

MODUL M4

Numerische Mathematik für Fachrichtung Informatik und Lehramt Wintersemester 2005/06

Martin Arnold
Martin-Luther-Universität Halle-Wittenberg
Fachbereich Mathematik und Informatik
Institut für Numerische Mathematik

13. Februar 2006

Modul M4

Numerische Mathematik für Fachrichtung Informatik und Lehramt

Institut für Numerische Mathematik

- Prof. Dr. M. Arnold (martin.arnold@mathematik.uni-halle.de)
- Dr. H. Podhaisky (helmut.podhaisky@mathematik.uni-halle.de)
- Dipl.-Tech. math. B. Bürgermeister ([bernhard.buergermeister@ ...](mailto:bernhard.buergermeister@...))

Georg-Cantor-Haus (Heide Süd), Theodor-Lieser-Str. 5, Räume 221, 222, 224

Internet: Stichwort „Lehre“ unter <http://www.mathematik.uni-halle.de/~arnold/>

Übungsbesprechung Mi 12.10.2005, 08.15 Uhr, HS 3.04

Abschlussklausur Freitag, den 17.02.06, 10-13 Uhr, HS I (Kröllwitz)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Hinweis Dieses Vorlesungsskript ist ausschließlich für den persönlichen Gebrauch der Hörerinnen und Hörer der Lehrveranstaltung „Modul M4: Numerische Mathematik für Fachrichtung Informatik und Lehramt“ (Wintersemester 2005/06) an der Martin-Luther-Universität Halle-Wittenberg bestimmt.

Teile des vorliegenden Materials wurden sinngemäß, z. T. auch wörtlich, der angegebenen Lehrbuchliteratur entnommen ohne dies in jedem Einzelfall durch Quellenangaben zu belegen.

Inhaltsverzeichnis

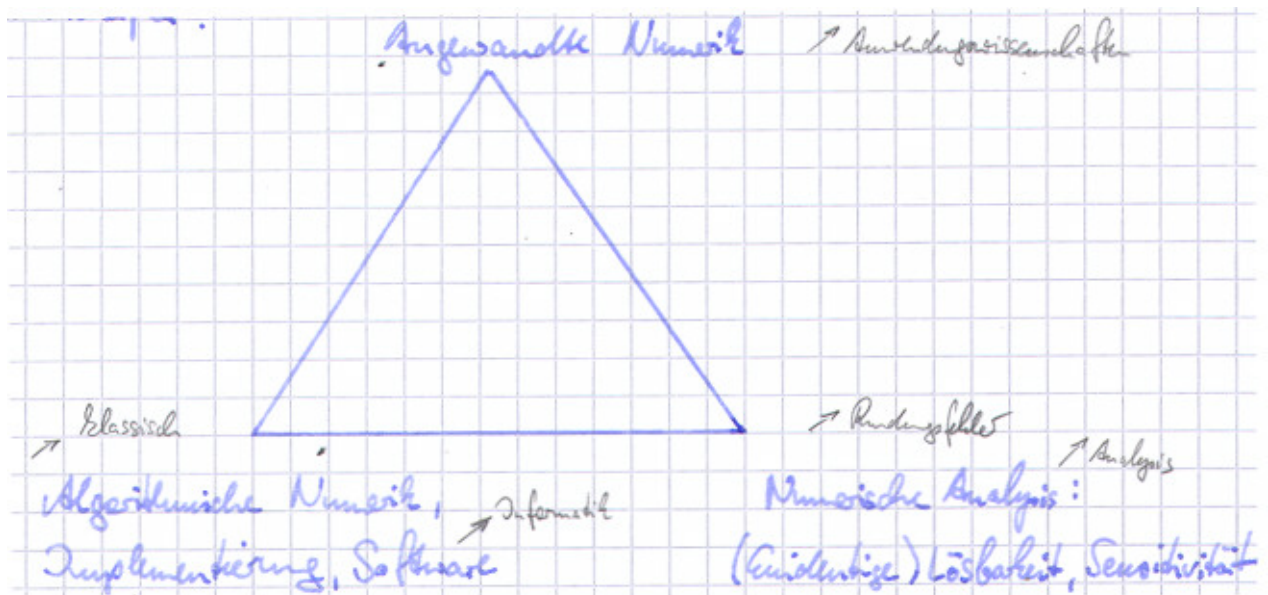
1	Einführung	1
1.1	Grundlagen	1
1.2	Klassische Polynominterpolation	3
2	Lineare Gleichungssysteme	18
2.1	Gaußscher Algorithmus	18
2.2	Lineare Ausgleichsrechnung	25
3	Rechnerarithmetik und Rundungsfehler	29
3.1	Gleitpunktarithmetik	29
3.2	Vektor- und Matrixnormen	36
3.3	Kondition und Stabilität	38
4	Interpolation (II)	42
4.1	Spline-Interpolation	42
4.2	Trigonometrische Interpolation – Schnelle Fouriertransformation	47
5	Quadratur	55
6	Iterationsverfahren für lineare und nichtlineare Gleichungssysteme	60
6.1	Nullstellen reeller Funktionen	60
6.2	Das Newtonverfahren	65

1 Einführung

1.1 Grundlagen

Bemerkung 1.1 (Numerische Mathematik)

- a) **Im engeren Sinn:** zahlenmäßige Auswertung mathematischer Zusammenhänge
z. B.
- Lösung von linearen und nichtlinearen Gleichungssystemen
 - Numerische Integration und Differentiation
 - Näherungsweise Auswertung reeller Funktionen
 - Numerische Lösung von Differentialgleichungen
 - Numerische Lösung von Optimierungsproblemen
- b) **Typisches Ziel:** Näherungen für die exakte Lösung eines mathematischen Problems, deren Fehler beliebig klein gemacht werden kann und für die verlässliche Fehlerschranken vorliegen.
- c) **praktisch:** wesentliche Komponente des Wissenschaftlichen Rechnens (engl.: scientific computing): Computersimulation auf der Grundlage mathematischer Modelle in den Anwendungswissenschaften: Naturwissenschaften, Ingenieurwissenschaften, Medizin, Wirtschaftswissenschaften.

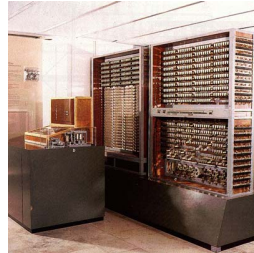


Bemerkung 1.2 (Entwicklung der Rechentechnik)

- Entwicklung der Numerik untrennbar verknüpft mit Entwicklung der Rechentechnik
- Grundlagen: verstärkt ab 18. Jahrhundert

Bemerkung 1.2: Entwicklung der Rechentechnik

Z3 (1941)



ENIAC (1946)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.1: Klassische Rechentechnik.

Bemerkung 1.2: Entwicklung der Rechentechnik (II)

Hitachi SR8000-F1



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.2: Moderner Hochleistungsrechner am LRZ München.

- 1941 Z3 (K. Zuse)
- 1946 ENIAC (J. v. Neumann)
- 1958 erster Mikrochip
- 1967 erster Taschenrechner
- 1976 Home-Computer „Apple“
- 1981 erster Personal Computer (PC)
- heute: leistungsfähige Arbeitsplatzrechner (PC, Workstation)
Vektor- und Parallelrechner für High performance computing
- seit 1971: Anzahl der elementaren Transistorfunktionen je Sekunde verdoppelt sich etwa nach jeweils 18 Monaten

Bemerkung 1.3 (Literatur)

- [1] Th. Huckle and S. Schneider. *Numerik für Informatiker*. Springer-Verlag, Berlin, 2002.
- [2] J. Stoer. *Numerische Mathematik 1*. Springer-Verlag, Berlin Heidelberg New York, 8th edition, 1999.
- [3] J. Stoer and R. Bulirsch. *Numerische Mathematik 2*. Springer-Verlag, Berlin Heidelberg New York, 4th edition, 2000.
- [4] P. Deuffhard and A. Hohmann. *Numerische Mathematik I. Eine algorithmisch orientierte Einführung*. Walter de Gruyter, Berlin New York, 3rd edition, 2002.
- [5] A. Quarteroni, R. Sacco, and F. Saleri. *Numerische Mathematik 1*. Springer-Verlag, Berlin, 2002.
- [6] A. Quarteroni, R. Sacco, and F. Saleri. *Numerische Mathematik 2*. Springer-Verlag, Berlin, 2002.
- [7] G.H. Golub and Ch.F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore London, 3rd edition, 1996.

1.2 Klassische Polynominterpolation

Bemerkung 1.4 (Problemstellung)

geg.: $n + 1$ Stützpunkte (x_j, y_j) , ($j = 0, 1, \dots, n$) mit Stützstellen x_j und Stützwerten y_j , zwischen denen ein (oft auch nur vermuteter) funktionaler Zusammenhang besteht: $y_j = f(x_j)$, ($j = 0, 1, \dots, n$). Praktisch oft: Messdaten y_j

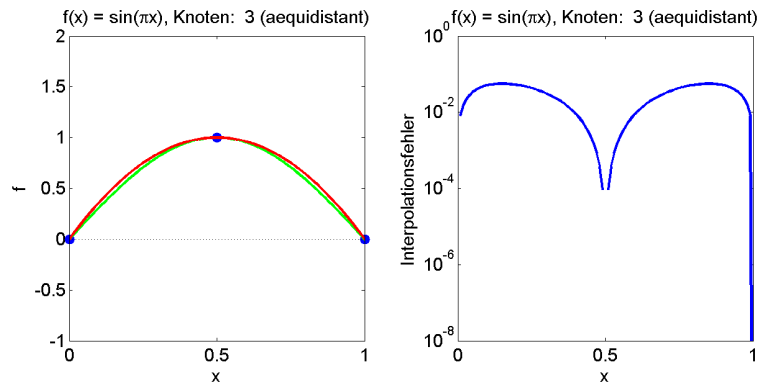
ges.: Polynom $\Phi^{(n)}(x)$ höchstens n -ten Grades, das die $n + 1$ Interpolationsbedingungen

$$y_j = \Phi^{(n)}(x_j), \quad (j = 0, 1, \dots, n)$$

erfüllt.

Bemerkung 1.4: Polynominterpolation

Beispiel $n = 2$, $x_0 = 0$, $x_1 = 1/2$, $x_2 = 1$
 $y_j = \sin \pi x_j$, ($j = 0, 1, 2$)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.3: Interpolationspolynom $\Phi^{(2)}(x)$ zu $f(x) = \sin \pi x$, ($x \in [0, 1]$).

Beispiel Tabellierte Daten, z. B.

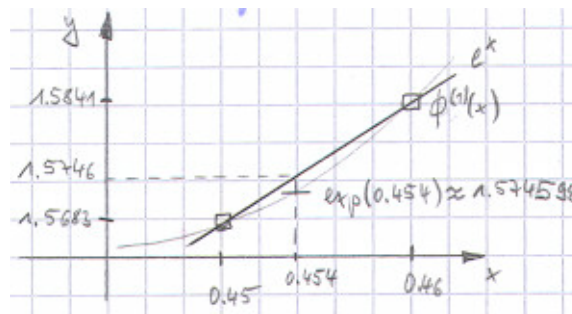
$n = 1$, $f(x) = e^x$, $x_0 = 0.45$, $x_1 = 0.46$, $y_0 = 1.5683 \approx \exp(0.45)$, $y_1 = 1.5841 \approx \exp(0.46)$

Lineare Interpolation

$$\exp(x) \approx \Phi^{(1)}(x) = 1.5683 + \frac{x - 0.45}{0.46 - 0.45}(1.5841 - 1.5683)$$

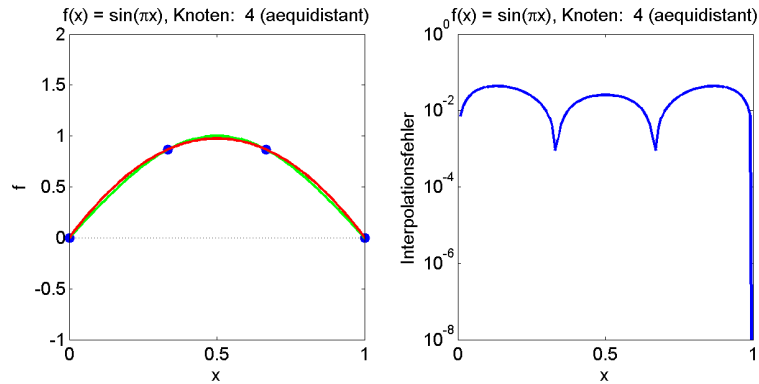
Beispiel

$$\Phi^{(1)}(0.454) = 1.5746$$



Bemerkung 1.4: Polynominterpolation (II)

Beispiel $n = 3$, $x_0 = 0$, $x_1 = 1/3$, $x_2 = 2/3$, $x_3 = 1$
 $y_j = \sin \pi x_j$, ($j = 0, 1, 2, 3$)

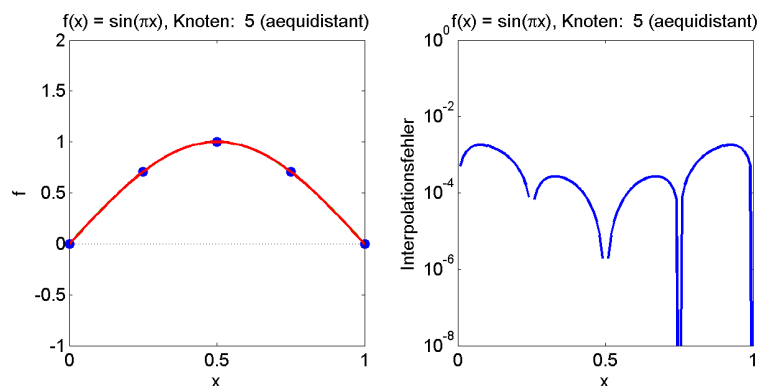


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.4: Interpolationspolynom $\Phi^{(3)}(x)$ zu $f(x) = \sin \pi x$, ($x \in [0, 1]$).

Bemerkung 1.4: Polynominterpolation (III)

Beispiel $n = 4$, $x_0 = 0$, $x_1 = 1/4$, $x_2 = 1/2$, $x_3 = 3/4$, $x_4 = 1$
 $y_j = \sin \pi x_j$, ($j = 0, 1, 2, 3, 4$)

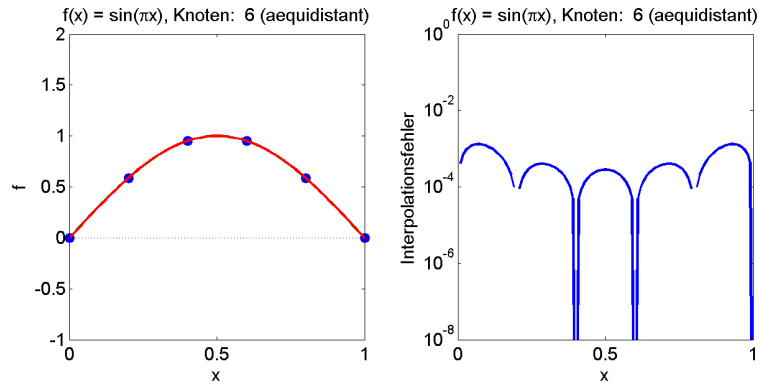


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.5: Interpolationspolynom $\Phi^{(4)}(x)$ zu $f(x) = \sin \pi x$, ($x \in [0, 1]$).

Bemerkung 1.4: Polynominterpolation (IV)

Beispiel $n = 5$, $x_j = j/5$, ($j = 0, 1, 2, 3, 4, 5$)
 $y_j = \sin \pi x_j$, ($j = 0, 1, 2, 3, 4, 5$)

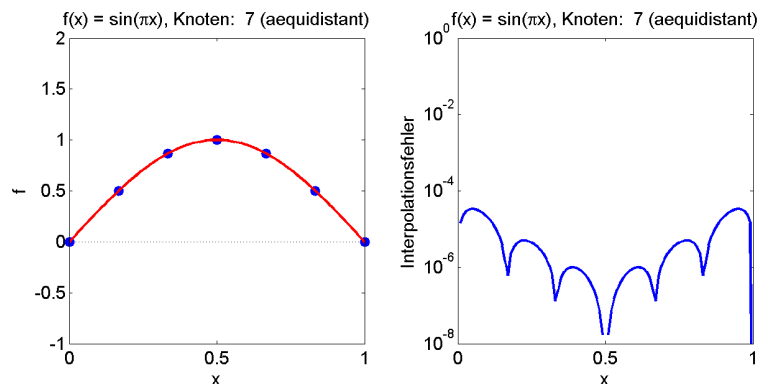


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.6: Interpolationspolynom $\Phi^{(5)}(x)$ zu $f(x) = \sin \pi x$, ($x \in [0, 1]$).

Bemerkung 1.4: Polynominterpolation (V)

Beispiel $n = 6$, $x_j = j/6$, ($j = 0, 1, 2, 3, 4, 5, 6$)
 $y_j = \sin \pi x_j$, ($j = 0, 1, 2, 3, 4, 5, 6$)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.7: Interpolationspolynom $\Phi^{(6)}(x)$ zu $f(x) = \sin \pi x$, ($x \in [0, 1]$).

Bemerkung 1.5 (Eindeutigkeit des Interpolationspolynoms)

Sind die Stützstellen x_0, x_1, \dots, x_n paarweise voneinander verschieden, so ist das Interpolationspolynom $\Phi^{(n)}(x)$ aus Bemerkung 1.4 eindeutig bestimmt, denn gilt für zwei Polynome $\Phi_1^{(n)}, \Phi_2^{(n)} \in \Pi_n$

$$\Phi_1^{(n)}(x_j) = \Phi_2^{(n)}(x_j) = y_j, \quad (j = 0, 1, \dots, n),$$

so ist $\Phi_1^{(n)} - \Phi_2^{(n)} \in \Pi_n$ ein Polynom mit den $n + 1$ Nullstellen x_0, x_1, \dots, x_n
 $\Rightarrow \Phi_1^{(n)} - \Phi_2^{(n)} = 0$ (Fundamentalsatz der Algebra), $\Phi_1^{(n)}(x) \equiv \Phi_2^{(n)}(x)$.

Bemerkung 1.6 (Elementarer Zugang)

Sei $\Phi^{(n)}(x) = \sum_{i=0}^n a_i x^i$ mit den zunächst unbekanntenen Koeffizienten a_0, a_1, \dots, a_n . Die Interpolationsbedingungen $y_j = \Phi^{(n)}(x_j)$, ($j = 0, 1, \dots, n$) sind äquivalent zu dem linearen Gleichungssystem

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

Lösung mit dem Gaußschen Algorithmus, vgl. Abschnitt 2.1.

Ziel: Transformation in ein äquivalentes lineares Gleichungssystem mit Dreiecksgestalt

Schritt 1 Addiere Vielfache der 1. Zeile zu Zeilen $2, \dots, n + 1$ so, dass in der 1. Spalte alle Elemente unterhalb der Hauptdiagonale verschwinden:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 0 & x_1 - x_0 & x_1^2 - x_0^2 & \cdots & x_1^n - x_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & x_n - x_0 & x_n^2 - x_0^2 & \cdots & x_n^n - x_0^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 - y_0 \\ \vdots \\ y_n - y_0 \end{pmatrix}$$

Rechenaufwand: $\approx n^2$ Rechenoperationen zur Transformation von n Zeilen mit je n Spalten

Schritt k Addiere Vielfache der k -ten Zeile zu Zeilen $k + 1, \dots, n + 1$ so, dass in der k -ten Spalte alle Elemente unterhalb der Hauptdiagonale verschwinden.

Rechenaufwand: $\approx (n + 1 - k)^2$ Rechenoperationen zur Transformation von $n + 1 - k$ Zeilen mit je $n + 1 - k$ Spalten

gesamt n Gaußschritte mit insgesamt

$$\sum_{k=1}^n (n + 1 - k)^2 = \sum_{l=1}^n l^2 = \frac{n(n + 1)(2n + 1)}{6}$$

Rechenoperationen.

Ergebnis: Rechenaufwand $\frac{n^3}{3} + \mathcal{O}(n^2)$ Rechenoperationen, wächst kubisch mit n .

Bemerkung 1.7 (Landau–Symbole)

Sei $x_0 \in \mathbb{R}$ und $g : I \rightarrow \mathbb{R}$ in einer Umgebung von x_0 definiert. Gibt es für ein $p \in \mathbb{R}$, $p \geq 0$ eine positive Konstante $\bar{c} \in \mathbb{R}$, so dass für alle x in einer hinreichend kleinen Umgebung von x_0 die Abschätzung

$$|g(x)| \leq \bar{c} \cdot |x - x_0|^p$$

erfüllt ist, so schreibt man

$$g(x) = \mathcal{O}((x - x_0)^p), \quad (x \rightarrow x_0)$$

spricht: „ $g(x)$ ist groß O von $(x - x_0)^p$ “.

Beispiel $\sin x = \mathcal{O}(x), \quad (x \rightarrow 0)$

Existiert $\lim_{x \rightarrow x_0} \frac{g(x)}{(x - x_0)^p}$ und ist $\lim_{x \rightarrow x_0} \frac{g(x)}{(x - x_0)^p} = 0$, so schreibt man

$$g(x) = o((x - x_0)^p), \quad (x \rightarrow x_0)$$

spricht: „klein o“.

Beispiel $\sqrt{x^3} = o(x), \quad (x \rightarrow 0)$

Entsprechend bedeutet

$$v(n) = \mathcal{O}(n^p), \quad (n \rightarrow \infty)$$

für eine Funktion $v : \mathbb{N} \rightarrow \mathbb{R}$, dass für alle $n \geq n_0$ die Abschätzung $|v(n)| \leq \bar{c}n^p$ mit einer gewissen positiven Konstanten $\bar{c} \in \mathbb{R}$ erfüllt ist, und

$$v(n) = o(n^p), \quad (n \rightarrow \infty)$$

steht für

$$\lim_{n \rightarrow \infty} \frac{v(n)}{n^p} = 0.$$

Beispiel $v(n) = \frac{n(n+1)(2n+1)}{6}$

Es gilt $v(n) < n \cdot 2n \cdot 3n / 6 = n^3$, also $v(n) = \mathcal{O}(n^3)$. Genauer gilt $v(n) = n^3/3 + w(n)$ mit $w(n) = \frac{1}{2}n^2 + \frac{1}{6}n = \mathcal{O}(n^2)$, man schreibt kurz: $v(n) \doteq n^3/3$.

Bemerkung 1.8 (Klassische Polynominterpolation: Lagrange–Darstellung)

Elementarer Zugang aus Bemerkung 1.6 \Rightarrow Interpolationspolynom $\Phi^{(n)}$ in monomialer Basis $\{1, x, x^2, \dots, x^n\}$:

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

Lagrange-Darstellung

$$\Phi^{(n)}(x) = \sum_{j=0}^n y_j L_j^{(n)}(x)$$

mit den *Lagrangeschen Basispolynomen*

$$L_j^{(n)}(x) := \prod_{\substack{i=0 \\ i \neq j}}^n \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_j - x_0)(x_j - x_1) \cdots (x_j - x_{i-1})(x_j - x_{i+1}) \cdots (x_j - x_n)},$$

die die Interpolationsbedingungen

$$L_j(x_k) = \delta_{kj} = \begin{cases} 1 & \text{falls } k = j, \\ 0 & \text{sonst,} \end{cases} \quad (k = 0, 1, \dots, n)$$

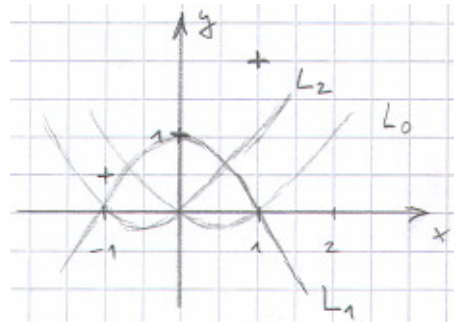
erfüllen.

Beispiel $n = 2$, $x_0 = -1$, $x_1 = 0$, $x_2 = 1$

$$L_0^{(2)}(x) = \frac{(x - 0)(x - 1)}{(-1 - 0)(-1 - 1)}$$

$$L_1^{(2)}(x) = \frac{(x + 1)(x - 1)}{(0 + 1)(0 - 1)}$$

$$L_2^{(2)}(x) = \frac{(x + 1)(x - 0)}{(1 + 1)(1 - 0)}$$



Bemerkung 1.9 (Horner-Schema)

Auswertung des Interpolationspolynoms mit $\mathcal{O}(n)$ Rechenoperationen:

$$\begin{aligned} \Phi^{(n)}(x) &= \sum_{j=0}^n a_j x^j = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \\ &= a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + x a_n) \dots)) \end{aligned}$$

Horner-Schema

	a_n	a_{n-1}	a_{n-2}	\dots	a_0
x		$x \cdot a_n$	$x \cdot (a_{n-1} + x a_n)$	\dots	$x \cdot (\dots)$
	$x \cdot a_n$	$a_{n-1} + x a_n$	$a_{n-2} + x(a_{n-1} + x a_n)$	\dots	$\dots = \Phi^{(n)}(x)$

Beispiel Auswertung von $\Phi(x) = (x - 1)^3 = x^3 - 3x^2 + 3x - 1$ an der Stelle $x = 5$:

	1	-3	3	-1
5		$5 \cdot 1 = 5$	$5 \cdot 2 = 10$	$5 \cdot 13 = 65$
	1	$-3 + 5 = 2$	$3 + 10 = 13$	$-1 + 65 = 64 = \Phi(5)$

Algorithmus

```

p := a_n
for i = n : -1 : 1
  | p := a_{i-1} + x * p

```

Matlab-Code Speicherschema für Vektoren in Matlab $(a_0, a_1, \dots, a_n) = \mathbf{a}(1:(n+1))$

```

p = a(n+1);
for i=n:-1:1,
  p = a(i) + x * p;
end;

```

Bemerkung 1.10 (Rekursive Auswertung des Interpolationspolynoms)

Sei $\Phi_{i,\dots,i+l} \in \Pi_l$ das Interpolationspolynom zu Stützpunkten

$$(x_k, y_k), \quad (k = i, i + 1, \dots, i + l).$$

Dann gilt

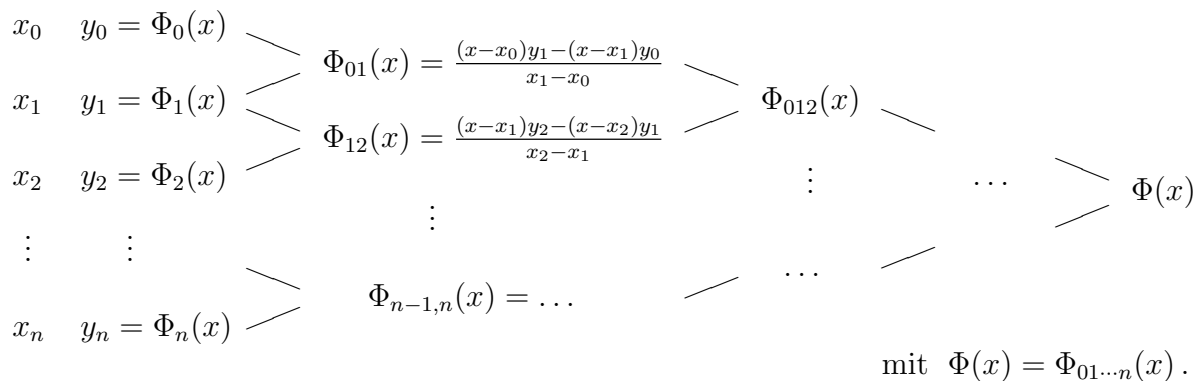
$$\begin{aligned} \Phi_{i+1,\dots,i+l}(x_k) &= y_k, \quad (k = i + 1, \dots, i + l), \\ \Phi_{i,\dots,i+l-1}(x_k) &= y_k, \quad (k = i, \dots, i + l - 1), \end{aligned}$$

also

$$\Phi_{i,\dots,i+l}(x) = \frac{(x - x_i)\Phi_{i+1,\dots,i+l}(x) - (x - x_{i+l})\Phi_{i,\dots,i+l-1}(x)}{x_{i+l} - x_i}, \quad (*)$$

denn der rechts stehende Ausdruck erfüllt $\Phi_{i,\dots,i+l}(x_k) = y_k$ für $k = i$, für $k = i + l$ und für $k = i + 1, \dots, i + l - 1$.

Neville-Schema



Beispiel 1.11 (Polynominterpolation: Neville–Schema)

Idee Approximiere $\sqrt{2} = 2^{1/2}$ durch $\Phi(1/2)$ mit dem Interpolationspolynom $\Phi \in \Pi_2$ zu $(-1, 1/2)$, $(0, 1)$ und $(1, 2)$.

Lagrange–Darstellung

$$\Phi\left(\frac{1}{2}\right) = \sum_{j=0}^2 y_j L_j^{(2)}\left(\frac{1}{2}\right) = \frac{1}{2} \cdot L_0^{(2)}\left(\frac{1}{2}\right) + 1 \cdot L_1^{(2)}\left(\frac{1}{2}\right) + 2 \cdot L_2^{(2)}\left(\frac{1}{2}\right)$$

mit den Lagrangeschen Basispolynomen $L_j^{(2)}(x)$ aus Bemerkung 1.8.

Neville

$$\begin{array}{rcl} -1 & \frac{1}{2} & \searrow \\ & & \frac{(\frac{1}{2} - (-1)) \cdot 1 - (\frac{1}{2} - 0) \cdot \frac{1}{2}}{0 - (-1)} = \frac{5}{4} \\ 0 & 1 & \nearrow \\ & & \searrow \\ & & \frac{(\frac{1}{2} - 0) \cdot 2 - (\frac{1}{2} - 1) \cdot 1}{1 - 0} = \frac{3}{2} \\ 1 & 2 & \nearrow \end{array} \quad \begin{array}{l} \searrow \\ \nearrow \end{array} \quad \frac{(\frac{1}{2} - (-1)) \cdot \frac{3}{2} - (\frac{1}{2} - 1) \cdot \frac{5}{4}}{1 - (-1)} = \frac{23}{16} \approx 1.4375$$

Bemerkung 1.12 (Klassische Polynominterpolation: Newton–Darstellung)

Problem Neville–Schema erfordert $\mathcal{O}(n^2)$ Rechenoperationen, ebenso die Auswertung von $\sum_j y_j L_j^{(n)}(x)$, aber

$$\sum_{j=0}^n a_j x^j = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + x a_n) \dots))$$

kann mit $\mathcal{O}(n)$ Rechenoperationen ausgewertet werden.

gesucht Darstellung des Interpolationspolynoms, die man effizient bestimmen kann und die mit Horner–artigem Schema ausgewertet werden kann.

Ansatz Es gilt

$$\Phi_{i,i+1,\dots,i+l}(x) = \Phi_{i,i+1,\dots,i+l-1}(x) + (x - x_i)(x - x_{i+1}) \cdots (x - x_{i+l-1}) f_{i,i+1,\dots,i+l}$$

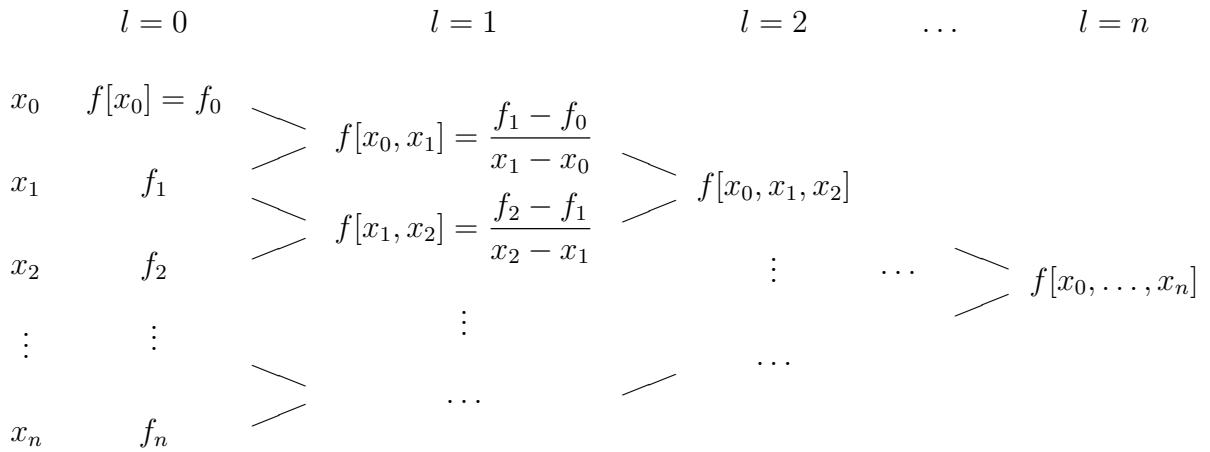
mit einem $f_{i,i+1,\dots,i+l} \in \mathbb{R}$, denn

$$\Phi_{i,i+1,\dots,i+l}(x_k) = \Phi_{i,i+1,\dots,i+l-1}(x_k), \quad (k = i, i + 1, \dots, i + l - 1).$$

Durch Vergleich der Koeffizienten von x^l in (*) aus Bemerkung 1.10 folgt

$$f_{i,i+1,\dots,i+l} = \frac{f_{i+1,\dots,i+l} - f_{i,\dots,i+l-1}}{x_{i+l} - x_i}.$$

Für $f_k = f(x_k)$ bezeichnet man diese *dividierten Differenzen* mit $f[x_i, \dots, x_{i+l}]$.
 Rekursive Berechnung mittels *Steigungsschema*



mit

$$f[x_0, x_1, x_2] = \frac{\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}, \dots$$

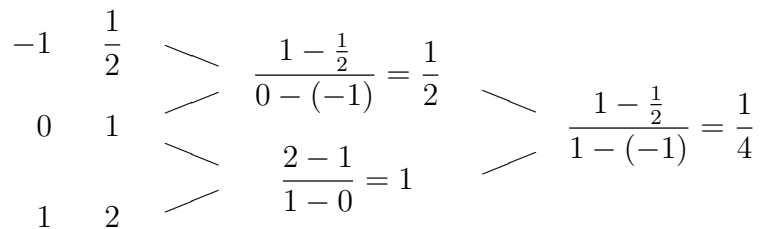
Newton'sche Darstellung

$$\Phi(x) = f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \dots + (x - x_0)(x - x_1) \dots (x - x_{n-1}) f[x_0, x_1, \dots, x_n]$$

Newton–Horner–Schema

$$\Phi(x) = f[x_0] + (x - x_0) (f[x_0, x_1] + (x - x_1) (f[x_0, x_1, x_2] + \dots + (x - x_{n-2}) (f[x_0, x_1, \dots, x_{n-1}] + (x - x_{n-1}) f[x_0, x_1, \dots, x_n]) \dots))$$

Beispiel vgl. Beispiel 1.11



$$\Phi(x) = \frac{1}{2} + (x - (-1)) \cdot \frac{1}{2} + (x - (-1))(x - 0) \cdot \frac{1}{4} = \frac{1}{2} + (x + 1) \cdot \left(\frac{1}{2} + x \cdot \frac{1}{4}\right)$$

$$\Phi\left(\frac{1}{2}\right) = \frac{1}{2} + \left(\frac{1}{2} + 1\right) \cdot \left(\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4}\right) = \frac{1}{2} + \frac{3}{2} \cdot \frac{5}{8} = \frac{23}{16}$$

Algorithmen

Steigungsschema

```
for i = n : -1 : 0
  | a_i := f_i
  | for j = i + 1 : n
  |   | a_j := (a_j - a_{j-1}) / (x_j - x_i)
```

Newton–Horner–Schema

```
p := a_n
for i = n - 1 : -1 : 0
  | p := a_i + (x - x_i) p
```

Satz 1.13 (Restglied der Polynominterpolation)

Sei $f \in C^{n+1}[a, b]$ und Φ das Interpolationspolynom zu den Stützstellen

$$a \leq x_0 < x_1 < \dots < x_n \leq b,$$

d. h. $\Phi(x_k) = f(x_k)$, ($k = 0, 1, \dots, n$). Dann gibt es zu jedem $\bar{x} \in [a, b]$ ein $\xi \in [a, b]$ mit

$$f(\bar{x}) - \Phi(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n).$$

Beweis Die Behauptung ist trivial für $x = x_k$, ($k = 0, 1, \dots, n$).

Andernfalls betrachtet man

$$g(x) := f(x) - \Phi(x) - K(x - x_0)(x - x_1) \cdots (x - x_n)$$

mit

$$K := \frac{f(\bar{x}) - \Phi(\bar{x})}{(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)}.$$

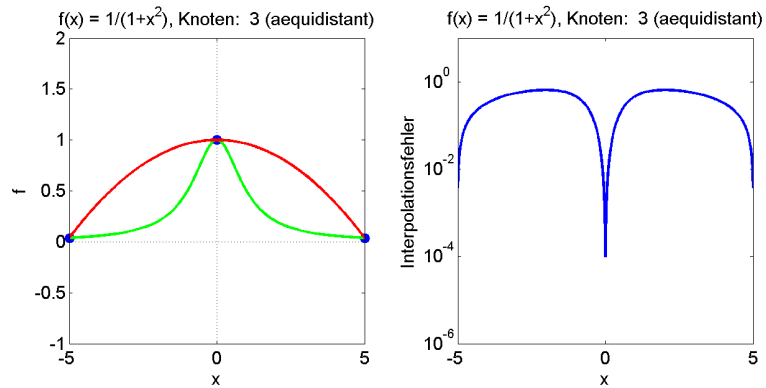
Die Funktion g hat in $[a, b]$ (mindestens) $n+2$ Nullstellen: $x_0, x_1, \dots, x_n, \bar{x}$. Nach dem Satz von Rolle hat g' mindestens $n+1$ Nullstellen usw. und schließlich $g^{(n+1)}(x)$ mindestens eine Nullstelle $\xi \in [a, b]$. Wegen

$$\begin{aligned} 0 &= g^{(n+1)}(\xi) \\ &= f^{(n+1)}(\xi) - \underbrace{\frac{d^{n+1}\Phi(x)}{dx^{n+1}} \Big|_{x=\xi}}_{= 0, \text{ da } \Phi \in \Pi_n} - K \cdot \frac{d^{n+1}}{dx^{n+1}} \left((x - x_0)(x - x_1) \cdots (x - x_n) \right) \Big|_{x=\xi} \\ &= f^{(n+1)}(\xi) - K \cdot (n+1)! \end{aligned}$$

folgt schließlich $K = \frac{f^{(n+1)}(\xi)}{(n+1)!}$ und hieraus die Behauptung. ■

Beispiel 1.15: Funktion von Runge

Beispiel $n = 2$, $x_0 = -5$, $x_1 = 0$, $x_2 = 5$
 $y_j = 1/(1 + x_j^2)$, ($j = 0, 1, 2$)

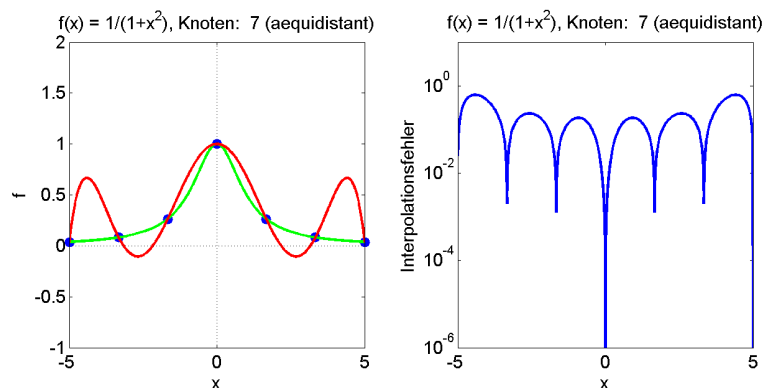


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.8: Interpolation der Funktion von Runge: $\Phi^{(2)}(x)$, Stützstellen äquidistant.

Beispiel 1.15: Funktion von Runge (II)

Beispiel $n = 6$, $x_j = -5 + 10j/6$, ($j = 0, 1, \dots, 6$)
 $y_j = 1/(1 + x_j^2)$, ($j = 0, 1, \dots, 6$)

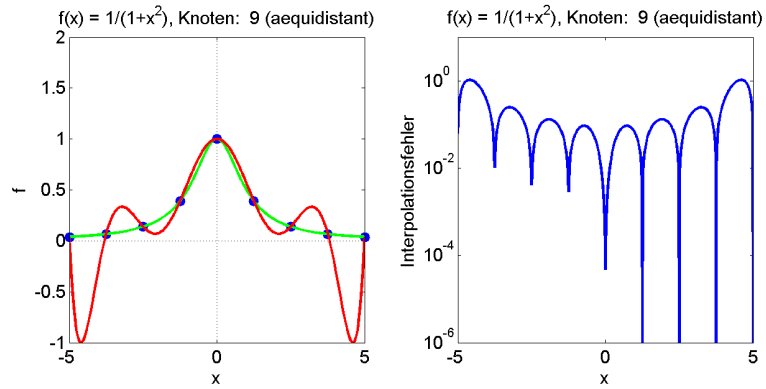


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.9: Interpolation der Funktion von Runge: $\Phi^{(6)}(x)$, Stützstellen äquidistant.

Beispiel 1.15: Funktion von Runge (III)

Beispiel $n = 8, x_j = -5 + 10j/8, (j = 0, 1, \dots, 8)$
 $y_j = 1/(1 + x_j^2), (j = 0, 1, \dots, 8)$

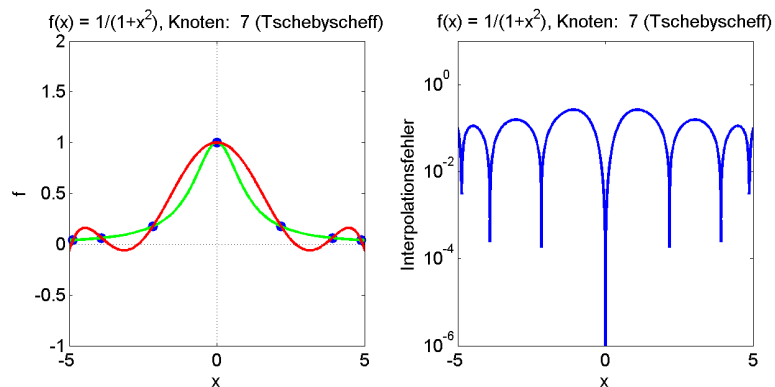


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.10: Interpolation der Funktion von Runge: $\Phi^{(8)}(x)$, Stützstellen äquidistant.

Beispiel 1.15: Funktion von Runge (IV)

Beispiel $n = 6, x_j = 5 \cos \frac{(2j+1)\pi}{14}, (j = 0, 1, \dots, 6)$
 $y_j = 1/(1 + x_j^2), (j = 0, 1, \dots, 6)$

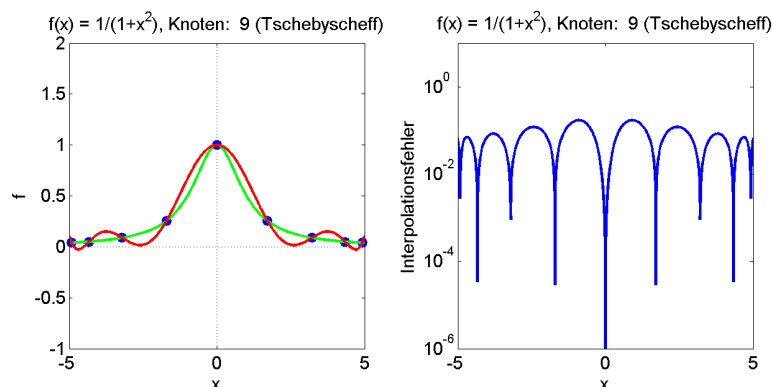


Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.11: Interpolation der Funktion von Runge: $\Phi^{(6)}(x)$, Tschebyscheff-Stützstellen.

Beispiel 1.15: Funktion von Runge (V)

Beispiel $n = 8$, $x_j = 5 \cos \frac{(2j+1)\pi}{18}$, ($j = 0, 1, \dots, 8$)
 $y_j = 1/(1+x_j^2)$, ($j = 0, 1, \dots, 8$)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 1.12: Interpolation der Funktion von Runge: $\Phi^{(8)}(x)$, Tschebyscheff-Stützstellen.

Bemerkung 1.14 (Wahl der Stützstellen)

a) Nach Satz 1.13 sollte man die Stützstellen möglichst so wählen, dass

$$\max_{x \in [a,b]} |(x - x_0)(x - x_1) \cdots (x - x_n)| \rightarrow \min$$

Typische Stützstellen:

- äquidistant: $x_j := a + jh$, Schrittweite $h := \frac{b-a}{n}$
- Tschebyscheff-Nullstellen: $[a, b] = [-1, 1]$

$$x_j := \cos \frac{(2j+1)\pi}{2n+2}, \quad (j = 0, 1, \dots, n)$$

- b) Zu jeder Folge von Stützstellen lässt sich ein $f \in C[a, b]$ angeben, so dass die zugehörige Folge der Interpolationspolynome nicht gleichmäßig konvergiert (Satz von Faber).
- c) Praktische Erfahrung: Polynome hohen Grades neigen zu Oszillationen und sollten vermieden werden.

Beispiel 1.15 (Funktion von Runge)

Äquidistante Stützstellen sind ungeeignet zur Interpolation der *Funktion von Runge*

$$f(x) = \frac{1}{1+x^2}, \quad (x \in [-5, 5])$$

(vgl. Abb. 1.8, 1.9 und 1.10). Bessere Ergebnisse für Tschebyscheff–Stützstellen (vgl. Abb. 1.11 und 1.12).

Bemerkung 1.16 (Hermite–Interpolation)

Klassische Polynominterpolation Bestimme zu gegebenen Stützstellen x_j und gegebenen Stützwerten y_j ein Polynom $\Phi \in \Pi_n$, das die $n + 1$ Interpolationsbedingungen

$$\Phi(x_j) = y_j, \quad (j = 0, 1, \dots, n)$$

erfüllt.

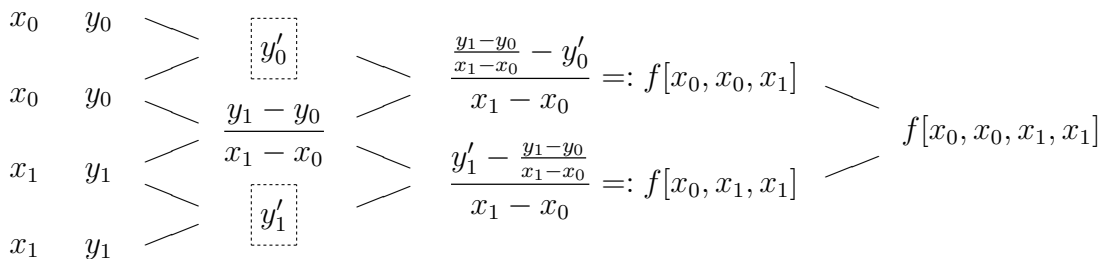
(Klassische) Hermite–Interpolation Bestimme zu gegebenen Stützstellen x_j und gegebenen Stützwerten (y_j, y'_j) ein Polynom $\Phi \in \Pi_{2n+1}$, das die $2(n + 1)$ Interpolationsbedingungen

$$\Phi(x_j) = y_j, \quad \Phi'(x_j) = y'_j, \quad (j = 0, 1, \dots, n)$$

erfüllt.

Newtonsche Darstellung Füge die Stützpunkte (x_j, y_j) , $(j = 0, 1, \dots, n)$ jeweils zweimal in das Steigungsschema ein und ersetze auftretende Quotienten „0/0“ durch die vorgegebenen Funktionswerte y'_j der Ableitung $\Phi'(x)$.

Beispiel $n = 1$



mit

$$f[x_0, x_0, x_1, x_1] := \frac{f[x_0, x_1, x_1] - f[x_0, x_0, x_1]}{x_1 - x_0}$$

Ergebnis

$$\Phi(x) = y_0 + (x - x_0) \cdot y'_0 + (x - x_0)^2 \cdot f[x_0, x_0, x_1] + (x - x_0)^2(x - x_1) \cdot f[x_0, x_0, x_1, x_1]$$

Effiziente Auswertung mit Newton–Horner–Schema.

Höhere Ableitungen Interpolationsbedingungen

$$\frac{d^k \Phi}{dx^k}(x_j) = y_j^{(k)}, \quad (j = 0, 1, \dots, n; k = 0, 1, \dots, c_j)$$

bestimmen das Hermite-Interpolationspolynom $\Phi \in \Pi_r$ mit $r + 1 = \sum_{j=0}^n (1 + c_j)$.

2 Lineare Gleichungssysteme

2.1 Gaußscher Algorithmus

Bemerkung 2.1 (Aufgabenstellung)

geg.: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $n = 10^1 \dots 10^7$

ges.: Lösung x des linearen Gleichungssystems $Ax = b$

formal $x = A^{-1}b$, numerisch ungeeignet (Rechenaufwand zur Auswertung von A^{-1})

Lösbarkeit $x \in \mathbb{R}^n$ genau dann eindeutig bestimmt, wenn A regulär

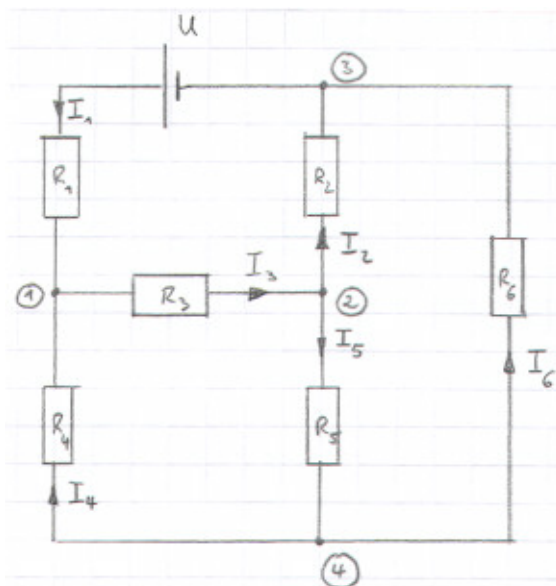
$n = 2, n = 3$ Lösung mittels Cramerscher Regel

Bemerkung 2.2 (Lineare Netzwerkmodelle)

Modellbildung komplexer Systeme (Technik, Umwelt)

Basisbausteine, verknüpft durch Ein-/Ausgangsbeziehungen und Erhaltungsgrößen

Beispiel elektrische Schaltungen, Chip-Design



Basisbaustein: Widerstand

Ohmsches Gesetz

$$U_i = R_i \cdot I_i, \quad (i = 1, 2, \dots, 6)$$

1. Kirchhoffsche Regel

In jedem Knoten: Summe der zufließenden gleich Summe der abfließenden Ströme

2. Kirchhoffsche Regel

In jeder Masche: Summe der Spannungsabfälle gleich Summe der Quellspannungen

Ergebnis Lineares Gleichungssystem

$$\begin{array}{l}
 \textcircled{1} \\
 \textcircled{2} \\
 \textcircled{3} \\
 \textcircled{4} \\
 [\textcircled{1}, \textcircled{4}, \textcircled{3}] \\
 [\textcircled{2}, \textcircled{3}, \textcircled{4}] \\
 [\textcircled{1}, \textcircled{2}, \textcircled{4}] \\
 [\textcircled{1}, \textcircled{2}, \textcircled{3}]
 \end{array}
 \begin{pmatrix}
 1 & 0 & -1 & 1 & 0 & 0 \\
 0 & -1 & 1 & 0 & -1 & 0 \\
 -1 & 1 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & -1 & 1 & -1 \\
 R_1 & 0 & 0 & -R_4 & 0 & R_6 \\
 0 & R_2 & 0 & 0 & -R_5 & -R_6 \\
 0 & 0 & R_3 & R_4 & R_5 & 0 \\
 R_1 & R_2 & R_3 & 0 & 0 & 0
 \end{pmatrix}
 \begin{pmatrix}
 I_1 \\
 I_2 \\
 I_3 \\
 I_4 \\
 I_5 \\
 I_6
 \end{pmatrix}
 =
 \begin{pmatrix}
 0 \\
 0 \\
 0 \\
 0 \\
 U \\
 0 \\
 0 \\
 U
 \end{pmatrix}$$

Streicht man die redundanten Gleichungen $\textcircled{4} = -\textcircled{1} - \textcircled{2} - \textcircled{3}$ und

$$[\textcircled{1}, \textcircled{2}, \textcircled{3}] = [\textcircled{1}, \textcircled{4}, \textcircled{3}] + [\textcircled{2}, \textcircled{3}, \textcircled{4}] + [\textcircled{1}, \textcircled{2}, \textcircled{4}],$$

so ergibt sich $Ax = b$ mit $A \in \mathbb{R}^{6 \times 6}$, $x = (I_1, I_2, \dots, I_6)^\top \in \mathbb{R}^6$, $b = (0, 0, 0, U, 0, 0)^\top \in \mathbb{R}^6$.

Beachte

- Anteil der Nichtnullelemente in A gering \Rightarrow „schwach besetzte“ Matrizen
- Position der Nichtnullelemente a_{ij} in A ergibt sich aus der sog. Topologie der Schaltung

Bemerkung 2.3 (Gestaffelte Gleichungssysteme)

Spezialfall $Rx = z$ mit regulärer oberer Dreiecksmatrix $R = (r_{ij}) \in \mathbb{R}^{n \times n}$,

d. h., $r_{ii} \neq 0$, $r_{ij} = 0$, ($i = 1, \dots, n$; $j = 1, \dots, i - 1$).

$$\left. \begin{array}{l}
 r_{11}x_1 + r_{12}x_2 + \dots + r_{1n}x_n = z_1 \\
 r_{22}x_2 + \dots + r_{2n}x_n = z_2 \\
 \quad \quad \quad \ddots \quad \quad \quad \vdots = \quad \quad \quad \vdots \\
 \quad \quad \quad \quad \quad \quad r_{nn}x_n = z_n
 \end{array} \right\} Rx = z$$

Beispiel

$$\left. \begin{aligned} 10x_1 - 7x_2 + 0 \cdot x_3 &= 7 \\ 2.5x_2 + 5 \cdot x_3 &= 2.5 \\ 6.2 \cdot x_3 &= 6.2 \end{aligned} \right\}$$

$$\Rightarrow x_3 = \frac{6.2}{6.2} = 1, \quad x_2 = \frac{2.5 - 5 \cdot 1}{2.5} = -1, \quad x_1 = \frac{7 + 7 \cdot (-1)}{10} = 0$$

$$x = (0, -1, 1)^\top \quad \dots \quad \text{Rückwärtssubstitution}$$

allgemein

$$x_n = \frac{z_n}{r_{nn}}, \quad x_i = \frac{z_i - \sum_{j=i+1}^n r_{ij}x_j}{r_{ii}}, \quad (i = n-1, n-2, \dots, 1)$$

Algorithmus

```
for i = n : -1 : 1
    s := z_i
    for j = (i + 1) : n
        s := s - r_ij * x_j
    end
    x_i := s / r_ii
end
```

Matlab-Code

```
x = zeros(size(z));
x(n) = z(n)/r(n,n);
for i=n-1:-1:1,
    x(i) = ( z(i) - r(i,i+1:n) * x(i+1:n) ) / r(i,i);
end;
```

analog $Lx = z$ mit regulärer unterer Dreiecksmatrix $L = (l_{ij}) \in \mathbb{R}^{n \times n}$,
d. h., $l_{ii} \neq 0$, $l_{ij} = 0$, ($i = 1, \dots, n$; $j = i + 1, \dots, n$) \Rightarrow „Vorwärtssubstitution“.

Bemerkung 2.4 (Gaußscher Algorithmus)

Idee Gleichungssystem $Ax = b$ in äquivalentes gestaffeltes Gleichungssystem umformen durch

- Multiplikation einer Gleichung mit einer von Null verschiedenen Zahl,
- Addition des Vielfachen einer Gleichung zu einer anderen Gleichung und / oder
- Vertauschung von Gleichungen.

Beispiel

$$\left. \begin{array}{rcl} 10x_1 - 7x_2 & = & 7 \\ -3x_1 + 2x_2 + 6x_3 & = & 4 \\ 5x_1 - x_2 + 5x_3 & = & 6 \end{array} \right\}$$

Schritt k Addiere Vielfache der k -ten Gleichung zu den Gleichungen $k+1, \dots, n$, so dass in der k -ten Spalte unterhalb der Hauptdiagonale die Nichtnullelemente eliminiert werden \rightsquigarrow **engl.:** Gaussian elimination

$k = 1$

$$\left. \begin{array}{rcl} 10x_1 - 7x_2 & = & 7 \\ \text{II}' = +\frac{3}{10} * \text{I} + \text{II} : & -0.1x_2 + 6x_3 & = 6.1 \\ \text{III}' = -\frac{5}{10} * \text{I} + \text{III} : & 2.5x_2 + 5x_3 & = 2.5 \end{array} \right\}$$

$k = 2$

$$\left. \begin{array}{rcl} 10x_1 - 7x_2 & = & 7 \\ & -0.1x_2 + 6x_3 & = 6.1 \\ \text{III}'' = +25 * \text{II}' + \text{III}' : & 155x_3 & = 155 \end{array} \right\}$$

Rücksubstitution $\Rightarrow x = (0, -1, 1)^\top$.

Problem Hauptdiagonalelement $a_{kk}^{(k)} = 0$ im k -ten Eliminationsschritt

Lösung Vertauschung der k -ten Gleichung mit einer der Gleichungen $k+1, \dots, n$ so, dass *Pivotelement* $a_{lk}^{(k)} \neq 0$ (stets möglich, falls A regulär).

Strategie Bestimme im k -ten Eliminationsschritt $p \in \{k, k+1, \dots, n\}$ so, dass

$$|a_{pk}^{(k)}| = \max \{ |a_{lk}^{(k)}| : l = k, k+1, \dots, n \}$$

und vertausche k -te und p -te Gleichung \Rightarrow (Spalten)-*Pivotisierung*

\rightsquigarrow auch vorteilhaft zur Verringerung des Einflusses von Rundungsfehlern

Beispiel $k = 2$, tausche Gleichungen $\text{II}' \Leftrightarrow \text{III}'$

$$\left. \begin{array}{rcl} 10x_1 - 7x_2 & = & 7 \\ \widetilde{\text{II}}' = \text{III}' : & 2.5x_2 + 5x_3 & = 2.5 \\ \widetilde{\text{III}}' = \text{II}' : & -0.1x_2 + 6x_3 & = 6.1 \end{array} \right\}$$

$$\left. \begin{array}{rcl} 10x_1 - 7x_2 & = & 7 \\ & 2.5x_2 + 5x_3 & = 2.5 \\ \widetilde{\text{III}}'' = +\frac{1}{25} * \widetilde{\text{II}}' + \widetilde{\text{III}}' : & 6.2x_3 & = 6.2 \end{array} \right\}$$

Algorithmus

```

for k = 1 : n - 1
    p := k; s := |akk|
    for i = k + 1 : n
        | if |aik| > s then p := i; s := |aik|
    for j = k : n
        | s := akj; akj := apj; apj := s
    s := bk; bk := bp; bp := s
    for i = k + 1 : n
        | lik := aik/akk;
        | bi := bi - lik · bk
        for j = k + 1 : n
            | aij := aij - lik · akj

```

Bemerkung 2.5 (LU-Zerlegung)

k-ter Eliminationsschritt in Matrixschreibweise (ohne Pivotisierung)

$A^{(k)} \rightsquigarrow A^{(k+1)}$ mit

$$A^{(k)} = \left(\begin{array}{c|c} k-1 & n-k+1 \\ \hline \begin{array}{c} \diagdown \\ \diagup \end{array} & \\ \hline n-k+1 & \end{array} \right)_{k-1}^{n-k+1}, \quad A^{(k+1)} = \left(\begin{array}{c|c} \begin{array}{c} \diagdown \\ \diagup \end{array} & \\ \hline * \\ 0 \\ \vdots \\ 0 & \\ \hline n-k & \end{array} \right)_{n-k}$$

$$A^{(k+1)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & -l_{k+1,k} & 1 & \\ & & & \vdots & & \ddots \\ & & & -l_{n,k} & & 1 \end{pmatrix} \cdot A^{(k)} = (I - L^{(k)})A^{(k)}$$

mit

$$L^{(k)} = \begin{pmatrix} 0 & & & & & \\ & \ddots & & & & \\ & & 0 & & & \\ & & l_{k+1,k} & 0 & & \\ & & \vdots & & \ddots & \\ & & l_{n,k} & & & 0 \end{pmatrix}, \quad A^{(1)} := A, \quad A^{(n)} := U \text{ (obere Dreiecksmatrix),}$$

$$(I - L^{(n-1)}) \dots (I - L^{(1)})A = U.$$

Auflösen nach A

$$A = LU \quad \text{mit} \quad L := (I - L^{(1)})^{-1} \cdots (I - L^{(n-1)})^{-1}.$$

Wegen $L^{(i)}L^{(j)} = 0$, ($i \leq j$), ist

$$(I - L^{(i)})(I + L^{(i)}) = I - L^{(i)} + L^{(i)} = I \quad \Rightarrow \quad (I - L^{(i)})^{-1} = (I + L^{(i)}).$$

Außerdem erhält man $(I + L^{(i)})(I + L^{(j)}) = I + L^{(i)} + L^{(j)}$, ($i \leq j$), also insgesamt

$$L = (I + L^{(1)}) \cdots (I + L^{(n-1)}) = I + L^{(1)} + L^{(2)} + \cdots + L^{(n-1)} \quad \dots \text{untere Dreiecksmatrix.}$$

Ergebnis Gauß-Algorithmus berechnet LU -Zerlegung von A : $A = L \cdot U$ mit der oberen Dreiecksmatrix U aus dem gestaffelten linearen Gleichungssystem und der unteren Dreiecksmatrix L , die die Eliminationskoeffizienten l_{ik} enthält und deren Hauptdiagonalelemente $= 1$ sind.

praktisch Abspeicherung der Nicht-Diagonalelemente von L unterhalb der Hauptdiagonalen von A .

Pivotisierung Spaltenpivotisierung $\Rightarrow LU = PA$ mit *Permutationsmatrix* P

Beispiel wie oben

$$\begin{array}{ccc} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} & = & \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ -3/10 & -1/25 & 1 \end{pmatrix} & \begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix} \\ P & A & & L & U \end{array}$$

Lösung linearer Gleichungssysteme

Schritt 1 Berechne mittels Gauß-Algorithmus mit Spaltenpivotisierung die LU -Zerlegung $PA = LU$.

Für jede rechte Seite: **Schritt 2** Vorwärtssubstitution

$$Ly = Pb \rightsquigarrow y$$

Schritt 3 Rückwärtssubstitution

$$Ux = y \rightsquigarrow x$$

Software • Matlab-Kommando `lu`

- LAPACK (FORTRAN, C), über <http://www.netlib.org>
DGETRF, DGETRS

Rechenaufwand Gemessen in flops ... Floating point operations
(1 Gleitpunktaddition, 1 Gleitpunktmultiplikation)

Schritt 1 $\frac{n^3}{3} + \mathcal{O}(n^2)$ Rechenoperationen, vgl. Bemerkung 1.6

Schritt 2+3 jeweils $\frac{n^2}{2} + \mathcal{O}(n)$ Rechenoperationen $\Rightarrow n^2 + \mathcal{O}(n)$ Rechenoperationen pro linearem Gleichungssystem

Bemerkung 2.6 (Symmetrische Koeffizientenmatrizen, Cholesky-Zerlegung)
 Wichtiger Spezialfall: $Ax = b$ mit $A = A^\top$, insbesondere auch symmetrische, positiv definite Koeffizientenmatrizen A , d. h.

$$x^\top Ax > 0, \quad (x \in \mathbb{R}^n \setminus \{0\}).$$

$A = A^\top$ Gauß-Algorithmus ohne Pivotisierung $\rightsquigarrow A = L \cdot U$

Sei $D := \text{diag}_{1 \leq i \leq n} u_{ii} \Rightarrow D^{-1}U$ ist obere Dreiecksmatrix mit Hauptdiagonalelementen = 1.

$$A^\top = (L \cdot D \cdot D^{-1}U)^\top = (D^{-1}U)^\top DL^\top$$

Aus der Eindeutigkeit der LU-Zerlegung (!) folgt $D^{-1}U = L^\top$, also $A = LDL^\top$.

Berechnung mit $\frac{n^3}{6} + \mathcal{O}(n^2)$ Rechenoperationen möglich.

Pivotisierung: gleichzeitiger Zeilen- und Spaltentausch, um Symmetrie zu erhalten.

A symmetrisch, positiv definit Man zeigt, dass der Gauß-Algorithmus ohne Pivotisierung stets durchführbar ist. Wegen $0 < y^\top Ay$ für $y := (L^\top)^{-1}x$ ist

$$0 < y^\top Ay = ((L^\top)^{-1}x)^\top (LDL^\top) ((L^\top)^{-1}x) = (L^\top (L^\top)^{-1}x)^\top D (L^\top (L^\top)^{-1}x) = x^\top Dx,$$

also ist auch D positiv definit: $D = \text{diag}_{1 \leq i \leq n} d_i$ mit $d_i > 0$.

Cholesky-Zerlegung von A

$$A = \hat{L}\hat{L}^\top \quad \text{mit} \quad \hat{L} := L \cdot D^{1/2}, \quad D^{1/2} = \text{diag}_i \sqrt{d_i}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} \hat{l}_{11} & & & \\ \hat{l}_{21} & \hat{l}_{22} & & \\ \vdots & \vdots & \ddots & \\ \hat{l}_{n1} & \hat{l}_{n2} & \cdots & \hat{l}_{nn} \end{pmatrix} \begin{pmatrix} \hat{l}_{11} & \hat{l}_{21} & \cdots & \hat{l}_{n1} \\ & \hat{l}_{22} & \cdots & \hat{l}_{n2} \\ & & \ddots & \vdots \\ & & & \hat{l}_{nn} \end{pmatrix}$$

Algorithmus

```

for k = 1 : n
    |
    |  $\hat{l}_{kk} := \left( a_{kk} - \sum_{j=1}^{k-1} \hat{l}_{kj}^2 \right)^{1/2}$ 
    |
    | for i = (k + 1) : n
    | |  $\hat{l}_{ik} := \left( a_{ik} - \sum_{j=1}^{k-1} \hat{l}_{ij} \hat{l}_{kj} \right) / \hat{l}_{kk}$ 
    |

```

Vorwärts- / Rückwärtssubstitution

wie im allgemeinen Fall, beachte jedoch, dass nur \hat{L} , nicht \hat{L}^\top abgespeichert wird.

Rechenaufwand $\frac{n^3}{6} + \mathcal{O}(n^2)$ Rechenoperationen, n Quadratwurzeln

2.2 Lineare Ausgleichsrechnung

Bemerkung 2.7 (Methode der kleinsten Quadrate)

geg.: $A \in \mathbb{R}^{m \times n}$, $m > n$, $\text{rank}(A) = n$, $b \in \mathbb{R}^m$

ges.: „Lösung“ $x \in \mathbb{R}^n$ von $Ax = b$

Methode der kleinsten Quadrate (Gauß)

Da i. Allg. keine *klassische Lösung* $x \in \mathbb{R}^n$ mit $Ax - b = 0$ existiert, sucht man als *verallgemeinerte Lösung* ein $x \in \mathbb{R}^n$ so, dass

$$\|Ax - b\|_2 \rightarrow \min . \quad (*)$$

Hierbei bezeichnet $\|v\|_2 := \sqrt{v^\top v} = \left(\sum_{i=1}^m v_i^2\right)^{1/2}$ die *euklidische Vektornorm* des Vektors $v = (v_i)_{i=1}^m \in \mathbb{R}^m$, vgl. Abschnitt 3.2.

Eigenschaft: Für Vektoren $y = (y_i) \in \mathbb{R}^n$, $z = (z_i) \in \mathbb{R}^{m-n}$ gilt

$$\left\| \begin{pmatrix} y \\ z \end{pmatrix} \right\|_2^2 = \sum_{i=1}^n y_i^2 + \sum_{i=1}^{m-n} z_i^2 = \|y\|_2^2 + \|z\|_2^2 .$$

Das *Kleinste-Quadrate-Problem* (*) ist äquivalent zu

$$\|Ax - b\|_2^2 = \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij}x_j - b_i \right)^2 \rightarrow \min .$$

Notwendige Bedingung für Minimum

$$0 \stackrel{!}{=} \frac{\partial}{\partial x_k} \|Ax - b\|_2^2 = 2 \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij}x_j - b_i \right) a_{ik} , \quad (k = 1, \dots, n) ,$$

$$\sum_{i=1}^m a_{ik} \left(\sum_{j=1}^n a_{ij}x_j \right) = \sum_{i=1}^m a_{ik}b_i , \quad (k = 1, \dots, n) ,$$

Gaußsche Normalgleichungen

$$A^\top Ax = A^\top b$$

Wegen $A^\top A = (A^\top A)^\top \in \mathbb{R}^{n \times n}$ und $\text{rank}(A) = n$ ist $A^\top A$ symmetrisch und positiv definit:

$$\xi \in \mathbb{R}^n \setminus \{0\} \Rightarrow \xi^\top (A^\top A) \xi = (\xi^\top A^\top)(A\xi) = (A\xi)^\top (A\xi) = \|A\xi\|_2^2 > 0 ,$$

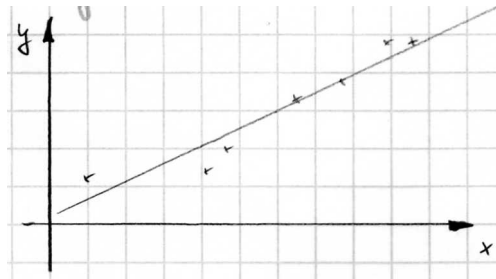
denn $A\xi \neq 0$ wegen $\xi \neq 0$ und $\text{rank}(A) = n$.

Lösung der Normalgleichungen mittels Cholesky-Zerlegung

Problem Bei Verwendung der Gaußschen Normalgleichungen reagiert die numerische Lösung oft sehr empfindlich auf Rundungsfehler.

Alternative Orthogonalisierungsverfahren, vgl. Bemerkung 2.10.

Beispiel 2.8 (Lineare Regression)



geg.: Messdaten (x_i, y_i) , $(i = 1, \dots, m)$, mit Messfehlern behaftet

ges.: Gerade $y = ax + b$, die die Messwerte „möglichst gut“ approximiert:
 $y_i \approx a + bx_i$, $(i = 1, \dots, m)$

Ansatz $\sum_{i=1}^m (a + bx_i - y_i)^2 \rightarrow \min$

Matrixschreibweise

$$A \begin{pmatrix} a \\ b \end{pmatrix} = y \quad \text{mit} \quad y = (y_1, \dots, y_m)^\top \in \mathbb{R}^m, \quad A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \in \mathbb{R}^{m \times 2}, \quad n = 2$$

Normalgleichungen $A^\top A \cdot \begin{pmatrix} a \\ b \end{pmatrix} = A^\top y$

$$\begin{pmatrix} m & \sum_{j=1}^m x_j \\ \sum_{j=1}^m x_j & \sum_{j=1}^m x_j^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^m y_j \\ \sum_{j=1}^m x_j y_j \end{pmatrix} \Rightarrow a, b \in \mathbb{R}$$

Bemerkung 2.9 (Orthogonale Transformationen)

Idee Verwende orthogonale Transformationen, um lineare Gleichungssysteme und lineare Ausgleichsprobleme in äquivalente Probleme einfacherer Gestalt umzuformen.

QR-Zerlegung $A = QR$ mit der orthogonalen Matrix

$$Q := G_{21}^\top G_{31}^\top \cdots G_{m,n-1}^\top G_{n+1,n}^\top \cdots G_{m,n}^\top.$$

Wegen $n = \text{rank}(A) = \text{rank}(R) = \text{rank}(\tilde{R})$ ist $\tilde{R} \in \mathbb{R}^{n \times n}$ regulär.

Lösung regulärer linearer Gleichungssysteme

1. QR-Zerlegung mittels Givens-Drehungen, $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ Rechenoperationen
2. $Ax = b \Leftrightarrow QRx = b \Leftrightarrow Rx = z, Qz = b$
 - 2a) $z = Q^\top b = G_{n,n-1}(G_{n,n-2}(G_{n-1,n-2}(\cdots G_{31}(G_{21}b) \cdots)))$
Berechnung durch sukzessive Auswertung von Matrix-Vektor-Produkten
 $G_{kl}w$, keine explizite Berechnung von Q
 - 2b) Löse $Rx = z$ mittels Rücksubstitution.

Besonders geeignet für Gleichungssysteme $Ax = b$, deren Lösung sehr empfindlich gegenüber Rundungsfehlern ist, und zur numerischen Rangbestimmung von A .

Lösung von Kleinste-Quadrate-Problemen

Wegen $\|w\|_2^2 = w^\top w$ ist $\|w\|_2$ invariant gegenüber orthogonalen Transformationen:

$$\|Qw\|_2^2 = (Qw)^\top (Qw) = w^\top (Q^\top Q)w = w^\top w = \|w\|_2^2$$

$$\Rightarrow \|Ax - b\|_2^2 = \|QRx - b\|_2^2 = \|Q(Rx - Q^\top b)\|_2^2 = \|Rx - Q^\top b\|_2^2 = \|\tilde{R}x - z_1\|_2^2 + \|z_2\|_2^2$$

$$\text{mit } Q^\top b = \begin{pmatrix} z_1 \\ \cdots \\ z_2 \end{pmatrix} \begin{matrix} n \\ \\ m-n \end{matrix} \in \mathbb{R}^m.$$

Lösung: Berechne $Q^\top b$ und löse $\tilde{R}x = z_1$ mittels Rücksubstitution.

Beispiel 2.11 (Neuronale Netze)

Idee Entscheidungen fallen auf Grundlage einer Vielzahl von Einzelinformationen in Anlehnung an Entscheidungsprozesse im menschlichen Hirn.

Beispiel Mensch Beim Verlassen des Hauses Regenschirm mitnehmen?

Aktuelles Wetter, Wettervorhersage, Wind, Wolken, voraussichtliche Dauer des Aufenthalts im Freien, eigener Gesundheitszustand (Erkältung), ...

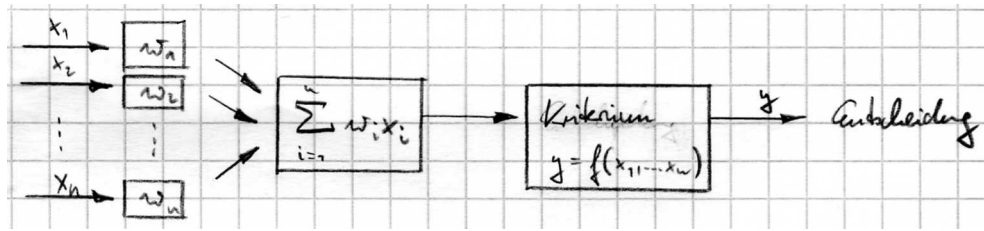
Grundlage der Entscheidung: Erfahrung

Konsequenz einer Fehlentscheidung: Lerneffekt

Beispiel Technik Wäsche in Waschmaschine „stark verschmutzt“

Messung der Wassertemperatur und Wassertrübung an verschiedenen Punkten

Neuronales Netz (Ein-Schicht-Modell, linear)



Verhalten des Netzes bestimmt durch *Gewichte* w_1, \dots, w_n

Training des Netzes Wähle w_1, \dots, w_n so, dass eine große Zahl von Tests mit vorgegebenen Eingangsdaten $(x_1^{(j)}, \dots, x_n^{(j)})^\top$ und bekannten Resultaten $y^{(j)}$ möglichst gut wiedergegeben wird:

$$\sum_{i=1}^n x_i^{(j)} w_i \approx y^{(j)}, \quad (j = 1, \dots, m)$$

↪ überbestimmtes lineares Gleichungssystem, Bestimmung von (w_1, \dots, w_n) als Kleinste-Quadrate-Lösung

praktisch Lösung der Normalgleichungen oder Lösung mittels QR-Zerlegung.

3 Rechnerarithmetik und Rundungsfehler

3.1 Gleitpunktarithmetik

Bemerkung 3.1 (Gleitpunktzahlen)

a) Ganzzahlige Datentypen (INTEGER) mit exakter Arithmetik

$$-\text{MaxInt} - 1, \dots, -1, 0, 1, \dots, \text{MaxInt},$$

z. B. für Indizes in Laufanweisungen.

b) *Normalisierte Gleitpunktdarstellung* (engl.: floating point numbers) zur Darstellung reeller Zahlen

$$F := \{ y : y = \pm m * \beta^{e-t} \} \subset \mathbb{R}$$

mit β ... Basis (meist 2, 8 oder 16),

t ... Mantissenlänge,

e ... Exponent, $e_{\min} \leq e \leq e_{\max}$

m ... Mantisse (ganzzahlig), $m = 0$ oder $\beta^{t-1} \leq m < \beta^t$

Schreibweise $y = \pm \beta^e * [0.d_1 d_2 \dots d_t]_\beta := \pm \beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t} \right)$

mit Mantisse $m = d_1 d_2 \dots d_t$ oder $m = 0$.

3. Rechnerarithmetik und Rundungsfehler



Beispiel Zahlendarstellung in Matlab

```

>> format long e           % Datenausgabe mit vielen Dezimalstellen

>> 1                       % Exakte Darstellung ganzer Zahlen
ans = 1

>> 1 - 1                   % Exakte Arithmetik für ganze Zahlen
ans = 0

>> 1 - 1 + 1.0e-15        % Beim Rechnen mit reellen Zahlen können
ans = 1.000000000000000e-015 % Rundungsfehler auftreten, müssen aber nicht.

>> 1 + 1.0e-15 - 1        % Reihenfolge der Rechenschritte ist wesentlich
ans = 1.110223024625157e-015

>> 1 + 1.0e-8 - 1        % Groessenordnung der Rundungsfehler: ca. 1.0e-16
ans = 9.999999939225290e-009

>> sqrt(2)^2 - 2          % Groessenordnung der Rundungsfehler: ca. 1.0e-16
ans = 4.440892098500626e-016

>> factorial(170)         % Darstellbarer Zahlenbereich nach oben beschaenkt
ans = 7.257415615307994e+306

>> factorial(171)         % Zahl 171! uebersteigt darstellbaren Zahlenbereich
ans = Inf
    
```



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

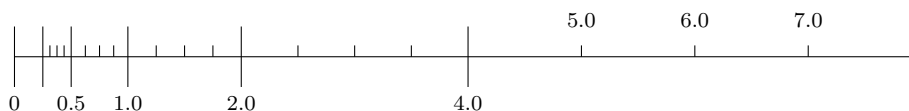
Abbildung 3.1: Rechnen in Gleitpunktarithmetik: Beispiel Matlab.

Beispiel $\beta = 2$, $t = 3$, $e_{\min} = -1$, $e_{\max} = 3$

$$F \cap [0, \infty) = \{ 0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875, \\
 1.0, 1.25, 1.50, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0 \}$$

$\begin{matrix} \text{---} e=-1 \text{---} & \text{---} e=0 \text{---} & \dots \\ \text{[0.100]}_2 & \text{[0.101]}_2 & \text{[0.110]}_2 & \text{[0.111]}_2 & \text{[0.100]}_2 \\ & & & & \text{[0.111]}_2 \\ & & & & \dots \text{---} e=3 \text{---} \end{matrix}$

Beachte: Gleitpunktzahlen sind auf der reellen Achse *nicht* gleichverteilt.



IEEE-Standard 754 [1985] Binäre Gleitpunktarithmetik, Quasi-Standard

einfache Genauigkeit (single precision)

4 Byte, $\beta = 2$, $t = 23$, $e_{\min} = -126$, $e_{\max} = 127$

Zahlenbereich: $[1.2_E - 38, 3.4_E + 38]$

doppelte Genauigkeit (double precision)

8 Byte, $\beta = 2$, $t = 52$, $e_{\min} = -1022$, $e_{\max} = 1023$

Zahlenbereich: $[2.2_E - 308, 1.8_E + 308]$

Abstand zweier positiver Gleitpunktzahlen x, \tilde{x} :

Maschinenepsilon $\text{eps} = \beta^{1-t}$... kleinste Maschinenzahl, die zu $1 = [0.10 \dots 0]_\beta * \beta^1$ addiert einen von 1 verschiedenen Wert ergibt

Sind x, \tilde{x} unmittelbar benachbart, so gilt

$$\frac{1}{\beta} \text{eps} |x| \leq |x - \tilde{x}| \leq \text{eps} |x|.$$

Bemerkung 3.2 (Rundung, Rundungsfehler)

a) Sei G die Menge aller y wie in Bemerkung 3.1, jedoch für beliebiges $e \in \mathbb{Z}$, und $\text{fl} : \mathbb{R} \rightarrow G$ eine Abbildung mit

$$|x - \text{fl}(x)| = \min_{\tilde{x} \in G} |x - \tilde{x}|, \quad (x \in \mathbb{R}).$$

Der Übergang $x \mapsto \text{fl}(x)$ heißt *runden*. fl ist nicht eindeutig, praktisch meist: *Gerade-Zahl-Regel*, d. h., für $\tilde{x}_1, \tilde{x}_2 \in G$ mit $\tilde{x}_1 \neq \tilde{x}_2$ und

$$|x - \tilde{x}_1| = |x - \tilde{x}_2| = \min_{\tilde{x} \in G} |x - \tilde{x}|$$

wählt man fl so, dass d_t geradzahlig.

b) praktisch $\text{fl}(x) \stackrel{!}{\in} F$

Exponentenüberlauf (engl.: overflow): $|\text{fl}(x)| > \max \{ |y| : y \in F \}$

Exponentenunterlauf (engl.: underflow): $0 < |\text{fl}(x)| < \min \{ |y| : y \in F, y \neq 0 \}$

c) Zu jedem $x \in \mathbb{R}$ mit $\text{fl}(x) \in F$ ist

$\text{fl}(x) = x(1 + \delta)$ mit einem δ mit $|\delta| < \varepsilon$

und

$\text{fl}(x) = x/(1 + \bar{\delta})$ mit einem $\bar{\delta}$ mit $|\bar{\delta}| \leq \varepsilon$,

wobei $\varepsilon := \frac{1}{2}\beta^{1-t}$ die *Maschinengenauigkeit* (engl.: *unit round-off*) bezeichnet:

$\varepsilon \approx 5.96_{\text{E}} - 8$ (single), $\varepsilon \approx 1.11_{\text{E}} - 16$ (double).

Begründung

Für $x > 0$ ist $x = \mu * \beta^{e-t}$ mit einem $\mu \in [\beta^{t-1}, \beta^t - 1]$ und $e \in [e_{\min}, e_{\max}]$. Unmittelbar benachbarte Gleitpunktzahlen: $\underline{\mu} * \beta^{e-t}, \bar{\mu} * \beta^{e-t}$ mit $\underline{\mu} \leq \mu \leq \bar{\mu}$. Es gilt

$$\begin{aligned} |x - \text{fl}(x)| &= \min \{ |\mu - \underline{\mu}|, |\mu - \bar{\mu}| \} * \beta^{e-t} \\ &\leq \frac{1}{2} |\bar{\mu} - \underline{\mu}| * \beta^{e-t} \leq \frac{1}{2} * \beta^{e-t} = x \cdot \frac{1}{2\mu} \leq x \cdot \varepsilon \end{aligned}$$

Bemerkung 3.3 (Absoluter und relativer Fehler)

a) Der *absolute Fehler* einer Größe mit Soll-Wert $\bar{\xi}$ und Ist-Wert ξ ist

$$\delta\xi := |\xi - \bar{\xi}|,$$

für $\bar{\xi} \neq 0$ ist der zugehörige *relative Fehler* $f_{\text{rel}}(\bar{\xi}) := \frac{|\xi - \bar{\xi}|}{|\bar{\xi}|}$.

b) Der in Bemerkung 3.2c) betrachtete Rundungsfehler erfüllt

$$f_{\text{rel}}(x) = \frac{|\mathbf{fl}(x) - x|}{|x|} \leq \varepsilon.$$

Bemerkung 3.4 (Gleitpunktarithmetik)

Grundrechenarten $\text{op} \in \{+, -, *, /\}$, $\text{op} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

Problem F nicht abgeschlossen bez. op

Anforderung („Standardmodell“)

$$\left. \begin{aligned} x \widetilde{\text{op}} y &= (x \text{ op } y)(1 + \delta) \\ x \widetilde{\text{op}} y &= \frac{x \text{ op } y}{1 + \bar{\delta}} \end{aligned} \right\}, \quad (x \in F),$$

mit $\delta = \delta(x, y; \text{op})$, $\bar{\delta} = \bar{\delta}(x, y; \text{op})$, $|\delta| \leq \varepsilon$, $|\bar{\delta}| \leq \varepsilon$.

praktisch $x \widetilde{\text{op}} y$ „unendlich genau“ (praktisch: „mit größerer Mantissenlänge“) auswerten, anschließend runden auf nächstgelegene Gleitpunktzahl (Gerade-Zahl-Regel).

Alternativen

- Runden auf nächst kleinere bzw. nächst größere Maschinenzahl \rightsquigarrow Intervallarithmetik
- „Abschneiden“ überzähliger Ziffern („chopping“)

Beispiel $\beta = 2$, $t = 3$, $x = \frac{7}{4} = [0.111]_2 * 2^1$, $y = \frac{3}{8} = [0.110]_2 * 2^{-1}$

$$\begin{aligned} x + y &= [0.111]_2 * 2^1 + [0.110]_2 * 2^{-1} = [11.100]_2 * 2^{-1} + [0.110]_2 * 2^{-1} \\ &= [100.010]_2 * 2^{-1} = [0.100010]_2 * 2^2 \end{aligned}$$

$$x \widetilde{+} y = [0.100]_2 * 2^2 = 2.00$$

absoluter Fehler: $|2.00 - \frac{17}{8}| = \frac{1}{8}$

relativer Fehler: $\frac{1}{8} : \frac{17}{8} \approx 6\%$

Maschinengenauigkeit: $\varepsilon = 2^{-3} = \frac{1}{8} = 12.5\%$

Bemerkung 3.5 (Rundungsfehleranalyse: Beispiel Addition)

geg.: $a, b, c \in F$

ges.: $s = a + b + c$ in Gleitpunktarithmetik

$$\underline{\tilde{s} := (a \tilde{+} b) \tilde{+} c}$$

$$\begin{aligned}\tilde{s} &= ((a \tilde{+} b) + c)(1 + \varepsilon_2) = ((a + b)(1 + \varepsilon_1) + c)(1 + \varepsilon_2) \\ &= s + (a + b)\varepsilon_1 + (a + b + c)\varepsilon_2 + (a + b)\varepsilon_1\varepsilon_2 \doteq s + (a + b)\varepsilon_1 + s \cdot \varepsilon_2\end{aligned}$$

mit $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon$. Terme höherer Ordnung werden vernachlässigt („ \doteq “).

$$f_{\text{rel}}(s) \doteq \left| \varepsilon_2 + \frac{a + b}{a + b + c} \varepsilon_1 \right| \leq \left(1 + \left| \frac{a + b}{a + b + c} \right| \right) \varepsilon$$

Beachte Fehler in Zwischenergebnissen (ε_1) können verstärkt werden, ebenso auch Fehler in Ausgangsdaten.

kritisch $|a + b + c| \ll |a + b|$

$$\underline{\tilde{\tilde{s}} := a \tilde{+} (b \tilde{+} c)}$$

$$\tilde{\tilde{s}} \doteq s + (b + c)\varepsilon_3 + s \cdot \varepsilon_4 \quad \text{mit} \quad |\varepsilon_3|, |\varepsilon_4| \leq \varepsilon,$$

$$f_{\text{rel}}(s) \doteq \left| \varepsilon_4 + \frac{b + c}{a + b + c} \varepsilon_3 \right| \leq \left(1 + \left| \frac{b + c}{a + b + c} \right| \right) \varepsilon$$

Beachte Gleitpunktoperationen sind in der Regel weder assoziativ noch kommutativ.

Beispiel 3.6 (Addition in Gleitpunktarithmetik)

geg.: $\beta = 2, t = 3$

$$a = [0.111]_2 * 2^0 = \frac{7}{8}, \quad b = -[0.110]_2 * 2^0 = -\frac{6}{8}, \quad c = [0.110]_2 * 2^{-2} = \frac{3}{16} \in F$$

$$\begin{aligned}\tilde{s} &= ([0.111]_2 * 2^0 \simeq [0.110]_2 * 2^0) \tilde{+} [0.110]_2 * 2^{-2} \\ &= [0.001]_2 * 2^0 \tilde{+} [0.110]_2 * 2^{-2} = [0.100]_2 * 2^{-2} \tilde{+} [0.110]_2 * 2^{-2} \\ &= [1.01]_2 * 2^{-2} = [0.101]_2 * 2^{-1} = \frac{5}{16} = s, \quad \text{exaktes Ergebnis}\end{aligned}$$

$$\begin{aligned}\tilde{\tilde{s}} &= [0.111]_2 * 2^0 \tilde{+} (-[0.110]_2 * 2^0 \simeq [0.110]_2 * 2^{-2}) \\ &= [0.111]_2 * 2^0 \simeq [0.100]_2 * 2^0 = [0.011]_2 * 2^0 = [0.110]_2 * 2^{-1} = \frac{3}{8}\end{aligned}$$

Ergebnis: $|\tilde{\tilde{s}} - s| = \frac{1}{16}$, relativer Fehler 20%

Relativer Fehler in $b \tilde{+} c$: $\frac{1}{8} = 12.5\%$

Verstärkung im Endergebnis $\tilde{\tilde{s}}$ wegen $\left| \frac{b + c}{a + b + c} \right| = \frac{9}{5} = 1.8$

Bemerkung 3.7 (Auslöschung)

Problem Subtraktion annähernd gleich großer Zahlen in Gleitpunktarithmetik

geg.: $a, b \in \mathbb{R}$, ges.: $a - b$

$$\tilde{a} := \text{fl}(a) = a(1 + \varepsilon_a), \quad \tilde{b} := \text{fl}(b) = b(1 + \varepsilon_b) \quad \text{mit } |\varepsilon_a|, |\varepsilon_b| \leq \varepsilon$$

$$\begin{aligned} \text{fl}(a - b) &= \tilde{a} - \tilde{b} = (\tilde{a} - \tilde{b})(1 + \varepsilon_-) \quad \text{mit } |\varepsilon_-| \leq \varepsilon \\ &= a - b + \varepsilon_- \cdot (a - b) + \varepsilon_a \cdot a - \varepsilon_b \cdot b \end{aligned}$$

$$f_{\text{rel}}(a - b) = \left| \frac{a}{a - b} \varepsilon_a - \frac{b}{a - b} \varepsilon_b + \varepsilon_- \right| \leq \left(1 + \frac{|a| + |b|}{|a - b|} \right) \varepsilon$$

Relative Fehler $\varepsilon_a, \varepsilon_b$ in den Ausgangsdaten können drastisch verstärkt werden, falls $|a - b| \ll |a|, |b|$, insbesondere für $a \approx b$.

Beispiel $\beta = 2, a = \frac{3}{5}, b = \frac{4}{7}$

$$t = 5 \quad \tilde{a} = [0.10011]_2 * 2^0, \quad \tilde{b} = [0.10010]_2 * 2^0$$

$$\varepsilon_a \approx 0.010, \quad \varepsilon_b \approx 0.016, \quad \varepsilon \approx 0.031$$

$$\tilde{a} - \tilde{b} = [0.10000]_2 * 2^{-4} = \frac{1}{32}$$

$$\text{absoluter Fehler: } \frac{1}{32} - \frac{1}{35}, \quad \text{relativer Fehler: } 8.6\%$$

$$t = 3 \quad \tilde{a} = \tilde{b} = [0.101]_2 * 2^0$$

$$\varepsilon_a \approx 0.042, \quad \varepsilon_b \approx 0.094, \quad \varepsilon = 0.125$$

$$\tilde{a} - \tilde{b} = 0, \quad \text{relativer Fehler: } 100\%$$

Führende Ziffern $[0.1001 \dots]_2$ in a und b sind gleich und werden bei Subtraktion „ausgelöscht“ \Rightarrow „Auslöschung“.

Faustregel Vermeide – falls möglich – die Subtraktion annähernd gleich großer Zahlen in numerischen Algorithmen.

Strategien und Tricks

a) Unvermeidbare Subtraktionen annähernd gleich großer Zahlen möglichst an den Anfang des Algorithmus stellen.

b) Konjugierte Wurzelausdrücke

$$\begin{aligned} \bullet \quad \sqrt{1+x} - \sqrt{1-x} &= \frac{(\sqrt{1+x} - \sqrt{1-x})(\sqrt{1+x} + \sqrt{1-x})}{\sqrt{1+x} + \sqrt{1-x}} \\ &= \frac{2x}{\sqrt{1+x} + \sqrt{1-x}}, \quad |x| \ll 1 \end{aligned}$$

Bemerkung 3.7c): (Vermeidbare) Auslöschung

Beispiel:

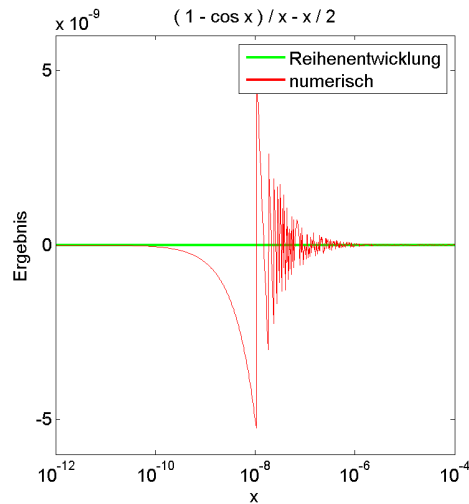
$$f(x) = \frac{1 - \cos x}{x}$$



see
numstab.m

```
% -> evaluate data
x = logspace (-12, -4, 801);
f = (1-cos(x))./x - x/2;
res = x/2 .* ( -x.^2/12 + ...
             x.^4/360 - x.^6/(360*7*8) );

% -> plot
semilogx ( x, res, 'g', x, f, 'r' );
xlabel ( 'x' );
ylabel ( 'Ergebnis' );
title ( '( 1 - cos x ) / x - x / 2' );
legend ( 'Reihenentwicklung', 'numerisch' );
```



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 3.2: Analytische Umformungen zur Vermeidung von Auslöschung.

- $x^2 + px + q = 0 \Rightarrow x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$

Auslöschung für $|q| \ll 1$, deshalb für $p \neq 0$

$$x_1 := -\frac{p}{2} - \operatorname{sgn}(p) \sqrt{\frac{p^2}{4} - q} \quad \text{mit} \quad \operatorname{sgn}(p) := \begin{cases} 1 & \text{für } p > 0, \\ 0 & \text{für } p = 0, \\ -1 & \text{für } p < 0, \end{cases}$$

$$x_2 := \frac{q}{x_1} \quad (\text{VIETAScher Wurzelsatz})$$

c) Analytische Umformungen, z. B. Reihenentwicklungen (vgl. Abb. 3.2)

$$\frac{1 - \cos x}{x} = \frac{1}{x} \left(1 - \left(1 - \frac{x^2}{2} + \frac{x^4}{24} \mp \dots \right) \right) = \frac{x}{2} \left(1 - \frac{x^2}{12} \pm \dots \right)$$

Fehler der Approximation $\frac{1 - \cos x}{x} \approx \frac{x}{2}$ betragsmäßig beschränkt durch $\frac{|x|}{2} \cdot \frac{x^2}{12}$

(Reihenrest alternierender Reihen, Satz von Leibniz)

Beispiel 3.9: Vektornormen

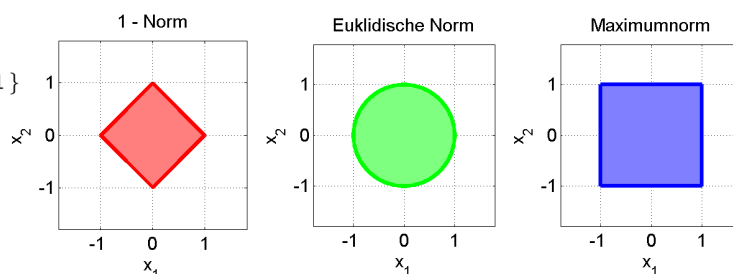
$$\|x\|_2 := \left(\sum_{i=1}^n x_i^2 \right)^{1/2} \quad \dots \quad \text{Euklidische Vektornorm}$$

$$\|x\|_1 := \sum_{i=1}^n |x_i| \quad \dots \quad \text{1-Norm}$$

$$\|x\|_\infty := \max_{i=1, \dots, n} |x_i| \quad \dots \quad \text{Maximumnorm, } \infty\text{-Norm}$$

Kugeln

$$\{x : \|x\| \leq 1\}$$



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 3.3: Einheitskugeln im \mathbb{R}^2 .

3.2 Vektor- und Matrixnormen

Definition 3.8 (Vektornorm)

Eine Abbildung $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ heißt Vektornorm auf \mathbb{R}^n , falls

1. $\|x\| \geq 0$, ($x \in \mathbb{R}^n$) und ($\|x\| = 0 \Leftrightarrow x = 0$) (Positivität),
2. $\|\alpha x\| = |\alpha| \|x\|$, ($\alpha \in \mathbb{R}$, $x \in \mathbb{R}^n$) (Homogenität),
3. $\|x + y\| \leq \|x\| + \|y\|$, ($x, y \in \mathbb{R}^n$) (Dreiecksungleichung).

Beispiel 3.9 (Vektornorm)

$$\text{a) } \|x\|_2 := \sqrt{\sum_{i=1}^n x_i^2} \quad \dots \quad \text{Euklidische Vektornorm}$$

$$\|x\|_1 := \sum_{i=1}^n |x_i| \quad \dots \quad \text{1-Norm}$$

$$\|x\|_\infty := \max_{i=1, \dots, n} |x_i| \quad \dots \quad \text{Maximumnorm, } \infty\text{-Norm}$$

b) Kugeln im \mathbb{R}^n : $\{x : \|x\| \leq 1\}$, vgl. Abb. 3.3.

Bemerkung 3.10 (Eigenschaften von Vektornormen)

a) Jedes Skalarprodukt $\langle \cdot, \cdot \rangle$ in \mathbb{R}^n erzeugt eine Vektornorm in \mathbb{R}^n :

$$\|x\| := \sqrt{\langle x, x \rangle}$$

mit $|\langle x, y \rangle| \leq \|x\| \cdot \|y\|$, ($x, y \in \mathbb{R}^n$) ... Cauchy-Schwarzsche Ungleichung.

b) Auf \mathbb{R}^n sind sämtliche Vektornormen äquivalent, d. h., zu beliebig vorgegebenen Vektornormen $\|\cdot\|_p, \|\cdot\|_q$ gibt es Konstanten $\underline{c}, \bar{c} > 0$ mit

$$\underline{c}\|x\|_q \leq \|x\|_p \leq \bar{c}\|x\|_q, \quad (x \in \mathbb{R}^n).$$

Definition 3.11 (Matrixnorm)

a) Eine Abbildung $\|\cdot\| : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ heißt Matrixnorm, falls

1. $\|A\| \geq 0$, ($A \in \mathbb{R}^{m \times n}$) und ($\|A\| = 0 \Leftrightarrow A = 0$) (Positivität),
2. $\|\alpha A\| = |\alpha| \cdot \|A\|$, ($\alpha \in \mathbb{R}, A \in \mathbb{R}^{m \times n}$) (Homogenität),
3. $\|A + B\| \leq \|A\| + \|B\|$, ($A, B \in \mathbb{R}^{m \times n}$) (Dreiecksungleichung).

b) Eine Matrixnorm $\|\cdot\|$ heißt submultiplikativ, falls

$$\|AB\| \leq \|A\| \cdot \|B\|, \quad (A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}).$$

c) Eine submultiplikative Matrixnorm $\|\cdot\|$ heißt verträglich (auch: konsistent) mit einer vorgegebenen Vektornorm $\|\cdot\|$, falls

$$\|Ax\| \leq \|A\| \cdot \|x\|, \quad (A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n).$$

Beispiel 3.12 (Frobeniusnorm)

$$\|A\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2} \quad \dots \text{ Frobeniusnorm}$$

- Submultiplikative Matrixnorm, verträglich mit $\|\cdot\|_2$:

$$\|Ax\|_2^2 = \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij} x_j \right)^2 \leq \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij} \right)^2 \left(\sum_{j=1}^n x_j \right)^2 = \|A\|_F^2 \cdot \|x\|_2^2$$

- $\|I_n\| = \sqrt{n}$

Satz 3.13 (Zugeordnete Matrixnorm)

Zu einer vorgegebenen Vektornorm $\|\cdot\|$ wird durch

$$A \mapsto \|A\| := \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|$$

eine submultiplikative, mit $\|\cdot\|$ verträgliche Matrixnorm definiert, die der Vektornorm $\|\cdot\|$ zugeordnete Matrixnorm. Es gilt $\|I_n\| = 1$.

Beweis vgl. Huckle/Schneider, Anhang B.2, z. B.

$$\|I_n\| = \sup_{x \neq 0} \frac{\|I_n x\|}{\|x\|} = 1. \quad \blacksquare$$

Beispiel 3.14 (Zugeordnete Matrixnorm)

a) *Zeilensummennorm*

$$\|A\|_\infty := \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}|$$

ist $\|x\|_\infty$ zugeordnet, denn

- $\|Ax\|_\infty = \max_{i=1,\dots,m} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}| |x_j| \leq \|A\|_\infty \cdot \|x\|_\infty,$
 - zu einem $i_0 \in \{1, \dots, m\}$ mit $\sum_{j=1}^n |a_{i_0,j}| = \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}|$ wählt man $x \in \mathbb{R}^n$ so, dass $\|x\|_\infty = 1$ und $x_j = 1$, falls $a_{i_0,j} > 0$, $x_j = -1$, falls $a_{i_0,j} < 0$
- $$\Rightarrow \|Ax\|_\infty \geq \sum_{j=1}^n |a_{i_0,j} x_j| = \sum_{j=1}^n |a_{i_0,j}| = \|A\|_\infty.$$

b) *Spaltensummennorm*

$$\|A\|_1 := \max_{j=1,\dots,n} \sum_{i=1}^m |a_{ij}| \quad \text{ist } \|x\|_1 \text{ zugeordnet.}$$

c) *Spektralnrm*

$$\|A\|_2 := \max_{i=1,\dots,n} \sqrt{\lambda_i(A^\top A)}$$

Für orthogonale Matrizen $U \in \mathbb{R}^{n \times n}$, also $U^\top U = I_n$, ist $\lambda_i(U^\top U) = 1$ und $\|U\|_2 = 1$.

3.3 Kondition und Stabilität

Bemerkung 3.15 (Eingabefehler und Fehler im Ergebnis)

analytisch Eingabe $x \rightarrow$ Algorithmus / Berechnungsvorschrift
 \rightarrow Resultat $f(x)$

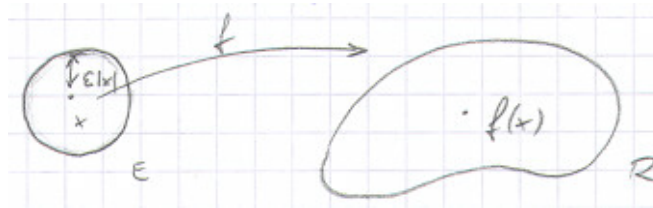
numerisch Fehler im Resultat entstehen durch

- Eingabefehler
- Fehler im Algorithmus

Numerische Eingabe $x \in F$ repräsentiert

Eingabemenge $E = \{ \tilde{x} \in \mathbb{R} : f(\tilde{x}) = x \}$

Resultatmenge $R = f(E) := \{ f(\tilde{x}) : \tilde{x} \in E \}$



„**Kondition**“ Maß für das Verhältnis von R zu E

Ziel Fehler in der Berechnungsvorschrift sollen Menge R nicht deutlich vergrößern

Fehler im Ergebnis $y = f(x)$:

$$\delta_y := f(x + \delta_x) - f(x) \approx f'(x) \delta_x$$

Relativer Fehler:

$$\frac{\|\delta_y\|}{\|y\|} \approx \frac{\|f'(x) \delta_x\|}{\|y\|} \leq \frac{\|x\| \|f'(x)\|}{\|y\|} \cdot \frac{\|\delta_x\|}{\|x\|}$$

Definition 3.16 (Konditionszahl)

Zu einem Problem $x \mapsto f(x)$ heißt

$$\text{cond}_x := \frac{\|x\| \|f'(x)\|}{\|f(x)\|}$$

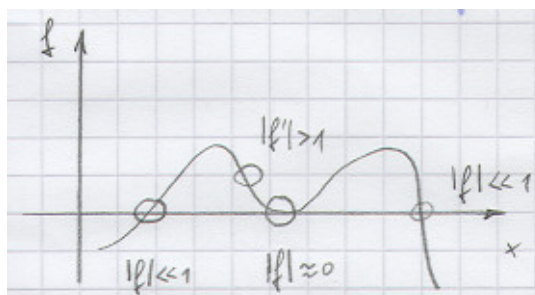
Konditionszahl. Das Problem ist gut konditioniert, wenn cond_x klein ist, und schlecht konditioniert für große Konditionszahlen cond_x .

Beispiel 3.17 (Kondition)

Exponentialfunktion $x \mapsto e^x$, $\text{cond}_x = \left| \frac{x e^x}{e^x} \right| = |x|$, gut konditioniert für $|x| \lesssim 1$.

Logarithmus $x \mapsto \ln x$, $\text{cond}_x = \left| \frac{x \cdot \frac{1}{x}}{\ln x} \right| = \frac{1}{|\ln x|}$, sehr schlecht konditioniert für $x \approx 1$.

Gute / schlechte Kondition



Polynomnullstellen häufig schlecht konditioniertes Problem

Beispiel: $\pi(t) = t^4 - 8t^3 + 24t^2 - 32t + 15.999\,999\,99 = (t - 2)^4 - 10^{-8}$

$$t_{1,2} = 2 \pm 0.01, \quad t_{3,4} = 2 \pm 0.01i$$

Relativer Fehler bei Darstellung von 15.999 999 99 durch 16.0: $\varepsilon_x = 6.25 \cdot 10^{-10}$, z. B. bei Maschinengenauigkeit $\varepsilon = 8 \cdot 10^{-10} \Rightarrow t_{1,2,3,4} = 2.0$

$$\text{cond}_{t_{1,2}} = \frac{0.01/2.01}{6.25 \cdot 10^{-10}} \approx 8.0 \cdot 10^6$$

Bemerkung 3.18 (Berechnungsvorschrift)

Zum mathematischen Problem $x \mapsto f(x)$ sei die Abbildung $x \mapsto \tilde{f}(x)$ gegeben zur Berechnung von $f(x)$ in Gleitpunktarithmetik (u. a. auch Reihenfolge der Rechenoperationen festgelegt) \rightsquigarrow *Berechnungsvorschrift*

Beispiel $f(x) = 1 - \sqrt{1 - x^2}$, für $|x| \ll 1$ gut konditioniert, $\text{cond}_x \approx 2$.

Berechnungsvorschrift 1: $\tilde{f}(x) := 1 - \left(\sqrt{1 - (x^2)}\right)$

Berechnungsvorschrift 2: $\tilde{f}(x) := \frac{(x^2)}{\left(1 + \left(\sqrt{1 - (x^2)}\right)\right)}$

Definition 3.19 (Numerische Stabilität)

Zu einem gut konditionierten Problem $x \mapsto f(x)$ heißt eine Berechnungsvorschrift $x \mapsto \tilde{f}(x)$ numerisch stabil, wenn die relativen Eingabefehler durch die Berechnungsvorschrift nicht vergrößert werden, und numerisch instabil sonst.

Bemerkung 3.20 (Lineare Gleichungssysteme: Kondition und Stabilität)

a) Betrachte zu gegebener regulärer Matrix $A \in \mathbb{R}^{n \times n}$ und gegebenem $b \in \mathbb{R}^n$ die Lösung des linearen Gleichungssystems $Ax = b$ als Abbildung $b \mapsto x := A^{-1}b$ mit gestörten Eingangsdaten b und exakten Matrizen $A, A^{-1} \Rightarrow$

$$\begin{aligned} A(x + \delta_x) &= b + \delta_b, & Ax &= b \\ \|\delta_x\| &= \|A^{-1}\delta_b\| \leq \|A^{-1}\| \|\delta_b\| \\ \|b\| &= \|Ax\| \leq \|A\| \|x\| \end{aligned}$$

Ergebnis $\frac{\|\delta_x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta_b\|}{\|b\|}$

- Empfindlichkeit der Lösung gegenüber Störungen in den Eingangsdaten wird beschrieben durch die *Konditionszahl* $\text{cond}(A) := \|A\| \cdot \|A^{-1}\|$.
- Ergebnis lässt sich übertragen auf Empfindlichkeit gegenüber Störungen in A .
- Wegen $1 \leq \|I_n\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$ gilt stets $\text{cond}(A) \geq 1$.

Gut konditioniert: $\text{cond}(A) \lesssim 10^3$

Schlecht konditioniert: $\text{cond}(A) \gg 10^6$

Beispiel 1

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1/\varepsilon \end{pmatrix} \quad \text{mit } 0 < \varepsilon \ll 1$$

$\Rightarrow \text{cond}_2(A) = 1 \cdot \frac{1}{\varepsilon} = \frac{1}{\varepsilon} \rightarrow \infty$ für $\varepsilon \rightarrow 0$, Fehlerverstärkung um Faktor $1/\varepsilon$ möglich

Beispiel 2

Hilbert-Matrizen $H^{(n)} = (h_{ij}^{(n)})_{i,j} \in \mathbb{R}^{n \times n}$ mit $h_{ij}^{(n)} := 1/(i + j - 1)$, $(i, j = 1, \dots, n)$.

Zu $b^{(n)} \in \mathbb{R}^n$, $b^{(n)} = (b_i^{(n)})_i$ mit $b_i^{(n)} := \sum_{j=1}^n \frac{1}{i + j - 1}$ ist die Lösung $x^{(n)}$ des linearen

Gleichungssystems $H^{(n)}x^{(n)} = b^{(n)}$ gegeben durch $x^{(n)} = (1, 1, \dots, 1)^\top$.

Fehler der mit Matlab ($\varepsilon = 1.1_E - 16$) berechneten Lösung $\tilde{x}^{(n)}$:

n	$\ \tilde{x}^{(n)} - x^{(n)}\ _2$	$\text{cond}_2(H^{(n)})$
2	9.0e-16	1.9e+01
4	4.6e-13	1.6e+04
6	3.5e-10	1.5e+07
8	1.3e-08	1.5e+10
10	3.0e-04	1.6e+13
12	1.6e+00	1.7e+16

b) Für orthogonale Matrizen $Q \in \mathbb{R}^{n \times n}$ ist $Q^{-1} = Q^\top$ und $\|Q\|_2 = \|Q^\top\|_2 = 1$

$\Rightarrow \text{cond}_2(Q) = 1$. Operationen mit orthogonalen Matrizen lassen die Kondition einer

Matrix unverändert: $A = QR \Rightarrow \text{cond}_2(R) = \text{cond}_2(A)$.

c) Realisierung des Gauß-Algorithmus in Gleitpunktarithmetik:

Fehlerschranke hängt linear ab von $\max_{i,k} |l_{ik}|$.

Spaltenpivotisierung: $|l_{ik}| \leq 1 \rightsquigarrow$ kleine Fehlerschranke

Numerische Stabilität: numerische Lösung \tilde{x} erfüllt

$$(A + \delta_A)\tilde{x} = b$$

mit

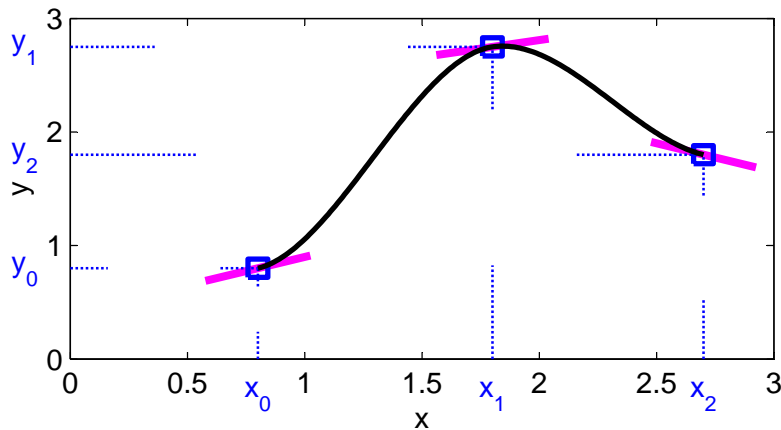
$$\frac{\|\delta_A\|_\infty}{\|A\|_\infty} \leq 8n^3 \cdot \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|} \varepsilon.$$

4 Interpolation (II)

Bemerkung 4.1 (Stückweise Hermite-Interpolation)

geg.: $r + 1$ Stützstellen x_0, x_1, \dots, x_r

Stützwerte (y_k, y'_k) , ($k = 0, 1, \dots, r$)



Definiert man die interpolierende Funktion Φ stückweise durch Hermite-Interpolationspolynome $\Phi|_{[x_{i-1}, x_i]}$, ($i = 1, \dots, r$) mit Interpolationsbedingungen

$$\Phi(x_{i-1}) = y_{i-1}, \quad \Phi'(x_{i-1}) = y'_{i-1}, \quad \Phi(x_i) = y_i, \quad \Phi'(x_i) = y'_i, \quad (i = 1, \dots, r),$$

so ist $\Phi \in C^1[a, b]$, aber $\deg \Phi|_{[x_{i-1}, x_i]} \leq 3$.

4.1 Spline-Interpolation

Bemerkung 4.2 (Kubische Spline-Interpolation)

Kubische Splines erreichen ähnlich wie zusammengesetzte Hermite-Interpolierende eine hohe globale Glattheit, jedoch mit deutlich niedrigerem Polynomgrad:

$$s \in C^2[a, b], \quad s|_{[x_i, x_{i+1}]} \in \Pi_3.$$

Splines der Ordnung k : $s \in C^{k-2}[a, b]$, $s|_{[x_i, x_{i+1}]} \in \Pi_{k-1}$.

Splinegitter $a = x_0 < x_1 < \dots < x_n = b$.

Zum kubischen Spline s ist

$$s|_{[x_i, x_{i+1}]} = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3, \quad (i = 0, 1, \dots, n-1)$$

\Rightarrow insgesamt $4n$ Parameter (a_i, b_i, c_i, d_i) , $(i = 0, 1, \dots, n-1)$.

$n + 1$ Interpolationsbedingungen $s(x_i) = y_i \Rightarrow a_i = y_i$, $(i = 0, 1, \dots, n-1)$,

$$s(x_n) = y_n \Rightarrow a_{n-1} + b_{n-1}(x_n - x_{n-1}) + \frac{c_{n-1}}{2}(x_n - x_{n-1})^2 + \frac{d_{n-1}}{6}(x_n - x_{n-1})^3 = y_n =: a_n.$$

$3(n - 1)$ Stetigkeitsbedingungen

$$s(x_{i+1} - 0) = s(x_{i+1} + 0) : \quad a_i + b_i h_i + \frac{c_i}{2} h_i^2 + \frac{d_i}{6} h_i^3 = a_{i+1}, \quad (i = 0, 1, \dots, n-2)$$

$$s'(x_{i+1} - 0) = s'(x_{i+1} + 0) : \quad b_i + c_i h_i + \frac{d_i}{2} h_i^2 = b_{i+1}, \quad (i = 0, 1, \dots, n-2)$$

$$s''(x_{i+1} - 0) = s''(x_{i+1} + 0) : \quad c_i + d_i h_i = c_{i+1}, \quad (i = 0, 1, \dots, n-2)$$

mit *Schrittweiten* $h_i := x_{i+1} - x_i$, $(i = 0, 1, \dots, n-1)$.

\Rightarrow insgesamt $4n - 2$ lineare Bedingungen an $4n$ Parameter

Zusatzbedingungen

(i) $s''(x_0) = s''(x_n) = 0 \dots$ natürlicher kubischer Spline

oder

(ii) $s'(x_0) = y'_0$, $s'(x_n) = y'_n \dots$ vollständiger kubischer Spline

oder

(iii) $s'(x_0) = s'(x_n)$, $s''(x_0) = s''(x_n) \dots$ periodischer kubischer Spline, für periodische Daten ($y_0 = y_n$) $\Rightarrow s(x_0) = s(x_n)$.

In jedem der drei Fälle ist die Splinefunktion eindeutig bestimmt.

Berechnung der Koeffizienten

$$d_i = \frac{c_{i+1} - c_i}{h_i},$$

$$b_i = \frac{a_{i+1} - a_i}{h_i} - \frac{c_i}{2} h_i - \frac{d_i}{6} h_i^2 = \frac{y_{i+1} - y_i}{h_i} - \frac{2c_i + c_{i+1}}{6} h_i$$

Stetigkeitsbedingung für $s'(x) \Rightarrow (i = 0, 1, \dots, n-2)$

$$\frac{y_{i+1} - y_i}{h_i} - \frac{2c_i + c_{i+1}}{6} h_i + c_i h_i + \frac{c_{i+1} - c_i}{2} h_i = \frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{2c_{i+1} + c_{i+2}}{6} h_{i+1}$$

$$\frac{h_i}{6} c_i + \frac{h_i + h_{i+1}}{3} c_{i+1} + \frac{h_{i+1}}{6} c_{i+2} = \frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}$$

Zusammen mit $c_0 = c_n = 0$ ergibt sich für den natürlichen kubischen Spline ein tridiagonales lineares Gleichungssystem der Dimension $n - 1$ zur Bestimmung von c_1, \dots, c_{n-1} \Rightarrow Gaußscher Algorithmus erfordert $\mathcal{O}(n)$ Rechenoperationen. Analoges Vorgehen für vollständigen und periodischen Spline.

Algorithmus 1 Bestimmung der Splinekoeffizienten.

1. $a_i := y_i, (i = 0, 1, \dots, n - 1)$
2. Berechne c_0, c_1, \dots, c_n als Lösung eines tridiagonalen Gleichungssystems.
3. Bestimme $b_i, d_i, (i = 0, 1, \dots, n - 1)$.

Algorithmus 2 Auswertung der Splinefunktion.

1. Bestimme Teilintervall (binäre Suche)

```

i := 0, ibar := n
repeat
  i* := ⌊  $\frac{i + i^{\text{bar}}}{2}$  ⌋
  if  $x \geq x_{i^*}$  then i := i* else ibar := i*
until ibar - i ≤ 1
i := i

```

Einfachster Spezialfall: äquidistantes Gitter $x_i = a + ih$ mit $h = \frac{b - a}{n}$

$$\Rightarrow i := \left\lceil \frac{x - a}{h} \right\rceil$$

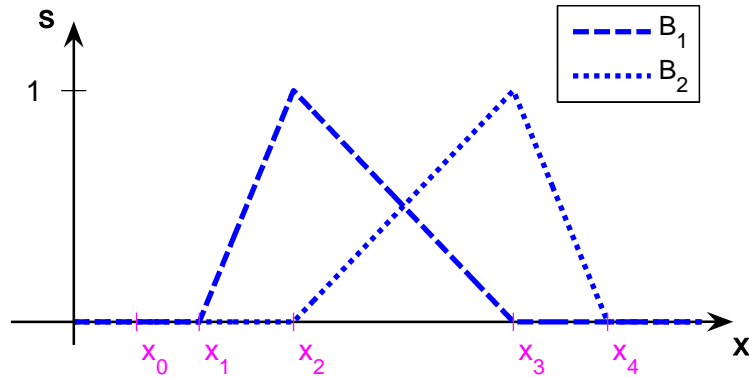
2. Splineauswertung $s(x) = a_i + (x - x_i)(b_i + (x - x_i)(\frac{1}{2}c_i + \frac{1}{6}d_i(x - x_i)))$

Bemerkung 4.3 (B-Splines)

Idee Darstellung der Splinefunktion als Linearkombination „einfacher“ Basisfunktionen des Vektorraums der Splinefunktionen \rightsquigarrow B-Splines.

Beispiel $k = 2$: stetige, stückweise lineare Funktion

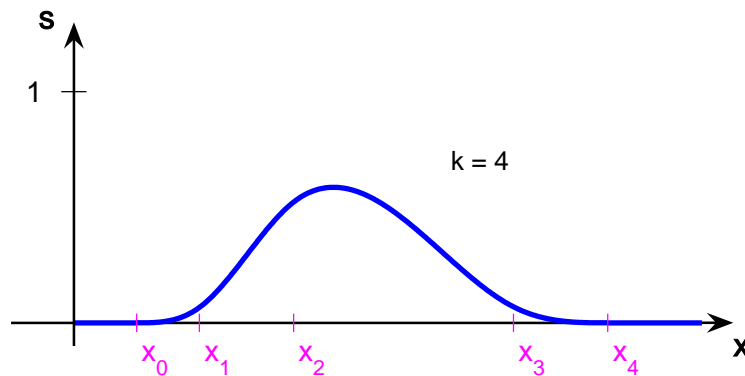
$$B_j(x) = \begin{cases} \frac{x - x_j}{x_{j+1} - x_j}, & (x \in [x_j, x_{j+1}]), \\ \frac{x - x_{j+2}}{x_{j+1} - x_{j+2}}, & (x \in [x_{j+1}, x_{j+2}]), \\ 0 & \text{sonst.} \end{cases}$$



Interpolierender linearer Spline $s(x) = \sum_{j=0}^n y_j B_{j-1}(x)$.

- allgemein**
- $B_j|_{[x_i, x_{i+1}]} \in \Pi_{k-1}$, ($i = 0, 1, \dots, n-1$)
 - $B_j \in C^{k-2}[a, b]$
 - $\sum_j B_j(x) = 1$, ($x \in [a, b]$)
 - $\text{supp } B_j = [x_j, x_{j+k}]$, d. h. $B_j(x) = 0$, ($x \leq x_j$ oder $x \geq x_{j+k}$)

Beispiel Kubischer B-Spline



Bestimmung der Koeffizienten α_j des interpolierenden Splines $\sum_j \alpha_j B_j(x)$ als Lösung eines linearen Gleichungssystems der Bandbreite $k-1$.

Satz 4.4 (Approximationseigenschaften kubischer Splines)

Gegeben sei eine Funktion $f \in C^4[a, b]$ mit $\max_{a \leq x \leq b} |f^{(4)}(x)| \leq M$ sowie ein Gitter

$$\Delta = \{ a = x_0 < x_1 < \dots < x_n = b \}$$

mit Schrittweiten $h_i := x_{i+1} - x_i$, ($i = 0, 1, \dots, n-1$) und einer Konstanten

$$K \geq \max_{0 \leq i \leq n-1} h_i / \min_{0 \leq i \leq n-1} h_i .$$

Dann gibt es zum vollständigen interpolierenden kubischen Spline s_Δ Konstanten C_0, C_1, C_2 und C_3 , die von Δ und K unabhängig sind und für die gilt

$$|f^{(k)}(x) - s_\Delta^{(k)}(x)| \leq C_k MK \left(\max_{0 \leq i \leq n-1} h_i \right)^{4-k}, \quad (x \in [a, b], k = 0, 1, 2, 3)$$

in jedem Punkt x , in dem $s_\Delta^{(k)}(x)$ definiert ist.

Beweisidee (i) $f(x_i) = s_\Delta(x_i)$ für Stützstellen x_i

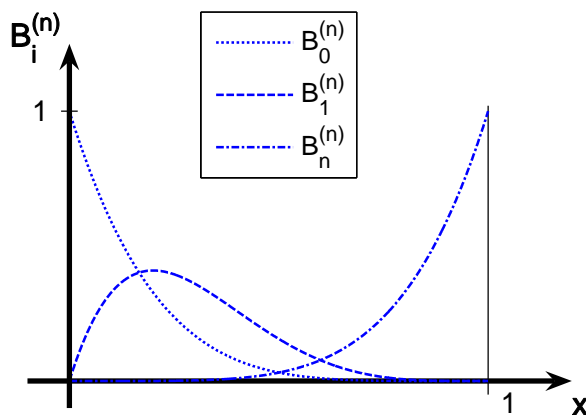
(ii) Abschätzung von $|f''(x_i) - s_\Delta''(x_i)|$ durch Einsetzen von f in das Gleichungssystem aus Bemerkung 4.2

(iii) Hieraus Abschätzungen für $x \neq x_i$. ■

Bemerkung 4.5 (Bernstein–Polynome und Bezier–Kurven)

a) Bernstein–Polynome

$$B_i^{(n)}(x) = \binom{n}{i} (1-x)^{n-i} x^i, \quad (i = 0, 1, \dots, n)$$



Eigenschaften

a) i -fache Nullstelle $x = 0$, $(n - i)$ -fache Nullstelle $x = 1$

b) $B_i^{(n)} \Big|_{[0,1]} \geq 0$

c) $\sum_{i=0}^n B_i^{(n)}(x) = \sum_{i=0}^n \binom{n}{i} (1-x)^{n-i} x^i = ((1-x) + x)^n = 1$

d) $B_i^{(n)}(x) = x \cdot B_{i-1}^{(n-1)}(x) - B_i^{(n-1)}(x)$

b) Bezier-Kurve

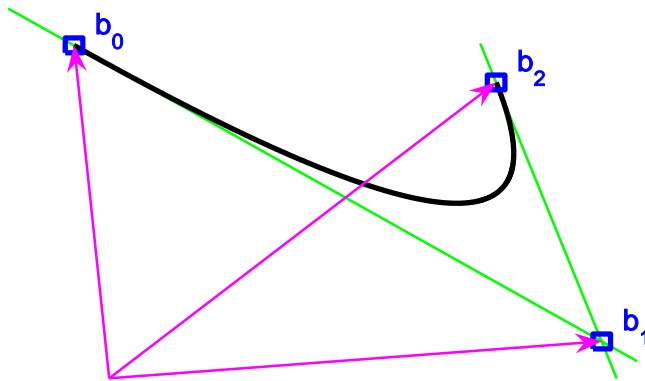
geg.: Kontrollpunkte $b_0, b_1, \dots, b_n \in \mathbb{R}^k$

Bezierkurve im \mathbb{R}^k : $\sum_{i=0}^n b_i B_i^{(n)}(x), (x \in [0, 1])$.

- Anfangspunkt b_0 , Endpunkt b_n
- Tangente in b_0 verläuft durch b_1 , denn

$$\frac{d}{dx} B_0^{(n)}(0) = -n, \quad \frac{d}{dx} B_1^{(n)}(0) = n, \quad \frac{d}{dx} B_i^{(n)}(0) = 0, \quad (i > 1)$$

- $n = 2$: Tangenten in b_0 und b_2 schneiden sich in b_1
- Kurve verläuft in der konvexen Hülle des von den b_i gebildeten Polygons
 \Rightarrow keine unerwünschten Oszillationen



4.2 Trigonometrische Interpolation – Schnelle Fouriertransformation

Bemerkung 4.6 (Problemstellung)

Interpolation periodischer Daten durch trigonometrische Polynome.

$N = 2n + 1$ ungerade

$$T_R^N := \left\{ \phi_{2n+1}(t) := \frac{a_0}{2} + \sum_{j=1}^n (a_j \cos jt + b_j \sin jt) \text{ mit } a_j, b_j \in \mathbb{R}, (j = 0, \dots, n) \right\}$$

$N = 2n$ gerade

$$T_R^N := \left\{ \phi_{2n}(t) := \frac{a_0}{2} + \sum_{j=1}^{n-1} (a_j \cos jt + b_j \sin jt) + \frac{a_n}{2} \cos nt \text{ mit } a_j, b_j \in \mathbb{R}, (j = 0, \dots, n) \right\}$$

Komplexe Darstellung

$$T_C^N := \left\{ \phi_N(t) := \sum_{j=0}^{N-1} c_j e^{ijt} : c_j \in \mathbb{C}, (j = 0, \dots, N-1) \right\}$$

Ansatzfunktionen $\{1, \cos jt, \sin jt\}$ bzw. $\{e^{ijt} : 0 \leq j \leq N-1\}$ sind linear unabhängig \rightsquigarrow Interpolationsaufgabe zu N Stützpunkten (t_k, f_k) , $(k = 0, 1, \dots, N-1)$ eindeutig lösbar.

Typische Aufgabenstellung Digitale Signalverarbeitung, Datenerfassung im festen Takt \Rightarrow äquidistante Knoten, normiert auf $t_k = k \cdot 2\pi/N$.

Bemerkung 4.7 (Trigonometrische Interpolation auf äquidistantem Gitter)

Zu $t_k = k \cdot 2\pi/N$, $(k = 0, 1, \dots, N-1)$ sind $\omega_k := e^{it_k} = e^{ik \cdot 2\pi/N}$ die N -ten Einheitswurzeln.

Interpolationsbedingungen $\phi_N(t_k) = f_k$, $(k = 0, 1, \dots, N-1)$ bestimmen ϕ_N mit

$$\begin{pmatrix} 1 & \omega_0 & \omega_0^2 & \cdots & \omega_0^{N-1} \\ 1 & \omega_1 & \omega_1^2 & \cdots & \omega_1^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_{N-1} & \omega_{N-1}^2 & \cdots & \omega_{N-1}^{N-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix}.$$

Bemerkung 4.8 (Komplexe und reelle trigonometrische Polynome)

Gilt für das (komplexe) trigonometrische Polynom

$$\phi_N(t) = \sum_{j=0}^{N-1} c_j e^{ijt}$$

$\phi_N(t) \in \mathbb{R}$, $(t \in \mathbb{R})$, so gilt $\phi_N \in T_R^N$ mit

$$a_j = 2 \operatorname{Re} c_j = c_j + c_{N-j}, \quad b_j = -2 \operatorname{Im} c_j = i(c_j - c_{N-j}).$$

Beweis Für die N äquidistanten Knoten $t_k = k \cdot 2\pi/N$, $(k = 0, 1, \dots, N-1)$ gilt

$$e^{-ijt_k} = \underbrace{e^{iNk \cdot 2\pi/N}}_{=1} \cdot e^{-ijt_k} = e^{i(N-j)t_k}$$

und

$$\sum_{j=0}^{N-1} c_j e^{ijt_k} = \phi_N(t_k) = \overline{\phi_N(t_k)} = \sum_{j=0}^{N-1} \overline{c_j} e^{-ijt_k} = \sum_{j=0}^{N-1} \overline{c_j} e^{i(N-j)t_k} = \sum_{l=1}^N \overline{c_{N-l}} e^{-ilt_k}$$

$\Rightarrow c_j = \overline{c_{N-j}}$, da Interpolationsaufgabe eindeutig lösbar.

Wegen $e^{i \cdot 0 \cdot t_k} = e^{i \cdot N \cdot t_k} = 1$ ist insbesondere $c_0 = \overline{c_0}$, also $c_0 \in \mathbb{R}$.

Ebenso folgt $c_n \in \mathbb{R}$ für gerade $N = 2n$.

Sei nun $N = 2n + 1$, so ist

$$\begin{aligned} \phi_N(t_k) &= c_0 + \sum_{j=1}^{2n} c_j e^{ij t_k} = c_0 + \sum_{j=1}^n c_j e^{ij t_k} + \sum_{l=1}^n c_{N-l} e^{i(N-l)t_k} \\ &= c_0 + \sum_{j=1}^n (c_j e^{ij t_k} + \overline{c_j} e^{-ij t_k}) \\ &= c_0 + 2 \sum_{j=1}^n \operatorname{Re}(c_j e^{ij t_k}) = c_0 + 2 \sum_{j=1}^n (\operatorname{Re} c_j \cdot \cos j t_k - \operatorname{Im} c_j \cdot \sin j t_k). \end{aligned}$$

Wegen der Eindeutigkeit des trigonometrischen Interpolationspolynoms folgt die Behauptung durch Koeffizientenvergleich. Analog für gerade $N = 2n$. ■

Satz 4.9 (Lösung der trigonometrischen Interpolationsaufgabe)

Für äquidistante Stützstellen $t_k = k \cdot 2\pi/N$, ($k = 0, 1, \dots, N-1$) ist die Lösung der trigonometrischen Interpolationsaufgabe

$$\phi_N(t_k) = f_k, \quad (k = 0, 1, \dots, N-1)$$

gegeben durch

$$\phi_N(t) = \sum_{j=0}^{N-1} c_j e^{ijt} \quad \text{mit} \quad c_j := \frac{1}{N} \sum_{k=0}^{N-1} f_k \omega_k^{-j}, \quad (j = 0, 1, \dots, N-1).$$

Beweis Wegen der Eindeutigkeit des trigonometrischen Interpolationspolynoms reicht es aus, für $l = 0, 1, \dots, N-1$ zu zeigen

$$f_l \stackrel{!}{=} \sum_{j=0}^{N-1} \underbrace{\left(\frac{1}{N} \sum_{k=0}^{N-1} f_k \omega_k^{-j} \right)}_{= c_j} e^{ijt_l} = \sum_{k=0}^{N-1} f_k \cdot \frac{1}{N} \sum_{j=0}^{N-1} \omega_k^{-j} \omega_l^j.$$

Nun ist

$$\sum_{j=0}^{N-1} \omega_k^{-j} \omega_l^j = \sum_{j=0}^{N-1} \omega_{l-k}^j = \begin{cases} N & \text{falls } k = l, \\ 0 & \text{sonst,} \end{cases}$$

denn

$$\frac{\omega_{l-k} - 1}{\omega_{l-k} - 1} \cdot \sum_{j=0}^{N-1} \omega_{l-k}^j = \frac{1}{\omega_{l-k} - 1} \sum_{j=0}^{N-1} (\omega_{l-k}^{j+1} - \omega_{l-k}^j) = \frac{\omega_{l-k}^N - 1}{\omega_{l-k} - 1} = 0,$$

falls $l \neq k$, denn ω_{l-k} ist N -te Einheitswurzel. Hieraus folgt die Behauptung. ■

Bemerkung 4.10 (Diskrete Fourier–Transformation)

Für 2π -periodische Funktionen $f \in L^2(\mathbb{R})$ erhält man mit den Fourierkoeffizienten

$$\hat{f}(j) := \frac{1}{2\pi} \int_0^{2\pi} f(t)e^{-ijt} dt, \quad (j \in \mathbb{Z})$$

die abgebrochenen Fourier–Reihen

$$f_n(t) := \sum_{j=-n}^n \hat{f}(j)e^{ijt}.$$

Setzt man für $N = 2n + 1$

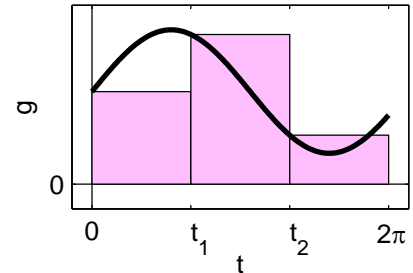
$$c_{-j} := c_{N-j}, \quad (j = 1, \dots, n),$$

so ist

$$\begin{aligned} \phi_N(t_k) &= \sum_{j=0}^{N-1} c_j e^{ijt_k} = \sum_{j=0}^n c_j e^{ijt_k} + \sum_{j=n+1}^{N-1} c_j e^{ijt_k} \\ &= \sum_{j=0}^n c_j e^{ijt_k} + \sum_{l=1}^n c_{N-l} e^{i(N-l)t_k} = \sum_{j=0}^n c_j e^{ijt_k} + \sum_{l=1}^n c_{-l} e^{i(-l)t_k} = \sum_{j=-n}^n c_j e^{ijt_k}. \end{aligned}$$

Approximiert man

$$\int_0^{2\pi} g(t) dt \approx \frac{2\pi}{N} \sum_{k=0}^{N-1} g(t_k),$$



so ergibt sich

$$\hat{f}(j) \approx \frac{1}{N} \sum_{k=0}^{N-1} f(t_k) e^{-ijt_k} = \frac{1}{N} \sum_{k=0}^{N-1} f_k \omega_k^{-j} = c_j.$$

Wegen der Analogie zur klassischen kontinuierlichen Fourier–Transformation heißt die Abbildung

$$\mathcal{F}_N : \mathbb{C}^N \rightarrow \mathbb{C}^N, \quad (f_k)_{k=0}^{N-1} \mapsto (c_j)_{j=0}^{N-1}$$

aus Satz 4.9 *Diskrete Fourier–Transformation* (DFT) und die Umkehrabbildung

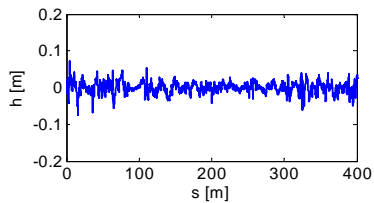
$$\mathcal{F}_N^{-1} : (c_j)_j \mapsto (f_k)_k, \quad f_k := \sum_{j=0}^{N-1} c_j \omega_j^k, \quad (k = 0, 1, \dots, N-1)$$

(Fourier–)Synthese oder *Inverse diskrete Fourier–Transformation* (IDFT).

Beispiel 4.11: Tiefpassfilter

Beispiel (Prof. Dr.-Ing. P. Pickel, U Halle, Landwirtschaftliche Fakultät)

Fahrgeweg eines landwirtschaftlichen Nutzfahrzeugs (vertikale Auslenkung)



Messdaten im Abstand $\Delta_s = 0.05$ m

Geschwindigkeit $v = 2.0$ m/s

$\Rightarrow \Delta_t = 0.025$ s $\Rightarrow \bar{f}_{\text{input}} = 40$ Hz



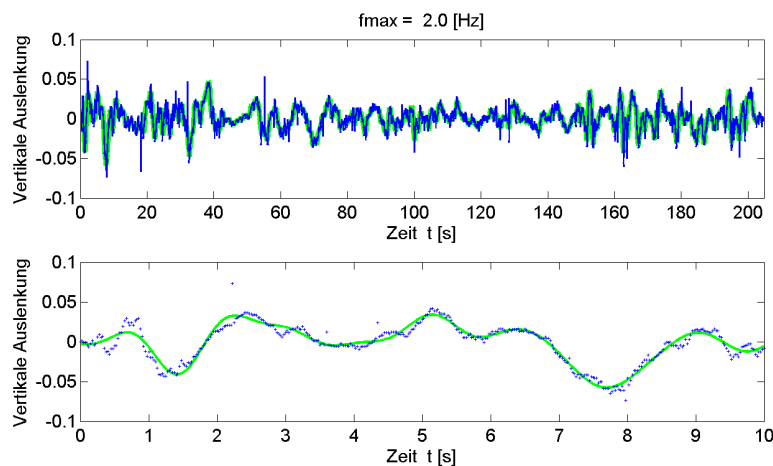
Programmieraufgabe
Übungsblatt 5



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 4.1: Anwendung der DFT in der Signalverarbeitung: Ausgangsdaten.

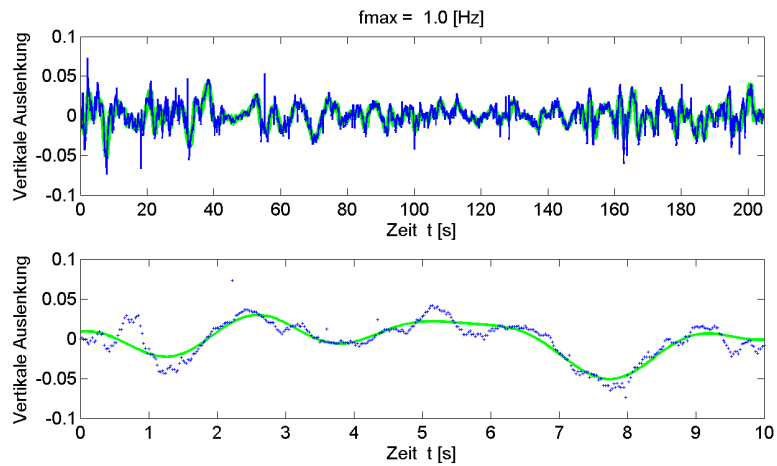
Beispiel 4.11: Tiefpassfilter (II)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 4.2: Anwendung der DFT in der Signalverarbeitung: $f_{\text{max}} = 2.0$ Hz.

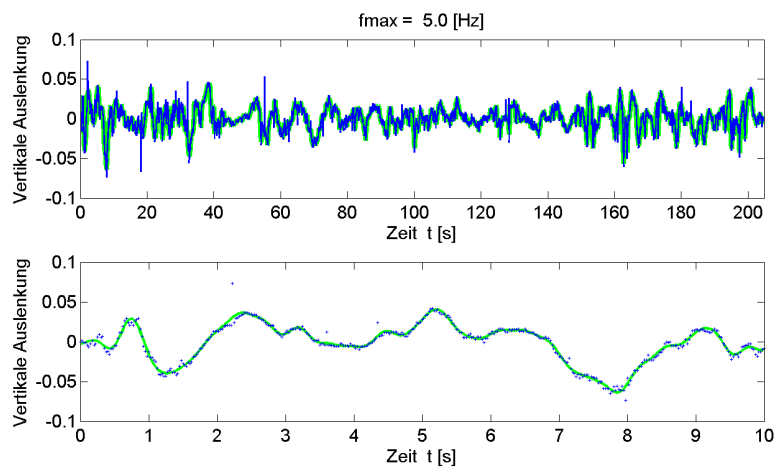
Beispiel 4.11: Tiefpassfilter (III)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 4.3: Anwendung der DFT in der Signalverarbeitung: $f_{\max} = 1.0 \text{ Hz}$.

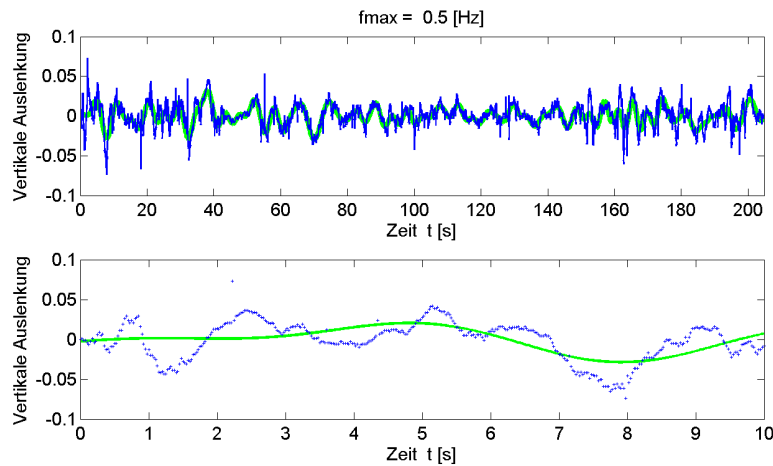
Beispiel 4.11: Tiefpassfilter (IV)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 4.4: Anwendung der DFT in der Signalverarbeitung: $f_{\max} = 5.0 \text{ Hz}$.

Beispiel 4.11: Tiefpassfilter (V)



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
 Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 4.5: Anwendung der DFT in der Signalverarbeitung: $f_{\max} = 0.5 \text{ Hz}$.

Beispiel 4.11 (Tiefpassfilter)

Anwendung der DFT in der Signalverarbeitung

Elimination hochfrequenter Anteile im Signal, die häufig durch Messfehler verfälscht („Messrauschen“) und darüberhinaus für die praktische Anwendung oft nicht relevant sind \Rightarrow „Tiefpassfilter“.

praktisch Berücksichtige in $\phi_N(t)$ nur die Terme $a_j \cos jt$ und $b_j \sin jt$, die zu den ersten $j_{\max} \ll n$ Frequenzen gehören (j_{\max} ist vom Anwender vorzugeben). Für $N = 2n + 1$ erhält man

$$\tilde{f}_k := c_0 + \sum_{j=1}^{j_{\max}} c_j \omega_j^k + \sum_{j=1}^{j_{\max}} c_{N-j} \omega_{N-j}^k = c_0 + \sum_{j=1}^{j_{\max}} (c_j \omega_j^k + c_{N-j} \bar{\omega}_j^k), \quad (k = 0, 1, \dots, N - 1).$$

Beispiel Rauigkeit einer Feldoberfläche

(Originaldaten von Prof. Dr.-Ing. P. Pickel, Institut für Agrartechnik und Landeskultur). Die Messdaten für die Höhe $u(x)$ einer Feldoberfläche am Ort x liegen im Abstand von $\Delta_x = 0.05 \text{ m}$ vor (Abb. 4.1). Bei konstanter Fahrgeschwindigkeit $v = 2.0 \text{ m/s}$ entspricht dies einer Frequenz von $1 / (0.05 \text{ m} / 2.0 \text{ m/s}) = 40 \text{ Hz}$. Die Abb. 4.2–4.5 zeigen für verschiedene maximale Frequenzen f_{\max} den Vergleich zwischen gefilterten Daten und Originaldaten. Die Elimination der hochfrequenten Anteile in $u(t)$ ist deutlich erkennbar.

Bemerkung 4.12 (Schnelle Fourier–Transformation)

engl.: Fast Fourier Transform(ation) (FFT)

Problem Standard–Algorithmus zur Auswertung von \mathcal{F}_N oder \mathcal{F}_N^{-1} würde $\mathcal{O}(N^2)$ Rechenoperationen erfordern (Matrix–Vektor–Multiplikation).

Cooley–Tuckey (1965) Sei $N = 2M$ gerade und $\omega = e^{i\frac{2\pi}{N}}$ oder $\omega = e^{-i\frac{2\pi}{N}}$. Dann gilt für

$$\alpha_j = \sum_{k=0}^{N-1} f_k \omega^{kj}, \quad (j = 0, 1, \dots, N-1)$$

$$\alpha_{2l} = \sum_{k=0}^{M-1} g_k \xi^{kl}, \quad \alpha_{2l+1} = \sum_{k=0}^{M-1} h_k \xi^{kl}, \quad (l = 0, 1, \dots, M-1)$$

mit $M := N/2$, $\xi := \omega^2$ und

$$g_k := f_k + f_{k+M}, \quad h_k := (f_k - f_{k+M}) \omega^k.$$

Mit $2M$ Additionen und $2M$ Multiplikationen ($\omega^k \rightarrow \omega^{k+1} = \omega \cdot \omega^k$, $h_k = (\dots) \cdot \omega^k$) wird die Berechnung von N Summen der Länge N zurückgeführt auf $2M = N$ Summen der Länge $M = N/2$.

Beweis

$$\alpha_{2l} = \sum_{k=0}^{N-1} f_k \omega^{k \cdot 2l} = \sum_{k=0}^{\frac{N}{2}-1} (f_k \omega^{2kl} + f_{k+\frac{N}{2}} \omega^{2(k+\frac{N}{2})l}) = \sum_{k=0}^{M-1} (f_k + f_{k+M}) \omega^{2kl}$$

$$\alpha_{2l+1} = \sum_{k=0}^{N-1} f_k \omega^{(2l+1)k} = \sum_{k=0}^{\frac{N}{2}-1} (f_k \omega^{2kl+k} + f_{k+\frac{N}{2}} \omega^{(2l+1)(k+\frac{N}{2})}) = \sum_{k=0}^{M-1} (f_k - f_{k+\frac{N}{2}}) \omega^k \cdot \omega^{2kl}$$

Rekursive Anwendung besonders einfach für $N = 2^p \Rightarrow$ Aufwand zur Berechnung von $\alpha_0, \alpha_1, \dots, \alpha_{N-1}$ (Analyse oder Synthese) beträgt $2N \log_2 N$ Multiplikationen.

Algorithmus 4.13 (Schnelle Fourier–Transformation)

Sei $N = 2^p$ und $\omega = e^{i\frac{2\pi}{N}}$ oder $\omega = e^{-i\frac{2\pi}{N}}$.

Eingabe: $f_0, f_1, \dots, f_{N-1} \in \mathbb{C}$

Ausgabe: $\alpha_0, \alpha_1, \dots, \alpha_{N-1} \in \mathbb{C}$ mit $\alpha_j := \sum_{k=0}^{N-1} f_k \omega^{kj}$.

```

 $N_{\text{red}} := N; \quad z := \omega$ 
while  $N_{\text{red}} > 1$  do
   $M_{\text{red}} := N_{\text{red}}/2$ 
  for  $j = 0 : (N/N_{\text{red}} - 1)$ 
     $l := jN_{\text{red}}$ 
    for  $k = 0 : M_{\text{red}} - 1$ 
       $a := f_{l+k} + f_{l+k+M_{\text{red}}}$ 
       $f_{l+k+M_{\text{red}}} := (f_{l+k} - f_{l+k+M_{\text{red}}})z^k$ 
       $f_{l+k} := a$ 
     $N_{\text{red}} := M_{\text{red}}; \quad z := z^2$ 
  for  $k = 0 : N - 1$ 
     $\alpha_{\sigma(k)} := f_k$ 

```

Vertauschung der Komponenten von α bestimmt durch Permutation $\sigma(k)$:

$$\sigma\left(\sum_{j=0}^{N-1} a_j 2^j\right) := \sum_{j=0}^{N-1} a_{N-j} 2^j \quad \text{mit } a_0, a_1, \dots, a_{N-1} \in \{0, 1\}.$$

↔ „bit reversal“, einfache Implementierung durch Bitmanipulationen

- Typisches Beispiel eines „divide-and-conquer“-Algorithmus,
- gut parallelisierbar,
- in Signalprozessoren hardwaremäßig verfügbar.

5 Quadratur

Bemerkung 5.1 (Problemstellung)

geg.: stückweise stetige Funktion $f : [a, b] \rightarrow \mathbb{R}$

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

Eigenschaften

- Linearität: $I(\alpha f + \beta g) = \alpha I(f) + \beta I(g)$
- Positivität: $f(x) \geq 0, (x \in [a, b]) \Rightarrow I(f) \geq 0$
- Additivität: $I_a^b(f) = I_a^c(f) + I_c^b(f), (c \in (a, b))$

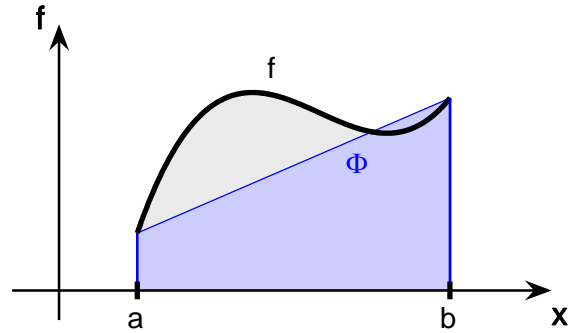
Spezialfall $\tilde{f}(x) = x^k \Rightarrow I_a^b(\tilde{f}) = \int_a^b x^k \, dx = \frac{1}{k+1} x^{k+1} \Big|_a^b$

Idee Approximiere $f : [a, b] \rightarrow \mathbb{R}$ durch ein (Interpolations-)Polynom \tilde{f} und verwende $I(\tilde{f})$ als Näherungswert für $I(f)$.

Beispiel 5.2 (Trapezregel)

a) Approximation von $f : [a, b] \rightarrow \mathbb{R}$ durch lineares Interpolationspolynom Φ mit Stützstellen a und b :

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



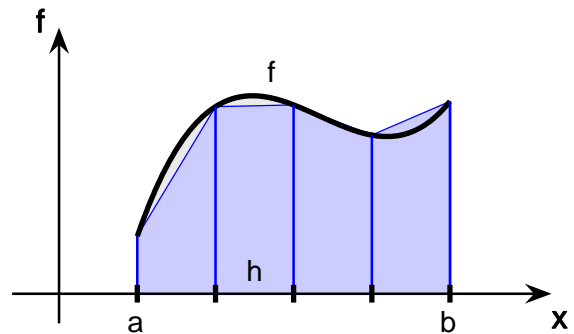
$$I(\Phi) = (b-a)f(a) + \frac{f(b) - f(a)}{b-a} \cdot \frac{1}{2}(x-a)^2 \Big|_a^b = \frac{b-a}{2}(f(a) + f(b))$$

b) Zusammengesetzte Trapezregel, *Trapezsumme*. Wähle ein Gitter

$$\Delta = \{a = x_0 < x_1 < x_2 < \dots < x_N = b\}$$

mit Schrittweiten

$$h_i := x_{i+1} - x_i, \quad (i = 0, 1, \dots, N-1).$$



$$I_a^b(f) = \sum_{i=0}^{N-1} I_{x_i}^{x_{i+1}}(f), \quad \text{Trapezregel: } I_{x_i}^{x_{i+1}}(f) \approx \frac{h_i}{2}(f(x_i) + f(x_{i+1}))$$

Aufsummieren \Rightarrow *zusammengesetzte Trapezregel*:

$$I_a^b(f) \approx \tilde{I}_a^b(f) := \frac{h_0}{2}f(x_0) + \sum_{i=1}^{N-1} \frac{h_{i-1} + h_i}{2}f(x_i) + \frac{h_{N-1}}{2}f(x_N)$$

Spezialfall äquidistantes Gitter $x_i := a + ih$ mit $i = 0, 1, \dots, N$ und $h := (b-a)/N$:

$$I_a^b(f) \approx \tilde{I}_a^b(f) = \frac{h}{2} \left(f(a) + 2 \sum_{i=1}^{N-1} f(a + ih) + f(b) \right).$$

Bemerkung 5.3 (Newton-Cotes-Formeln)

Wähle Interpolationspolynom Φ zu $n + 1$ äquidistanten Stützstellen $x_i := a + ih$ mit $h := (b-a)/n$, ($i = 0, \dots, n$):

$$\Phi(x) = \sum_{i=0}^n f(x_i)L_i^{(n)}(x)$$

$$\int_a^b f(x) dx \approx \int_a^b \Phi(x) dx = \sum_{i=0}^n \left(\int_a^b L_i^{(n)}(x) dx \right) f(x_i) = \sum_{i=0}^n w_i f(x_i)$$

mit *Knoten* x_0, x_1, \dots, x_n und *Gewichten* $w_i := \int_a^b L_i^{(n)}(x) dx$.

Allgemeine Struktur einer *Quadraturformel*: $\tilde{I}(f) = \sum_{i=0}^n w_i f(x_i)$.

Positivität, falls $w_i > 0$, ($i = 0, 1, \dots, n$)

Gewichte der *Newton-Cotes-Quadraturformeln* liegen für $[a, b] = [0, 1]$ tabelliert vor:

n	w_i	Name	Fehler
1	1/2, 1/2	Trapezregel	$h^3/12 f''(\tau)$
2	1/6, 2/3, 1/6	Keplersche Fassregel	$h^5/90 f^{IV}(\tau)$
3	1/8, 3/8, 3/8, 1/8	Newtonsche 3/8-Regel	$3h^5/80 f^{IV}(\tau)$

Positivität nur bis $n \leq 7$.

Für $n \geq 8$ treten auch negative Gewichte auf \rightsquigarrow praktisch unbrauchbar.

Analog zu Beispiel 5.2 definiert man *zusammengesetzte Newton-Cotes-Formeln*:

Simpson-Regel Zusammengesetzte Keplersche Fassregel.

$$S(h) := \frac{h}{3} \left(f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + 2f(a+4h) + \dots + 4f(a+(2N-1)h) + f(b) \right) \quad \text{mit } h := \frac{b-a}{2N}.$$

Bemerkung 5.4 (Fehlerabschätzungen für Newton-Cotes-Formeln)

Trapezregel Restglied der Polynominterpolation, vgl. Satz 1.13 \Rightarrow

$$f(x) - \Phi(x) = \frac{f''(\xi)}{2} (x-a)(x-b) \quad \text{mit einem } \xi \in [a, b]$$

$$\begin{aligned} \left| \underbrace{\int_a^b f(x) dx}_{I(f)} - \underbrace{\int_a^b \Phi(x) dx}_{\tilde{I}(f)} \right| &= \left| \int_a^b \frac{f''(\xi)}{2} \underbrace{(x-a)(x-b)}_{<0} dx \right| \\ &\leq \max_{\xi \in [a,b]} |f''(\xi)| \cdot \frac{1}{2} \int_a^b (x-a)(b-x) dx \\ &= \max_{\xi \in [a,b]} |f''(\xi)| \cdot \frac{(b-a)^3}{12} \end{aligned}$$

Polynome ersten Grades werden durch die Trapezregel exakt integriert.

Keplersche Fassregel $n = 2$, $h = \frac{b-a}{2}$, $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$

$$f(x) - \Phi(x) = \frac{f'''(\xi)}{3!} (x-a) \left(x - \frac{a+b}{2}\right) (x-b)$$

Ähnlich wie bei der Trapezregel zeigt man

$$|I(f) - \tilde{I}(f)| \leq \frac{h^5}{90} \max_{\xi \in [a,b]} |f^{IV}(\xi)|.$$

Polynome bis zum Grad 3 (!) werden exakt integriert.

Bemerkung 5.5 (Gauß–Christoffel–Quadraturformeln)

Idee Wähle Gewichte w_i und Knoten x_i so, dass Polynome möglichst hohen Grades exakt integriert werden:

$$\sum_{i=0}^n w_i x_i^j = \tilde{I}(x^j) \stackrel{!}{=} I(x^j) = \int_a^b x^j dx = \frac{b^{j+1} - a^{j+1}}{j+1}, \quad (j = 0, 1, \dots, m)$$

$m+1$ nichtlineare Gleichungen für $2n+2$ Unbekannte $x_0, x_1, \dots, x_n, w_0, w_1, \dots, w_n$.

Ergebnisse

- $m = 2n + 1$ ist stets erreichbar,
- Knoten x_i und Gewichte w_i sind für $m = 2n + 1$ eindeutig bestimmt,
- Knoten x_i sind Nullstellen der sog. Legendre–Polynome.

Beispiel $n = 1 \Rightarrow x_{0,1} = \frac{a+b}{2} \pm \frac{b-a}{2\sqrt{3}}$, $w_{0,1} = \frac{b-a}{2}$

$$\tilde{I}(f) = \frac{b-a}{2} (f(x_0) + f(x_1))$$

integriert Polynome bis zum Grad $2n+1 = 3$ exakt.

Erweiterung $\int_a^b \omega(x) f(x) dx$ mit Gewichtsfunktion $\omega(x) \geq 0$, ($x \in [a, b]$).

Die optimalen Knoten x_0, x_1, \dots, x_n ergeben sich auch hier als Nullstellen orthogonaler Polynome.

Beispiel Tschebyscheff–Quadratur

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}} dx \approx \sum_{i=0}^n w_i f(x_i) \quad \text{mit} \quad x_i = \cos \frac{2i+1}{2n+2} \pi, \quad w_i = \frac{\pi}{n+1}, \quad (i = 0, 1, \dots, n)$$

Literatur Deuffhard/Hohmann, Kapitel 9.3.

Bemerkung 5.6 (Romberg–Quadratur)

Die zusammengesetzte Trapezregel $T(h)$ aus Beispiel 5.2b) ergibt (theoretisch) für $h \rightarrow 0$ den exakten Wert $I_a^b(f)$.

Idee Berechne $T(h_i)$ für einige endliche Schrittweiten $h_i > 0$, interpoliere die Stützpunkte $(h_1, T(h_1)), \dots, (h_k, T(h_k))$ durch ein Polynom $\pi(h)$ mit $\deg \pi \leq k - 1$ und setze $I_a^b(f) \approx \tilde{I}_a^b(f) := \pi(0)$.

Theorie Asymptotische Entwicklung des Fehlers der Trapezsumme (Euler–Maclaurinische Summenformel):

$$T(h) = \int_a^b f(x) dx + \sum_{k=1}^M c_k h^{2k} + \mathcal{O}(h^{2M+2}) \quad \text{mit Konstanten } c_k.$$

praktisch Wähle $h_i = H/n_i$ mit Grundschriftweite H und $n_i = 2^{i-1}$ und bestimme Interpolationspolynom π mit $\pi(h_i^2) = T(h_i)$, ($i = 1, \dots, k$). Berechnung von $\pi(0)$ mittels Neville–Schema, vgl. Bemerkung 1.10:

$$T_{i, \dots, i+l} = T_{i+1, \dots, i+l} - \frac{T_{i+1, \dots, i+l} - T_{i, \dots, i+l-1}}{\left(\frac{h_i}{h_{i+l}}\right)^2 - 1} \quad \text{mit} \quad \left(\frac{h_i}{h_{i+l}}\right)^2 = \left(\frac{n_{i+l}}{n_i}\right)^2$$

und $T_i := T(h_i)$. Ergebnis: $I_a^b(f) \approx T_{1, \dots, k}$ (*Romberg–Quadratur, Romberg–Schema*).

wichtig Für $n_i = 2^{i-1}$ ist $h_i = 2h_{i+1}$ und

$$\begin{aligned} T(h_{i+1}) &= \frac{h_{i+1}}{2} (f(a) + 2 \sum_{j=1}^{n_{i+1}-1} f(a + jh_{i+1}) + f(b)) \\ &= \frac{1}{2} T(h_i) + h_{i+1} \sum_{k=0}^{n_i-1} f(a + (2k+1)h_{i+1}) \end{aligned}$$

allgemein Romberg–Quadratur ist Spezialfall von *Extrapolationsverfahren*, die immer dann angewendet werden können, wenn der Fehler eines numerischen Verfahrens eine asymptotische Entwicklung in Potenzen von h hat.

Beispiel 5.7 (Extrapolation und Differenzenquotient)

a) Rechtsseitiger Differenzenquotient erster Ordnung

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Taylorentwicklung von $f(x+h)$ ergibt asymptotische h –Entwicklung des Fehlers.

b) Zentraler Differenzenquotient zur Approximation der ersten Ableitung

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Fehler hat h^2 -Entwicklung (Taylorentwicklung von $f(x+h)$ und $f(x-h)$).

6 Iterationsverfahren für lineare und nichtlineare Gleichungssysteme

6.1 Nullstellen reeller Funktionen

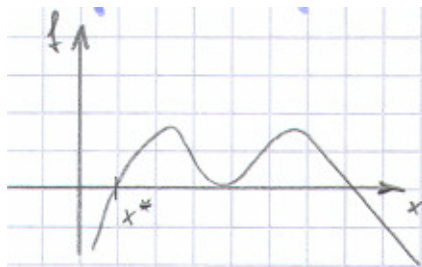
Bemerkung 6.1 (Problemstellung)

geg.: $f \in C[a, b]$

ges.: $x^* \in [a, b]$ mit $f(x^*) = 0$

Lösungstheorie

- f linear $\Rightarrow f(x) = 0$ genau dann eindeutig lösbar in \mathbb{R} , falls $f' \neq 0$.
- f nichtlinear \Rightarrow i. Allg. nur Aussagen über lokale Eindeutigkeit der Lösung



- Satz über die implizite Funktion: Ist $f(x^*) = 0$, $f \in C^1[a, b]$ und $f'(x^*) \neq 0$, so ist $y = f(x)$ in einer Umgebung von x^* eindeutig nach x auflösbar: $x = x(y)$.
- $f'(x^*) = 0 \Rightarrow$ mehrfache Nullstelle, numerische Bestimmung oft kompliziert



- $f(a) \cdot f(b) < 0 \Rightarrow$ es existiert ein x^* mit $f(x^*) = 0$ (Zwischenwertsatz)

Bemerkung 6.2 (Bisektionsverfahren)

geg.: $f \in C[a, b]$, Intervallenden a, b mit $f(a) \cdot f(b) < 0$, Abbruchschranke TOL

```
Initialisierung:  $f_a := f(a), f_b := f(b)$ .
repeat
   $c := \frac{a+b}{2}, f_c := f(c)$ 
  if  $f_a \cdot f_c < 0$  then  $b := c, f_b := f_c$ 
  else  $a := c, f_a := f_c$ 
until  $|b - a| \leq \text{TOL}$ 
Ergebnis:  $x^* \approx x_{\text{bi}} := \frac{a+b}{2}$ 
```

Konvergenz stets gesichert: Werden mindestens $1 + \log_2 \frac{b-a}{\text{TOL}}$ Iterationsschritte ausgeführt, so gilt

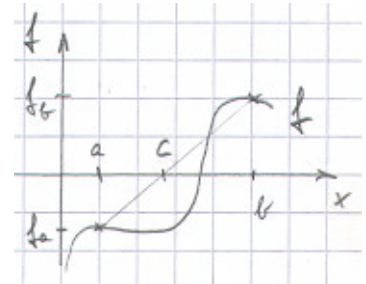
$$|x_{\text{bi}} - x^*| \leq \text{TOL}.$$

beachte Wegen der Rundungsfehler bei der Auswertung von f kann x^* außerhalb des numerisch bestimmten Intervalls $[a, b]$ liegen.

praktisch sehr robust, einfach zu implementieren, aber sehr langsame Konvergenz

Bemerkung 6.3 (Regula falsi)

Idee Bestimme wie im Bisektionsverfahren immer kleinere Intervalle, die x^* enthalten, berücksichtige bei der Wahl von c jedoch den Lösungsverlauf.



praktisch Wähle c als Nullstelle des (linearen) Interpolationspolynoms zu den Stützpunkten (a, f_a) und (b, f_b) .

geg.: $f \in C[a, b]$, Intervallenden a, b mit $f(a) \cdot f(b) < 0$, Abbruchschranke TOL

```
Initialisierung:  $f_a := f(a), f_b := f(b)$ .
repeat
   $c := \frac{af_b - bf_a}{f_b - f_a}, f_c := f(c)$ 
  if  $f_a \cdot f_c < 0$  then  $b := c, f_b := f_c$ 
  else  $a := c, f_a := f_c$ 
until  $|b - a| \leq \text{TOL}$ 
Ergebnis:  $x^* \approx x_{\text{rf}} := \frac{a+b}{2}$ 
```

Regula falsi konvergiert i. Allg. deutlich schneller als das Bisektionsverfahren.

Problem Langsame Konvergenz, wenn eines der beiden Intervallenden stets unverändert bleibt wie z. B. für $f(x) := x^{10} - 1/2$, ($x \in [0, 1]$).

Alternative Bleibt eines der beiden Intervallenden a bzw. b über mehr als einen Iterationsschritt unverändert, so ersetze $f_a := \frac{1}{2}f_a$ bzw. $f_b := \frac{1}{2}f_b$.

Bemerkung 6.4 (Sekantenverfahren)

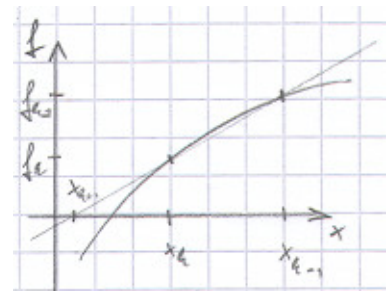
Verzichtet man auf eine Einschließung der Nullstelle x^* , so ergibt sich ausgehend von $(x_{k-1}, f(x_{k-1}))$, $(x_k, f(x_k))$ in der Regel eine wesentlich bessere Näherung für f :

Betrachte das (lineare) Interpolationspolynom

$$f(x) \approx f_k(x) := f_{k-1} + \frac{x - x_{k-1}}{x_k - x_{k-1}} (f_k - f_{k-1})$$

und bestimme aus $f_k(x) \stackrel{!}{=} 0$ die neue Näherung

$$x_{k+1} := x_k - f_k \frac{x_k - x_{k-1}}{f_k - f_{k-1}}, \quad f_{k+1} := f(x_{k+1}).$$



Konvergenzordnung des Sekantenverfahrens $q := (1 + \sqrt{5})/2$, d. h.

$$0 \leq \lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^q} < \infty$$

praktisch Schnelle Konvergenz für gute Startwerte, jedoch Gefahr der Divergenz für schlechte Startwerte

Verallgemeinerung Inverse Interpolation.

Hinzunahme weiterer Stützpunkte $(x_i, f(x_i))$, Bestimmung des Interpolationspolynoms $\pi(y)$ zu den Stützstellen $y = f_k, f_{k-1}, f_{k-2}, \dots$ und Stützwerten $\pi = x_k, x_{k-1}, x_{k-2}, \dots$ und Wahl von c als $c := \pi(0)$.

Bemerkung 6.5 (Newtonverfahren)

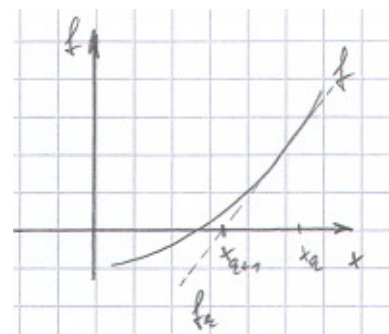
geg.: $f \in C^1[a, b]$

Linearisierung von f in x_k :

$$f(x) \approx f_k(x) := f(x_k) + f'(x_k)(x - x_k)$$

Bestimmung von x_{k+1} als Nullstelle von $f_k(x)$:

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}$$

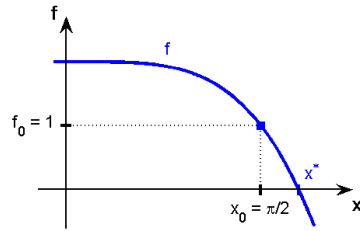


Beispiel 6.6: Newtonverfahren

gesucht

Kleinste positive Lösung von

$$f(x) = \cos x \cosh x + 1 = 0$$



Newtonverfahren

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad \text{Startwert } x_0 := \frac{\pi}{2}$$

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	1.0000E+00	-2.5092	3.0431E-01
1	1.969333142133283	-4.1751E-01	-4.7298	9.4229E-02
2	1.881060554590512	-2.4757E-02	-4.1744	5.9565E-03
3	1.875129963043149	-1.0716E-04	-4.1383	2.5894E-05
4	1.875104069204172	-2.0368E-09	-4.1381	4.9221E-10
5	1.875104068711961	2.2204E-16	-4.1381	< 1.0E-16
6	1.875104068711961			



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 6.1: Quadratische Konvergenz des Newtonverfahrens.

Quadratisch konvergent für einfache Nullstellen, d. h.

$$0 \leq \lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < \infty,$$

linear konvergent für mehrfache Nullstellen.

Beispiel 6.6 (Newtonverfahren)

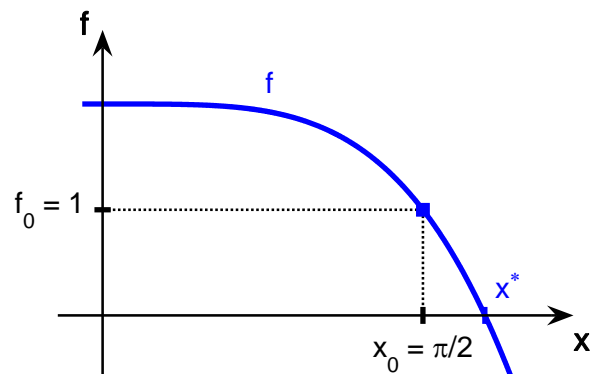
Berechnung der kleinsten positiven Lösung von $f(x) = \cos x \cosh x + 1 = 0$.

Newtonverfahren

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

mit Startwert $x_0 := \pi/2$ und

$$f'(x) = \cos x \sinh x - \sin x \cosh x.$$



k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	1.0000 _E + 00	-2.5092	3.0431 _E - 01
1	1.969333142133283	-4.1751 _E - 01	-4.7298	9.4229 _E - 02
2	1.881060554590512	-2.4757 _E - 02	-4.1744	5.9565 _E - 03
3	1.875129963043149	-1.0716 _E - 04	-4.1383	2.5894 _E - 05
4	1.875104069204172	-2.0368 _E - 09	-4.1381	4.9221 _E - 10
5	1.875104068711961	2.2204 _E - 16	-4.1381	< 1.0 _E - 16
6	1.875104068711961			

Ergebnis Folge (x_k) konvergiert quadratisch gegen $x^* = 1.875104068711961 \dots$

Beispiel 6.8: Vereinfachtes Newtonverfahren

gesucht Kleinste positive Lösung von $f(x) = \cos x \cosh x + 1 = 0$.

Vereinfachtes Newtonverfahren $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}$

Startwert $x_0 = 2.0$: (x_k) konvergiert linear mit $\alpha \approx 0.16$.

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	2.0000000000000000	-5.6563E-01	-4.9303	1.2490E-01
1	1.885274674997890	-4.2402E-02		1.0171E-02
2	1.876674249774155	-6.5051E-03		1.5702E-03
3	1.875354824372530	-1.0379E-03		2.5076E-04
4	1.875144317977280	-1.6656E-04		4.0249E-05
5	1.875110534418510	-2.6756E-05		6.4657E-06
6	1.875105107508024	-4.2987E-06		1.0388E-06
7	1.875104235610924	-6.9065E-07		1.6690E-07
8	1.875104095527000	-1.1096E-07		2.6815E-08
9	1.875104073020237	-1.7828E-08		4.3083E-09
10	1.875104069404156	-2.8644E-09		6.9220E-10



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 6.2: Lineare Konvergenz des vereinfachten Newtonverfahrens, einmalige Auswertung der Ableitung $f'(x_k)$, Startwert $x_0 = 2.0$.

6.2 Das Newtonverfahren

Bemerkung 6.7 (Newton–Raphson–Verfahren)

Beispiel 6.8 (Vereinfachtes Newtonverfahren)

Betrachte zur Funktion $f(x)$ aus Beispiel 6.6 das vereinfachte Newtonverfahren

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}.$$

Startwert $x_0 = 2.0$ (x_k) konvergiert linear mit $\alpha \approx 0.16$

Beispiel 6.8: Vereinfachtes Newtonverfahren (II)

gesucht Kleinste positive Lösung von $f(x) = \cos x \cosh x + 1 = 0$.

Vereinfachtes Newtonverfahren
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}$$

Startwert $x_0 = \pi/2$: (x_k) konvergiert linear mit $\alpha \approx 0.65$.

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	1.0000E+00	-2.5092	3.0431E-01
1	1.969333142133283	-4.1751E-01		9.4229E-02
2	1.802938787863725	2.8309E-01		7.2165E-02
3	1.915761753759818	-1.7332E-01		4.0658E-02
4	1.846688134029678	1.1515E-01		2.8416E-02
5	1.892580923809663	-7.3253E-02		1.7477E-02
6	1.863386753854851	4.8072E-02		1.1717E-02
7	1.882545193155160	-3.0961E-02		7.4411E-03
8	1.870206143078939	2.0195E-02		4.8979E-03
9	1.878254787141379	-1.3068E-02		3.1507E-03
10	1.873046598671781	8.5012E-03		2.0575E-03



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 6.3: Lineare Konvergenz des vereinfachten Newtonverfahrens, einmalige Auswertung der Ableitung $f'(x_k)$, Startwert $x_0 = \pi/2$.

Beispiel 6.8: Vereinfachtes Newtonverfahren (III)

gesucht Kleinste positive Lösung von $f(x) = \cos x \cosh x + 1 = 0$.

Vereinfachtes Newtonverfahren

mit Neuberechnung von $f'(x_k)$ in jedem fünften Iterationsschritt

Deutlich besseres Konvergenzverhalten: $\alpha \approx 0.025$ für $k \geq 5$

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	1.0000E+00	-2.5092	3.0431E-01
1	1.969333142133283	-4.1751E-01		9.4229E-02
2	1.802938787863725	2.8309E-01		7.2165E-02
3	1.915761753759818	-1.7332E-01		4.0658E-02
4	1.846688134029678	1.1515E-01		2.8416E-02
5	1.892580923809663	-7.3253E-02	-4.2450	1.7477E-02
6	1.875324505147979	-9.1234E-04		2.2044E-04
7	1.875109582992217	-2.2819E-05		5.5143E-06
8	1.875104207501406	-5.7433E-07		1.3879E-07
9	1.875104072205700	-1.4458E-08		3.4937E-09
10	1.875104068799909	-3.6394E-10	-4.1381	8.7948E-11



Martin-Luther-Universität Halle-Wittenberg, FB Mathematik und Informatik
Martin Arnold: Numerische Mathematik für Fachrichtung Informatik und Lehramt (WiS 2005/06)

Abbildung 6.4: Lineare Konvergenz des vereinfachten Newtonverfahrens, Auswertung der Ableitung $f'(x_k)$ in jedem fünften Iterationsschritt, Startwert $x_0 = \pi/2$.

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	2.000000000000000	$-5.6563_{\text{E}} - 01$	-4.9303	$1.2490_{\text{E}} - 01$
1	1.885274674997890	$-4.2402_{\text{E}} - 02$		$1.0171_{\text{E}} - 02$
2	1.876674249774155	$-6.5051_{\text{E}} - 03$		$1.5702_{\text{E}} - 03$
3	1.875354824372530	$-1.0379_{\text{E}} - 03$		$2.5076_{\text{E}} - 04$
4	1.875144317977280	$-1.6656_{\text{E}} - 04$		$4.0249_{\text{E}} - 05$
5	1.875110534418510	$-2.6756_{\text{E}} - 05$		$6.4657_{\text{E}} - 06$
6	1.875105107508024	$-4.2987_{\text{E}} - 06$		$1.0388_{\text{E}} - 06$
7	1.875104235610924	$-6.9065_{\text{E}} - 07$		$1.6690_{\text{E}} - 07$
8	1.875104095527000	$-1.1096_{\text{E}} - 07$		$2.6815_{\text{E}} - 08$
9	1.875104073020237	$-1.7828_{\text{E}} - 08$		$4.3083_{\text{E}} - 09$
10	1.875104069404156	$-2.8644_{\text{E}} - 09$		$6.9220_{\text{E}} - 10$

Startwert $x_0 = \pi/2$ (x_k) konvergiert linear mit $\alpha \approx 0.65$

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	$1.0000_{\text{E}} + 00$	-2.5092	$3.0431_{\text{E}} - 01$
1	1.969333142133283	$-4.1751_{\text{E}} - 01$		$9.4229_{\text{E}} - 02$
2	1.802938787863725	$2.8309_{\text{E}} - 01$		$7.2165_{\text{E}} - 02$
3	1.915761753759818	$-1.7332_{\text{E}} - 01$		$4.0658_{\text{E}} - 02$
4	1.846688134029678	$1.1515_{\text{E}} - 01$		$2.8416_{\text{E}} - 02$
5	1.892580923809663	$-7.3253_{\text{E}} - 02$		$1.7477_{\text{E}} - 02$
6	1.863386753854851	$4.8072_{\text{E}} - 02$		$1.1717_{\text{E}} - 02$
7	1.882545193155160	$-3.0961_{\text{E}} - 02$		$7.4411_{\text{E}} - 03$
8	1.870206143078939	$2.0195_{\text{E}} - 02$		$4.8979_{\text{E}} - 03$
9	1.878254787141379	$-1.3068_{\text{E}} - 02$		$3.1507_{\text{E}} - 03$
10	1.873046598671781	$8.5012_{\text{E}} - 03$		$2.0575_{\text{E}} - 03$

Startwert $x_0 = \pi/2$, Neuberechnung von $f'(x_k)$ in jedem 5. Iterationsschritt
 Deutliche Verbesserung des Konvergenzverhaltens: $\alpha \approx 0.025$ für $k \geq 5$.

k	x_k	$f(x_k)$	$f'(x_k)$	$ x_k - x^* $
0	1.570796326794897	$1.0000_{\text{E}} + 00$	-2.5092	$3.0431_{\text{E}} - 01$
1	1.969333142133283	$-4.1751_{\text{E}} - 01$		$9.4229_{\text{E}} - 02$
2	1.802938787863725	$2.8309_{\text{E}} - 01$		$7.2165_{\text{E}} - 02$
3	1.915761753759818	$-1.7332_{\text{E}} - 01$		$4.0658_{\text{E}} - 02$
4	1.846688134029678	$1.1515_{\text{E}} - 01$		$2.8416_{\text{E}} - 02$
5	1.892580923809663	$-7.3253_{\text{E}} - 02$	-4.2450	$1.7477_{\text{E}} - 02$
6	1.875324505147979	$-9.1234_{\text{E}} - 04$		$2.2044_{\text{E}} - 04$
7	1.875109582992217	$-2.2819_{\text{E}} - 05$		$5.5143_{\text{E}} - 06$
8	1.875104207501406	$-5.7433_{\text{E}} - 07$		$1.3879_{\text{E}} - 07$
9	1.875104072205700	$-1.4458_{\text{E}} - 08$		$3.4937_{\text{E}} - 09$
10	1.875104068799909	$-3.6394_{\text{E}} - 10$	-4.1381	$8.7948_{\text{E}} - 11$

Lemma 6.9 (Kontrahierende Abbildungen)

Satz 6.10 (Banachscher Fixpunktsatz)

Sei $E \subset \mathbb{R}^n$ kompakt und $\Phi : E \rightarrow E$ kontrahierend. Dann gilt:

- a) Φ hat genau einen Fixpunkt x^* in E : $\Phi(x^*) = x^*$.
- b) Für jeden Startwert $x_0 \in E$ konvergiert die Fixpunktiteration $x_{k+1} = \Phi(x_k)$ gegen x^* und es gilt:

$$(i) \quad \|x_k - x^*\| \leq \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\|,$$

$$(ii) \quad \|x_{k+1} - x^*\| \leq \frac{\alpha}{1 - \alpha} \|x_{k+1} - x_k\|.$$

Beweis Wegen $\Phi : E \rightarrow E$ folgt mittels vollständiger Induktion $x_k \in E$ und

$$\|x_{k+1} - x_k\| = \|\Phi(x_k) - \Phi(x_{k-1})\| \leq \alpha \|x_k - x_{k-1}\| \leq \dots \leq \alpha^k \|x_1 - x_0\|, \quad (k \geq 0).$$

Aus der Dreiecksungleichung folgt

$$\begin{aligned} \|x_{k+m} - x_k\| &= \|x_{k+m} - x_{k+m-1} + x_{k+m-1} - x_{k+m-2} + x_{k+m-2} - \dots - x_k\| \\ &\leq \|x_{k+m} - x_{k+m-1}\| + \|x_{k+m-1} - x_{k+m-2}\| + \dots + \|x_{k+1} - x_k\| \\ &\leq (\alpha^{k+m-1} + \alpha^{k+m-2} + \dots + \alpha^k) \|x_1 - x_0\| \\ &\leq \alpha^k \sum_{i=0}^{\infty} \alpha^i \cdot \|x_1 - x_0\| = \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\|, \end{aligned}$$

also ist $(x_k)_k$ eine Cauchy-Folge, denn wählt man zu vorgegebenem $\varepsilon > 0$ ein k_0 mit

$$\frac{\alpha^{k_0}}{1 - \alpha} \|x_1 - x_0\| \leq \varepsilon,$$

so gilt für alle $k \geq k_0$ und alle $m \geq 0$ die Abschätzung $\|x_{k+m} - x_k\| \leq \varepsilon$. Als Cauchy-Folge im Kompaktum E hat $(x_k)_k$ einen Häufungspunkt $x^* = \lim_{k \rightarrow \infty} x_k \in E$ mit

$$\|x^* - x_k\| = \lim_{m \rightarrow \infty} \|x_{k+m} - x_k\| \leq \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\|.$$

Dieser Häufungspunkt ist Fixpunkt von Φ , denn

$$\begin{aligned} \|x^* - \Phi(x^*)\| &= \|x^* - x_{k+1} + \Phi(x_k) - \Phi(x^*)\| \\ &\leq \|x^* - x_{k+1}\| + \|\Phi(x_k) - \Phi(x^*)\| \leq \|x^* - x_{k+1}\| + \alpha \|x_k - x^*\| \rightarrow 0. \end{aligned}$$

Der Fixpunkt ist eindeutig bestimmt, denn aus $x_1^* = \Phi(x_1^*)$ und $x_2^* = \Phi(x_2^*)$ folgt

$$\|x_2^* - x_1^*\| = \|\Phi(x_2^*) - \Phi(x_1^*)\| \leq \alpha \|x_2^* - x_1^*\|,$$

also $\underbrace{(1 - \alpha)}_{>0} \|x_2^* - x_1^*\| \leq 0$ und $x_2^* = x_1^*$.

Zum Beweis von (ii) verwendet man

$$\begin{aligned}\|x_{k+1+m} - x_{k+m}\| &= \|\Phi(x_{k+m}) - \Phi(x_{k+m-1})\| \\ &\leq \alpha \|x_{k+m} - x_{k+m-1}\| \leq \dots \leq \alpha^m \|x_{k+1} - x_k\|, \quad (m \geq 1),\end{aligned}$$

um wie oben unter Verwendung der Dreiecksungleichung die Abschätzung

$$\|x_{k+1+m} - x_{k+1}\| \leq (\alpha^m + \alpha^{m-1} + \dots + \alpha) \|x_{k+1} - x_k\| \leq \frac{\alpha}{1 - \alpha} \|x_{k+1} - x_k\|, \quad (m \geq 1),$$

zu zeigen, aus der die Behauptung durch Grenzübergang $m \rightarrow \infty$ folgt. ■

Bemerkung 6.11 (Konvergenz des Newtonverfahrens)

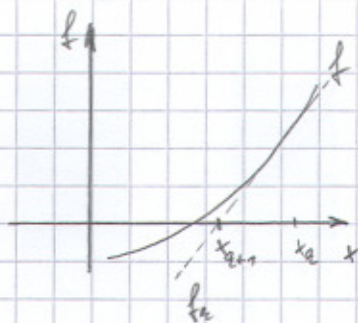
Bemerkung 6.12 (Gedämpftes Newtonverfahren)

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}$$

Quadratische Konvergenz für einfache Nullstellen, d.h.

$$0 \equiv \lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < \infty,$$

linear konvergent für mehrfache Nullstellen.



Arnold

5.2. Das Newtonverfahren

Bemerkung 5.6: Newton-Raphson-Verfahren.

geg.: $F: D \rightarrow \mathbb{R}^n$ mit $D \subset \mathbb{R}^n$, F stetig differenzierbar

Anfangsnäherung $x_0 \in D$

ges.: $x^* \in D$ mit $F(x^*) = 0$

Algorithmus:

Schritt 0

$k := 0$

↗ Jacobimatrix

↗ Bem. 2.5

Schritt 1

Berechne $J := F_x(x_k)$ und LU-Zerlegung von J

Schritt 2

Berechne $F := F(x_k)$

Schritt 3

Berechne p_k mit $J \cdot p_k = -F$ mittels Gaußsches und Rückwärtssubstitution.

Schritt 4

$x_{k+1} := x_k + p_k$, $k := k+1$

if konvergent then stop

else go to Schritt 1

x_{k+1} ist Nullstelle der linearisierten Funktion $F(x_k) + F_x(x_k)(x - x_k)$.

praktisch: Berechne Ableitungen $F_x(x_k)$ analytisch unter Verwendung mathematischer Hilfsprogramme (Maple, Mathematica) oder numerisch mittels Differenzquotienten. ↗ vgl. Beispiel 4.7

Ver Einfachtes Newtonverfahren: Vermeidung der häufigen Neuberechnung und LU-Zerlegung der Jacobimatrix, indem f über mehrere Iterationsauswertungen konstant gehalten wird.

Arnold

? Auswirkungen auf das Konvergenzverhalten und auf die Konvergenzgeschwindigkeit

? Kriterium für die Neuberechnung der Jacobimatrix

Lemma 5.7: Kontrahierende Abbildungen.

Sei $D \subset \mathbb{R}^n$ eine konvexe offene Menge und $\Phi: D \rightarrow \mathbb{R}^n$ stetig differenzierbar. Existiert $\bar{\alpha} := \sup_{x \in D} \|\Phi'_x\|$ und ist $\bar{\alpha} < 1$, so ist Φ kontrahierend.

d.h., es gibt ein $\bar{\alpha} \in [0, 1)$ so, dass $\|\Phi(y) - \Phi(x)\| \leq \bar{\alpha} \|y - x\|$, $(x, y \in D)$.

Beweis: Für $\varphi(\vartheta) := \Phi(x + \vartheta(y-x))$, $(\vartheta \in [0, 1])$ ist $\varphi'(\vartheta) = \Phi'_x(x + \vartheta(y-x)) \cdot (y-x)$ und $\varphi(1) - \varphi(0) = \int_0^1 \varphi'(\vartheta) d\vartheta$, also

$$\|\Phi(y) - \Phi(x)\| \leq \int_0^1 \|\Phi'_x(x + \vartheta(y-x))\| \cdot \|y-x\| d\vartheta \leq \bar{\alpha} \|y-x\|.$$

Satz 5.8: Banachsches Fixpunktsatz.

Sei $E \subset \mathbb{R}^n$ kompakt und $\Phi: E \rightarrow E$ kontrahierend. Dann gilt:

a) Φ hat genau einen Fixpunkt x^* in E : $\Phi(x^*) = x^*$.

b) Für jeden Startwert $x_0 \in E$ konvergiert die Fixpunktiteration $x_{k+1} = \Phi(x_k)$

gegen x^* und es gilt:

(i) $\|x^* - x_k\| \leq \frac{\alpha^k}{1-\alpha} \|x_1 - x_0\|$

(ii) $\|x^* - x_{k+1}\| \leq \frac{\alpha}{1-\alpha} \|x_{k+1} - x_k\|$

Beweis: Liegen $\Phi: E \rightarrow E$ folgt mittels vollständiger Induktion $x_k \in E$ und

$$\|x_{k+1} - x_k\| = \|\Phi(x_k) - \Phi(x_{k-1})\| \leq \alpha \|x_k - x_{k-1}\| \leq \dots \leq \alpha^k \|x_1 - x_0\|,$$

($k \geq 0$).

Aus der Dreiecksungleichung folgt

Arnold

$$\begin{aligned}\|x_{k+m} - x_k\| &= \|x_{k+m} - x_{k+m-1} + x_{k+m-1} - x_{k+m-2} + x_{k+m-2} - \dots - x_k\| \\ &\leq \|x_{k+m} - x_{k+m-1}\| + \|x_{k+m-1} - x_{k+m-2}\| + \dots + \|x_{k+1} - x_k\| \\ &\leq (\alpha^{k+m-1} + \alpha^{k+m-2} + \dots + \alpha^k) \|x_1 - x_0\| \\ &\leq \alpha^k \sum_{i=0}^{\infty} \alpha^i \cdot \|x_1 - x_0\| = \frac{\alpha^k}{1-\alpha} \|x_1 - x_0\|,\end{aligned}$$

also ist $(x_k)_k$ eine Cauchy-Folge, denn wählt man zu vorgegebenem $\varepsilon > 0$ ein k_0 mit $\frac{\alpha^{k_0}}{1-\alpha} \|x_1 - x_0\| \leq \varepsilon$, so gilt für alle $k \geq k_0$ und alle $m \geq 0$

$\|x_{k+m} - x_k\| \leq \varepsilon$. Als Cauchy-Folge im Kompaktum E hat $(x_k)_k$ einen Häufungspunkt $x^* = \lim_{k \rightarrow \infty} x_k \in E$ mit $\|x^* - x_k\| \leq \frac{\alpha^k}{1-\alpha} \|x_1 - x_0\|$.

Dieser Häufungspunkt ist Fixpunkt von Φ^* , denn

$$\begin{aligned}\|x^* - \Phi(x^*)\| &\stackrel{!}{=} \|x^* - x_{k+1} + \Phi(x_k) - \Phi(x^*)\| \\ &\leq \|x^* - x_{k+1}\| + \|\Phi(x_k) - \Phi(x^*)\| \leq \|x^* - x_{k+1}\| + \alpha \|x_k - x^*\| \rightarrow 0.\end{aligned}$$

Der Fixpunkt ist eindeutig bestimmt, denn aus $x_1^* = \Phi(x_1^*)$ und $x_2^* = \Phi(x_2^*)$ folgt $\|x_2^* - x_1^*\| = \|\Phi(x_2^*) - \Phi(x_1^*)\| \leq \alpha \|x_2^* - x_1^*\|$,

also $\underbrace{(1-\alpha)}_{>0} \|x_2^* - x_1^*\| \leq 0$ und $x_1^* = x_2^*$.

Die zweite Fehlerschranke folgt wie oben. ■

↑ 8.1.04

Satz 5.9: Konvergenz des Newtonverfahrens.

(Vereinfachtes) Newtonverfahren entspricht Fixpunktabbildung $x_{k+1} = \Phi(x_k)$

mit $\Phi(x) := \tilde{\Phi}(x) := x - (F_x(x))^{-1} F(x)$

bzw. $\Phi(x) = \tilde{\Phi}_z(x) := x - (F_x(x_z))^{-1} F(x)$.

Konvergenz:

Anzahl

$$a) \frac{\partial}{\partial x} \tilde{\Phi}(x) = I - (F_x(x))^{-1} F_x(x) - \frac{\partial}{\partial x} \left((F_x(x))^{-1} v \right) \Big|_{v=F(x)}$$

$$= \sigma(\|F(x)\|),$$

falls F zweimal stetig differenzierbar und $\|F_{xx}(x)\|$ und $\|(F_x(x))^{-1}\|$ gleichmäßig beschränkt \Rightarrow konvergiert für $\|x - x^*\| \ll 1$ wegen $F(x^*) = 0$.
 \nearrow sehr schnelle Konvergenz \rightarrow für $\|x_k - x^*\| \ll 1$

$$b) \frac{\partial}{\partial x} \tilde{\Phi}_{\frac{1}{2}}(x) = I - (F_x(x_{\frac{1}{2}}))^{-1} F_x(x) = (F_x(x_{\frac{1}{2}}))^{-1} (F_x(x_{\frac{1}{2}}) - F_x(x)).$$

Da $(F_x(x_{\frac{1}{2}}))^{-1} F_x(x)$ Lipschitz-stetig mit Konstante L und $\|x - x_{\frac{1}{2}}\| \leq \Delta_0$, so folgt

$$\left\| \frac{\partial}{\partial x} \tilde{\Phi}_{\frac{1}{2}}(x) \right\| \leq L \cdot \Delta_0 < 1, \text{ falls } \Delta_0 \text{ hinreichend klein.}$$

Ergebnis: Für hinreichend gute Startwerte x_0 konvergiert das Newtonverfahren sehr schnell, man zeigt: $\|x_{k+1} - x^*\| \leq C \cdot \|x_k - x^*\|^2$ mit $C > 0$ und \nearrow quadratische Konvergenz mit einer Konstante $C > 0$. Das vereinfachte Newtonverfahren konvergiert, sofern $\tilde{J} = F_x(x_{\frac{1}{2}})$ verwendet wird, falls $\|x_{\frac{1}{2}} - x^*\| > \Delta_0$.

Praktisch: Abschätzung der Konvergenzgeschwindigkeit mit Banach'schem Fixpunktsatz, Neuberechnung der Jacobimatrix, falls $\alpha > \alpha_0$ mit einem $\alpha_0 \in (0, 1)$, z.B. $\alpha_0 = 1/4$.

Schätzwert für α :

$$\|x_{k+2} - x_{k+1}\| = \|\tilde{\Phi}(x_{k+1}) - \tilde{\Phi}(x_k)\| \leq \alpha \|x_{k+1} - x_k\|$$

$$\Rightarrow \alpha \approx \alpha_k := \frac{\|x_{k+2} - x_{k+1}\|}{\|x_{k+1} - x_k\|}$$

Abbruch der Iteration, falls $\frac{\alpha_k}{1 - \alpha_k} \|x_{k+1} - x_k\| \leq \text{ATOL} + \text{RTOL} \|x_k\|$ \nearrow vgl. Banach'sches Fixpunktsatz

und $\alpha_k \leq \alpha_0$ mit vorgegebenen Fehlergrenzen ATOL, RTOL für den absoluten und relativen Fehler.

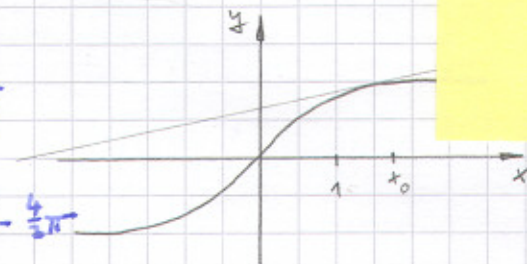
Bemerkung 5.10: Gedämpftes Newtonverfahren.

Arnold

Trotz des sehr guten lokalen Konvergenzverhaltens divergiert das Newtonverfahren häufig, wenn man geeigneten Startwerte x_0 wählt.

Beispiel: $f(x) = \arctan x$, $x_0 = \sqrt{3}$, $\arctan x_0 = \frac{\pi}{3}$

$$x_{k+1} = x_k - \frac{\arctan x_k}{\frac{1}{1+x_k^2}}, \quad x_1 = \sqrt{3} - \frac{4}{3}\pi$$



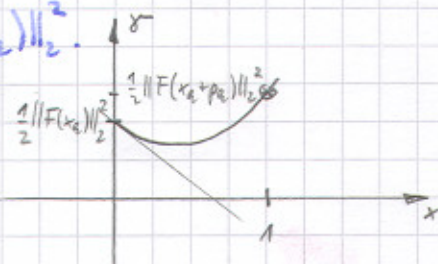
Satz: Modifiziere das Verfahren so, dass $\|F(x_{k+1})\|_2 < \|F(x_k)\|_2$.

Betrachte hierzu $\gamma(\lambda) := \frac{1}{2} \|F(x_k + \lambda p_k)\|_2^2$.

~~Wegen $\gamma'(\lambda) = F_x(x_k + \lambda p_k)^T p_k$ ist~~

$$\gamma'(0) = F_x(x_k)^T p_k = -$$

~~Wegen $\gamma'(\lambda) = \left(F_x^T(x_k) F_x(x_k) \right) p$~~



Wegen $\gamma'(\lambda) = F^T(x_k + \lambda p_k) F_x(x_k + \lambda p_k) p_k$ ist

$$\gamma'(0) = F^T(x_k) \cdot (-F(x_k)) = -\|F(x_k)\|_2^2 < 0, \text{ falls } F(x_k) \neq 0.$$

Für hinreichend kleines $\lambda > 0$ ist also stets $\|F(x_k + \lambda p_k)\|_2 < \|F(x_k)\|_2$.

praktisch:

$$\lambda_k^{(0)} := 1, \quad l := 0$$

while $\|F(x_k + \lambda_k^{(l)} p_k)\|_2 > (1 - \frac{1}{2} \lambda_k^{(l)}) \|p_k\|_2$ do

$$\lambda_k^{(l+1)} := c \cdot \lambda_k^{(l)}, \quad l := l+1$$

$$\lambda_k := \lambda_k^{(l)}, \quad x_{k+1} := x_k + \lambda_k p_k$$

typischer Wert: $c = \frac{1}{2}$

↑ 12.01.06
3'

Ist x^* einfache Nullstelle, so ergibt sich für ein $\delta_0 > 0$

$$\lambda_k = 1, \quad (k \geq k_0),$$

d.h., für große k fallen gedämpftes und klassisches Newtonverfahren

zusammen \Rightarrow beide Konvergenzgeschwindigkeit bei vorhandenem globalem Konvergenz

5.3. Freie Minimierungsprobleme

Amold

Bemerkung 5.11: Problemstellung.

geg.: $f: \mathbb{R}^n \rightarrow \mathbb{R}$, f stetig differenzierbar

ges.: x^* mit $f(x^*) = \min f(x)$

Notwendige Bedingung für lokalen Extremwert:

$$\text{in } \mathbb{R}^1: f'(x^*) = 0$$

$$\text{in } \mathbb{R}^n: \nabla f(x^*) = 0 \text{ mit dem Gradienten } \nabla f(x) := \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$$

Nichtlineares Gleichungssystem $F(x^*) = 0$ mit $F(x) := \nabla f(x)$.

$$\text{Newtonverfahren in } \mathbb{R}^1: x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$\text{in } \mathbb{R}^n: x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

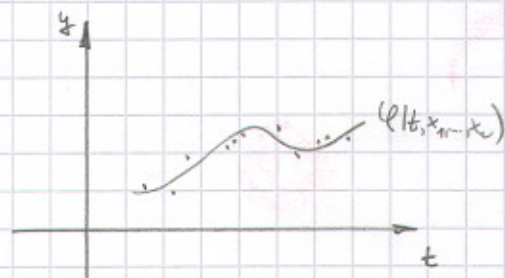
$$\text{mit der Hesse-Matrix } \nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{i,j=1}^n.$$

Bemerkung 5.12: Nichtlineare Ausgleichsprobleme.

→ vgl. Abschnitt 2.2

ziel: zurückführen auf den linearen Fall

geg.: Daten (t_k, y_k) , $(k=1, \dots, m)$, die
über (vermuteten) funktionalen
Zusammenhang $y = \varphi(t, x_1, \dots, x_n)$
gegeben



ges.: Parameter x_1, \dots, x_n , für die $\varphi(t_k, x_1, \dots, x_n)$
„möglichst gut“ y_k entspricht, $(k=1, \dots, m)$

$$\|F(x)\|_2 \rightarrow \min$$

$$\text{mit } F(x_1, \dots, x_n) = \begin{pmatrix} y_1 - \varphi(t_1, x_1, \dots, x_n) \\ \vdots \\ y_m - \varphi(t_m, x_1, \dots, x_n) \end{pmatrix}$$

Gauß-Newton-Verfahren

Bestimme x_{opt} als Lösung von $\|F(x_k) + F_x(x_k)(x - x_k)\|_2 \rightarrow \min$
 mit den Methoden aus Bemerkung 2.9 oder Bemerkung 2.11c)
 angewandt auf $\|Ax - b\|_2 \rightarrow \min$ mit $A := F_x(x_k)$, $b := F_x(x_k)x_k - F(x_k)$.

Schnelle lokale Konvergenz, falls $\min \|F'(x)\|_2 \ll 1$. ↗ praktisch interessantes Fall

praktisch: gedämpftes Gauß-Newton-Verfahren

↑ 14.1.04

5.4. Klassische Iterationsverfahren für große lineare Gleichungssysteme

Bemerkung 5.13: Lineare Gleichungssysteme mit schwach besetzter Koeffizientenmatrix.

Direkte Lösungsverfahren

Klassisches Gauß-Algorithmus: $O(\frac{n^3}{3})$ Rechenoperationen

Für große n , ($n > 10^4$) ist die Annahme der (in der Regel vorliegenden)

speziellen Struktur der Koeffizientenmatrix zwingend erforderlich. ↗ ausführlich, mit Matlab-Beispiel

Beispiel: ~~5-Stone~~ 5-Punkte-Stein-Differenzapproximation von $-u(x,y) = f(x,y)$

Strategie 1: Annahme des Blockstrukturbruchs in direkten Lösungsverfahren für ↑ 18.01.06

schwach besetzte Matrizen. ↗ Beispiel: Bandstruktur, auch: graphentheoretische Hilfsmittel zur optimalen Sortierung von Zeilen und Spalten

Strategie 2: Nähungsweise Lösung des Gleichungssystems mittels iterativer Verfahren.

↗ dies Abbruchzeit

Strategie 3: Partielles Hintergrund definiert nicht lineares Gleichungssystem,

Sondern eine Klasse von Gleichungssystemen \rightarrow Annahme in der numerischen Lösung, z.B. Mehrgitterverfahren. ↗ vgl. numerische Lösung partieller Differentialgleichung

Bemerkung 5.14: Jacobi- oder Gauß-Seidelverfahren.

geg.: $A = (a_{ij}) \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$

ge.: (klassische) Lösung x von $Ax = b$

Komponentenweise Darstellung in Fixpunktform:

Arnold

$$a_{ii} x_i = b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j, \quad (i=1, \dots, n)$$

Algorithmus:

Initialisierung: Anfangsnäherung $x^{(0)}$, $k:=0$

repeat

for $i=1:n$

$$x_i^{(k+1)} := \frac{b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j^{(k)}}{a_{ii}}$$

↑ parallelisierbar

until „Konvergenz“

↑ Aufwand pro Iterationsschritt (Beste-Praxis-Strategie!)

↑ an obiger Fixpunktformulierung erläutern

Matrix Umformung: Sei $D := \text{diag } a_{ii} \Rightarrow x^{(k+1)} = D^{-1}(b - (A-D)x^{(k)})$

$$x^{(k+1)} = \Phi(x^{(k)}) \quad \text{mit} \quad \Phi(x) = -D^{-1}(A-D)x + D^{-1}b, \\ \|\Phi_x\| = \|D^{-1}(A-D)\|$$

Gibt es eine Matrixnorm $\|\cdot\|$ mit $\alpha := \|D^{-1}(A-D)\| < 1$, so ist das Verfahren linear konvergent. Konvergenztest:

Matrix- und Vektornorm müssen verträglich sein

$$\frac{\alpha_k}{1-\alpha_k} \|x^{(k+1)} - x^{(k)}\| \leq \text{ATOL} + \text{RTOL} \|x^{(k)}\| \quad \text{vgl. Bem. 5.9}$$

mit $\alpha_k := \frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)} - x^{(k-1)}\|}$. Beispiel Modellproblem: $\alpha = 1 - \pi^2 h^2 + O(h^4)$

Hinreichend für Konvergenz: A ^{stabil} diagonaldominant, d.h. $\sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| < |a_{ii}|$, $(i=1, \dots, n)$

$$\Rightarrow \|D^{-1}(A-D)\|_{\infty} < 1.$$

Anfangsnäherung: z.B. $x^{(0)} := 0 \Rightarrow x_i^{(1)} = \frac{b_i}{a_{ii}}, (i=1, \dots, n)$

Speicherbedarf: n Werte der Länge n

$x_i^{(k+1)}$ wird auf dem Platz von $x_i^{(k)}$ abgespeichert.

Bemerkung 5.15: Gauß-Seidel- oder Einzelschrittverfahren, überrelaxation. Abund

Ersetze im Jacobi-Verfahren $x_i^{(k+1)} = \dots$ durch

$$x_i^{(k+1)} := \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)}}{a_{ii}}, \quad (i=1, \dots, n)$$

↑ auch: Einzelschritt
Gauß-Seidel-Verfahren

pro Iterationsschritt gleiches Aufwand wie im Jacobi-Verfahren,

aber i. Allg. doppelt so schnelle Konvergenz, jedoch: nicht parallelisierbar.

Beispiel Modellproblem: $\alpha = 1 - 2\pi^2 h^2 + O(h^4)$

↑ 28.04.06

Überrelaxation (Successive overrelaxation, SOR)

$$x_i^{(k+1)} := x_i^{(k)} + \omega \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)}}{a_{ii}}, \quad (i=1, \dots, n)$$

mit geeignetem Überrelaxationsparameter ω . Meist: $\omega \in (1, 2)$.

Für optimales ω & Konvergenz i. Allg. um eine Größenordnung schneller als beim Gauß-Seidel-Verfahren. \nearrow Stör/Bubisch, Kapitel 8

Beispiel Modellproblem: $\omega_{\text{opt}} = 2 - 2\pi h + O(h^2)$, $\alpha(\omega_{\text{opt}}) = 1 - 2\pi h + O(h^2)$ ↑ 21.11.

Bemerkung 5.16: CG-Verfahren.

geg.: $A = (a_{ij}) \in \mathbb{R}^{n \times n}$, A symmetrisch positiv definit, $b \in \mathbb{R}^n$

\nearrow Beispiel: Modellproblem

ges.: Lösung x von $Ax = b$

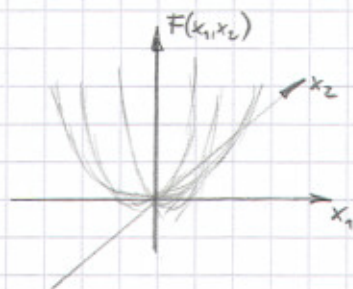
Analyse zu Minimierungsproblemen: $Ax = b \Leftrightarrow F(x) = \min_{\xi \in \mathbb{R}^n} F(\xi)$

$$\text{mit } F(\xi) := \frac{1}{2} (A\xi - b)^T (A\xi - b),$$

da mit A ist auch A^T symmetrisch positiv definit, also $F(\xi) = 0 \Leftrightarrow A\xi - b = 0$.

Lösungsansatz: Minimierung von F mittels eines iterativen Verfahrens, das pro Iterationsschritt nur geringen Aufwand erfordert

$n=2$



\nearrow nicht: Newton-Verfahren (würde in 1 Iterationsschritt konvergieren, erfordert aber die dritte Lösung von $A\xi = b$)

Idee: Verfahren des steilsten Abstiegs.

Wunder

geg.: $x^{(k)}$ mit $0 \neq \nabla F(x^{(k)})$ \nearrow sonst: notwendige Bedingung für lokales Extremum erfüllt

ges.: Richtung $d^{(k)}$ mit $F(x^{(k)} + s d^{(k)}) \leq F(x^{(k)} + s d)$, $(s \rightarrow 0)$
für jede Richtung $d \in \mathbb{R}^n$ mit $\|d\|_2 = \|\nabla F(x^{(k)})\|_2$

$$\begin{aligned} F(x^{(k)} + s \cdot d) &= F(x^{(k)}) + s \nabla F(x^{(k)})^T d + \mathcal{O}(s^2) \\ &\stackrel{\text{Schwarz'sche Ungleichung, Gleichheit für } d = -\nabla F(x^{(k)})}{\geq} F(x^{(k)}) - s \|\nabla F(x^{(k)})\|_2 \|d\|_2 + \mathcal{O}(s^2) \\ &= F(x^{(k)}) - s \|\nabla F(x^{(k)})\|_2^2 + \mathcal{O}(s^2) \end{aligned}$$

$$\Rightarrow \underline{d^{(k)} = -\nabla F(x^{(k)}) = r_k := b - Ax^{(k)}} \quad \dots \text{Gradientenverfahren}$$

\nearrow Wahl des optimalen Schrittweite $s^{(k)}$: Minimierung einer skalaren quadratischen Funktion

EinfaCh zu implementieren, aber oft nur langsame Konvergenz.

\nearrow Nachteil: Schritt $x^{(k)} \rightarrow x^{(k+1)}$ berücksichtigt nur Information in $x^{(k)}$, aber nicht in $x^{(l)}$ mit $l < k$.

Alternative: $x^{(k+1)} = x^{(k)} + s_0 r_0 + s_1 r_1 + \dots + s_k r_k$

mit geeigneten $s_i \in \mathbb{R}$ und $r_i := b - Ax^{(i)}$, $(i = 0, 1, \dots, k)$

Bestimmung der s_0, s_1, \dots, s_k so, dass $F(x^{(k+1)}) \rightarrow \min$.

Ergebnis: Verfahren der konjugierten Gradienten (engl.: conjugate gradient, CG-Verfahren, Hestenes und Steiha, 1972)

\nearrow 26.02.05

Algorithmus: Initialisierung: Wähle Startwert $x^{(0)} \in \mathbb{R}^n$, $k := 0$,

$$p_0 := r_0 := b - Ax^{(0)}$$

~~Repeat~~ while $p_k \neq 0$

$$a_k := \frac{r_k^T r_k}{p_k^T A p_k}, \quad x^{(k+1)} := x^{(k)} + a_k p_k,$$

$$r_{k+1} := r_k - a_k A p_k, \quad b_k := \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

$$p_{k+1} := r_{k+1} + b_k p_k$$

$\nearrow \sum_{i=0}^k s_i r_i$

Aufwand pro Iterationsschritt: 1 Matrix-Vektor-Produkt Ax_k ,
2 Skalarprodukte, 3 Additionen. Skalares Vielfaches von Vektoren.

Wunder

→ trotz komplizierteres Struktur der Householdermatrix Aufwand pro Schritt vergleichbar mit klassische Iterationsverfahren

Konvergenz: (i) Bezüglich des durch A induzierten Skalarprodukts

$\langle \xi, \eta \rangle_A := \xi^T A \eta$ sind die Vektoren p_k orthogonal

(„ A -~~konjugiert~~ konjugiert“ \leftrightarrow „konjugierte Gradienten“) → Gauss/QR-Verfahren

(ii) Konvergenz nach höchstens n Schritten → höchstens n A -konjugierte Vektoren p_k (exakte Mathematik)

→ praktisch interessant: Konvergenz nach $k \ll n$ Schritten

$$(iii) \frac{\|x^{(k)} - x\|_A}{\|x^{(0)} - x\|_A} \leq 2 \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k$$

→ Konvergenz umso schneller,
je kleiner $\kappa_2(A)$, aber:
Modellprobleme $\kappa_2(A) = \mathcal{O}(N^2)$