) = y(b) = 0$. Wir landen also bei einem Randwertproblem, bei dem die festen Bedingungen nicht nur am Randpunkt a, sondern auch am Randpunkt b vorgegeben sind.

Definition 6.1 *Ein Randwertproblem sucht Funktionen $y(x) \in C^1([a, b] \mapsto \mathbb{R}^n)$ mit*

$$y'(t) = f(t, y(t)),\ g(y(a), y(b)) = 0$$

für eine Funktion $g : \mathbb{R}^{2n} \mapsto \mathbb{R}$. 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). Für $n = 2$:

1. $y(a) = y_0$, $y(b) = y_1$: *Dirichlet–Randbedingungen. Falls* $y_0 = y_1 = 0$*: Homogene Dirichlet–Randbedingungen.*

2. $y'(a) = y_0$, $y'(b) = y_2$: *Neumann–Randbedingungen. Falls* $y_0 = y_1 = 0$*: Homogene Neumann–Randbedingungen.*

3. $g_1(y(a), y'(a)) = 0$, $g_2(y(b), y'(b)) = 0$*: Gemischte Randbedingungen.*

Die Lösbarkeit dieses Systems ist leider erheblich komplizierter als bei Anfangswertaufgaben. Wir betrachten die lineare Differentialgleichung zweiter Ordnung mit konstanten Koeffizienten und Dirichlet–Randbedingungen:

$$y''(t) - \alpha^2 y(t) = 0,\ \alpha \in \mathbb{R},\ y(a) = y_1,\ y(b) = y_2,\ b > a.$$

Alle Lösungen der Differentialgleichung sind von der Form

$$y(t) = c_1 y_1(t) + c_2 y_2(t) = 0,\ y_1(t) = \exp(\alpha t),\ y_2(t) = \exp(-\alpha t)$$

und zur Erfüllung der Randwerte erhalten wir die Gleichungen

$$c_1 y_1(a) + c_2 y_2(a) = 0,\ c_1 y_1(b) + c_2 y_2(b) = 0.$$

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 tatsächlich

$$W = \exp(a) \begin{pmatrix} 1 & \exp(\alpha(b-a)) \\ 1 & \exp(-\alpha(b-a)) \end{pmatrix}$$

und da die Exponentialfunktion monoton ist, sind die Zeilen voneinander unabhängig, damit ist die Matrix invertierbar und die Dirichlet–Aufgabe eindeutig lösbar. Tatsächlich sind lineare Dirichlet–Randwertaufgaben zweiter Ordnung der Form

$$y''(t) + p(t)y'(t) - q(t)y(t) = 0, y(a) = y_0,\ y(b) = y_1$$

mit $q(t) \geq 0$ immer eindeutig lösbar (Walter).
Anders ist es im Fall $q(t) \leq 0$. Wir betrachten nun das fast gleiche Randwertproblem

$$y''(t) + \alpha^2 y(t) = 0,\ y(0) = 0,\ y(\pi) = 0,\ \alpha \in \mathbb{R}.$$

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. 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 $\alpha \notin \mathbb{Z}$.
An dem einfachen Beispiel der linearen Differentialgleichungen zweiter Ordnung mit homogenen Randbedingungen lasen sich bereits alle Schwierigkeiten studieren. Wir betrachten also im Folgenden das Problem (LRWP)

$$y''(t) + p(t)y'(t) - q(t)y(t) = f(t),\ y(a) = y(b) = 0,\ q(t) \geq 0.$$

Durch die Transformation $Y(x) = y(p(x))$ können wir sogar noch $p = 0$ erreichen, d.h. es reicht die vereinfachte Form

$$y''(t) - q(t)y(t) = f(t),\ y(a) = y(b) = 0$$

zu betrachten.Wir werden nun vier verschiedene Ansätze zur 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 DGL ist es also sehr nützlich, die Interpretation als gewöhnliche DGL im Kopf zu behalten. Wir werden daher hier immer einige Beziehungen zu den PDGL angeben, auch wenn Sie diese bisher nicht gehört haben sollten.

6.1 Analytische Lösung

Ein Verfahren zur Lösung der homogenen Differentialgleichung mit $f = 0$ haben wir oben bereits angegeben: Wir bestimmen die Grundlösungen, die Lösung von LRWP 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 entsprechenden Anfangswertaufgabe bei b.

Die Funktion

$$G(t,t') = \frac{1}{W(t)} \begin{cases} y_1(t)y_2(t'), & t \le t' \\ y_2(t)y_1(t'), & t \ge t' \end{cases}$$

mit

$$W(t) = \begin{vmatrix} y_1(t) & y_2(t) \\ y_1'(t) & y_2'(t) \end{vmatrix}$$

heißt Greensche Funktion von (LRWP), $W(x)$ *heißt Wronski–Determinante und ist konstant.*
Falls $y = 0$ *einzige Lösung von (LRWP) mit* $f = 0$ *ist, so ist (LRWP) für alle stetigen* f *eindeutig lösbar. Die Lösung ist gegeben durch*

$$y(t) = \int_a^b G(t,t')f(t')\text{dt'}.$$

Beweis:

$$W(t) = y_1(t)y_2'(t) - y_2(t)y_1'(t),$$

also

$$W'(t) = y_1'(t)y_2'(t) + y_1(t)y_2''(t) - y_2'(t)y_1(t) - y_2(t)y_1''(t) = q(t)y_1(t)y_2(t) - q(t)y_1(t)y_2(t) = 0$$

und damit $W(x) = c$ konstant. Wäre $y_1(b) = 0$, so wäre y_1 nichttriviale Lösung des homogenen Problems. Nach Voraussetzung gilt also

$$c = W(b) = y_1(b) \neq 0.$$

Zur Interpretation von G im Sinne von partiellen Differentialgleichungen betrachten wir es bei festem t'. Dann ist es für $t < t'$ ein Vielfaches von $y_1(t)$, für $t > t'$ ein Vielfaches von $y_2(t)$ und erfüllt damit die Differentialgleichung. Am Punkt $t = t'$ ist G gar nicht differenzierbar, der Differenzenquotient wächst über alle Maßen. Tatsächlich löst G in einem erweiterten (distributionellem) Sinne die Differentialgleichung

$$G''(t,t') - q(t)G'(t,t') = \delta(t - t')$$

wobei δ eine Funktion (Distribution) ist mit der Eigenschaft

$$\int_{\mathbb{R}} \delta(x) = 1$$

und δ verschwindet außerhalb des Nullpunkts – dies kann offensichtlich von keiner normalen Funktion erfüllt werden. Unter der Annahme, dass es eine solche Funktion gibt, so gilt aber natürlich sofort

$$y''(t) - q(t)y(t) = \frac{1}{W(t)} \int_a^b \delta(t')f(t')\text{dt'} = f(t)$$

und y ist Lösung von (LRWP).
Wir zeigen den Satz aber natürlich direkt. Nachrechnen liefert sofort, dass überraschenderweise für stetiges f y differenzierbar ist, obwohl G nicht differenzierbar ist, und

$$y''(t) = y(t)q(t) + \frac{1}{c}W(t)f(t)$$

und damit ist y Lösung der Differentialgleichung, hat homogene Randwerte, ist also Lösung von (LRWP). □

Damit haben wir die erste (analytische) Methode zur Lösung von (LRWP) gefunden:

1. Bestimme die Lösungen y_0, y_1 der Anfangswertprobleme.
2. Berechne die Greensche Funktion.
3. Berechne $y(t) = \int_a^b f(t')G(t,t')\mathrm{d}t'$.

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 von (LRWP). Für jedes α besitzt die Anfangswertaufgabe mit der vorgegebenen DGL 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 Analysis kennen, etwa das Newton–Verfahren. Voraussetzung zur Anwendung dieser Verfahren ist, dass die Funktion mindestens einmal stetig differenzierbar ist. Tatsächlich gilt der Satz:

Satz 6.3 *Sei (LRWP) gegeben. Sei $y_\alpha(t)$ die Lösung der Anfangswertaufgabe der DGL für $y'_\alpha(a) = \alpha$, $y_\alpha(a) = 0$. Sei $F(\alpha) = y_\alpha(b)$. Dann ist F differenzierbar.*

Dies liefert uns das zweite (numerische) Verfahren zur Lösung von (LRWP):

1. Implementiere die Funktion $F(\alpha)$, die numerisch 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.

6.3 Diskretisierungsverfahren

Wir können auch die Diskretisierungsidee aus dem letzten Kapitel zur Lösung einsetzen. Sei $t_0, \ldots, t_N$ wie üblich ein zulässiges Gitter auf $[a, b]$. Wir suchen Approximationen $y_0, \ldots, y_N$ mit $y(t_k) \sim y_k$.
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.
Wir betrachten als Modellproblem die eindimensionale Poisson–Gleichung

$$-y''(t) = f(t)$$

für t in $[0, 1]$ auf einem äquidistanten Gitter. Zur Diskretisierung nutzen wir die Formeln aus 3.27. Wir erhalten das lineare Gleichungssystem

$$\begin{aligned} -\frac{y_0 - 2y_1 + y_2}{h^2} &= f_1 = f(t_1) \\ &\vdots \\ -\frac{y_{N-2} - 2y_{N-1} + y_N}{h^2} &= f_{N-1} = f(t_{N-1}) \end{aligned}$$

Setzen wir die Randbedingung $y_0 = y_N = 0$ in das Gleichungssystem ein, so erhalten wir für die Unbekannten $y_1, \ldots, y_{N-1}$ das lineare Gleichungssystem

$$\underbrace{\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}}_{L_h} \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix}}_{y_h} = h^2 \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{pmatrix}}_{f_h}.$$

Die Diskretisierung ist offensichtlich konsistent. 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 $(y_k(t_j)$, $k = 1, \ldots, N-1$, zu den Eigenwerten

$$\lambda_k = \frac{4}{h^2} \sin^2(kh\pi/2).$$

und damit

$$\lambda_k \geq \lambda_1 \geq \frac{4}{h^2} \left(\frac{2}{\pi}\right)^2 \frac{h^2\pi^2}{4} = 4$$

mit der Abschätzung

$$\sin x \geq \frac{2}{\pi}x,\ x \in [0, \pi/2].$$

Insbesondere ist L_h also invertierbar und $||L_h^{-1}||_2 \leq 1/4$. Das lineare Gleichungssystem ist also lösbar, und sogar sehr effizient lösbar, denn es ist eine Tridiagonalmatrix.
Für die Konvergenz gilt:

$$\begin{array}{rcl} ||y_{I_h} - y_h||_2 & = & ||L_h^{-1}(L_h y|_{I_h} - f_h)||_2 \\ & \leq & \underbrace{||L_h^{-1}||}_{\text{Stabilitätsfaktor}} \cdot \underbrace{||L_h(y|_{I_h}) - f_h||_2}_{\text{Konsistenzfehler}} \\ & \mapsto_{h \mapsto 0} & 0 \end{array}$$

Mit dieser Definition des Stabilitätsfaktors und der offensichtlichen Definition der Stabilität (der Stabilitätsfaktor ist unabhängig von h nach oben beschränkt) gilt wieder der Satz:

Satz 6.4 *Aus Stabilität und Konsistenz folgt Konvergenz (bzgl. der euklidischen Norm).*

Leider steht hier nur die euklidische Norm. 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 6.5 *Sei A eine reelle, invertierbare $N \times N$–Matrix. A heißt M–Matrix genau dann, wenn $A_{ii} > 0 \forall_i$, $A_{ik} \leq 0 \forall_{i \neq k}$, und $A_{ik}^{-1} \geq 0 \forall_{i,k}$.*

Definition 6.6 *Eine Matrix A heißt schwach diagonaldominant, falls in jeder Zeile die Summe der Beträge der Außerdiagonalelemente nicht größer ist als der Betrag des Hauptdiagonalelements.*

Satz 6.7 *Sei A eine reelle, invertierbare, schwach diagonaldominante $N \times N$–Matrix mit positiven Hauptdiagonaleelementen und nicht–positiven Außerdiagonalelementen. Dann ist A eine M–Matrix.*

Beweis: Sei $b < 0$, $Ax = b$. Angenommen, $x_j = max\{x_1, \ldots, x_N\} > 0$. Dann gilt

$$\begin{array}{rcl} A_j j x_j & = & b_j - \sum_{k \neq j} A_{jk} x_k \\ & < & -\sum_{k \neq j} A_{jk} x_j \\ & \leq & A_{jj} x_j. \end{array}$$

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 6.8 *Das Standard–Differenzenschema zur Berechnung der Lösung der eindimensionalen Poisson–Gleichung ist stabil bezüglich der Maximumnorm.*

Beweis: Todo: Zu zeigen ist: Die Norm von L_h ist unabhängig von h beschränkt. □

Korollar 6.9 *Das Standardverfahren zur Berechnung der Lösung der eindimensionalen Poisson–Gleichung ist konvergent von der Ordnung 2 (bezüglich der Maximumnorm).*

Beweis: Klar nach den Vorbemerkungen. □

Kapitel 7

Differenzenverfahren für partielle Differentialgleichugen

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings