Einführung in die Numerische Mathematik

Roland Pulch, Michael Günther

Skript zur Vorlesung im Sommersemester 2009
korrigierte Version März 2011

Bergische Universität Wuppertal
Lehrstuhl für Angewandte Mathematik und Numerische Mathematik

Inhalt:

Literatur:

Stoer, J.: Numerische Mathematik 1, Springer Verlag.
Deuflhard, P.; Hohmann, A.: Numerische Mathematik I, de Gruyter Verlag.
Quarteroni, A.; Sacco, R.; Saleri, F.: Numerical Mathematics, Springer Verlag.

Inhaltsverzeichnis

Kapitel 1

Numerische Mathematik – Was ist das?

Die *Numerische Mathematik* beschäftigt sich mit Entwicklung und Analyse von Algorithmen/Rechenmethoden zur zahlenmäßigen Lösung mathematischer Probleme.

In der Analysis und der Linearen Algebra zeigt man die Existenz und Eindeutigkeit von Lösungen zu mathematischen Problemen.

Um diese Charakterisierung mit Leben zu erfüllen, betrachten wir zwei Beispiele: Lineare Gleichungssysteme und bestimmte Integrale.

I) *Lineare Gleichungssysteme*
Wir betrachten das Gleichungssystem

$$Ax = b,$$

wobei $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$ gegeben sind und die Lösung $x \in \mathbb{R}^n$ gesucht wird. In der linearen Algebra zeigt man: Das System besitzt genau dann eine eindeutige Lösung, wenn für die Determinante $\det(A) \neq 0$ gilt. Die Cramersche Regel liefert eine Formel für die Lösung. Diese Formel ist für die Praxis ungeeignet. Die zahlenmäßige Bestimmung der Lösung kann geeignet mit dem Gauß-Algorithmus erfolgen.

II) *Bestimmte Integrale*
Gegeben sei eine Funktion $f : [0, 1] \to \mathbb{R}$ und gesucht wird das Integral

$$I(f) := \int_0^1 f(x)\,\mathrm{d}x,$$

d.h. die reelle Zahl $I(f)$ ist (näherungsweise) zu bestimmen. In der Analysis zeigt man: Ist die Funktion f stetig, dann existiert das Integral und ist eindeutig. Bei vielen Funktionen f existiert keine explizite Formel für die zugehörige Stammfunktion.

Daher wird das Integral zahlenmäßig näherungsweise bestimmt durch z.B. Rechteck-Summen, Trapez-Summen u.a.

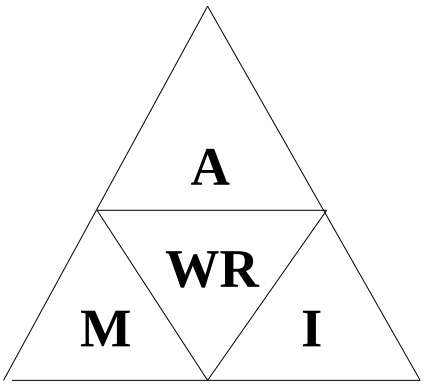
Fazit: Der Algorithmus/Rechenmethode (Rechteck-Summen, Gauß-Elimination, etc.) liefert *Zahlen* als Ergebnis eines konkreten mathematischen Problems.

Beim Einsatz eines Computers ist allerdings zu beachten, dass nur endlich viele Stellen zur Durchführung der arithmetischen Operationen vorhanden sind. Bei jedem Rechenschritt können sogenannte *Rundungsfehler* auftreten und die Lösung verfälschen.

Einordnung der Numerischen Mathematik

Nach dem bisher gesagten ist klar, dass die Numerik zur Angewandten Mathematik zählt. In den letzten Jahren hat ihre Bedeutung noch zugenommen, da eine neue Disziplin, das *Wissenschaftliche Rechnen*, aus ihr hervorgegangen ist.

Das Wissenschaftliche Rechnen (WR) oder *Scientific Computing* begreift sich als interdisziplinäre Methode, um Aufgaben aus Wissenschaft und Technik mit dem Computer zu lösen. Im Diagramm:



A: Anwendung

M: Mathematik

I: Informatik

Abbildung 1: Wissenschaftliches Rechnen im Spannungsfeld von Anwendung, Mathematik und Informatik.

Zentral beim WR ist der Begriff der *Simulation.*

Definition 1.1: *VDI-Norm 3633, Simulation*
Simulation ist das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.

Im Kontext des WR bedeutet Simulation das Experiment auf dem Rechner.

Reale Experimente sind oft aus physikalischen oder ökonomischen Gründen nicht möglich. Darüberhinaus spart die Simulation Zeit- und Entwicklungskosten.

Gegeben sei ein Anwendung aus den Naturwissenschaften oder der Technik. Typischerweise geht man im WR nach den folgenden Schritten vor:

1. (Mathematische) Modellbildung
   → Erzeugen von mathematische Gleichungen
2. (Numerische) Simulation
   → Benutzung von Algorithmen auf Rechner
3. Auswertung/Post-Processing (der Ergebnisse)
   z.B. graphische Ausgabe, Vergleich untereinander, etc.

Die einzelnen Schritte zeigen, wie Anwendung und Mathematik miteinander verzahnt sind. Die Informatik leistet ebenfalls einen wichtigen Beitrag durch schnelle Hardware und neue Rechnertechnologien. Um die Bedeutung der Mathematik hervorzuheben, verwendet man auch gerne das Schlagwort

High Tech = Math Tech

und bezeichnet die Mathematik als Schlüsseltechnologie (key technology).

Inzwischen gibt es die ersten Studiengänge WR in Deutschland. An der Bergischen Universität Wuppertal existiert der Master-Studiengang „Computer Simulation in Science“.

Rechnerentwicklung

Numerik wie auch WR profitierten und profitieren sehr stark von der rasanten Entwicklung der Rechnertechnologie. Immer leistungsfähigere Computer ermöglichen inzwischen die Simulation komplexer Zusammenhänge in mannigfachen Anwendungen.

ABER: Erst ein effizienter und präziser Algorithmus liefert die Lösung eines Problems – die numerische Software ist noch wichtiger als eine schnelle Hardware!

Historischer Abriss:

1936: Der Bauingenieur Konrad Zuse beginnt mit dem Bau eines noch rein mechanisch arbeitenden, durch Lochstreifen gesteuerten Rechenautomaten, der Z1.

1941: Ein weiterer, in Relaistechnik gebauter Rechenautomat wurde im Mai 1941 fertiggestellt. Die Z3 war der erste funktionsfähige, frei programmierbare, programmgesteuerte Rechenautomat. In den USA fanden gleichzeitig mehrere ähnliche Entwicklungen statt wie z.B. der von Howard H. Aiken ab 1939 bei IBM gebaute programmgesteuerte, automatische Relaisrechner Mark I.

1946: Entscheidend für die Entwicklung des Universalrechners wurde der an der Moore School of Electrical Engeneering der Universität Pennsylvania von John Mauchly und John Presper Eckert entwickelte Röhrenrechner ENIAC (150 kW Stromverbrauch, 80 qm Aufstellfläche). Noch während des Baus wurde der Mathematiker John von Neumann auf den ENIAC aufmerksam. Er entwickelt die Idee einer speicherprogrammierten universellen Maschine. Das Konzept des Computers ist geboren. Der von ihm erfundene logische Aufbau der Maschine wird heute als von-Neumann-Architektur bezeichnet.

1948: Frederic C. Williams bringt an der Universität Manchester (England) den Prototypen des Manchester Mark I zum Laufen.

Ende der 50er Jahre: Weltweit sind etwa 8000 Computersysteme installiert, Hauptanwender sind Forschungseinrichtungen, Behörden, zunehmend auch Banken (IBM).

1964: DEC bringt den ersten Minicomputer, die PDP-8, das "Model-T" der Computerindustrie, heraus.

1969: INTEL entwickelt den ersten Mikroprozessor.

70er Jahre: Der Computer wird in Wirtschaft und Beruf immer wichtiger.

1981: IBM stellt den ersten Personal Computer (PC) vor.

1984: Apple Macintosh – Fenster und Maus als neue Bedienungselemente. IBM, Apple und weitere Hersteller bringen den Computer auf jeden Schreibtisch.

90er Jahre : Internet und Multimedia eröffnen neue Dimensionen.

Die Hardware moderner Hochleistungsrechner bietet inzwischen ungeahnte Möglichkeiten.

Eine der arithmetischen Grundoperationen (+,-,*,/) bezeichnet man als Operation mit Gleitkommazahl oder floating point operation (FLOP). Die Rechengeschwindigkeit wird gemessen in FLOPs pro Sekunde. Normale PCs besitzen Leistungen von über 6 GFLOPs/s (Giga-FLOPs/s). Hochleistungsrechner (Supercomputer) erreichen heute Leisungen von 1000 TFLOPs/s (Tera-FLOPs/s). Ein Geschwindigkeitsvergleich ist in Abbildung 2 enthalten.

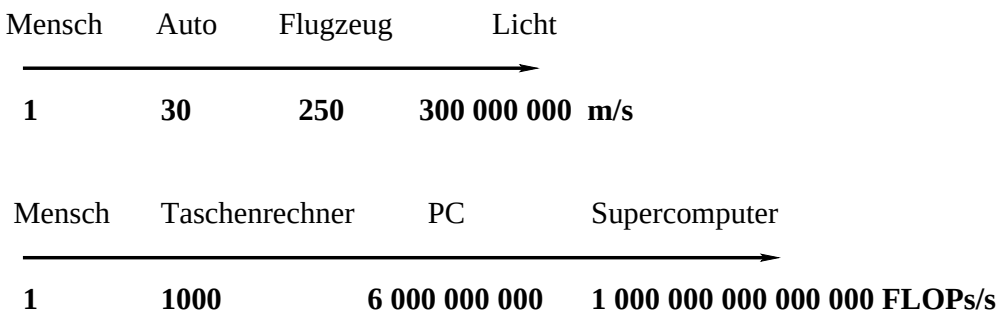


Abbildung 2: Beschleunigungsfaktoren bei physikalischer Geschwindingkeit und Rechnerleistung, FLOPs/s = Floating Point Operations Per Second.

Weitere Informationen im Internet:

Anwendungen der Mathematik / Wissenschaftliches Rechnen:

`www.mathematik-21.de/index.shtml`

Arbeitsgemeinschaft Simulation der Gesellschaft für Informatik:

`www.asim-gi.org`

Supercomputer - aktuelle weltweite Rangliste
(neue Liste im November 2008!):

`www.top500.org`

Geschichte des Computers: Heinz-Nixdorf-Museum Paderborn

`www.hnf.de`

Beispiel: Integral-Rekursion

Bei elementaren Rechenoperationen entstehen auf dem Computer Rechenfehler bzw. Rundungsfehler. Beispielsweise gilt die Gleichheit $\frac{1}{3^5} = \frac{1}{243}$. Auf einem Rechner (unter Software MATLAB 7.5.0) erhalten wir jedoch:

$$\begin{aligned} (1/3)*(1/3)*(1/3)*(1/3)*(1/3) &= 4.115226337448559e-03 \\ 1/243 &= 4.115226337448560e-03 \end{aligned}$$

Wir bemerken, dass die letzte Nachkommastelle unterschiedlich ist. Mindestens eines der beiden Ergebnisse ist somit (leicht) falsch. Ursache hierfür sind die Rundungsfehler auf einem Computer/Rechner. Der Fehler hier ist unbedeutend, da wir im allgemeinen nur an den führenden (3-4) Stellen in den Anwendungen interessiert sind, welche hier übereinstimmen. Es gibt allerdings bereits einfache Fälle, wo elementare Rechenoperationen schon nach wenigen Schritten zu vollkommen falschen Ergebnissen führen – wie das nächste Beispiel zeigt.

Zu berechnen seien die bestimmten Integrale

$$I_n = \frac{1}{\mathrm{e}} \int_0^1 x^n \mathrm{e}^x \,\mathrm{d}x \qquad \text{für } n = 0, 1, 2, 3, \ldots .$$

Elementare Integration versagt hier. Stattdessen kann man die Aufgabe durch partielle Integration umformen:

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

Ergebnis: *Zwei-Term-Rekursion*

$$I_n = 1 - n I_{n-1} \tag{1.1}$$

Ist I_0 gegeben, so lassen sich aus (1.1) beliebige Integrale I_n, $n > 0$, bestimmen.

Der Startwert

$$I_0 = \frac{1}{\mathrm{e}} \int_0^1 \mathrm{e}^x \,\mathrm{d}x = \frac{\mathrm{e} - 1}{\mathrm{e}} \doteq 0.632120558\ldots \tag{1.2}$$

ist eine transzendente Zahl, d.h. insbesondere gilt $I_0 \notin \mathbb{Q}$.

Die *Vorwärtsrekursion* besitzt daher das Schema

$$I_0 \to I_1 \to I_2 \to \cdots \to I_n \to \cdots ,$$

welches beliebig weit durchgeführt werden kann.

Um diese Resultate zu überprüfen, wird das Integral abgeschätzt:

$$I_n = \frac{1}{\mathrm{e}} \int_0^1 x^n \mathrm{e}^x \, \mathrm{d}x \geq 0$$

wegen $x^n \mathrm{e}^x > 0$ für alle $x \in (0,1)$ und

$$I_n = \frac{1}{\mathrm{e}} \int_0^1 x^n \mathrm{e}^x \, \mathrm{d}x \leq \frac{1}{\mathrm{e}} \mathrm{e} \int_0^1 x^n \, \mathrm{d}x = \frac{x^{n+1}}{n+1}\Big|_0^1 = \frac{1}{n+1} \, .$$

durch $\mathrm{e}^x \leq \mathrm{e}$ für alle $x \in [0,1]$.

Damit folgt die Einschachtelung

$$0 \leq I_n \leq \frac{1}{n+1} \qquad \text{für alle } n \geq 0 \, .$$

Insbesondere gilt $0 \leq I_n \leq 1$ für jedes $n \in \mathbb{N}$.

Die Vorwärtsrekursion liefert auf dem Rechner die in Tabelle 1 angegebenen Werte. Wir erkennen, das der Wert I_{18} und alle folgenden Werte deutlich falsch sind. Der Fehler wächst dabei mit alternierendem Vorzeichen extrem an.

Zur Erklärung: Es seien I_n die exakten Werte und $\tilde{I}_n$ die auf einem Rechner erhaltenen Ergebnisse. Der Fehler ist daher $\Delta I_n = \tilde{I}_n - I_n$. Für den Startwert (1.2) gilt

$$\Delta I_0 = \tilde{I}_0 - I_0 \neq 0,$$

da die transzendente Zahl I_0 nicht exakt auf dem Rechner dargestellt werden kann. Jedoch ist $|\Delta I_0| \approx 10^{-16}$, d.h. sehr klein. Wir erhalten eine Rekursion für den Fehler:

$$\begin{array}{rcl} I_n &=& 1 - nI_{n-1} \\ \tilde{I}_n &=& 1 - n\tilde{I}_{n-1} \\ \hline \tilde{I}_n - I_n &=& -n(\tilde{I}_{n-1} - I_{n-1}) \\ \Delta I_n &=& -n\Delta I_{n-1} \end{array}$$

Sukzessives Einsetzen liefert

$$\Delta I_n = -n\Delta I_{n-1} = -n(-(n-1))\Delta I_{n-2} = -n(-(n-1))(-(n-2))\Delta I_{n-3} = \cdots \, .$$

Somit folgt die Fehlerformel

$$\Delta I_n = (-1)^n n! \Delta I_0, \tag{1.3}$$

welche das Verhalten der Vorwärtsrekursion erklärt. Trotz kleinem ΔI_0 wächst der Fehler wegen dem Term $n!$ extrem an. Der Vorfaktor $(-1)^n$ erklärt das alternierende Vorzeichen in den Ergebnissen.

Tabelle 1: Zwei-Term-Rekusion zu Integralen.

Vorwärtsrekursion:		Rückwärtsrekursion:	
n	I_n	n	I_n
0	6.321205588285577e-01	0	6.321205588285577e-01
1	3.678794411714423e-01	1	3.678794411714423e-01
2	2.642411176571153e-01	2	2.642411176571153e-01
3	2.072766470286540e-01	3	2.072766470286539e-01
4	1.708934118853840e-01	4	1.708934118853843e-01
5	1.455329405730801e-01	5	1.455329405730786e-01
6	1.268023565615195e-01	6	1.268023565615284e-01
7	1.123835040693635e-01	7	1.123835040693008e-01
8	1.009319674450921e-01	8	1.009319674455933e-01
9	9.161229299417073e-02	9	9.161229298966059e-02
10	8.387707005829270e-02	10	8.387707010339417e-02
11	7.735222935878028e-02	11	7.735222886266420e-02
12	7.177324769463667e-02	12	7.177325364802957e-02
13	6.694777996972334e-02	13	6.694770257561571e-02
14	6.273108042387321e-02	14	6.273216394138015e-02
15	5.903379364190187e-02	15	5.901754087929777e-02
16	5.545930172957014e-02	16	5.571934593123560e-02
17	5.719187059730757e-02	17	5.277111916899477e-02
18	-2.945367075153627e-02	18	5.011985495809427e-02
19	1.559619744279189e+00	19	4.772275579620888e-02
20	-3.019239488558378e+01	20	4.554488407582235e-02
21	6.350402925972594e+02	21	4.355743440773061e-02
22	-1.396988643713971e+04	22	4.173644302992650e-02
23	3.213083880542133e+05	23	4.006181031169058e-02
24	-7.711400313301118e+06	24	3.851655251942608e-02
25	1.927850088325280e+08	25	3.708618701434793e-02
26	-5.012410228645727e+09	26	3.575913762695372e-02
27	1.353350761744346e+11	27	3.450328407224959e-02
28	-3.789382132883170e+12	28	3.390804597701149e-02
29	1.098920818536129e+14	29	1.666666666666667e-02
30	-3.296762455608386e+15	30	5.000000000000000e-01

Statt der *Vorwärtsrekursion* $I_n = 1 - nI_{n-1}$ kann man eine *Rückwärtsrekursion*

$$I_{n-1} = \frac{1 - I_n}{n}$$

versuchen, wobei man sukzessive bestimmt

$$I_n \to I_{n-1} \to I_{n-2} \to \cdots \to I_1 \to I_0.$$

Ein korrekter Startwert ist dabei unbekannt. Daher setzen wir einfach $I_n = 0.5$ als Startwert, d.h. absichtlich einen falschen Wert. Jedoch gilt dann mit den obigen Abschätzungen zumindest $|\Delta I_n| < 0.5$.

Die Rückwärtrekursion liefert auf dem Rechner die in Tabelle 1 aufgelistete Folge für den Startwert I_{30}. Das Ergebnis für I_0 ist nun richtig auf allen Stellen!

Auch hier liefert die Fehlerformel (1.3) eine Erklärung, denn es folgt direkt

$$\Delta I_0 = \frac{1}{(-1)^n n!}\, \Delta I_n.$$

Somit wird der bezüglich I_n gemachte Fehler durch den Faktor $1/n!$ erheblich verkleinert. Dieses Prinzip kann auch genutzt werden, um I_m für ein $m > 0$ in einer Rückwärtrekursion aus einem I_n mit $n > m$ genauer zu erhalten.

In diesem Beispiel bewirken Rundungsfehler und eine fehlerverstärkende (*instabile*) Vorwärtsrekursion total verfälschte Ergebnisse. Interessanterweise werden die Fehler bei der (*stabilen*) Rückwärtsrekursion dagegen herausgehoben.

Nach den einführenden Beispielen wollen wir die charakteristischen Eigenschaften der Numerischen Mathematik zusammenfassen.

Kennzeichen der Numerischen Mathematik:

Konstruktiv: Nicht nur Existenz/Eindeutigkeit einer Lösung sind von Bedeutung, sondern eine konkrete Lösung (Zahl) wird beschafft.

Approximativ: Näherungen für die exakte Lösung werden als gleichwertig akzeptiert, wenn deren Fehler beliebig klein gemacht werden können und verlässliche Fehlerschranken vorliegen.

Anwendungsorientiert: Es werden Probleme gelöst, die aus Anwendungen (Wirtschaft/Technik/Naturwissenschaften/...) stammen, und die entwickelten Methoden werden als Software bereitgestellt.

Als Hilfsmittel dienen nur die vier Grundoperationen $+, -, \cdot, /$ und die sechs arithmetischen Vergleiche, wahlweise auch $\sqrt{\ }$. Die arithmetischen Operationen werden nur mit einer endlichen Stellenzahl ausgeführt. Rundungsfehler sind zugelassen, und ihr Einfluss ist zu untersuchen und abzuschätzen.

Kapitel 2

Rechnerarithmetik und Fehleranalyse

In diesem Kapitel geht es zunächst um die Zahldarstellung auf dem Rechner und die zugehörigen arithmetischen Operationen. Rundungsfehler und der Begriff der Kondition eines Problems werden danach behandelt. Zur Motivation betrachte man nochmals das Beispiel der Integralrekursion aus dem vorhergehenden Kapitel.

2.1 Gleitpunkt- und Maschinenzahlen, Rundung

Das System $\mathbb{R}$ der reellen Zahlen ist lückenlos: Zu jeder reellen Zahl gibt es noch größere und noch kleinere Zahlen, und zu jedem Paar von zwei verschiedenen Zahlen gibt es weitere Zahlen, die dazwischen liegen.

Die in einer Maschine exakt darstellbaren Zahlen sind dagegen finit, also beschränkt und diskret. Man unterscheidet zwischen Gleitpunktzahlen $\mathbb{G}$ und Maschinenzahlen $\mathbb{M}$.

Definition 2.1: Gleitpunktzahlen (*floating point numbers*)

Die normalisierten t-stelligen Gleitpunktzahlen zur Basis B sind die Menge

$$\mathbb{G} = \left\{ g = M \cdot B^E : \ M = 0 \ \textit{ oder } \ B^{t-1} \leq |M| < B^t \right\}$$

mit dem Exponenten E und der Mantisse M. $B > 1$ und $t > 0$ sind vorgegebene natürliche Zahlen, M und E sind ganze Zahlen.

Man fordert die Bedingung $B^{t-1} \leq |M| < B^t$, um die Darstellung zu normalisieren. Auf diese Weise kann keine führende 0 auftreten. Die Zuordnung $g \mapsto M, E$ ist für $g \neq 0$ eineindeutig.

Beispiel 2.1: 4-stellige, normalisierte Dezimalzahlen

$$\mathbb{G} = \left\{ M \cdot 10^E : \ M = 0 \ \text{ oder } \ 10^3 \leq |M| < 10^4 \right\}$$

Die Mantisse liegt damit zwischen $1000 \leq |M| \leq 9999$. Beispiele für Zahlen in dieser Darstellung sind:

$$19.25 = 1925 \cdot 10^{-2}, \qquad 0.004589 = 4589 \cdot 10^{-6}, \qquad 4.01 = 4010 \cdot 10^{-3}, \quad \ldots$$

Beispiel 2.2: 7-stellige, normalisierte Dualzahlen

$$\mathbb{G} = \left\{ M \cdot 2^E : \ M = 0 \ \text{ oder } \ 2^6 \leq |M| < 2^7 \right\}$$

Umrechnung von 19.25 ins Dualsystem:

$$19.25 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = L00LL.0L = L00LL0L \cdot 2^{-L0}$$

Maschinenzahlen bilden eine Teilmenge der Gleitpunktzahlen. Bei ihnen ist der Exponentenbereich eingeschränkt.

Definition 2.2: Maschinenzahlen

$$\mathbb{M} = \left\{ g = M \cdot B^E \in \mathbb{G} : \ \alpha \leq E \leq \beta \right\}$$

Das System der Maschinenzahlen ist damit finit. Die Größen B, t, α, β sind durch die Implementierung festgelegt und werden nicht gespeichert. Mantisse M und Exponent E legen die Maschinenzahl eindeutig fest. Die Abhängigkeit der Mengen von den Parametern kann man schreiben als

$$\mathbb{M}(B, t, \alpha, \beta) \subset \mathbb{G}(B, t).$$

Im allgemeinen System der Maschinenzahlen führt man als charakteristische Parameter ein

$$\begin{aligned} \sigma &:= B^{t-1} \cdot B^{\alpha} \quad \text{kleinste positive Maschinenzahl,} \\ \lambda &:= (B^t - 1) \cdot B^{\beta} \quad \text{größte Maschinenzahl.} \end{aligned}$$

Abbildung 3: Dichte der Maschinenzahlen in normalisierten Darstellung.

Man spricht von *Bereichsüberschreitung* oder *Exponentenunter/-überlauf*, wenn sich im Lauf einer Rechnung Werte ergeben, die außerhalb des Bereichs $[-\lambda, -\sigma] \cup \{0\} \cup [+\sigma, +\lambda]$ liegen.

Die Maschinenzahlen sind im übrigen nicht äquidistant verteilt. Um die null herum klafft eine riesige Lücke, und nach außen hin wird die Unterteilung immer gröber.

Etwas formaler: Zwei aufeinander folgende positive Gleitpunktzahlen haben die Werte $g = M \cdot B^E$ und $g' = (M+1) \cdot B^E$. Der relative Abstand ist also $(g' - g)/g = 1/M$. Gleiches gilt für zwei negative Nachbarn. Der größte relative Abstand ist $\rho := \max(1/M) = 1/B^{t-1}$, diese Größe bezeichnet man als die Auflösung der Arithmetik (*resolution*).

Wir fassen die Eigenschaften der Maschinenzahlen im Gegensatz zu den reellen Zahlen zusammen:

- Maschinenzahlen stellen eine endliche Menge da.
- Maschinenzahlen liegen nicht äquidistant.
- Es existiert eine große Lücke um die null.

IEEE-Standard 754

Die Kodierung von M und E erfolgt nach gewissen Standards. Seit 1984 gibt es den IEEE-Standard 754 (IEEE: Institute of Electrical and Electronics Engineers), der die Zahldarstellung auf handelsüblichen Mikroprozessoren (Bsp. Pentium) reglementiert. Basis ist $B = 2$ (Dualsystem).

Zur Verfügung stehende Maschinenzahlen:

Zahlentyp	Bits VZ	Exponent	Mantisse
4 byte - short real (single precision)	1	8	23
8 byte - long real (double precision)	1	11	52
10 byte - temp real (extended precision)	1	15	64

Kennzeichen der jeweiligen Arithmetik:

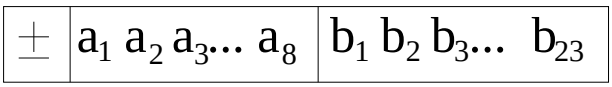
Zahlentyp	σ	λ	ρ
short real	$1.2 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$	$1.2 \cdot 10^{-7}$
long real	$2.2 \cdot 10^{-308}$	$1.8 \cdot 10^{308}$	$2.2 \cdot 10^{-16}$
temp real	$3.4 \cdot 10^{-4932}$	$1.2 \cdot 10^{4932}$	$1.1 \cdot 10^{-19}$

Hidden bit: Normalisierte Dualzahlen besitzen immer L als führende Ziffer, die daher nicht abgespeichert werden muss.

Beispiel 2.3: Short Real

1 bit für Vorzeichen, 1 byte = 8 bit für den Exponenten, 3 byte = 1+23 bit für die Mantisse ($t = 24$ wegen hidden bit).

Schematische Abspeicherung:



Sonderoperationen im IEEE-Standard:

$+\infty$	größer als jede reelle Zahl
$-\infty$	kleiner als jede reelle Zahl
`quiet NaN (Not a Number)`	unbestimmter Wert, wird vererbt
`signalizing NaN`	löst als Operand einen Alarm aus

Rundung

Beim Runden geht man vom Kontinuum $\mathbb{R}$ zur diskreten Menge der Maschinenzahlen $\mathbb{M}$ bzw. Gleitpunktzahlen $\mathbb{G}$ über.

Definition 2.3: Rundung

Eine korrekte Rundung ist eine Abbildung $\mathrm{rd} : \mathbb{R} \to \mathbb{G}$*, die jeder reellen Zahl* x *eine nächstgelegene Gleitpunktzahl* $\mathrm{rd}(x)$ *zuordnet, d.h.*

$$|\mathrm{rd}(x) - x| \leq |g - x| \quad \textit{für alle } g \in \mathbb{G}\,.$$

Die Rundung ist eine surjektive, idempotente und monotone Abbildung.

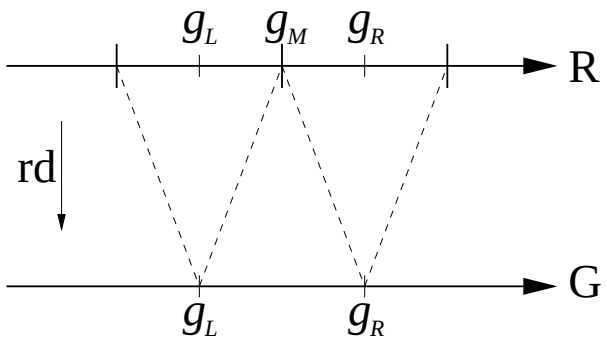


Abbildung 4: Skizze zur Rundung.

Sei g_L der nächstgelegene linke Nachbar von x und g_R der nächstgelegene rechte Nachbar sowie $g_M = (g_L + g_R)/2$ die Mitte zwischen beiden. Dann gilt für $g_L \leq x \leq g_R$ mit $x \neq g_M$

$$\mathrm{rd}(x) = \begin{cases} g_L & \text{falls } x \leq g_M, \\ g_R & \text{falls } x \geq g_M. \end{cases}$$

Im Fall der Mitte $x = g_M$ ist die korrekte Rundung nicht eindeutig festgelegt. Der IEEE-Standard schreibt in dieser Situation vor:

$$\mathrm{rd}(x) = \begin{cases} g_L & \text{falls } g_L \text{ gerade Mantisse}, \\ g_R & \text{falls } g_L \text{ ungerade Mantisse}. \end{cases}$$

Neben dem korrekten Runden gibt es noch die Begriffe

$$\begin{aligned} \textit{Abrunden:} \quad & x \mapsto g_L \\ \textit{Aufrunden:} \quad & x \mapsto g_R \\ \textit{Abschneiden:} \quad & x \mapsto g_L, \text{ falls } x \geq 0, \;\; x \mapsto g_R \text{ sonst.} \end{aligned}$$

Die Rundung verursacht einen Fehler. Man definiert

$$\Delta x := |\mathrm{rd}(x) - x| \quad \text{Absoluter Fehler}$$

und (für $x \neq 0$)

$$\varepsilon_x := \frac{\Delta x}{|x|} = \frac{|\mathrm{rd}(x) - x|}{|x|} \quad \text{Relativer Fehler}\,.$$

Für den absoluten Fehler findet man

$$\Delta x = |\mathrm{rd}(x) - x| \leq \frac{1}{2}|g_L - g_R| = \frac{1}{2}|M \cdot B^E - (M+1) \cdot B^E| = \frac{1}{2}B^E$$

und für den relativen Fehler

$$\varepsilon_x = \frac{|\mathrm{rd}(x) - x|}{|x|} \leq \frac{B^E}{2|M|B^E} = \frac{1}{2|M|} \leq \frac{1}{2}B^{1-t}\,.$$

Der relative Rundungsfehler ist damit immer kleiner oder gleich der halben Auflösung der Arithmetik. Formal hat man den Zusammenhang

$$\mathrm{rd}(x) = x(1+\varepsilon) \quad \text{mit } \; |\varepsilon| = \varepsilon_x \leq \frac{1}{2}B^{1-t}\,. \tag{2.1}$$

Die Schranke $B^{1-t}/2$ charakterisiert die Arithmetik und wird deshalb mit einem Namen belegt.

Definition 2.4: Maschinengenauigkeit

$$\varepsilon_0 := \frac{1}{2}B^{1-t}$$

Beim Datentyp short real ist die Maschinengenauigkeit $\varepsilon_0 = 5.96 \cdot 10^{-8}$, bei long real dagegen $\varepsilon_0 = 1.11 \cdot 10^{-16}$. (In MATLAB wird die Maschinengenauigkeit definiert als der Abstand von 1 zur nächst höheren Maschinenzahl, d.h. $\varepsilon_0 := \min\{g - 1 : g \in \mathbb{M}, g > 1\}$, wodurch sich $\varepsilon_0 = 2.2 \cdot 10^{-16}$ ergibt.)

Gleitpunktoperationen

Im vorhergehenden Kapitel wurde bei der Analyse der Integralrekursion vereinfachend angenommen, dass bei den arithmetischen Grundoperationen keine weiteren Fehler entstehen. In der Praxis ist dies natürlich nicht erfüllt. Im allgemeinen gilt für die Grundoperationen $\odot \in \{+, -, \cdot, /\}$

$$a,\, b \in \mathbb{G} \;\not\Rightarrow\; a \odot b \in \mathbb{G},$$

d.h. das Ergebnis muss gerundet werden.

Man führt nach Wilkinson (1958) die Notation $\mathrm{fl}(a \odot b)$ für das Ergebnis der Gleitpunktoperation ein und verlangt für eine *ideale Arithmetik*

$$\mathrm{fl}(a \odot b) = \mathrm{rd}(a \odot b) \quad \text{für } a, b \in \mathbb{G}\,. \tag{2.2}$$

Das Ergebnis der Gleitpunktoperation soll gleich dem korrekt gerundeten exakten Ergebnis sein. Diese starke Forderung ist kein Wunschtraum der Numeriker, sondern kann ohne großen Aufwand realisiert werden. Der IEEE-Standard 754 verlangt genau (2.2) von den heute verbreiteten Mikroprozessoren.

Aus der Forderung (2.2) folgt sofort die Eigenschaft

$$\mathrm{fl}(a \odot b) = (a \odot b)(1 + \varepsilon) \quad \text{mit } |\varepsilon| \le \varepsilon_0\,. \tag{2.3}$$

Man bezeichnet (2.3) auch als die *starke Hypothese* für Rundungsfehler. Sie wird von einer idealen Arithmetik erfüllt, und damit auch von jedem Mikroprozessor, der dem Standard genügt.

Die Eigenschaft (2.3) bildet die Basis der Rundungsfehleranalyse. Während ε von den Operanden $a, b \in \mathbb{G}$ abhängt, ist die Schranke ε_0 a priori bekannt und erlaubt Abschätzungen.

Bemerkungen

- Bei der Analyse von Gleitpunktoperationen vernachlässigt man den Exponentenüber- bzw. -unterlauf und betrachtet ausschließlich Gleitpunktzahlen.

- Eine korrekt rundende Arithemtik läßt sich mit einem $t+2$-stelligen Rechenwerk realisieren. Man benötigt also 2 Schutzziffern.
- Die Grundoperationen werden durch Mikroprogramme implementiert. Eine Multiplikation ist dabei ca. 2 mal teurer als eine Addition.

  Betrachten wir hierzu als Beispiel für ein Mikroprogramm noch die Addition der beiden Gleitpunktzahlen ($B = 10$, $t = 4$)

  $$0.6134 \cdot 10^4 \quad \text{und} \quad 0.3865 \cdot 10^1 :$$

  i) Exponentenangleich, d.h.: Shift der Mantissen durch Vergrößerung des kleineren Exponenten bei 2 Schutzziffern:
  $$0.613400 \cdot 10^4$$
  $$0.000386 \cdot 10^4$$
  ii) Mantissenaddition: $0.613786 \cdot 10^4$

  iii) Rundung ($t = 4$): $0.6138 \cdot 10^4$

  iv) eventuell Normierung (Mantissenshift)

  Bei der Subtraktion kann *Auslöschung* der führenden Ziffern auftreten. Ein Runden ist dann nicht mehr notwendig!
- Spezielle Funktionen wie $\sqrt{}, \sin, \cos, \ldots$ werden nach einem divisionsähnlichen Schema berechnet (*Pseudodivision, CORDIC-Verfahren*). Klassische Approximationstechniken oder Taylorreihen sind deutlich aufwendiger.

2.2 Rundungsfehleranalyse

Unter einem Algorithmus (Rechenverfahren) versteht man in der Numerik eine endliche Folge von Grundoperationen, deren Reihenfolge beim Ablauf eindeutig festliegt. In Gleitpunktarithmetik ausgeführt, verfälschen Rundungsfehler die Zwischen- und Endresultate.

Die Reihenfolge der Operationen kann bei der Rundungsfehleranalyse entscheidend sein, da eine Gleitpunktarithmetik weder Assoziativ- noch Distributivgesetz erfüllt. Im allgemeinen ist also

$$\begin{aligned} \mathrm{fl}(a + \mathrm{fl}(b + c)) &\neq \mathrm{fl}(\mathrm{fl}(a + b) + c), \\ \mathrm{fl}(a \cdot \mathrm{fl}(b + c)) &\neq \mathrm{fl}(\mathrm{fl}(a \cdot b) + \mathrm{fl}(a \cdot c)). \end{aligned}$$

Jedoch sind Addition und Multiplikation der Gleitpunktzahlen kommutativ.

Wie geht man bei der Rundungsfehleranalyse vor? Naheliegend ist eine *Vorwärtsanalyse*, welche das in Gleitpunktarithmetik berechnete mit dem exakten Resultat vergleicht. Oft ist diese Technik jedoch schwierig, weswegen sich die *Rückwärtsanalyse* als Alternative durchgesetzt hat. Sie interpretiert das berechnete Resultat als exaktes Resultat zu geeignet veränderten Eingangsdaten.

Beispiele:

$$\mathrm{fl}(a+b) = (a+b)(1+\alpha), \quad \text{d.h. exakt zu Operanden } a(1+\alpha), b(1+\alpha)$$

sowie

$$\mathrm{fl}(a\cdot b) = (a\cdot b)(1+\beta), \quad \text{d.h. exakt zu Operanden } a\sqrt{1+\beta}, b\sqrt{1+\beta}\,.$$

Als typisches Beispiel einer Vorwärtsanalyse der Rundungsfehler betrachten wir die Hintereinanderausführung einer Addition und einer Multiplikation: $(a+b)c$. In Gleitpunktarithmetik folgt mit $|\alpha|, |\beta| \leq \varepsilon_0$

$$\begin{aligned}
\mathrm{fl}(\mathrm{fl}(a+b)\cdot c) &= ((a+b)(1+\alpha)\cdot c)(1+\beta) \\
&= ((a+b)\cdot c)(1+\alpha)(1+\beta) \\
&= ((a+b)\cdot c)(1+\alpha+\beta+\alpha\beta) \\
&\doteq ((a+b)\cdot c)(1+\alpha+\beta).
\end{aligned}$$

Dabei wurde im letzte Schritt eine Linearisierung verwendet, wobei der Term $\alpha\beta$ wegen $|\alpha\beta| \leq \varepsilon_0^2 \ll \varepsilon_0$ vernachläßigt wurde. Für den relativen Fehler können wir abschätzen $|\alpha+\beta| \leq |\alpha| + |\beta| \leq 2\varepsilon_0$, d.h. der Fehler ist in der Größenordnung der Maschinengenauigkeit.

Allgemein kann man ansetzen

$$\text{Gleitpunktergebnis} = (\text{exaktes Ergebnis})(1 + \text{relativer Fehler}).$$

Sei ε der relative Fehler. Gilt $\varepsilon = \varepsilon(\alpha_1, \ldots, \alpha_n)$ mit kleinen Werten α_i, dann entspricht die Linearisierung einer mehrdimensionalen Taylor-Entwicklung

um $\alpha_i = 0$ und dem Weglassen aller Terme von zweiter und höherer Ordnung. Unter der Annahme $\varepsilon(0,\ldots,0) = 0$ folgt somit

$$\varepsilon(\alpha_1,\ldots,\alpha_n) \doteq \sum_{i=1}^{n} \frac{\partial \varepsilon}{\partial \alpha_i}(0,\ldots,0)\alpha_i.$$

Ist $\varepsilon(\alpha_1,\ldots,\alpha_n)$ ein Polynom in den Veränderlichen $\alpha_1,\ldots,\alpha_n$, dann verwendet man somit gerade den Anteil mit Polynomgrad 1.

Beispiel: Hornerschema

Die Rundungsfehleranalyse wird nun anhand des Hornerschemas vorgeführt. Dabei erreichen wir eine Rückwärtsanalyse.

Auszuwerten ist das Polynom vom Grad n

$$P(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

an der Stelle x, mit gegebenen Koeffizienten $c_0,\ldots,c_n$. Eine direkte Auswertung nach dieser Formel benötigt n (ADD+MULT) sowie die Potenzen x^k.

Nach Horner verwendet man dagegen das Schema

$$P(x) = (\cdots((c_n x + c_{n-1})x + c_{n-2})\cdots + c_1)x + c_0\,, \tag{2.4}$$

das mit n (ADD+MULT) auskommt und wesentlich weniger anfällig für Exponentenüberlauf ist.

Algorithmus 2.1: *Hornerschema*

$y := c_n$;
für $k = n-1 : -1 : 0$
    $y := y \cdot x + c_k$;

Als Resultat liefert der Algorithmus $y = P(x)$.

Für die Rundungsfehleranalyse nehmen wir an, dass die Koeffizienten c_i sowie das Argument x exakt sind. Unter Beachtung der starken Hypothese (2.3) gilt für die in Gleitpunktarithmetik anfallenden Werte

$\tilde{y} := c_n$;
für $k = n-1 : -1 : 0$
$\quad \tilde{y} := ((\tilde{y} \cdot x) \cdot (1+\mu_k) + c_k)(1+\alpha_k)$;

mit $|\mu_k|, |\alpha_k| \leq \varepsilon_0$. Zusammen also

$$\tilde{y} = \tilde{c}_0 + \tilde{c}_1 x + \cdots + \tilde{c}_n x^n$$

mit gestörten/veränderten Koeffizienten

$$\tilde{c}_k := c_k \cdot (1+\alpha_k) \cdot (1+\mu_{k-1}) \cdot (1+\alpha_{k-1}) \cdot (1+\mu_{k-2}) \cdot \ldots \cdot (1+\alpha_0)$$

und $\alpha_n := 0$. Der berechnete Polynomwert kann demnach als exakter Wert eines Polynoms mit leicht veränderten Koeffizienten gedeutet werden.

Der am stärksten betroffene Koeffizient $\tilde{c}_n$ soll noch näher betrachtet werden. Die gehäuft auftretenden Modifikatoren $1+\varepsilon$ lassen sich durch eine *Linearisierungstechnik* vereinfachen. Für $|\alpha|, |\mu| \leq \varepsilon_0$ setzt man

$$(1+\alpha) \cdot (1+\mu) \doteq 1 + \alpha + \mu$$

unter Vernachlässigung des 'kleinen' Produktes $\alpha \cdot \mu$. Damit ist

$$\tilde{c}_n \doteq c_n \cdot (1 + \mu_{n-1} + \alpha_{n-1} + \cdots + \mu_0 + \alpha_0)$$

und der absolute Fehler in $\tilde{c}_n$ genügt

$$|\Delta c_n| = |\tilde{c}_n - c_n| \doteq |\mu_{n-1} + \alpha_{n-1} + \cdots + \mu_0 + \alpha_0| \cdot |c_n| \leq 2n\varepsilon_0 |c_n| \,.$$

Für die anderen Koeffizienten zeigt man analog $|\Delta c_k| \leq (2k+1)\varepsilon_0 |c_k|$.

Im Sinne der Rückwärtsanalyse ist das Hornerschema damit gutartig: Der Fehler in den Koeffizienten bleibt bei moderatem Polynomgrad n klein.

Statt mit der Linearisierungstechnik kann man auch mit einer scharfen Abschätzung zu einem Ergebnis gelangen. Es gilt nämlich die Aussage:

$$\text{Sei } \prod_{i=1}^{m} (1+\alpha_i)^{\pm 1} =: 1+\varepsilon \,. \text{ Falls alle } |\alpha_i| \leq \varepsilon_0 \;\Rightarrow\; |\varepsilon| \leq \frac{m\varepsilon_0}{1 - m\varepsilon_0}. \quad (2.5)$$

(Zeige: $(1-\alpha_i)(1+\alpha_i) \leq 1 \;\Rightarrow\; 1-\varepsilon_0 \leq (1+\alpha_i)^{\pm 1} \leq (1-\varepsilon_0)^{-1}$. Dann gilt

$$(1-\varepsilon_0)^m \leq 1+\varepsilon \leq \frac{1}{(1-\varepsilon_0)^m} \quad \Rightarrow \quad (1-\varepsilon_0)^m - 1 \leq \varepsilon \leq \frac{1}{(1-\varepsilon_0)^m} - 1.$$

Damit ergibt sich aus der Bernoullischen Ungleichung

$$1 - m\varepsilon_0 \leq (1-\varepsilon_0)^m$$

für $m\varepsilon_0 < 1$ die gewünschte Abschätzung:

$$\varepsilon \leq \frac{1}{(1-\varepsilon_0)^m} - 1 \leq \frac{1}{1-m\varepsilon_0} - 1 = \frac{m\varepsilon_0}{1-m\varepsilon_0},$$

$$\varepsilon \geq (1-\varepsilon_0)^m - 1 \geq -m\varepsilon_0 \geq \frac{-m\varepsilon_0}{1-m\varepsilon_0}.)$$

Die Abschätzung (2.5) korrigiert den Fehler des Koeffizienten c_n etwas und liefert

$$\tilde{c}_n = c_n(1+\varepsilon) \quad \text{mit } |\varepsilon| \leq \frac{2n\varepsilon_0}{1-2n\varepsilon_0}$$

(Im Vergleich dazu vernachlässigt die Linearisierungstechnik den Nenner $1-2n\varepsilon_0$.)

Die am Hornerschema vorgeführte Rundungsfehleranalyse ist eine *a priori* Analyse. Man geht von einer realistischen Hypothese über die Rundungsfehler aus und ermittelt daraus Schranken für den Gesamtfehler. Weil dabei immer die ungünstigsten Umstände angenommen werden, ist der tatsächliche Fehler in der Anwendung des Algorithmus meist viel kleiner.

Neben der a priori Analyse werden manchmal auch *a posteriori* Fehlerschranken zu einer numerisch berechneten Näherung ermittelt. Solch eine Kontrollrechnung kann z.B. durch Einsetzen der Lösung und Nachrechnen in höherer Genauigkeit erfolgen. Dazu zählen auch die Techniken der *Intervallarithmetik*. Dort sind die Eingabedaten Intervalle, und im Laufe des Algorithmus werden daraus neue Intervalle berechnet, die die exakte Lösung sowie alle potentiellen Rundungsfehler einschließen. Die Intervallarithmetik kann allerdings Korrelationen der Rundungsfehler nicht berücksichtigen, weswegen die Intervalle oft stark anwachsen und nicht brauchbar sind.

2.3 Fehlerfortpflanzung und Kondition

In diesem Abschnitt untersuchen wir, wie sich Unsicherheiten in den Eingabedaten auf die Resultate auswirken. Es geht also um das vorliegende Problem und nicht um den mit Rundungsfehlern behafteten Algorithmus. Die Aussagen haben aber große Konsequenzen für die Numerik bzw. die Fortpflanzung von (Rundungs-)Fehlern:

1. Bei der Zerlegung des Lösungswegs in mehrere Teilschritte lassen sich die Rundungsfehler des i-ten Schritts ansehen als Eingabefehler für den $i+1$-ten Schritt.
2. Die Kondition eines Problems gibt einen möglichen Maßstab für die Beurteilung der Rundungsfehler.

Abstrakt wird ein zu lösendes Problem wie folgt charakterisiert:

Eingabedaten	$x = (x_1, \ldots, x_n)^\top \in \mathbb{D} \subseteq \mathbb{R}^n$
Resultate	$y = (y_1, \ldots, y_m)^\top \in \mathbb{R}^m$
Problem	Funktion (Abbildung) φ, die allen zulässigen Eingabedaten das eindeutige Resultat zuordnet: $\varphi : \mathbb{D} \to \mathbb{R}^m, \quad x \mapsto y = \varphi(x)$

Das Problem lösen heißt, den Wert von φ an der Stelle x zu berechnen.

Wie empfindlich sind die Resultate y gegenüber Änderungen in den Eingabedaten x?

Die *differentielle Fehleranalyse* bzw. *Störungstheorie 1. Ordnung* betrachtet die partiellen Ableitungen $\partial\varphi_i/\partial x_j$, um die Empfindlichkeit gegenüber Änderungen der Eingabedaten zu quantifizieren.

Es bezeichne $\tilde{x}_i$ den abgeänderten Wert x_i und analog $\tilde{x} \in \mathbb{D}$ bzgl. $x \in \mathbb{D}$. Sei $\Delta x_i := \tilde{x}_i - x_i$ die absolute Störung und $\varrho x_i := \Delta x_i / x_i$ die relative Störung der Komponente i. In erster Ordnung gilt für die Auswirkungen

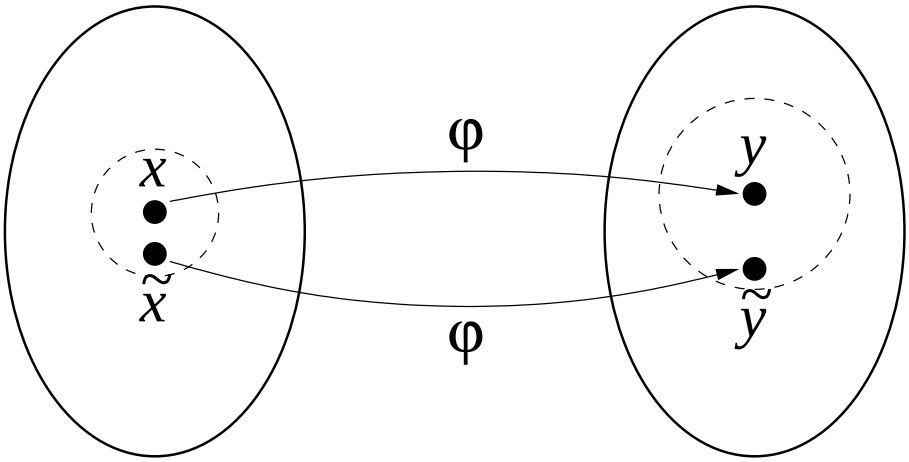


Abbildung 5: Skizze zu "gut konditioniert".

$\Delta y_i := \varphi_i(\tilde{x}) - \varphi_i(x)$ und $\varrho y_i := \Delta y_i / y_i$

$$\Delta y_i \doteq \sum_{j=1}^{n} \frac{\partial \varphi_i}{\partial x_j}(x) \Delta x_j, \tag{2.6}$$

$$\varrho y_i \doteq \sum_{j=1}^{n} \frac{x_j}{y_i} \frac{\partial \varphi_i}{\partial x_j}(x) \varrho x_j. \tag{2.7}$$

Für $n = m = 1$ mit Taylorentwicklung an Stelle x:

$$\begin{aligned} \Delta y &= \varphi(\tilde{x}) - \varphi(x) \\ &= \varphi(x) + \frac{\partial \varphi}{\partial x}(x)\Delta x + \frac{\partial^2 \varphi}{\partial x^2}(x + \vartheta \Delta x)(\Delta x)^2 - \varphi(x) \\ &\doteq \frac{\partial \varphi}{\partial x}(x)\Delta x \end{aligned}$$

sowie

$$\varrho y = \frac{\Delta y}{y} = \frac{\partial \varphi}{\partial x}(x) \frac{\Delta x}{y} \frac{x}{x} = \frac{x}{y} \frac{\partial \varphi}{\partial x}(x) \varrho x.$$

Statt der komponentenweisen Notation schreibt man für (2.6) auch $\Delta y = \partial \varphi / \partial x \cdot \Delta x$ mit der Funktionalmatrix $\partial \varphi / \partial x$.

$$\Delta y = \frac{\partial \varphi}{\partial x} \cdot \Delta x : \quad \begin{pmatrix} \Delta y_1 \\ \vdots \\ \Delta y_m \end{pmatrix} = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \varphi_m}{\partial x_1} & \cdots & \frac{\partial \varphi_m}{\partial x_n} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{pmatrix}$$

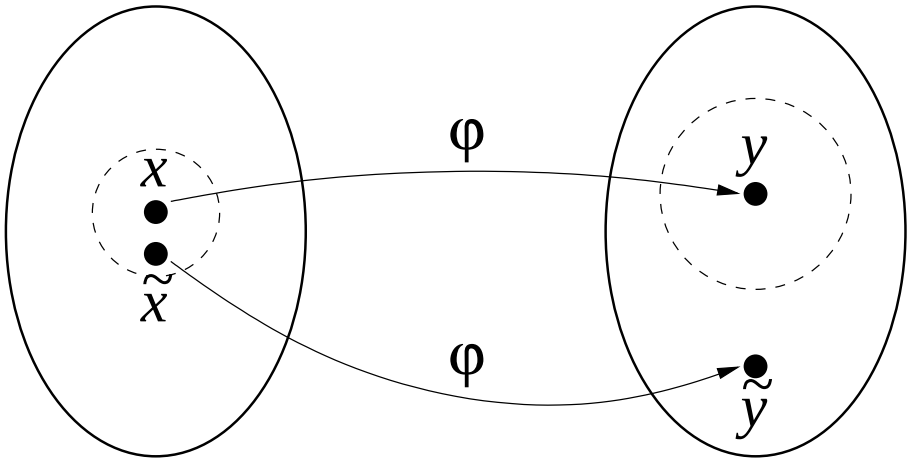


Abbildung 6: Skizze zu "schlecht konditioniert".

Die Größe $x_j/y_i \cdot \partial\varphi_i/\partial x_j$, siehe (2.7), misst die Empfindlichkeit, mit der ein relativer Fehler in x_j den relativen Fehler in y_i beeinflußt. Genauso misst $\partial\varphi_i/\partial x_j$ die Empfindlichkeit gegenüber absoluten Fehlern. Man nennt solche Maßzahlen *Konditionszahlen*.

Definition 2.5: Kondition

Ein Problem heisst gut konditioniert, falls kleine Änderungen Δx bzw. ϱx nur in kleinen Änderungen Δy bzw. ϱy resultieren, andernfalls schlecht konditioniert. Als Konditionszahlen hat man

$$\left|\frac{\partial\varphi_i}{\partial x_j}\right|, \quad \left|\frac{x_j}{y_i}\frac{\partial\varphi_i}{\partial x_j}\right| .$$

Auch die vier arithmethischen Grundoperationen sind nach Definition ein Problem und haben deshalb eine Kondition. Für den absoluten Fehler hat man die Auswirkungen:

$$\Delta(a \pm b) = \Delta a \pm \Delta b \tag{2.8}$$
$$\Delta(a \cdot b) = b\Delta a + a\Delta b \tag{2.9}$$
$$\Delta(a/b) = \Delta a/b - a\Delta b/b^2 \tag{2.10}$$

Für die relativen Fehler dagegen (mit $\varrho x = \Delta x/x$):

$$\varrho(a \pm b) = \varrho a \frac{a}{a \pm b} + \varrho b \frac{b}{a \pm b} \tag{2.11}$$

$$\varrho(a \cdot b) = \varrho a + \varrho b \tag{2.12}$$

$$\varrho(a/b) = \varrho a - \varrho b \tag{2.13}$$

Wie man sieht, sind die relativen Konditionszahlen für Multiplikation und Division 1, so dass diese Operationen als gut konditioniert bezeichnet werden. Bei der Addition/Subtraktion sind die absoluten Konditionszahlen klein, die relativen dagegen *unbegrenzt*. Die sogenannte *Auslöschung* kann hier eine gefährliche Situation darstellen: Falls $a \pm b \approx 0$, heben sich die gemeinsamen führenden Ziffern weg.

Beispiel 2.4: Auslöschung

Gegeben seien Gleitpunktzahlen zur Basis $B = 10$ mit $t = 5$ Stellen. Die folgenden Zahlen x, y sind keine solchen Gleitpunktzahlen und müßen daher gerundet werden:

$$x = 0.\underline{120}587 \quad \longrightarrow \quad \tilde{x} := \mathrm{rd}(x) = 0.12059$$

$$y = 0.\underline{120}942 \quad \longrightarrow \quad \tilde{y} := \mathrm{rd}(y) = 0.12094.$$

Es folgen die relativen Fehler

$$\left|\frac{\tilde{x} - x}{x}\right| = 2.5 \cdot 10^{-5} < 10^{-4}, \qquad \left|\frac{\tilde{y} - y}{y}\right| = 1.7 \cdot 10^{-5} < 10^{-4},$$

welche wie erwartet in der Größenordnung der Maschinengenauigkeit liegen. Da die führenden Stellen von x und y übereinstimmen ergibt sich bei Subtraktion Auslöschung:

$$y - x = 0.\underline{000}355 = 0.355 \cdot 10^{-3}$$

$$\tilde{y} - \tilde{x} = 0.\underline{000}35 = 0.35 \cdot 10^{-3}.$$

Der relative Fehler vergrößert sich deutlich:

$$\left|\frac{(\tilde{y} - \tilde{x}) - (y - x)}{y - x}\right| = \frac{0.35 - 0.355}{0.355} = 1.4 \cdot 10^{-2} > 10^{-4}.$$

Grund hierfür ist, dass die kleinen Rundungsfehler in den hintersten Stellen durch die Auslöschung in weiter vorne liegende Stellen übertragen wurden.

Für die Wurzelberechnung findet man übrigens noch

$$\Delta(\sqrt{a}) = \Delta a/(2\sqrt{a}), \quad \varrho(\sqrt{a}) = \tfrac{1}{2}\varrho a\,.$$

Die Quadratwurzel ist demnach gut konditioniert.

Beispiel 2.5: Wie man am Problem des Schnittpunktes zweier Geraden erkennt, siehe Abbildung 7, kann ein schlecht konditioniertes Problem durch keinen Algorithmus gerettet werden.

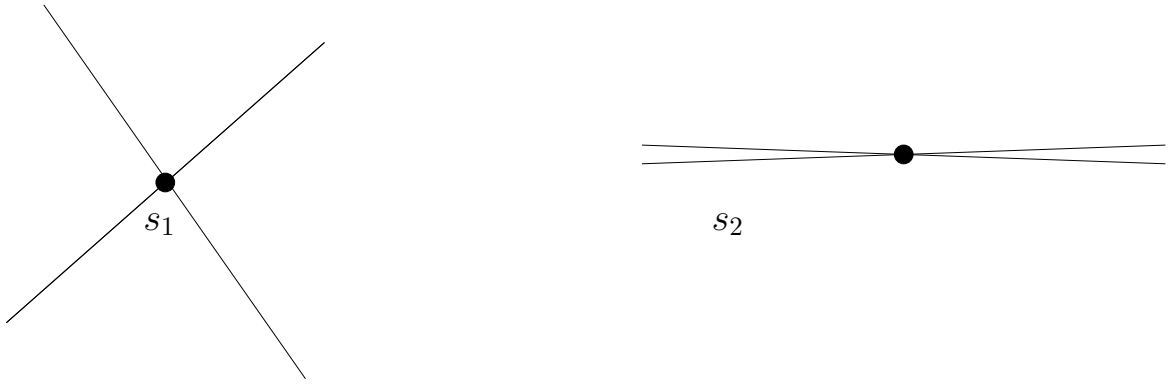


Abbildung 7: Berechnung des Schnittpunktes s_i zweier Geraden $y_i = a_i x + b_i$. Kleine Änderungen in den Daten a_i, b_i führen links auf kleine, und rechts auf große Änderungen in s_1 bzw. s_2.

Beispiel 2.6: Sei $y = \varphi(p,q) := -p + \sqrt{p^2+q}$. Die partiellen Ableitungen lauten

$$\frac{\partial \varphi}{\partial p} = -1 + \frac{p}{\sqrt{p^2+q}} = \frac{-y}{\sqrt{p^2+q}}, \quad \frac{\partial \varphi}{\partial q} = \frac{1}{2\sqrt{p^2+q}}\,.$$

Als relative Fehlerauswirkung ergibt sich

$$\varrho y = \frac{-p}{\sqrt{p^2+q}}\varrho p + \frac{q}{2y\sqrt{p^2+q}}\varrho q = \frac{-p}{\sqrt{p^2+q}}\varrho p + \frac{p+\sqrt{p^2+q}}{2\sqrt{p^2+q}}\varrho q$$

Für $q > 0$ ist das Problem gut konditioniert mit

$$|\varrho y| \leq |\varrho p| + |\varrho q|\,.$$

Für $q \approx -p^2$ ist es dagegen schlecht konditioniert.

Wenn das Problem φ in Teilprobleme zerlegt wird, hat jedes Teilproblem seine eigene Kondition. Nach der Kettenregel für $\varphi = \sigma \circ \tau$

$$\frac{\partial \varphi}{\partial x} = \left(\frac{\partial \sigma}{\partial z}\right)_{z=\tau(x)} \cdot \frac{\partial \tau}{\partial x}$$

folgt somit die Merkregel

$$\mathrm{cond}_{\sigma\circ\tau} = \mathrm{cond}_\sigma \cdot \mathrm{cond}_\tau \,.$$

Man beachte, dass die Kondition des Gesamtproblems unabhängig von der Zerlegung in Teilprobleme ist. Die Dinge wären nun unproblematisch, wenn für $\mathrm{cond}_{\sigma\circ\tau}$ klein auch die Faktoren cond_σ und cond_τ klein wären.

Leider kommt es aber häufig vor, dass bestimmte Varianten/Zerlegungen $\varphi = \hat{\sigma} \circ \hat{\tau}$ auf einen Faktor $\mathrm{cond}_{\hat{\sigma}}$ oder $\mathrm{cond}_{\hat{\tau}}$ führen, der sehr groß ist (und der zweite Faktor sehr klein). Rundungsfehler, die sich als Eingabefehler eines Teilschritts interpretieren lassen, können dann den Algorithmus in dieser Variante unbrauchbar machen.

Beispiel 2.7: Wurzelberechnung bei quadratischen Gleichungen
Die im Beispiel 2.4 behandelte Berechnung von

$$y = \varphi(p,q) := -p + \sqrt{p^2+q}$$

liefert die Wurzel kleinsten Betrages der quadratischen Gleichung

$$y^2 + 2py - q = 0\,.$$

Das Gesamtproblem war für $q > 0$ gut konditioniert. Wir untersuchen nun zwei Algorithmen für diesen Fall,

$$A1: \quad s = p^2,\ t = s + q,\ u = \sqrt{t},\ y = -p + u;$$

sowie

$$A2: \quad s = p^2,\ t = s + q,\ u = \sqrt{t},\ v = p + u,\ y = q/v.$$

Welcher Algorithmus ist besser?

Antwort: Das hängt vom Vorzeichen von p ab!
Zuerst erkennen wir, dass sich beide Algorithmen in den ersten drei (stabilen!) Schritten nicht unterscheiden. Es ist also nur der letzte Schritt $\bar{\varphi}(p,u) = -p + u$ von A1 bzw. die letzten beiden Schritte $\varphi_1(p,u) = p + u$, $\varphi_2(q,v) = q/v$ von A2 für die beiden Fälle $p < 0$ und $p > 0$ zu untersuchen (Wieso ist $p = 0$ harmlos?).

1. Fall $p < 0$:
Für den letzten Schritt von A1 gilt (Addition von zwei positiven Zahlen):

$$|\varrho\bar{\varphi}| < |\varrho p| + |\varrho u|\,.$$

Beim vorletzten Schritt von A2 tritt jedoch für $|p| \gg q$ Auslöschung auf:

$$\varrho\varphi_1 = \frac{p}{p+u}\varrho p + \frac{u}{p+u}\varrho u$$

mit $p + u \approx -|p| + \sqrt{p^2+q} \approx 0$.

2. Fall $p > 0$:
Hier ist es genau umgekehrt. Nun tritt beim letzten Schritt von A1 für $p \gg q$ Auslöschung auf. Die letzten beiden Schritte von A2 erfüllen jedoch

$$|\varrho\varphi_1| \leq |\varrho p| + |\varrho u|, \quad \varrho\varphi_2 = \varrho q - \varrho v\,.$$

Allgemeiner sei nun $\varphi = \sigma \circ \tau$ und $\tilde{\varphi}, \tilde{\sigma}, \tilde{\tau}$ die Auswertungen unter Einfluss der Rundungsfehler im Fall $n = m = 1$. Somit gilt

$$\tilde{\sigma}(z) = \sigma(z) + \Delta\sigma, \qquad \tilde{\tau}(x) = \tau(x) + \Delta\tau$$

mit den Fehlertermen $\Delta\sigma, \Delta\tau$. Sei $\Delta x = \tilde{x} - x$ ein Fehler in den Eingangsdaten. Als Plausibilitätsüberlegung folgt mit zwei Linearisierungen

$$\begin{aligned}
\tilde{\varphi}(x+\Delta x) &= \tilde{\sigma}(\tilde{\tau}(x+\Delta x)) \\
&= \sigma(\tau(x+\Delta x) + \Delta\tau) + \Delta\sigma \\
&\doteq \sigma(\tau(x) + \tau'(x)\Delta x + \Delta\tau) + \Delta\sigma \\
&\doteq \sigma(\tau(x)) + \sigma'(\tau(x))\tau'(x)\Delta x + \sigma'(\tau(x))\Delta\tau + \Delta\sigma \\
&= \varphi(x) + \mathrm{cond}_\varphi \Delta x + \mathrm{cond}_\sigma \Delta\tau + \Delta\sigma,
\end{aligned}$$

wobei die Konditionszahlen bzgl. der absoluten Fehler auftreten. Der Fehler in den Anfangsdaten wird wie üblich mit der Kondition von φ verstärkt. Der Rundungsfehler aus der Auswertung von τ wird jedoch zusätzlich noch mit der Kondition von σ verstärkt. Der Rundungsfehler aus der Auswertung von σ bleibt dagegen unverändert und somit im allgemeinen unproblematisch.

Man nennt einen Algorithmus zur Berechnung von $y = \varphi(x)$ *numerisch stabiler* als einen zweiten Algorithmus, falls der Gesamteinfluß der Rundungsfehler kleiner ist. In diesem Zusammenhang noch zwei Begriffe:

Definition 2.6: Akzeptables Resultat

Eine numerisch berechnete Lösung $\tilde{y}$ mit Eingangsdaten x heißt akzeptabel, falls sie sich interpretieren lässt als exakte Lösung zu modifizierten Daten $\tilde{x}$, die innerhalb gegebener Toleranzen bleiben:

$$\tilde{y} \text{ akzeptabel } :\Leftrightarrow \exists \tilde{x} : \tilde{y} = \varphi(\tilde{x}) \text{ und } \|\tilde{x} - x\| \leq \Delta x \text{ bzw. } \frac{\|\tilde{x} - x\|}{\|x\|} \leq \varepsilon_x$$

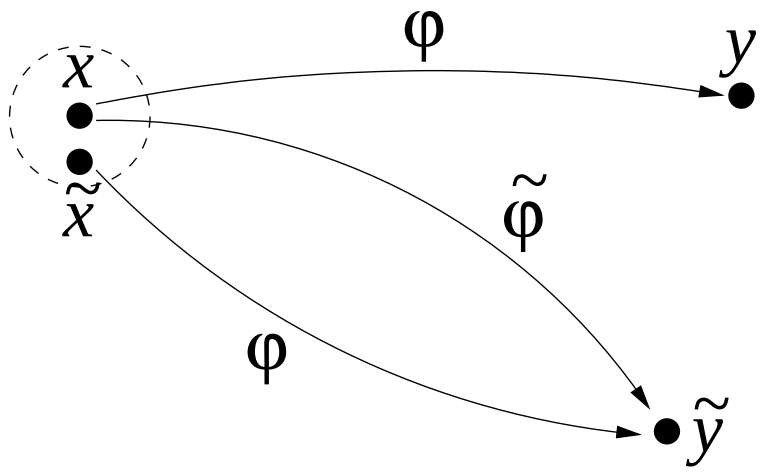


Abbildung 8: Skizze zu "$\tilde{y}$ akzeptable Näherung"

Eine akzeptable Näherung oder Lösung muß sich also interpretieren lassen (im Sinne der Rückwärtsanalyse) als exakt zu Eingabedaten, die im Rahmen vorgegebener Toleranzen Δx bzw. ε_x geeignet abgeändert worden sind.

Die Definition der akzeptablen Näherung ist unabhängig davon, ob das korrespondierende Problem gut oder schlecht konditioniert ist. Ist ein Problem schlecht konditioniert, dann könnte ein Algorithmus dennoch akzeptable Ergebnisse liefern.

Definition 2.7: Numerisch stabiler (gutartiger) Algorithmus

Ein Algorithmus heißt numerisch stabil, wenn er unter dem Einfluß der Rundungsfehler Näherungslösungen $\tilde{y}$ liefert, die für alle Daten x akzeptabel sind.

Bemerkungen

- Man kann die Definition 2.6 abschwächen und nur fordern, dass ein y' existiert mit $\|\tilde{y} - y'\|$ klein und y' exakt zu den Daten $\tilde{x}$.
- Das Hornerschema zur Polynomauswertung ist ein stabiler Algorithmus, genauso die vier arithmetischen Grundoperationen.
- Zwei stabile Algorithmen hintereinander ausgeführt ergeben keineswegs immer wiederum einen stabilen Algorithmus.
- Um zu zeigen, dass ein Algorithmus instabil ist, genügt ein einziges Gegenbeispiel. Um zu zeigen, dass er stabil ist, muß man neben den Verfahrensfehlern auch die Rundungsfehler abschätzen.
- Bei gut konditionierten Problemen kann man eine Vorwärtsanalyse der Rundungsfehler noch durchführen. Bei schlecht konditionierten Problemen kommt dagegen nur die Rückwärtsanalyse in Frage.
- Die Definition (2.7) der relativen Konditionszahlen hat den Nachteil, dass sie nur für nicht verschwindende y_i, x_j sinnvoll ist. Häufig ist sie auch recht unhandlich, weswegen man dann auf eine geeignete Norm für die Matrix mit den Konditionszahlen ausweicht.

Wir beenden dieses Kapitel mit einer Diskussion der Fehlerquellen, die auf dem Weg von der wissenschaftlich/technischen Aufgabenstellung bis zur algorithmischen Realisierung einer numerischen Lösung auftreten können.

Grobe Klassifikation der Fehlerquellen:

- *Modellfehler*: Die exakte Lösung des mathematischen Modells/Gleichungen weicht vom Verhalten in der Realität ab. Grund sind häufig Vereinfachungen/Idealisierungen, die im Modell gemacht wurden.
- *Fehler in Eingabedaten*: Die Eingabedaten sind häufig nur näherungsweise bekannt: etwa aus Messungen, welche mit Messfehlern behaftet sind.

- *Verfahrensfehler/Approximationsfehler*: Oft werden numerische Verfahren eingesetzt, deren exakte Lösung (d.h. ohne Rundungsfehler) nur eine Näherung der exakten Lösung des Modells darstellen. Beispielsweise werden Integrale durch Rechtecksummen approximiert.
- *Rundungsfehler*: Die Lösung eines numerischen Verfahrens kann auf einem Rechner nicht exakt erhalten werden, da die Gleitpunktoperationen mit Rundungsfehlern behaftet sind.

Wer — Anwender/Mathematiker/Informatiker — ist dafür verantwortlich, die jeweilige Fehlerquelle wenn nicht zu eliminieren, so doch zu verkleinern bzw. zu beschränken?

	Anwender	Mathematiker	Informatiker
Modellfehler	++	+	
Mess- und Eingabefehler	++		
Approximationsfehler		++	
Rundungsfehler		++	+

Einige Faustregeln: Die Abschätzung für die Größe der vorhandenen Modell-, Mess- und Eingabefehler legt fest, ob ein Resultat akzeptabel ist, d.h., definiert Δx bzw. ε_x für die Rückwärtsanalyse. Rundungsfehler werden in Abhängigkeit der Kondition akzeptiert, falls die Abschätzung

$$\frac{||\widetilde{\varphi}(x) - \varphi(x)||}{||\varphi(x)||} \approx \text{cond}_\varphi \cdot \varepsilon_0 \cdot n$$

gilt, wobei die Kondition im relativen Sinn benutzt wird und n die Anzahl der Elementaroperationen bezeichne. Als Motivation dieser Formel setzen wir im Sinne der Rückwärtsanalyse $\widetilde{\varphi}(x) = \varphi(\tilde{x})$ mit $\Delta x = \tilde{x} - x$ an. Es folgt, vergleiche (2.7),

$$\frac{||\varphi(\tilde{x}) - \varphi(x)||}{||\varphi(x)||} \approx \text{cond}_\varphi \cdot \|\Delta x\|.$$

Wir tolerieren Abweichungen $\|\Delta x\| \leq \varepsilon_0 \cdot n$, da bei jeder Elementaroperation häufig der Fehler um die Maschinengenauigkeit ε_0 anwachsen kann.

Zum Abschluß noch ein Literaturhinweis zur Rundungsfehleranalyse:

Christian Reinsch: *Die Behandlung von Rundungsfehlern in der numerischen Analysis.* In: Jahrbuch Überblicke Mathematik 1979, Bibliographisches Institut, S. 43–62.

Die Sätze, mit denen Christian Reinsch seine Ausführungen beschließt, standen auch der Zielrichtung dieses Kapitels Pate:

> *Die feinen Punkte einer Rundungsfehler-Analyse werden die meisten Anwender numerischer Methoden gerne dem Experten überlassen. Um aber dessen Aussagen zu verstehen und seine Produkte zuverlässig einzusetzen, ist das Vertrautsein mit einigen grundsätzlichen Punkten unbedingt erforderlich.*

Kapitel 3

Lineare Gleichungssysteme

In diesem Kapitel beschäftigen wir uns mit der numerischen Lösung von linearen Gleichungssystemen. Die Kondition dieses Problems wird charakterisiert. Grundlegender Algorithmus ist die Gaußelimination, welche einer Rundungsfehleranalyse unterzogen wird.

3.1 Motivation

Wir betrachten ein lineares Gleichungssystem (LGS)

$$A \cdot x = b \tag{3.1}$$

mit quadratischer Matrix $A \in \mathbb{R}^{n \times n}$ und rechter Seite $b \in \mathbb{R}^n$. Gesucht ist die Lösung $x \in \mathbb{R}^n$. Eine eindeutige Lösung existiert genau dann, wenn die Determinante $\det(A) \neq 0$ erfüllt. Formal kann die Lösung von (3.1) in diesem Fall dargestellt werden als

$$x = A^{-1} b$$

mit der inversen Matrix $A^{-1} \in \mathbb{R}^{n \times n}$.

Aus der Cramerschen Regel erhält man

$$x_i = \frac{\det(A_{i,b})}{\det(A)} \qquad \text{für } \; i = 1, \ldots, n \; ,$$

wobei in der Matrix $A_{i,b}$ die i-te Spalte von A durch den Vektor b ausgetauscht wurde. Formal sind somit $n+1$ Determinanten auszurechnen um die Lösung x zu bestimmen. Für die Determinanten liegt die Leibniz-Formel vor:

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sign}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Die Menge S_n der Permutationen einer n-elementigen Menge besitzt die Mächtigkeit $|S_n| = n!$. Somit sind $(n-1)n!$ Multiplikationen zur Berechnung der Determinante notwendig. Die Anwendung der Cramersche Regel erfordert also insgesamt $(n+1)(n-1)n!$ Multiplikationen. Beispielsweise folgt somit der Aufwand:

n	Anzahl Mult.
10	$359 \cdot 10^6$
11	$4790 \cdot 10^6$
12	$68497 \cdot 10^6$

Der Rechenaufwand steigt exponentiell mit n an. Zudem stellt die Cramersche Regel einen instabilen Algorithmus dar. Wünschenswert ist ein stabiler Algorithmus, dessen Rechenaufwand nur polynomial mit n anwächst.

Aus der Linearen Algebra bekannte Zusammenhänge sind somit für die numerische Realisierung auf dem Rechner völlig ungeeignet. Als Merkregel beachte man auf dem Rechner:

Grundsatz 1: Die numerische Lösung von $A \cdot x = b$ niemals mittels der Cramerschen Regel!

Grundsatz 2: Niemals eine Matrix A explizit numerisch invertieren (d.h. A^{-1} bestimmen), sondern stets ein LGS lösen!

Grundsatz 3: Niemals $\det(A)$ verwenden!

3.2 Elementarmatrizen

Wir geben eine Liste von Matrizen an, welche elementare Operationen beschreiben. In der numerischen linearen Algebra können Algorithmen oft durch sukzessive Operationen mit diesen Matrizen beschrieben werden. Jedoch stellt diese Beschreibung nur ein theoretisches Hilfsmittel dar.

a) Skalierung:

Wir betrachten eine quadratische Diagonal- oder *Skalierungsmatrix*

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_n). \tag{3.2}$$

Operation auf Vektoren: $(Dx)_j = d_j x_j, \quad j = 1, \ldots, n.$

Operation auf Matrizen:

$(DA)_{ij} = d_i a_{ij}, \quad i$-te Zeile wird skaliert mit $d_i,\ i = 1, \ldots, n.$

$(AD)_{ij} = a_{ij} d_j, \quad j$-te Spalte wird skaliert mit $d_j,\ j = 1, \ldots, n.$

Inverse: $D^{-1} = \operatorname{diag}(d_1^{-1}, d_2^{-1}, \ldots, d_n^{-1}) \quad (d_j \neq 0,\ j = 1, \ldots, n)$

Ähnlichkeitstransf.: $A \mapsto DAD^{-1},\ a_{ij} \mapsto (d_i/d_j) a_{ij},\ i, j = 1, \ldots, n$

b) Transposition:

Transpositionsmatrizen P_{ij} sind spezielle Permutationsmatrizen. Sie stimmen mit der Einheitsmatrix I bis auf die vier Einträge $(i,i), (i,j), (j,i), (j,j)$ überein, wobei $i, j \in \{1, \ldots, n\}$ eine spezielle Wahl bezeichnet.

$$P_{ij} := \begin{pmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 0 & & & 1 & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & 1 & & & & 0 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{pmatrix} \tag{3.3}$$

Operation auf Vektoren: $P_{ij}x$ vertauscht Komponenten i und j.

Operation auf Matrizen:

$P_{ij}A$ vertauscht Zeilen i und j.

AP_{ij} vertauscht Spalten i und j.

Inverse: $P_{ij}^{-1} = P_{ij}$ (Involution)

Ähnlichkeitstransf.: $A \mapsto P_{ij}AP_{ij}$ vertauscht Zeilen und Spalten.

Insbesondere: $\begin{matrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{matrix} \mapsto \begin{matrix} a_{jj} & a_{ji} \\ a_{ij} & a_{ii} \end{matrix}$ 'kreuzweise'

Da P_{ij} symmetrisch ist ($P_{ij} = P_{ij}^\top$), ist P_{ij} orthogonal wegen $P_{ij}^{-1} = P_{ij}$.

c) Permutation:

Sei $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ eine bijektive Abbildung, d.h. σ beschreibt eine Permutation der n-elementigen Menge. Die korrespondierende *Permutationsmatrix* ergibt sich aus einer Permutation der Zeilen der Ein-

heitsmatrix

$$I = \begin{pmatrix} - & e_1{}^\top & - \\ & \vdots & \\ - & e_n{}^\top & - \end{pmatrix} \quad \longrightarrow \quad P_\sigma := \begin{pmatrix} - & e_{\sigma(1)}{}^\top & - \\ & \vdots & \\ - & e_{\sigma(n)}{}^\top & - \end{pmatrix}. \tag{3.4}$$

Operation auf Vektoren: $P_\sigma x$ permutiert Komponenten von x.

Operation auf Matrizen:

$P_\sigma A$ permutiert Zeilen von A.

AP_σ permutiert Spalten von A.

Inverse: $P_\sigma^{-1} = P_{\sigma^{-1}} = P_\sigma^\top$

Jede Permutationsmatrix kann als Produkt von bestimmten Transpositionsmatrizen dargestellt werden, d.h.

$$P_\sigma = \prod_{l=1}^{L} P_{i(l),j(l)}. \tag{3.5}$$

Umgekehrt stellt ein beliebiges Produkt von Transpositionsmatrizen eine Permutationsmatrix dar.

d) Zeilen/Spalten-Operatoren:

Die elementare Operation, ein Vielfaches einer Zeile/Spalte zu einer anderen Zeile/Spalte hinzuzuaddieren, wird beschrieben durch die Matrix

$$N_{ij}(\alpha) := \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & \alpha & \ddots & \\ & & & 1 \end{pmatrix} \tag{3.6}$$

mit $i \neq j$. $N_{ij}(\alpha)$ ist identisch mit der Einheitsmatrix außer der Komponente (i, j), die das Element α enthält.

Operation auf Vektoren:

$$N_{ij}(\alpha)x: \; x_i \mapsto x_i + \alpha \cdot x_j \qquad \alpha\text{-mal Komp. } j \text{ addiert zu Komp. } i$$

Operation auf Matrizen:

$N_{ij}(\alpha)A$: α-mal Zeile j addiert zu Zeile i

$AN_{ij}(\alpha)$: α-mal Spalte i addiert zu Spalte j

Inverse: $N_{ij}(\alpha)^{-1} = N_{ij}(-\alpha)$

Ähnlichkeitstransformationen sind hier nicht von Interesse.

Produkte der Matrizen $N_{ij}(\alpha)$ sind von Bedeutung, da sie bei der Gauß-Elimination auftreten.

$$\prod_{i=1, i\neq j}^{n} N_{ij}(\alpha_i) = \begin{pmatrix} 1 & & & \alpha_1 & & & \\ & \ddots & & \vdots & & & \\ & & \ddots & \alpha_{i-1} & & & \\ & & & \ddots & & & \\ & & & \alpha_{i+1} & \ddots & & \\ & & & \vdots & & \ddots & \\ & & & \alpha_n & & & 1 \end{pmatrix} \tag{3.7}$$

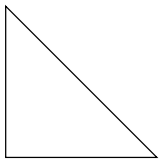
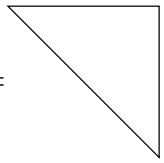
Das Ergebnis ist unabhängig von der Reihenfolge der Faktoren im Produkt. Desweiteren folgt (unter Beachtung der Reihenfolge)

$$\begin{aligned} &\prod_{i>1} N_{i1}(\alpha_{i1}) \prod_{i>2} N_{i2}(\alpha_{i2}) \dots \prod_{i>n-1} N_{i,n-1}(\alpha_{i,n-1}) \\ &= \begin{pmatrix} 1 & & & \\ \alpha_{21} & \ddots & & \\ \vdots & & \ddots & \\ \alpha_{n,1} & & \alpha_{n,n-1} & 1 \end{pmatrix}. \end{aligned} \tag{3.8}$$

Somit kann man eine Dreiecksmatrix erzeugen.

3.3 Gauß-Elimination und Pivotsuche

Spezialfälle von linearen Gleichungssystemen (3.1) liegen bei Dreiecksmatrizen vor:

$A = (a_{ij}) =$  $(a_{ij} = 0 \text{ for } j > i)$ $A =$  $(a_{ij} = 0 \text{ for } i > j)$

Vorausgesetzt ist jeweils $a_{ii} \neq 0$ für alle i. Hier kann die Lösung x sukzessive bestimmt werden durch Vorwärtssubstitution bei linken unteren Dreiecksmatrizen und Rückwärtssubstitution bei rechten oberen Dreiecksmatrizen.

Algorithmus 3.1: Vorwärtssubstitution

for $i = 1 : n$

$$x_i := \Big(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j\Big)/a_{ii}\,;$$

end

Algorithmus 3.2: Rückwärtssubstitution

for $i = n : -1 : 1$

$$x_i := \Big(b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j\Big)/a_{ii}\,;$$

end

Die Bezeichnungen erklären sich daraus, dass die Vorwärtssubstitution die Sequenz $x_1, x_2, \ldots, x_n$ sukzessive erzeugt, während die Rückwärtssubstitution die Reihenfolge $x_n, x_{n-1}, \ldots, x_1$ liefert.

Liegt eine Zerlegung $A = L \cdot R$ in linke untere und rechte obere Dreiecksmatrix vor, dann kann die Lösung von $A \cdot x = b$ erhalten werden über $(L \cdot R)x = b$, $L(Rx) = b$, $y := Rx$ mittels:

1. Löse $Ly = b$ mit Vorwärtssubstitution,
2. Löse $Rx = y$ mit Rückwärtssubstitution.

Die Gauß-Elimination liefert eine Zerlegung der gesuchten Form.

Algorithmus 3.3: Gauß-Elimination (ohne Pivotsuche)

for $j = 1 : n - 1$
  for $i = j + 1 : n$
    for $k = j + 1 : n$
      $a_{ik} := a_{ik} - (a_{ij}/a_{jj})\, a_{jk}$; $\qquad (\star)$
    end
    $b_i := b_i - (a_{ij}/a_{jj})\, b_j$;
  end
end

for $i = n : -1 : 1$
  $x_i := \Big(b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j\Big)/a_{ii}$;
end

Die in der Gauß-Elimination entstehenden Diagonalelemente a_{jj} werden *Pivots* genannt. Der obige Algorithmus ist durchführbar sofern alle Pivots ungleich null sind.

Die markierte Zeile $(\star)$ in Algorithmus 3.3 definiert den Eliminationsschritt. Dieser ist äquivalent zu

$$A := N_{ij}(-l_{ij}) \cdot A \tag{3.9}$$

mit $l_{ij} := a_{ij}/a_{jj}$ und der Elementarmatrix N_{ij} (siehe Abschnitt 3.2) – die Summe von $(-l_{ij}) \times j$-te Zeile und i-te Zeile.

Man beachte, dass im j-ten Schritt die führenden Elemente in jeder Zeile $a_{i,1}, \ldots, a_{i,j-1}$ $(i = j, \ldots, n)$ bereits zu null umgeformt wurden durch die vorhergehenden Eliminationsschritte. Diese haben somit keine Einfluss auf das Produkt $N_{ij}(-l_{ij}) \cdot A$.

Der gesamte Algorithmus kann beschrieben werden durch ein Produkt von Elementarmatrizen: Für die resultierende obere Dreiecksmatrix R erhalten wir

$$R = N_{n,n-1}(-l_{n,n-1}) \cdot \ldots \cdot N_{21}(-l_{21}) \cdot A\,.$$

Im Detail:

$$\widetilde{A}_1 := N_{n1}(-l_{n1}) \cdot \ldots \cdot N_{21}(-l_{21}) \cdot A = \left(\begin{array}{c|ccc} \star & \star & \cdots & \star \\ \hline 0 & \star & \cdots & \star \\ \vdots & \vdots & & \vdots \\ 0 & \star & \cdots & \star \end{array}\right)$$

$$\widetilde{A}_2 := N_{n2}(-l_{n2}) \cdot \ldots \cdot N_{32}(-l_{32}) \cdot \widetilde{A}_1 = \left(\begin{array}{cc|ccc} \star & \star & \star & \cdots & \star \\ 0 & \star & \star & \cdots & \star \\ \hline 0 & 0 & \star & \cdots & \star \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \star & \cdots & \star \end{array}\right)$$

usw.

Wegen $N_{ij}(\alpha)^{-1} = N_{ij}(-\alpha)$ folgt

$$A = N_{21}(l_{21}) \cdot \ldots \cdot N_{n,n-1}(l_{n,n-1}) \cdot R\,,$$

und wir erhalten eine untere Dreiecksmatrix $L = (l_{ij})$:

$$N_{21}(l_{21}) \cdot \ldots \cdot N_{n,n-1}(l_{n,n-1}) = \left(\prod_{i>1} N_{i1}(l_{i1})\right) \cdot \ldots \cdot \left(\prod_{i>n-1} N_{i,n-1}(l_{i,n-1})\right) = L$$

Desweiteren sind alle Diagonalelemente in L gleich 1, d.h. $l_{ii} = 1$ für alle i. Eine Dreiecksmatrix mit dieser Eigenschaft nennen wir *normiert*.

Die obigen Überlegungen zeigen den folgenden Satz.

Satz 3.1: (LR-Zerlegung)
Die Gauß-Elimination (Algorithmus 3.3) liefert eine eindeutige Zerlegung $A = L \cdot R$ mit normierter unterer Dreiecksmatrix L und oberer Dreiecksmatrix R genau dann, wenn alle Pivotelemente ungleich null sind.

Es verbleibt nur die Eindeutigkeit der Zerlegung zu zeigen. Die Abfolge der Rechenoperationen im Algorithmus ist dagegen nicht eindeutig.

Die Eindeutigkeit folgt aus

$$A = L_1 R_1 = L_2 R_2 \quad \Rightarrow \quad L_2^{-1} L_1 = R_2 R_1^{-1}.$$

Die Inversen von unteren/oberen Dreiecksmatrizen sind wieder vom gleichen Typ. Das Produkt aus zwei unteren/oberen Dreiecksmatrizen ist wieder eine untere/obere Dreiecksmatrix. Zudem ist das Produkt aus zwei normierten unteren Dreiecksmatrizen wieder eine normierte Matrix. Somit ist $L_2^{-1} L_1$ eine normierte untere und $R_2 R_1^{-1}$ eine obere Dreiecksmatrix. Beide müssen daher eine Diagonalmatrix mit nur Einsen sein. Es folgt

$$L_2^{-1} L_1 = R_2 R_1^{-1} = I \quad \Rightarrow \quad L_1 = L_2,\ R_1 = R_2.$$

Der Algorithmus der LR-Zerlegung geht direkt aus dem Algorithmus 3.3 der Gauß-Elimination hervor. Die Matrizen L und R werden dabei explizit bestimmt.

Algorithmus 3.4: LR-Zerlegung (ohne Pivotsuche)

```
for j = 1 : n
        l_jj = 1 ;
        for k = j : n
                r_jk := a_jk ;
        end
        for i = j + 1 : n
                l_ij := a_ij / r_jj ;
                for k = j + 1 : n
                        a_ik := a_ik − l_ij r_jk ;
                end
        end
end
```

Pivotsuche

Bisher wurde davon ausgegangen, dass die bei der Gauß-Elimination entstehenden Pivotelemente stets ungleich null sind. Dies ist selbst bei regulären Matrizen nicht immer der Fall.

Beispiel: Bei der Matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$$

kann der Gauß-Algorithmus nicht direkt angewendet werden. Eine Zeilenvertauschung löst jedoch sofort das Problem.

Desweiteren treten Komplikationen bei kleinen Pivotelementen auf, da Rundungsfehler dann erheblich verstärkt werden.

Beispiel: Gegeben sei das LGS

$$\begin{pmatrix} 0.005 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

in 2-stelliger Arithmetik. Die exakte Lösung ist

$$x_1 = \frac{5000}{9950} = 0.503\ldots, \qquad x_2 = \frac{4950}{9950} = 0.497\ldots .$$

Der übliche Gauß-Algorithmus, wobei der erste Pivot $a_{11} = 0.005$ ist, ergibt

$$\begin{pmatrix} 0.005 & 1 \\ 0 & -200 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ -99 \end{pmatrix} \quad \Rightarrow \tilde{x}_1 = 0, \quad \tilde{x}_2 = 0.50\,.$$

Vertauschung der ersten mit der zweiten Zeile liefert dann $a_{21} = 1$ als Pivot. Es folgt

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} \quad \Rightarrow \bar{x}_1 = 0.50, \quad \bar{x}_2 = 0.50\,.$$

Das letzte Beispiel zeigt, dass ein Element mit dem betragsmäßig höchsten Wert ein guter Kandidat für den Pivot ist. Je größer der Pivot, desto kleiner sind die Einträge in L. Dadurch erreichen wir, dass die Elemente in der Matrix nicht stark anwachsen.

Unter *Pivoting/Pivotisierung* verstehen wir einen Zeilen- und/oder Spaltentausch in der (Rest-)Matrix mit dem Ziel ein betragsmäßig relativ großes Pivotelement zu erhalten. Wir unterscheiden zwei Formen von Pivotisierung:

- *Partielles Pivoting* oder auch *Spaltenpivotsuche* wählt das vom Betrag her größte Element in der aktuellen Spalte unterhalb des ursprünglichen Pivots a_{jj}. Falls dies das Element a_{ij} $(i > j)$ ist, dann ist die Zeile i mit der Zeile j zu vertauschen.
- *Totales Pivoting* oder auch *vollständige Pivotsuche* wählt das betragsmäßig größte Element in der gesamten Restmatrix. Ist dies das Element a_{ik} $(i, k \geq j)$, dann sind die Zeilen i und j sowie die Spalten j und k zu vertauschen. Der Aufwand zur Suche des größten Eintrags macht sich jedoch bemerkbar, wodurch dieses Verfahren nur selten angewendet wird.

Der folgende Satz beschreibt die LR-Zerlegung einer beliebigen regulären Matrix.

Satz 3.2: Für jede reguläre Matrix A existiert eine Zerlegung

$$P \cdot A = L \cdot R$$

mit normierter unterer Dreiecksmatrix L, oberer Dreiecksmatrix R und Permutationsmatrix P. Dabei kann P derart gewählt werden, dass alle Einträge in L vom Betrag kleiner oder gleich 1 sind, d.h. $|l_{ij}| \leq 1$.

Beweis:

Die Elimination von allen Elementen in der j-ten Spalte unterhalb des Pivots wird beschrieben durch Multiplikation mit der Matrix

$$K_j = \prod_{i>j} N_{ij}(-l_{ij}) = I + q_j \cdot {e_j}^\top \quad \text{with} \quad q_j := (0, \ldots, 0, -l_{j+1,j}, \ldots, -l_{n,j})^\top.$$

Da diese Matrizenprodukte kommutieren, folgt

$$K_j^{-1} = \prod_{i>j} N_{ij}(l_{ij}),$$

welches genau die j-te Spalte der Matrix L ergibt. Ohne Pivotisierung ist die LR-Zerlegung gegeben durch

$$R = K_{n-1} \cdot \ldots \cdot K_1 \cdot A\,, \quad \text{d.h.} \quad L \cdot R = K_1^{-1} \cdot \ldots \cdot K_{n-1}^{-1} \cdot R = A.$$

Vertauschung der Zeile i mit Zeile j kann beschrieben werden durch Multiplikation mit der Transpositionsmatrix P_{ij}. Im ersten Schritt wählt die Spaltenpivotisierung ein Element $a_{k_1,1}$

mit $|a_{k_1,1}| \geq |a_{i,1}|$ für alle i. Es folgt $a_{k_1,1} \neq 0$, da eine Spalte mit nur Nullen eine singuläre Matrix implizieren würde. Die Zeilenvertauschung und die nachfolgende Elimination kann geschrieben werden als

$$A \rightsquigarrow K_1 \cdot P_{1,k_1} \cdot A = \left(\begin{array}{c|ccc} a_{k_1,1} & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{array}\right)$$

mit der Transpositionsmatrix P_{1,k_1} und einer Matrix K_1 mit der gleichen Struktur wie oben. Die Wahl des Pivots garantiert $|l_{j,1}| = |a_{j,1}/a_{k_1,1}| \leq 1$ für $j = 2, \ldots, n$. Wegen $\det(A) = a_{k_1,1} \cdot \det(A')$ ist die verbleibende Matrix A' regulär.

Dementsprechend liefern die folgenden Eliminationsschritte

$$K_{n-1} \cdot P_{n-1,k_{n-1}} \cdot K_{n-2} \cdot P_{n-2,k_{n-2}} \cdot \ldots \cdot K_1 \cdot P_{1,k_1} \cdot A = R$$

mit oberer Dreiecksmatrix R. Das Produkt aus Transpositions- und Eliminationsmatrizen kann modifiziert werden. Mit der Abkürzung $P_i := P_{i,k_i}$ gilt für $j < i$ (beachte $P_i^2 = I$)

$$P_i K_j P_i = I + \tilde{q}_j e_j{}^\top \qquad \text{mit} \quad \tilde{q}_j = P_i q_j,$$

welches eine untere Dreiecksmatrix mit der gleichen Form wie K_j ergibt. Wegen $P_i^{-1} = P_i$ können wir erweitern

$$\begin{aligned} K_{n-1} \cdot \underbrace{(P_{n-1} \cdot K_{n-2} \cdot P_{n-1})}_{=:\tilde{K}_{n-2}} \cdot \underbrace{(P_{n-1} \cdot P_{n-2} \cdot K_{n-3} \cdot P_{n-2} \cdot P_{n-1})}_{=:\tilde{K}_{n-3}} \\ \cdot \ldots \cdot \underbrace{(P_{n-1} \cdot \ldots \cdot P_2 \cdot K_1 \cdot P_2 \cdot \ldots \cdot P_{n-1})}_{=:\tilde{K}_1} \cdot \underbrace{(P_{n-1} \cdot \ldots \cdot P_1)}_{=:P} \cdot A = R. \end{aligned}$$

Mit der Definition $\tilde{K}_i := P_{n-1} \cdots P_{i+1} K_i P_{i+1} \cdots P_{n-1}$ und $P := P_{n-1} \cdots P_1$ erhalten wir

$$K_{n-1} \cdot \tilde{K}_{n-2} \cdot \ldots \cdot \tilde{K}_1 \cdot P \cdot A = R.$$

Die Matrizen $\tilde{K}_i$ besitzen die gleiche Form wie die Matrizes K_i (nur die Elemente in der i-ten Spalte unterhalb des Pivot werden verändert). Dadurch repräsentiert

$$L := \tilde{K}_1^{-1} \cdot \ldots \cdot \tilde{K}_{n-2}^{-1} \cdot K_{n-1}^{-1}$$

eine normierte untere Dreiecksmatrix. Die Matrix P ist als Produkt von Transpositionsmatrizen eine Permutationsmatrix, welche die Zeilenvertauschungen beschreibt. □

Mit der Zerlegung $PA = LR$ lösen sich ein LGS über

$$Ax = b \quad \rightarrow \quad PAx = Pb \quad \rightarrow \quad LR = Pb$$

mit Vorwärts- und Rückwärtssubstitution sowie einer Permutation der rechten Seite.

Bei totaler Pivotisierung (Zeilen- und Spaltenvertauschungen) erhalten wir eine Zerlegung der Form

$$P \cdot A \cdot Q = L \cdot R$$

mit normierter unterer Dreiecksmatrix L, oberer Dreiecksmatrix R und Permutationsmatrizen P, Q. Ein LGS löst sich über

$$Ax = b \;\rightarrow\; PAQ(Q^\top x) = Pb \;\rightarrow\; LR(Q^\top x) = Pb \;\rightarrow\; LRz = Pb$$

mit $z := Q^\top x$. Aus der Lösung z erhalten wir durch Permutation dann $x = Qz$.

Komplexität

Wir bestimmen noch den Rechenaufwand der Algorithmen. Als eine Operation definieren wir $a \cdot b + c$. Die relativ wenigen Divisionen werden nicht gezählt. Vertauschungen von Zeilen und Spalten stellen keinen Rechenaufwand dar. (Lediglich die Suche beim totalen Pivoting kann zeitintensiv werden.)

Für die Gauß-Elimination ergibt sich der Aufwand an Operationen

$$\sum_{k=1}^{n}(k-1)k = \sum_{k=1}^{n} k^2 - k = \tfrac{1}{6}(2n^3 + 3n^2 + n) - \tfrac{1}{2}(n^2 + n) \doteq \tfrac{1}{3}n^3.$$

Bei einer Vorwärts- oder Rückwärtssubstitution erhalten wir die Anzahl

$$\sum_{k=1}^{n} k - 1 = \tfrac{1}{2}(n^2 + n) - n = \tfrac{1}{2}(n^2 - n) \doteq \tfrac{1}{2}n^2.$$

Somit stellt die Gauß-Elimination bzw. die LR-Zerlegung den Hauptanteil des Rechenaufwands dar. Jedoch steigen die Rechenkosten nur polynomial mit n im Gegensatz zur Cramerschen Regel (siehe Abschnitt 3.1). Vorwärts- und Rückwärtssubstitution sind nur so teuer wie eine Matrix-Vektor-Multiplikation.

3.4 Normen für Vektoren und Matrizen

Normen sind in der Numerik wichtig, um Größenvergleiche und Abschätzungen wie 'x ist nahe an y' oder 'Δx vernachlässigbar klein gegenüber x' sauber und zugleich pauschal über alle Komponenten zu formulieren.

Definition: *(Vektor-)Norm*

Eine Abbildung $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ (bzw. $\mathbb{C}^n \to \mathbb{R}$) heißt Norm, wenn sie die drei Eigenschaften erfüllt:

(i) definit: $x \neq 0 \Rightarrow \|x\| > 0$

(ii) homogen: $\|\lambda x\| = |\lambda| \cdot \|x\|$ für alle $\lambda \in \mathbb{R}$ bzw. $\mathbb{C}$

(iii) subadditiv: $\|x + y\| \leq \|x\| + \|y\|$ (Dreiecksungleichung)

Als Vektornormen sind gebräuchlich:

Summennorm: $$\|x\|_1 := \sum_{i=1}^{n} |x_i|$$

Euklidische Norm: $$\|x\|_2 := \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

Maximum-Norm: $$\|x\|_\infty := \max_{i=1,\ldots,n} |x_i|$$

Die Euklidische Norm ist von besonderer Bedeutung. Sie wird durch das Euklidische Skalarprodukt

$$\langle x, y\rangle := \sum_{i=1}^{n} x_i y_i = x^\top y \quad \text{bzw.} \quad \langle x, y\rangle := \sum_{i=1}^{n} \bar{x}_i y_i = x^{\mathrm{H}} y$$

induziert. Der Nachweis der Normaxiome ist in allen drei Fällen trivial bis auf die Dreiecksungleichung bei der $\|\cdot\|_2$. Hier benötigt man als Hilfsmittel die

Cauchy-Schwarzsche Ungleichung $$|x^\top y| \leq \|x\|_2 \cdot \|y\|_2 \,. \qquad (3.10)$$

Aus der Dreiecksungleichung folgt

$$\|x\| = \|(x \pm y) \mp y\| \leq \|x \pm y\| + \|y\|$$

und somit die nützliche Abschätzung

$$\|x\| - \|y\| \leq \|x \pm y\| \leq \|x\| + \|y\|.$$

Insbesondere ergibt sich daraus

$$\big|\|x\| - \|y\|\big| \leq \|x - y\|.$$

Mit Normen kann man den Abstand zweier Punkte des Vektorraums definieren und damit eine *Metrik* einführen bzw. den Raum *topologisieren* bzw. *Umgebungen* definieren. Man bezeichnet in diesem Zusammenhang die Menge

$$\{x \in \mathbb{R}^n \text{ bzw. } \mathbb{C}^n : \ \|x\|_p \leq 1\}$$

als *p-Normkugel*. Die Normkugel muss nicht besonders 'rund' sein!

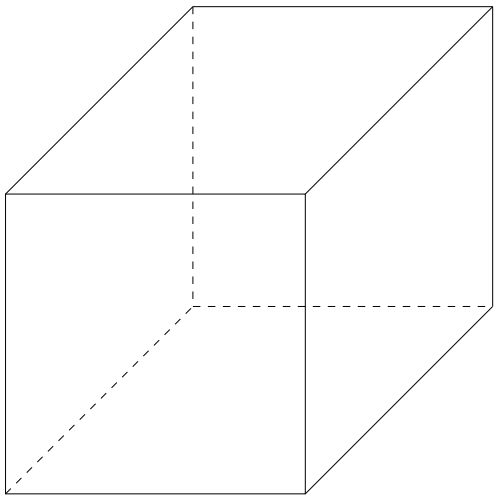


Abbildung 9: Skizze der Normkugel für $n = 3$ und $\|\cdot\|_\infty$

In endlichdimensionalen Vektorräumen sind alle Normen äquivalent, d.h. zu jeder Norm $\|\cdot\|$ existieren zwei positive Konstanten α_n, β_n, so dass

$$\alpha_n \|x\|_\infty \leq \|x\| \leq \beta_n \|x\|_\infty \quad \text{für alle } x \in \mathbb{R}^n \text{ bzw. } \mathbb{C}^n\,.$$

Wie kann man den Normbegriff für Vektoren auf lineare Abbildungen bzw. Matrizen übertragen? Zweckmäßig ist das Konzept der *Matrixnorm* und *Operatornorm*.

Definition: *Matrixnorm*

Eine Abbildung $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ (bzw. $\mathbb{C}^{m\times n} \to \mathbb{C}$) heißt *Matrixnorm*, falls die folgenden vier Eigenschaften erfüllt sind:

(i) definit: $A \neq 0 \Rightarrow \|A\| > 0$

(ii) homogen: $\|\lambda A\| = |\lambda| \cdot \|A\|$ für alle $\lambda \in \mathbb{R}$ bzw. $\mathbb{C}$

(iii) subadditiv: $\|A + B\| \leq \|A\| + \|B\|$ (Dreiecksungleichung)

(iv) submultiplikativ: $\|AB\| \leq \|A\| \cdot \|B\|$ (falls $m = n$)

Eine Matrixnorm heißt *konsistent* bezüglich der Vektornormen $\|\cdot\|_a, \|\cdot\|_b$ auf $\mathbb{R}^n$ bzw. $\mathbb{R}^m$ falls gilt

$$\|Ax\|_b \leq \|A\| \cdot \|x\|_a \quad \text{für alle } A \in \mathbb{R}^{m\times n}, \ x \in \mathbb{R}^n.$$

Ein Beispiel für eine Matrixnorm ist die *Schur-Norm*

$$\|A\|_S := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}.$$

Diese Matrixnorm ist konsistent zur Euklischen (Vektor-)Norm.

Sei nun $\Phi : V \to W$ eine lineare Abbildung (Operator) zwischen zwei normierten Vektorräumen.

Definition: *Operatornorm*

Die Operatornorm einer linearen Abbildung $\Phi : V \to W$ zwischen normierten Vektorräumen $(V, \|\cdot\|_V)$ und $(W, \|\cdot\|_W)$ ist gegeben durch

$$\|\Phi\| := \sup_{x\neq 0} \frac{\|\Phi(x)\|_W}{\|x\|_V},$$

sofern das Supremum existiert.

Wir betrachten hier ausschließlich die endlichdimensionalen Vektorräume $V = \mathbb{R}^n$ und $W = \mathbb{R}^m$. Es seien Normen $\|\cdot\|_V, \|\cdot\|_W$ gegeben. Eine lineare Abbildung $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ wird eineindeutig durch eine Matrix $A \in \mathbb{R}^{m\times n}$ beschrieben, d.h. $\Phi(x) = Ax$.

In diesem Fall ist die Operatornorm gegeben durch

$$\mathrm{lub}(A) := \max_{x\neq 0} \frac{\|Ax\|_W}{\|x\|_V}$$

(lub: least upper bound).

Es gelten die folgenden Eigenschaften:

- Die Vektornormen $\|\cdot\|_V, \|\cdot\|_W$ definieren lub auf eindeutige Weise.
- Die Abbildung $\mathrm{lub} : \mathbb{R}^{m\times n} \to \mathbb{R}$ ist eine Matrixnorm.
- Die Matrixnorm lub ist konsistent bezüglich der gegebenen Vektornormen $\|\cdot\|_V, \|\cdot\|_W$, denn für $x \neq 0$ gilt

  $$\|Ax\|_W = \frac{\|Ax\|_W}{\|x\|_V}\|x\|_V \leq \mathrm{lub}(A)\|x\|_V.$$

  Zudem ist lub die kleinste Matrixnorm mit dieser Eigenschaft.
- Alternativ hat man

  $$\mathrm{lub}(A) = \max_{\|x\|_V=1} \|Ax\|_W.$$

  Es gilt nämlich $\frac{1}{\|x\|_V}\|Ax\|_W = \|A\frac{x}{\|x\|_V}\|_W$ und $\|\frac{x}{\|x\|_V}\|_V = 1$.
  Das Maximum existiert somit (das Supremum wird zum Maximum).
- Für die Einheitsmatrix $I \in \mathbb{R}^{n\times n}$ folgt $\mathrm{lub}(I) = 1$ falls $\|\cdot\|_V = \|\cdot\|_W$.

Typische Matrixnormen, die als Operatornormen von Vektornormen induziert werden, sind

Spaltenbetragssumme: $\|A\|_1 = \max\limits_{j=1,\dots,n} \sum\limits_{i=1}^{m} |a_{ij}|$, induziert durch $\|\cdot\|_1$

Zeilenbetragssumme: $\|A\|_\infty = \max\limits_{i=1,\dots,m} \sum\limits_{j=1}^{n} |a_{ij}|$, induziert durch $\|\cdot\|_\infty$

Spektralnorm: $\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)}$, induziert durch $\|\cdot\|_2$

Die Spektralnorm ist die Wurzel aus dem maximalen Eigenwert der positiv semi-definiten symmetrischen Matrix $A^\top A$. Zur Berechnung der Spektralnorm muss man ein Eigenwertproblem lösen, was in der Praxis zu teuer ist. Die Auswertung von Zeilen- oder Spaltensummen ist dagegen einfach durchführbar. Bei bestimmten Matrizen ist die Spektralnorm aber von vornherein bekannt. Für z.B. A orthogonal hat man $\|A\|_2 = 1$ wegen $A^\top A = I$.

Definition: *Konditionszahl*

Die Konditionszahl einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ lautet

$$\kappa(A) := \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|x\|=1} \|Ax\|}$$

bezüglich einer vorgegebenen Vektornorm $\|\cdot\|$ auf $\mathbb{R}^n$.

Die Konditionszahl gibt an, wieweit die Normkugel unter der Abbildung A verzerrt wird. Es gilt stets $\kappa(A) \geq 1$ (Maximum größergleich Minimum). Im Idealfall ist $\kappa(A) = 1$ (z.B. bei Einheitsmatrix), was auch von den orthogonalen/unitären Matrizen erfüllt wird bei Verwendung der Euklidischen Norm.

Es gilt $\kappa(A) = \infty$ für singuläres A, da dann $Ax = 0$ für ein $x \neq 0$. Falls aber A^{-1} existiert, kann $\kappa(A)$ umgeformt werden, denn

$$\frac{1}{\min\limits_{\|x\|=1} \|Ax\|} = \max_{\|x\|=1} \frac{1}{\|Ax\|} = \max_{x\neq 0} \frac{\|x\|}{\|Ax\|} = \max_{y\neq 0} \frac{\|A^{-1}y\|}{\|y\|} = \mathrm{lub}(A^{-1})\,.$$

Die Konditionszahl einer quadratischen invertierbaren Matrix ist folglich durch die prägnante Formel

$$\kappa(A) = \|A\| \cdot \|A^{-1}\| \tag{3.11}$$

gegeben, wobei $\|\cdot\|$ die Matrixnorm lub bezeichnet. Desweiteren folgt die Aussage $\kappa(A^{-1}) = \kappa(A)$ wegen $(A^{-1})^{-1} = A$.

3.5 Kondition und Rundungsfehler

Ist die Lösung eines LGS ein gut konditioniertes Problem? Und wie wirken sich Rundungsfehler auf die Gauß-Elimination bzw. Dreieckszerlegung aus? Diesen Fragen wollen wir nun nachgehen.

Zunächst zur Kondition: Wir vergleichen x in $Ax = b$ mit dem gestörten Problem $(A+\Delta A)(x+\Delta x) = b+\Delta b$. Dabei sind die Abweichungen $\Delta A, \Delta b$ vorgegeben und eine Abschätzung für das resultierende Δx ist gesucht.

Annahmen: A ist regulär und die Störung ΔA klein genug, so dass $A + \Delta A$ immer noch nicht singulär ist. Das ist der Fall laut folgendem Lemma.

Lemma 3.1: Ist $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix und $\Delta A \in \mathbb{R}^{n\times n}$ mit

$$\|\Delta A\| < \frac{1}{\|A^{-1}\|}$$

in einer beliebigen Matrixnorm, die konsistent zu einer Vektornorm ist, dann ist die Matrix $A + \Delta A$ ebenfalls regulär.

Beweis: Aus $(A + \Delta A)x = 0$ folgt $x = -A^{-1}\Delta A x$ und weiter

$$\|x\| \le \|A^{-1}\| \cdot \|\Delta A\| \cdot \|x\| \quad \Rightarrow \quad (1 - \|A^{-1}\| \cdot \|\Delta A\|) \cdot \|x\| \le 0.$$

Mit $\|\Delta A\| < 1/\|A^{-1}\|$ schließt man nun auf $1 - \|A^{-1}\| \cdot \|\Delta A\| > 0$ und somit $\|x\| = 0$, d.h. $x = 0$. Also ist $A + \Delta A$ regulär. □

Mit dem Ansatz $(A + \Delta A)(x + \Delta x) = b + \Delta b$ ergibt sich

$$\Delta x = A^{-1}(\Delta b - \Delta A \cdot x - \Delta A \cdot \Delta x)$$

und in der beliebigen Vektornorm und induzierter Matrixnorm (lub)

$$\|\Delta x\| \leq \|A^{-1}\| \cdot (\|\Delta b\| + \|\Delta A\| \cdot \|x\| + \|\Delta A\| \cdot \|\Delta x\|)$$

oder äquivalent

$$(1 - \|A^{-1}\| \cdot \|\Delta A\|) \cdot \|\Delta x\| \leq \|A^{-1}\| \cdot (\|\Delta b\| + \|\Delta A\| \cdot \|x\|)\,.$$

Somit folgt für den absoluten Fehler

$$\|\Delta x\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \cdot (\|\Delta b\| + \|\Delta A\| \cdot \|x\|)\,.$$

Für den relativen Fehler erhalten wir für $x \neq 0$

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A\| \cdot \|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \cdot \left(\frac{\|\Delta b\|}{\|A\| \cdot \|x\|} + \frac{\|\Delta A\|}{\|A\|} \right).$$

Mit $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$ folgt für $b \neq 0$

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A\| \cdot \|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \cdot \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right).$$

Man kann obigen Ausdruck mithilfe der Konditionszahl $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ noch weiter umformen. Sei $\|\Delta A\| \leq \varepsilon_A \|A\|$ und $\|\Delta b\| \leq \varepsilon_b \|b\|$. Dann gilt

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \varepsilon_A \kappa(A)} \cdot (\varepsilon_A + \varepsilon_b) \tag{3.12}$$

sofern $\varepsilon_A \kappa(A) < 1$. Für $\varepsilon_A \approx \varepsilon_b \approx \varepsilon$ erhalten wir als grobe Schätzung

$$\frac{\|\Delta x\|}{\|x\|} \approx \kappa(A) \cdot \varepsilon\,, \tag{3.13}$$

wobei der Nenner in (3.12) stark vereinfacht wurde.

Die Konditionszahl $\kappa(A) \geq 1$ gibt laut (3.12) den Verstärkungsfaktor an, mit dem sich Änderungen in den Daten auf die Änderung der Lösung auswirken. Beachte: Wie immer beim Begriff der Kondition, geht es hier nicht um Rundungsfehler, sondern um eine prinzipielle Eigenschaft des Problems!

Der Anwender muss die Datenunsicherheit und die Konditionszahl abschätzen und dann entscheiden, ob $\kappa \cdot \varepsilon \ll 1$ gewährleistet ist. Nur dann macht es Sinn, x zu berechnen.

Wir nehmen nun an, dass die Eingangsdaten in A und b exakt vorliegen. Sei $\tilde{x} = x + \Delta x$ die aus dem Gauß-Algorithmus mit Rundungsfehlern behaftete Näherung. Im Sinne der Rückwärtsanalyse interpretieren wir $\tilde{x}$ als exakte Lösung des LGS $(A + \Delta A)\tilde{x} = b + \Delta b$. Dabei nehmen wir $\varepsilon_A \approx \varepsilon_b \approx \varepsilon_0$ mit der Maschinengenauigkeit $\varepsilon_0 \approx 10^{-d}$ an. Gilt $\kappa(A) \approx 10^t$, dann folgt aus (3.13) ein relativer Fehler von 10^{t-d}. Wir verlieren somit die hinteren t Dezimalstellen in der Genauigkeit.

Ist eine LR-Zerlegung $A = LR$ oder $PA = LR$ bereits berechnet, dann erhalten wir mit den Pivotelementen in R eine (sehr) grobe Schätzung der Größenordnung der Kondition von A in der Euklidischen Norm:

$$\kappa_2(A) \approx \frac{\max\limits_{j=1,\ldots,n} |r_{jj}|}{\min\limits_{i=1,\ldots,n} |r_{ii}|}.$$

Beispiel: Hilbert-Matrizen

Die Hilbert-Matrizen sind für beliebiges $n \in \mathbb{N}$ definiert durch

$$H_n \in \mathbb{R}^{n\times n}, \quad H_n = \left(\tfrac{1}{i+j-1}\right)_{i,j=1,\ldots,n} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Für ein LGS $H_n x = b$ wählen wir die rechte Seite

$$b \in \mathbb{R}^n, \quad b = \left(\sum_{i=1}^{n} \tfrac{1}{i+j+1}\right)_{j=1,\ldots,n},$$

Tabelle 2: Ergebnisse zu LGSen mit Hilbert-Matrizen.

n	$\kappa_2(H_n)$	$\frac{\max_j \lvert r_{jj}\rvert}{\min_i \lvert r_{ii}\rvert}$	$\varepsilon_0 \cdot \kappa_2(H_n)$	$\frac{\lVert\tilde{x}-x\rVert_2}{\lVert x\rVert_2}$
1	1	1	$2 \cdot 10^{-16}$	0
2	$2 \cdot 10^{1}$	$1 \cdot 10^{1}$	$4 \cdot 10^{-15}$	$6 \cdot 10^{-16}$
3	$5 \cdot 10^{2}$	$2 \cdot 10^{2}$	$1 \cdot 10^{-13}$	$8 \cdot 10^{-15}$
4	$2 \cdot 10^{4}$	$3 \cdot 10^{3}$	$3 \cdot 10^{-12}$	$1 \cdot 10^{-13}$
5	$5 \cdot 10^{5}$	$9 \cdot 10^{4}$	$1 \cdot 10^{-10}$	$3 \cdot 10^{-12}$
6	$1 \cdot 10^{7}$	$2 \cdot 10^{6}$	$3 \cdot 10^{-9}$	$3 \cdot 10^{-10}$
7	$5 \cdot 10^{8}$	$3 \cdot 10^{7}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-8}$
8	$2 \cdot 10^{10}$	$9 \cdot 10^{8}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-7}$
9	$5 \cdot 10^{11}$	$2 \cdot 10^{10}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$
10	$2 \cdot 10^{13}$	$4 \cdot 10^{11}$	$4 \cdot 10^{-3}$	$2 \cdot 10^{-4}$
11	$5 \cdot 10^{14}$	$8 \cdot 10^{12}$	$1 \cdot 10^{-1}$	$4 \cdot 10^{-3}$
12	$2 \cdot 10^{16}$	$2 \cdot 10^{14}$	$4 \cdot 10^{0}$	$8 \cdot 10^{-2}$
13	$2 \cdot 10^{18}$	$1 \cdot 10^{15}$	$4 \cdot 10^{2}$	$1 \cdot 10^{1}$
14	$4 \cdot 10^{17}$	$2 \cdot 10^{16}$	$1 \cdot 10^{2}$	$2 \cdot 10^{0}$
15	$7 \cdot 10^{17}$	$3 \cdot 10^{15}$	$2 \cdot 10^{2}$	$3 \cdot 10^{0}$

wodurch die exakte Lösung $x = (1, 1, \ldots, 1, 1)^\top$ ist. Der Gauß-Algorithmus liefert eine Näherung $\tilde{x}$ in MATLAB, die bei Rechnung mit Maschinengenauigkeit $\varepsilon_0 = 2.2 \cdot 10^{-16}$ erhalten wird. Tabelle 2 zeigt die Konditionen der ersten 15 Hilbert-Matrizen und die erzielte Genauigkeit in den LGSen. Mit steigendem n erhöht sich die Kondition erheblich. Wir erkennen, dass die Schätzwerte die tatsächlichen Verhältnisse grob widerspiegeln.

Residuum

Zu einer Näherung $\tilde{x}$ der Lösung x definiert man das *Residuum* bzw. den Rest $r := b - A\tilde{x}$. Was kann man über die Qualität von $\tilde{x}$ aussagen, wenn man r kennt?

Der Schluss "kleines Residuum $r \Rightarrow$ kleine Differenz $\|x - \tilde{x}\|$" ist falsch!

Es gilt dagegen

$$x - \tilde{x} = A^{-1}b - A^{-1}A\tilde{x} = A^{-1}(b - A\tilde{x}) = A^{-1}r.$$

Richtig ist daher die Abschätzung

$$\|x - \tilde{x}\| \leq \kappa(A) \frac{\|r\|}{\|A\|} \,. \tag{3.14}$$

Die Matrix A kann man sich in (3.14) normiert vorstellen, d.h. $\|A\| = 1$. Dann sagt der Faktor $\kappa(A)$ aus, inwiefern man vom Residuum r auf die Lösungsgüte rückschließen kann. Für große Konditionszahlen $\kappa(A)$ kann $\|x - \tilde{x}\|$ beliebig falsch sein und trotzdem $\|r\|$ klein!

Ist das Residuum deshalb gar nichts wert? Doch, denn aus $r = b - A\tilde{x}$ erhält man

$$A\tilde{x} = b - r \,,$$

d.h. $\tilde{x}$ ist exakte Lösung zur rechten Seite $b - r$. Wenn das Residuum nun in der Größenordnung der Unsicherheit Δb der rechten Seite ist, kann man $\tilde{x}$ als akzeptabel betrachten, vergl. Def. 2.6.

Es gibt noch einen weiteren Grund, warum das Residuum von Bedeutung ist, nämlich im Verfahren der *Nachiteration*, bei dem man die Näherungslösung $\tilde{x}$ noch verbessern kann, siehe unten.

Rundungsfehlereinfluss

Während die Aussagen zur Kondition eines linearen Gleichungssystems unabhängig von der tatsächlichen Realisierung auf dem Rechner sind, wollen wir nun noch kurz den Einfluss der Rundungsfehler studieren.

O.B.d.A. sei $A = LR$ die exakte Zerlegung (ohne Pivotsuche), denn man kann die Matrix A a priori so permutieren, dass keine späteren Vertauschungen notwendig sind. Unter dem Einfluss der Rundungsfehler erhält man stattdessen die Zerlegung in Faktoren $\tilde{L}$ und $\tilde{R}$. Für ein Element von L gilt die Berechnungsvorschrift

$$l_{ik} = \big(\cdots ((a_{ik} - l_{i1} r_{1k}) - l_{i2} r_{2k}) - \cdots - l_{i,k-1} r_{k-1,k} \big) / r_{kk} \,,$$

die unter der starken Hypothese (2.3) übergeht in

$$\begin{aligned}\tilde{l}_{ik} &= \big[\big(\cdots((a_{ik} - \tilde{l}_{i1}\tilde{r}_{1k}(1+\mu_1))(1+\sigma_1) - \cdots \\ &\qquad -\tilde{l}_{i,k-1}\tilde{r}_{k-1,k}(1+\mu_{k-1})\big)(1+\sigma_{k-1})/\tilde{r}_{kk}\big]\cdot(1+\delta)\end{aligned}$$

mit $|\mu_i|, |\sigma_i|, |\delta| \le \varepsilon_0$. Die Größen $\tilde{l}_{ij}, \tilde{r}_{jk}$ sind in den vorherigen Schritten ebenfalls unter dem Einfluss der Rundungsfehler berechnet worden.

Mit einer auf Sautter (1971) zurückgehenden Rückwärtsanalyse kann man nun zeigen, dass die numerische Lösung $\tilde{x}$ exakt ist zu den Daten bzw. der Koeffizientenmatrix $\tilde{A}$ und der rechten Seite b,

$$\tilde{A}\cdot\tilde{x} = b\,.$$

Für die geänderte Koeffizientenmatrix $\tilde{A}$ gilt dabei die Abschätzung

$$|\tilde{A} - A| \le |\tilde{L}|\cdot|\tilde{R}|\cdot(3\varepsilon + \varepsilon^2)\,, \qquad \varepsilon := \frac{n\varepsilon_0}{1-n\varepsilon_0}\,. \tag{3.15}$$

Der Betrag $|A|$ ist hier komponentenweise zu verstehen ($|A| = (|a_{ij}|)_{1\le i,j\le n}$).

Die Schranke (3.15) reicht nicht aus, um die Gaußelimination als per se numerisch stabilen Prozess zu bezeichnen. Dafür bräuchte man die Eigenschaft $|\tilde{L}|\cdot|\tilde{R}| \approx |A|$. Es gibt aber Zerlegungen mit

$$|\tilde{L}|\cdot|\tilde{R}| \gg |\tilde{L}\cdot\tilde{R}| \approx |L\cdot R| = |A|\,,$$

und dann liefert (3.15) eine eventuell große Abweichung $|\tilde{A} - A|$.

Das Ziel der Pivotsuche muss es also sein, eine Zerlegung zu bestimmen, bei der das Produkt $|\tilde{L}|\cdot|\tilde{R}|$ möglichst klein wird. Die Spaltenpivotsuche garantiert $|l_{ij}| \le 1$ nach Satz 3.2, für die Koeffizienten r_{ij} existiert aber nur die Schranke

$$|r_{ij}| \le 2^{n-1}a_0 \qquad \text{mit} \qquad a_0 := \max_{1\le i,j\le n}|a_{ij}|\,,$$

die meist viel zu pessimistisch ist (es gibt jedoch Matrizen, für die die Abschätzung scharf ist – Beispiel?).

Für spezielle Matrizen fällt die Abschätzung günstiger aus. Beispielsweise ergibt sich bei

$$\begin{array}{ll} \text{Tridiagonalmatrizen:} & |r_{ij}| \leq 2a_0, \\ \text{Hessenbergmatrizen:} & |r_{ij}| \leq (n-1)a_0. \end{array}$$

Bei totaler Pivotsuche sind die Schranken bei vollbesetzten Matrizen etwas besser, aber bis heute hat man keine Pivotstrategie gefunden, die $|\tilde{L}| \cdot |\tilde{R}|$ minimiert.

Die Gauß-Elimination mit eventueller Pivotsuche ist daher (streng genommen) ein instabiler Algorithmus. Die Fälle von LGSen mit ungünstigem Verhalten sind jedoch eher selten in der Praxis. Für Tridiagonalmatrizen und Hessenbergmatrizen ist der Gauß-Algorithmus stabil.

Nachiteration

Bei der Lösung eines LGS erhält man eine Näherung $\tilde{x}$ der exakten Lösung $x = A^{-1}b$, da die Zerlegung $A = LR$ bzw. $PA = LR$ mit Rundungsfehlern behaftet ist. Für die Differenz $x - \tilde{x} =: \Delta x$ gilt bei exakter Rechnung

$$A\Delta x = Ax - A\tilde{x} = b - A\tilde{x} = r$$

mit dem Residuum r. Diesen Zusammenhang macht man sich bei der *Nachiteration* zunutze. Mit ihr kann die Näherung $\tilde{x}$ verbessert werden:

Algorithmus 3.5: Nachiteration

Schritt 1: $r := b - A\tilde{x}$; Residuum in doppelter Genauigkeit

Schritt 2: Löse $A\Delta x = r$ nach Δx;

Schritt 3: $x := \tilde{x} + \Delta x$;

Der Verbesserungsschritt kann mehrfach wiederholt (iteriert) werden, und solange Δx mindestens eine korrekte Stelle hat, ist $\tilde{x} + \Delta x$ besser als $\tilde{x}$.

Schritt 1 in normaler Genauigkeit ergibt keine verwertbare Information aufgrund der Rundungsfehler, weil Auslöschung entsteht ($A\tilde{x} \approx b$). Die doppelt

genaue Ausführung dagegen liefert ein sehr präzises r, das erst am Schluss zur normalen Stellenzahl gerundet wird.

Für den Schritt 2 ist die Zerlegung $A = L \cdot R$ schon vorhanden, so dass nur Vorwärts-Rückwärtssubstitutionen anfallen. Δx weist einen relativen Fehler von $c\kappa(A)\varepsilon_0$ mit bestimmter Konstante c auf (vergl. (3.12)) und nimmt pro Iteration um diesen Faktor ab.

Als Motivation gehen wir davon aus, dass mit der rundungsfehlerbehafteten LR-Zerlegung Gleichungssysteme für beliebige rechte Seite mit einem relativen Fehler von 0.1 gelöst werden (d.h. nur erste Dezimalstelle korrekt). Die Gauß-Elimination liefet also (ohne Nachiteration)

$$x = \tilde{x} + \Delta x, \qquad \frac{\|\Delta x\|}{\|x\|} \approx 0.1.$$

Sei $\tilde{\tilde{x}}$ die Näherung aus der Nachiteration, d.h. $\tilde{\tilde{x}} = \tilde{x} + \widetilde{\Delta x}$ mit der rundungsfehlerbehafteten Lösung $\widetilde{\Delta x}$ aus dem 2. Schritt der Nachiteration. Da $\widetilde{\Delta x}$ die Lösung eine linearen Gleichungssystems zur ursprünglichen LR-Zerlegung war, folgt

$$\Delta x = \widetilde{\Delta x} + \Delta(\Delta x), \qquad \frac{\|\Delta(\Delta x)\|}{\|\Delta x\|} \approx 0.1.$$

Somit ergibt sich der Fehler

$$\frac{\|x - \tilde{\tilde{x}}\|}{\|x\|} = \frac{\|\Delta(\Delta x)\|}{\|x\|} = \frac{\|\Delta(\Delta x)\|}{\|\Delta x\|} \cdot \frac{\|\Delta x\|}{\|x\|} \approx 0.1 \cdot 0.1 = 0.01,$$

d.h. wir haben eine Dezimalstelle in der Genauigkeit gewonnen. Dieser Ansatz kann iterativ fortgeführt werden, bis kein Genauigkeitsgewinn mehr aufgrund der Kondition des Problems entsteht.

Die Nachiteration ist eine billige Möglichkeit, um die Genauigkeit der Lösung zu verbessern, verlangt aber entsprechende Hard- oder Software, um doppelt genau rechnen zu können.

Abbildung 10 illustriert den Genauigkeitsgewinn bei der Nachiteration.

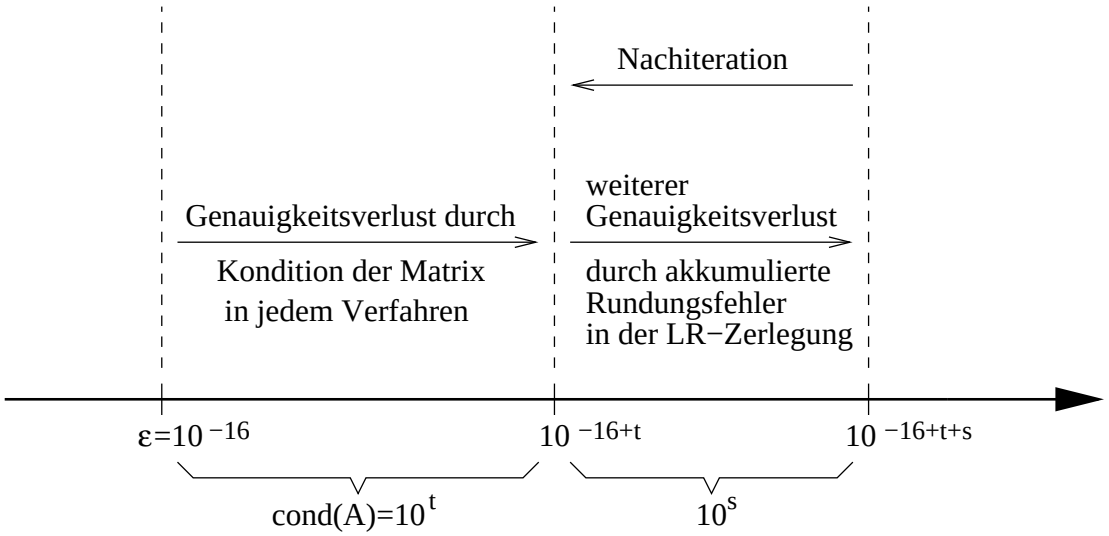


Abbildung 10: Genauigkeitsgewinn bei der Nachiteration. (Die Achse stellt die relative Genauigkeit im Ergebnis dar.)

Abschliessende Bemerkungen zur Gaußelimination:

- Die Gaußelimination bzw. LR-Zerlegung mit Spaltenpivotsuche ist *der Algorithmus der Wahl zur Lösung von LGSen.*
- Gaußelimination und LR-Zerlegung sind auch unter dem Einfluss der Rundungsfehler identische Verfahren. Sie liefern ein kleines Residuum $r = Ax - b$, das aber i.A. keine Rückschlüsse auf $\|\tilde{x} - x\|$ zulässt. Nur im Sinne der Rückwärtsanalyse liegt eine Schranke vor.
- Falls die Konditionszahl $\kappa(A)$ bekannt ist, kann man a priori die Genauigkeit der Lösung grob angeben.
  Faustregel: $\kappa(A) = 10^t \Rightarrow t$ Dezimalstellen gehen verloren.
- Zur Berechnung der Konditionszahl einer Matrix existieren spezielle Algorithmen. Am zuverlässigsten ist die *Singulärwertzerlegung*, die allerdings $O(n^3)$ Operationen erfordert.
- Eine *Vorabskalierung* $A \mapsto D_1 A D_2$ mit Diagonalmatrizen D_1, D_2 kann die Kondition verbessern. Ziel muss es dabei sei, alle Elemente von A auf ungefähr die gleiche Größenordnung zu bringen.

- Numerische Software: LINPACK (1979) war die erste Sammlung von Codes rund um die Numerische Lineare Algebra. LINPACK basiert auf BLAS Level 1 und 2. Spezielle Routinen für die LR-Zerlegung und dazugehörige Substitutionen: SGEFA, SGESL (bzw. DGEFA, DGESL für doppelte Genauigkeit). Die Routine SGECO schätzt zusätzlich die Kondition. LINPACK liegt als C- und FORTRAN Sourcecode vor.

  Das Nachfolgepaket LAPACK ist besser an die heutigen Rechnerarchitekturen angepasst, da es zusätzlich BLAS Level 3 verwendet. Routinen SGETRF, SGETRS bzw. DGETRF, DGETRS.

  Internet-Adresse: `http://www.netlib.org`

3.6 Cholesky-Zerlegung

Bei der speziellen Klasse der symmetrisch positiv definiten Matrizen, die in den Anwendungen durchaus auftreten, kann ein günstiges Verfahren zur direkten Lösung von linearen Gleichungssystemen konstruiert werden. Dabei ist insbesondere keine Pivotsuche notwendig.

Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist symmetrisch genau dann, wenn $A^\top = A$ gilt. Eine symmetrische Matrix heißt positiv definit, falls

$$x^\top Ax > 0 \qquad \text{für alle } \; x \in \mathbb{R}^n\backslash\{0\}.$$

Dies gilt genau dann, wenn alle Eigenwerte von A positiv sind. Eine positiv definite Matrix ist insbesondere regulär ($\det(A) = \lambda_1 \cdot \ldots \cdot \lambda_n$).

Beispiel: Gramsche Matrix
Wir betrachten einen reellen Hilbert-Raum $(V, \langle\cdot,\cdot\rangle)$. Zu einem Teilraum $U \subset V$ sei eine Basis $u_1, \ldots, u_n$ gegeben. Die korrespondierende Gramsche Matrix ist $A = (\langle u_i, u_j\rangle)$, d.h.

$$A = \begin{pmatrix} \langle u_1,u_1\rangle & \langle u_1,u_2\rangle & \cdots & \langle u_1,u_n\rangle \\ \langle u_2,u_1\rangle & \langle u_2,u_2\rangle & \cdots & \langle u_2,u_n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle u_n,u_1\rangle & \langle u_n,u_2\rangle & \cdots & \langle u_n,u_n\rangle \end{pmatrix}.$$

Die Symmetrie des Skalarprodukts liefert $\langle u_i, u_j\rangle = \langle u_j, u_i\rangle$, wodurch $A = A^\top$ gilt. Zudem

haben wir für $x \in \mathbb{R}^n \backslash \{0\}$ mit $x = (x_1, \ldots, x_n)^\top$

$$x^\top A x = \sum_{i,j=1}^{n} x_i x_j \langle u_i, u_j \rangle = \left\langle \sum_{i=1}^{n} x_i u_i, \sum_{j=1}^{n} x_j u_j \right\rangle = \left\| \sum_{i=1}^{n} x_i u_i \right\|_V^2 > 0,$$

d.h. die Matrix A ist positiv definit.

Wir betrachten nun den ersten Eliminationsschritt aus dem Gauß-Verfahren, angewandt auf die Matrix A. Wegen $e_1^\top A e_1 = a_{11} > 0$ für A positiv definit ist das erste Diagonalelement als Pivot akzeptabel, und man erhält

$$A = \left(\begin{array}{c|c} a_{11} & v^\top \\ \hline v & B \end{array} \right) \quad \rightsquigarrow \quad K_1 \cdot A = \left(\begin{array}{c|c} a_{11} & v^\top \\ \hline 0 & B' \end{array} \right), \quad v \in \mathbb{R}^{n-1}.$$

Die Restmatrix B' setzt sich zusammen aus

$$B' = B - \frac{1}{a_{11}} v v^\top .$$

Schema:

$$v v^\top = (v_i \cdot v_j)_{1 \le i,j \le n-1} = \left(\begin{pmatrix} v_1 \cdot v_1 & \ldots & v_1 \cdot v_{n-1} \\ \vdots & & \vdots \\ v_{n-1} \cdot v_1 & \ldots & v_{n-1} \cdot v_{n-1} \end{pmatrix} \right)$$

Damit ist B' wiederum symmetrisch. Zum Nachweis der Definitheit setzt man

$$x := \begin{pmatrix} -v^\top y / a_{11} \\ y \end{pmatrix} \quad \text{mit } y \in \mathbb{R}^{n-1} \text{ beliebig}$$

und zeigt

$$A x = \begin{pmatrix} 0 \\ B' y \end{pmatrix} \quad \Rightarrow \quad 0 < x^\top A x = y^\top B' y \, .$$

Mit Induktion für die weiteren Schritte folgt:

Satz 3.3: *Sei A symmetrisch und positiv definit. Dann ist bei der Gauß-Elimination bzw. LR-Zerlegung jede Restmatrix wiederum symmetrisch und positiv definit. Das jeweilige Diagonalelement kann als Pivot herangezogen werden.*

Man hat aber bei symmetrisch positiv definiten Matrizen nicht nur den Vorteil, keine Pivotsuche durchführen zu müssen. Die LR-Zerlegung kann hier unter Ausnutzung der Symmetrieeigenschaft mit dem halben Aufwand berechnet werden. Betrachte

$$A = L \cdot R = LDD^{-1}R\,, \quad D := \mathrm{diag}(r_{11}, \ldots, r_{nn})\,.$$

Die Matrix $\tilde{R} := D^{-1}R$ hat nur Einsen in der Diagonale, ist also normiert, und $\tilde{R}^\top$ ist demnach eine normierte untere Dreiecksmatrix. Aus der Symmetrie ergibt sich

$$LD\tilde{R} = A = A^\top = \tilde{R}^\top D L^\top\,.$$

Da die Dreieckszerlegung aber eindeutig ist (siehe Satz 3.1), muss notwendigerweise die Gleichheit $\tilde{R}^\top = L$ gelten. Somit hat man die sogenannte *rationale Cholesky-Zerlegung*

$$A = L \cdot D \cdot L^\top\,. \tag{3.16}$$

Die Diagonalfaktoren $d_i = r_{ii}$ sind positiv, da die Pivotelemente positiv sind (siehe oben). Man definiert nun die 'Wurzel'

$$D^{1/2} := \mathrm{diag}(\sqrt{r_{11}}, \ldots, \sqrt{r_{nn}})$$

aus der Diagonalmatrix D und erhält die eigentliche *Cholesky-Zerlegung*

$$A = \hat{L} \cdot \hat{L}^\top\,, \quad \hat{L} := L \cdot D^{1/2}\,. \tag{3.17}$$

Bei ihrer Berechnung kann die Symmetrie ausgenutzt werden. Man benötigt nur den halben Speicher und den halben Aufwand, d.h. $n^3/6$ Operationen.

Nun konstruieren wir einen effizienten Algorithmus zur Cholesky-Zerlegung. Es sei $\hat{L} = (\hat{\lambda}_{ij})$ und $\hat{L}^\top = (\hat{\hat{\lambda}}_{ij})$ in $A = \hat{L}\hat{L}^\top$. Wir erhalten

$$a_{ii} = \sum_{k=1}^{i} \hat{\lambda}_{ik}\,\hat{\hat{\lambda}}_{ki} \qquad \Rightarrow \qquad |\hat{\lambda}_{ii}|^2 = a_{ii} - \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2 \qquad (>0)\,,$$

$$j > i: \quad a_{ji} = \sum_{k=1}^{i} \hat{\lambda}_{jk}\,\hat{\hat{\lambda}}_{ki} \qquad \Rightarrow \qquad \hat{\lambda}_{ji} = \big(a_{ji} - \sum_{k=1}^{i-1} \hat{\lambda}_{jk}\,\hat{\lambda}_{ik}\big)/\hat{\lambda}_{ii}\,.$$

Dieser Ansatz ist durchführbar im Fall $\hat{\lambda}_{ii} > 0$ für alle i, was bei positivdefiniten Matrizen gilt. Man kann $\hat{L}$ spaltenweise ($i = 1, \ldots,\ i = n$) berechnen. Es folgt das Verfahren:

Algorithmus 3.6: Cholesky-Zerlegung

for $i = 1 : n$

$$\hat{\lambda}_{ii} := \big(a_{ii} - \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2\big)^{1/2}$$

for $j = (i+1) : n$

$$\hat{\lambda}_{ji} := \big(a_{ji} - \sum_{k=1}^{i-1} \hat{\lambda}_{jk}\,\hat{\lambda}_{ik}\big)/\hat{\lambda}_{ii}$$

end

end

Wir bestimmen die Größenordnung des Rechenaufwands. Eine Operation sei wieder $a \cdot b + c$. Die Summation in der inneren Schleife muss durch eine weitere Schleife realisiert werden. Es folgt der Aufwand

$$\begin{aligned}\sum_{i=1}^{n}(n-i)(i-1) &= n\sum_{i=1}^{n}(i-1) - \sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i \\ &= n\tfrac{1}{2}n(n-1) - \tfrac{1}{6}(2n^3+3n^2+n) + \tfrac{1}{2}n(n+1) \doteq \tfrac{1}{6}n^3.\end{aligned}$$

Der Rechenaufwand ist somit nur ungefähr $n^3/6$ Operationen gegenüber dem Aufwand $n^3/3$ in der Gauß-Elimination, da hier die Symmetrie ausgenutzt wird. Zudem spart man Speicherplatz, da nur eine Dreiecksmatrix abzulegen ist.

Aus dem Algorithmus lesen wir die Eigenschaft

$$a_{ii} = |\hat{\lambda}_{ii}|^2 + \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2 > 0$$

ab. Dadurch folgt

$$\sqrt{a_{ii}} \geq |\hat{\lambda}_{ik}| \qquad \text{für alle } \; k \leq i,$$

d.h. die Elemente in $\hat{L}$ sind klein gegenüber den Diagonalelementen in A. Dadurch hat die Cholesky-Zerlegung sehr gute Stabilitätseigenschaften. Eine Pivotsuche zur Reduzierung des Rundungsfehlereinfluss ist somit nicht notwendig.

Ein LGS $Ax = b$ löst sich dann mit der Cholesky-Zerlegung $A = \hat{L}\hat{L}^\top$ durch Vorwärtssubstitution $\hat{L}y = b$ und Rückwärtssubstitution $\hat{L}^\top x = y$, siehe Algorithmus 3.1 un 3.2.

Liegt eine negativ definite, symmetrische Matrix A vor, dann ist $-A$ positiv definit, symmetrisch und die Cholesky-Zerlegung kann ausgeführt werden. Ein LGS $Ax = b$ wird dann über $(-A)x = -b$ gelöst. Für indefinite symmetrische Matrizen existiert kein spezielles Verfahren, d.h. Gauß-Elimination ist anzuwenden.

Kapitel 4

Lineare Ausgleichsrechnung

Inhalt dieses Kapitels ist die Lösung überbestimmter linearer Gleichungssysteme nach der Methode der kleinsten Fehlerquadrate. Man spricht auch von der Ausgleichsrechnung nach Gauß (von diesem zur Minimierung von Vermessungsfehlern um 1800 erfunden) oder von *linear least squares*. Eine spezielle Zerlegung $A = Q \cdot R$ mit orthogonaler Matrix Q eignet sich hier sehr gut zur Lösung und lierfert gleichzeitig eine Alternative zur $L \cdot R$-Zerlegung bei linearen Gleichungssystemen.

4.1 Problemstellung

Ein LGS kann mehr Gleichungen als Unbekannte aufweisen und damit *überbestimmt* sein:

$$A \cdot x = b \quad \text{mit} \;\; A \in \mathbb{R}^{m \times n}, \;\; x \in \mathbb{R}^n, \;\; b \in \mathbb{R}^m, \;\; m > n \,. \tag{4.1}$$

Solche Problemstellungen treten beim Anpassen ('Fitten') von Messdaten an ein mathematisches Modell auf. x sind die unbekannten Modellparameter, b die Messungen, und A beschreibt den Zusammenhang zwischen beiden.

Beispiel: Radioaktiver Zerfall

Gegeben ist eine radioaktive Substanz (etwa aus der Medizin) bestehend aus 2 Isotopen mit Zerfallskonstanten λ_1, λ_2. Zu den Zeitpunkten t_j, $j = 1, \ldots, m$ erfolgen Messungen der Radioaktivität, die als Daten $b = (b_1, \ldots, b_m)^\top$ vorliegen.

Gesucht sind die Ausgangskonzentrationen x_1 und x_2 der beiden Isotope.

Um diese Aufgabe zu lösen, benötigt man zunächst einen *funktionalen Zusammenhang* zwischen den Zerfallswerten b_j und den Konzentrationen x_1, x_2. Im Beispiel ist dieser Zusammenhang durch Exponentialfunktionen gegeben. Eine radioaktive Substanz hat die *Zerfallskurve* $y(t) = x \exp(\lambda t)$ mit Konstante $\lambda < 0$.

Bei zwei Substanzen gilt entsprechend

$$y(t) = x_1 \exp(\lambda_1 t) + x_2 \exp(\lambda_2 t)\,.$$

Zu den Messzeitpunkten t_j erhält man damit die Beziehungen

$$y(t_j) = x_1 \exp(\lambda_1 t_j) + x_2 \exp(\lambda_2 t_j)\,, \quad j = 1, \ldots, m$$

bzw. in Matrixnotation $\bar{b} = A \cdot x$ mit

$$\bar{b} := \begin{pmatrix} y(t_1) \\ \vdots \\ y(t_m) \end{pmatrix}, \quad A := \begin{pmatrix} \exp(\lambda_1 t_1) & \exp(\lambda_2 t_1) \\ \vdots & \vdots \\ \exp(\lambda_1 t_m) & \exp(\lambda_2 t_m) \end{pmatrix}, \quad x := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Obwohl dieses System überbestimmt ist, besitzt es die Lösung x. Gegeben sind nun aber nicht die exakten Werte $\bar{b}$, sondern nur die Messungen b, von denen man auf die Ausgangskonzentrationen x rückschliessen möchte. Zu lösen ist demnach das System

$$A \cdot x = b\,,$$

das deutlich mehr Gleichungen als Unbekannte aufweist.

Aufgrund der Messfehler wird man normalerweise kein x finden, so dass $A \cdot x = b$ wirklich erfüllt wird. Stattdessen gibt man sich mit der Forderung

$$\|A \cdot x - b\| \to \min!$$

zufrieden.

Genauer liefert eine Messung der Radioaktivität eine Aussage über die Zeitableitung

$$\dot{y}(t) = x_1 \lambda_1 \exp(\lambda_1 t) + x_2 \lambda_2 \exp(\lambda_2 t).$$

Die lineare Ausgleichsrechnung verwendet in diesem Fall die Matrix

$$A := \begin{pmatrix} \lambda_1 \exp(\lambda_1 t_1) & \lambda_2 \exp(\lambda_2 t_1) \\ \vdots & \vdots \\ \lambda_1 \exp(\lambda_1 t_m) & \lambda_2 \exp(\lambda_2 t_m) \end{pmatrix}$$

und rechte Seite $\bar{b} = (\dot{y}(t_1), \ldots, \dot{y}(t_m))^\top$.

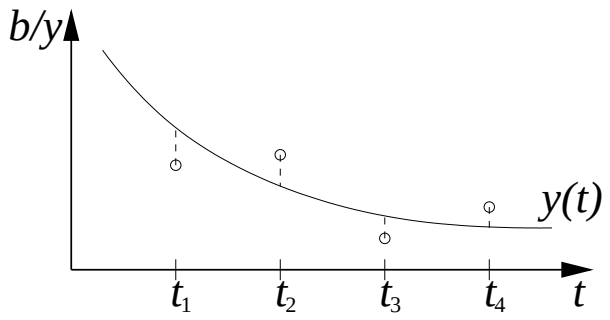


Abbildung 11: Schema der linearen Ausgleichsrechnung für eine Modellfunktion $y(t)$.

Wie am Beispiel veranschaulicht, macht es keinen Sinn, eine Lösung x zu suchen, die (4.1) exakt erfüllt. Stattdessen fordert man, dass das Residuum $r(x) := b - Ax$ in einer geeigneten Norm möglichst klein werden soll.

Die Euklidische Norm erweist sich hierfür als besonders geeignet (warum?) und führt auf das *lineare Ausgleichsproblem*

$$\text{Finde } \hat{x} \text{ mit} \quad \|r(\hat{x})\|_2 \leq \|r(x)\|_2 \quad \text{für alle } x \in \mathbb{R}^n. \tag{4.2}$$

Mit exakten (aber unbekannten) Daten $\bar{b} := Ax$ ist (4.2) äquivalent mit

$$\sum_{i=1}^{m} (\bar{b}_i - b_i)^2 \rightarrow \text{ min!} \tag{4.3}$$

Man nennt diesen Ansatz deshalb auch die Ausgleichung der Widersprüche nach der *Methode der kleinsten Quadrate.*

Die Aufgabenstellung der *linearen Ausgleichsrechnung* lautet nun zusammengefasst:

Gegeben sind

- Messdaten (t_j, b_j), $j = 1, \ldots, m$
- Ansatzfunktionen $g_i(t)$, $i = 1, \ldots, n$, $n < m$
- Funktionaler (linearer) Zusammenhang $y(t) = x_1 g_1(t) + \cdots + x_n g_n(t)$

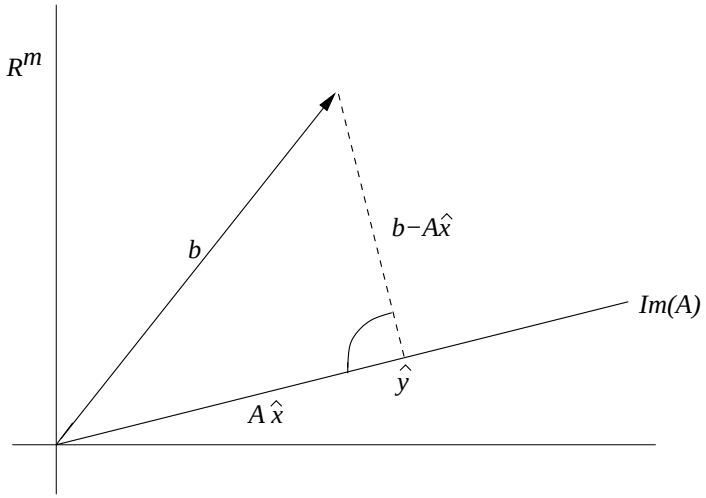


Abbildung 12: Normalgleichungen: Das kürzeste Residuum $r(\hat{x}) := b - A\hat{x}$ steht senkrecht zum Bild von A

Gesucht sind die linear auftretenden Koeffizienten $x_1, \ldots, x_n$, so dass

$$F(x_1, \ldots, x_n) := \sum_{i=1}^{m} (y(t_i) - b_i)^2 \rightarrow \text{ min!}$$

Da der funktionale Zusammenhang eine Linearkombination der Ansatzfunktionen darstellt, ist das gesamte Problem linear.

4.2 Normalgleichungen

Die Lösung des linearen Ausgleichsproblems (4.1) ist durch die *Normalgleichungen* gegeben. Zur Herleitung kann man geometrisch oder analytisch vorgehen.

Geometrischer Zugang:

Das Bild der linearen Abbildung mit A ist $\text{Im}(A) := \{Ax : x \in \mathbb{R}^n\}$. Sei $\hat{y} \in \mathbb{R}^m$ Lotfußpunkt von b auf Im A (senkrechte Projektion bezüglich des euklidischen Skalarprodukts). Dann existiert $\hat{x} \in \mathbb{R}^n$ mit $A\hat{x} = \hat{y}$. Falls A den vollen Rang n hat, ist $\hat{x}$ eindeutig. Wir fordern also $b - A\hat{x} \perp \text{Im}(A)$,

d.h.

$$\langle b - A\hat{x}, Ax\rangle = 0 \quad \text{für alle } \; x \in \mathbb{R}^n. \tag{4.4}$$

Abschätzung mit der Norm $\|\cdot\|_2$ liefert für beliebiges $x \in \mathbb{R}^n$:

$$\begin{aligned}
\|Ax - b\|^2 &= \|Ax - b + A\hat{x} - A\hat{x}\|^2 \\
&= \|A\hat{x} - b\|^2 + \|Ax - A\hat{x}\|^2 + 2\langle A\hat{x} - b, Ax - A\hat{x}\rangle \\
&= \|A\hat{x} - b\|^2 + \|Ax - A\hat{x}\|^2 + 2\langle A\hat{x} - b, A(x - \hat{x})\rangle \\
&= \|A\hat{x} - b\|^2 + \|Ax - A\hat{x}\|^2 \quad \geq \quad \|A\hat{x} - b\|^2.
\end{aligned}$$

Folglich löst $\hat{x}$ das Ausgleichsproblem. Umgekehrt kann man auch zeigen, dass eine Lösung des Ausgleichsproblems die Bedingung (4.4) erfüllt. Aus der Orthogonalität (4.4) haben wir desweiteren

$$\begin{aligned}
(Ax)^\top (b - A\hat{x}) &= 0 \\
x^\top (A^\top b - A^\top A\hat{x}) &= 0 \\
\langle x, A^\top b - A^\top A\hat{x}\rangle &= 0
\end{aligned}$$

jeweils für alle $x \in \mathbb{R}^n$. Somit muss $A^\top b - A^\top A\hat{x} = 0$ gelten. Es folgen die Normalgleichungen

$$A^\top A\hat{x} = A^\top b, \tag{4.5}$$

welche ein LGS für die Lösung $\hat{x}$ darstellen. (Beachte die Äquivalenz mit $A^\top r(\hat{x}) = 0$.) Die beteiligte Matrix $A^\top A \in \mathbb{R}^{n\times n}$ ist symmetrisch und positiv semi-definit. Für vollen Spaltenrang von A ist $A^\top A$ dann positiv definit.

Minimierungszugang:

Die Forderung (4.2) ist gleichbedeutend mit der Minimierung der Funktion $r(x)^\top r(x)$, d.h.

$$F(x) = r(x)^\top r(x) = x^\top A^\top Ax - 2x^\top A^\top b + b^\top b \;\rightarrow\; \text{min!}$$

Die notwendige Bedingung für ein Minimum der Funktion $F : \mathbb{R}^n \to \mathbb{R}$ an der Stelle $\hat{x}$ ist $\text{grad}F(\hat{x}) = 0$. Ableitung von F nach den Unbekannten x ergibt

$$\text{grad } F(x) = \left(\frac{\partial F(x)}{\partial x_1}, \ldots, \frac{\partial F(x)}{\partial x_n}\right)^\top = 2A^\top Ax - 2A^\top b\,.$$

Komponentenweise:

$$\begin{aligned}\frac{\partial F(x)}{\partial x_j} &= \frac{\partial}{\partial x_j}\left(x^\top A^\top A x + 2x^\top A^\top b\right) = e_j^\top A^\top A x + x^\top A^\top A e_j - 2e_j^\top A^\top b \\ &= 2\left(\left[A^\top A x\right]_j - \left[A^\top b\right]_j\right)\end{aligned}$$

Daraus folgt (4.5). Zudem ist die Hesse-Matrix von F gerade $2A^\top A$, d.h. eine positiv-semidefinite Matrix. Ein Extremum von F muss daher ein Minimum sein.

Die Normalgleichungen sind nicht nur notwendig, sondern auch hinreichend. Schreibe $r(x) = r(\hat{x}) + A(x - \hat{x})$, dann folgt

$$r(x)^\top r(x) = r(\hat{x})^\top r(\hat{x}) + 0 + (x - \hat{x})^\top A^\top A(x - \hat{x}) \geq r(\hat{x})^\top r(\hat{x})\,,$$

d.h., $\hat{x}$ ist ein Minimum. Gleichheit gilt nur für $A(x - \hat{x}) = 0$.

Fazit:

Satz 4.1: *Die Lösungsmengen des linearen Ausgleichsproblems (4.2) und der Normalgleichungen (4.5) sind identisch. Der Minimierer $\hat{x}$ ist genau dann eindeutig, wenn die Spalten von A linear unabhängig sind, d.h. wenn gilt $\mathrm{rang}(A) = n$. Das minimale Residuum $r(\hat{x})$ ist immer eindeutig.*

Beweis:

Wir haben bereits gesehen, dass gilt

$$A^\top A\hat{x} = A^\top b \quad \Leftrightarrow \quad \langle b - A\hat{x}, Ax\rangle = 0 \ \text{ für alle } \ x \in \mathbb{R}^n.$$

Wir zeigen nun, dass dies äquivalent ist zu

$$\|b - Ax\|^2 \geq \|b - A\hat{x}\|^2 \quad \text{für alle } \ x \in \mathbb{R}^n.$$

Die eine Richtung wurde bereits oben gezeigt. Sei daher umgekehrt die letztere Bedingung gegeben. Es folgt

$$\|Ax - A\hat{x}\|^2 + 2\langle A\hat{x} - b, A(x - \hat{x})\rangle \geq 0 \quad \text{für alle } \ x \in \mathbb{R}^n$$

und somit

$$\|Az\|^2 + 2\langle A\hat{x} - b, Az\rangle \geq 0 \quad \text{für alle } \ z \in \mathbb{R}^n.$$

Wenn $\langle A\hat{x} - b, Az^*\rangle \neq 0$ gilt für ein z^*, dann setzen wir $z = \mu z^*$ mit $\mu \in \mathbb{R}$ und es folgt

$$\mu^2\|Az^*\| + 2\mu\langle A\hat{x} - b, Az^*\rangle \geq 0.$$

Mit $\mu \to 0$ bei entsprechendem Vorzeichen wird die linke Seite negativ, d.h. ein Widerspruch ergibt sich. Somit gilt stets $\langle A\hat{x} - b, Ax\rangle = 0$ für alle x.

Zur Eindeutigkeit des Residuums seien x_1 und x_2 zwei Lösungen von (4.2). Dann erfüllen sie auch $A^\top A x_l = A^\top b$ für $l = 1, 2$. Subtraktion liefert $A^\top A(x_1 - x_2) = 0$. Damit folgt $A(x_1 - x_2) \perp \mathrm{Im}(A)$. Jedoch ist auch $A(x_1 - x_2) \in \mathrm{Im}(A)$. Somit muss $A(x_1 - x_2) = 0$ gelten, d.h. $Ax_1 = Ax_2$. Es folgt $r(x_1) = r(x_2)$. □

Im Fall des vollen Rangs, $\mathrm{rang}(A) = n$, ist $A^\top A$ positiv definit, und damit haben die Normalgleichungen immer eine eindeutige Lösung $\hat{x}$. Diesen Fall setzen wir im folgenden voraus.

Das folgende Beispiel zeigt jedoch, dass die Normalgleichungen auf einem Rechner Probleme bei der Lösung bereiten können.

Beispiel: Sei die Matrix $A \in \mathbb{R}^{4\times 3}$ gegeben durch

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}.$$

Exakte Rechnung ergibt $\mathrm{rang}(A^\top A) = 3$, da

$$A^\top A = \begin{pmatrix} 1+\varepsilon^2 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1+\varepsilon^2 \end{pmatrix}.$$

Mit $\varepsilon = 10^{-9}$ und der Machinengenauigkeit $\varepsilon_0 = 2 \cdot 10^{-16}$ folgt $\mathrm{fl}(1 + \varepsilon^2) = 1$ und damit $\mathrm{rang}(\mathrm{fl}(A^\top A)) = 1$, d.h. die Normalgleichungen sind singulär auf dem Rechner.

Daher hat sich ein anderer Weg zur Lösung des linearen Ausgleichsproblems durchgesetzt. Er basiert auf einer *orthogonalen Eliminationstechnik*.

Für eine orthogonale Matrix $Q \in \mathbb{R}^{m\times m}$ gilt, dass sie die Norm des Residuums nicht verändert:

$$\|r(x)\|_2 = \|Q \cdot r(x)\|_2 = \|Q \cdot (b - A \cdot x)\|_2 = \|Q \cdot b - Q \cdot A \cdot x\|_2.$$

(Beachte: $\|Qy\|^2 = (Qy)^\top (Qy) = y^\top Q^\top Q y = y^\top y = \|y\|^2$ f.a. y.)

Angenommen, Q kann so gewählt werden, dass $Q \cdot A$ eine reguläre obere Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$ und einen Nullblock $0 \in \mathbb{R}^{(m-n)\times n}$ ergibt,

$$Q \cdot A = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

dann kann das lineare Ausgleichsproblem neu geschrieben werden als

$$\|Q \cdot b - Q \cdot A \cdot x\|_2 = \left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2 \to \min !$$

Wegen

$$\left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2^2 = \|c - R \cdot x\|_2^2 + \|d\|_2^2$$

kann ein Minimum in $\hat{x}$ nur erreicht werden, wenn

$$\|c - R \cdot \hat{x}\|_2^2 = 0 \quad \Leftrightarrow \quad R \cdot \hat{x} = c\,.$$

Die Lösung $\hat{x}$ des Ausgleichsproblems würde dann aus einer einfachen Rückwärtssubstitution folgen.

Gibt es solch eine orthogonale Transformation, und lässt sie sich konstruktiv und stabil berechnen? Die Antwort lautet ja!

4.3 Householder-Transformation und QR-Zerlegung

Definition 4.1: *Die Matrix*

$$T := I - 2 \cdot v \cdot v^\top$$

mit $v \in \mathbb{R}^m$, $\|v\|_2 = 1$ *und Einheitsmatrix* I *heisst Spiegelungsmatrix.*

Schema für T:

$$T = \begin{pmatrix} 1 - 2v_1^2 & -2v_1v_2 & \dots & -2v_1v_{m-1} & -2v_1v_m \\ -2v_2v_1 & 1 - 2v_2^2 & \dots & -2v_2v_{m-1} & -2v_2v_m \\ & & \ddots & & \\ -2v_{m-1}v_1 & -2v_{m-1}v_2 & \dots & 1 - 2v_{m-1}^2 & -2v_{m-1}v_m \\ -2v_mv_1 & -2v_mv_2 & \dots & -2v_mv_{m-1} & 1 - 2v_m^2 \end{pmatrix}$$

Man beachte, dass $\|v\|_2 = 1$ äquivalent ist zu $v^\top v = 1$.

Einen beliebigen Vektor $x \in \mathbb{R}^m$ kann man zerlegen in $x = p + s$ mit $p := (v^\top x)v$ parallel zu v und $s := x - p$ senkrecht zu v.

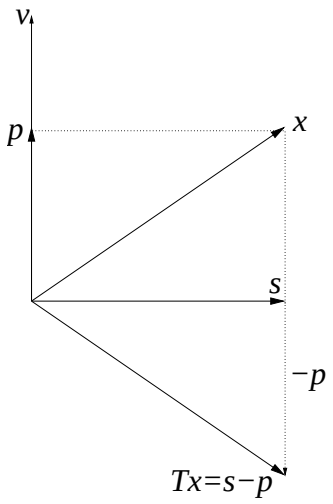


Abbildung 13: Geometrische Interpratation der Matrixspiegelung mit T.

Die Matrix T spiegelt dann den parallelen Anteil p an s,

$$T \cdot (s+p) = T \cdot (s + v(v^\top x)) = (I - 2vv^\top) \cdot (s + v(v^\top x)) = s - p\,.$$

Die Matrix T beschreibt somit die Spiegelung an der (eindeutigen) Hyperebene, die orthogonal zu v liegt.

Die Spiegelungsmatrix T hat besondere Eigenschaften:

(i) T ist symmetrisch: $T^\top = T$,

(ii) T ist involutorisch: $T^{-1} = T$,

(iii) T ist orthogonal: $T^\top T = I$.

Statt mittels eines normierten Vektors v kann man T auch durch

$$T := I - \tfrac{1}{\kappa}\, uu^\top\,, \quad \kappa := \tfrac{1}{2} u^\top u$$

mit $u \in \mathbb{R}^m \backslash \{0\}$ definieren.

Die Anwendung von T auf einen Vektor x vermöge

$$y = T \cdot x = x - \tfrac{1}{\kappa} uu^\top x = x - \tfrac{u^\top x}{\kappa} u$$

benötigt keine Matrix-Vektor-Multiplikation. Man formt zunächst das Skalarprodukt $\sigma := u^\top x/\kappa$ und dann $y := x - \sigma u$. Analog geht man bei der Anwendung auf Matrizen vor:

$$T \cdot A = A - \tfrac{1}{\kappa} u(u^\top A)\,, \quad \text{jede Spalte wie zuvor.}$$

Die Matrix A muss dabei nicht quadratisch sein.

Bei linearen Ausgleichsproblemen hat man es immer mit einem Spezialfall zu tun. T bzw. u ist so zu bestimmen, dass die Anwendung auf ein gegebenes x gerade ein Vielfaches des ersten Einheitsvektors e_1 ergibt,

$$T \cdot x = y = -\zeta e_1 = \begin{pmatrix} -\zeta \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Die folgende Wahl leistet das Gewünschte:

$$\begin{aligned} \zeta &:= \begin{cases} \|x\|_2 \cdot x_1/|x_1| & \text{für } x_1 \neq 0 \\ \|x\|_2 & \text{für } x_1 = 0 \end{cases} \\ u &:= x + \zeta e_1 = (x_1 + \zeta, x_2, \ldots, x_m)^\top \\ \kappa &:= x^\top x + \|x\|_2 \cdot |x_1| \\ T &:= I - \tfrac{1}{\kappa} u u^\top \end{aligned} \tag{4.6}$$

Man nennt die Spiegelung T dann eine *Householdertransformation.*

Satz 4.2: *Householdertransformation*
Die Householdertransformationsmatrix T aus (4.6) spiegelt x auf $Tx = -\zeta e_1 = (-\zeta, 0, \ldots, 0)^\top$.

Beweis:

Mit $u^\top x = x^\top x + \zeta x_1 = x^\top x + \|x\|_2 |x_1| = \kappa$ folgt sofort

$$Tx = x - \tfrac{1}{\kappa} u u^\top x = x - u = -\zeta e_1\,. \qquad \square$$

Durch eine Sequenz von Householdertransformationen kann man nun eine Matrix A auf obere Dreiecksform bringen.

Schritt 1: Als x in (4.6) wähle die erste Spalte der Matrix $A \in \mathbb{R}^{m\times n}$ und bilde T_1. Dann ist

$$T_1 A = \left(\begin{array}{c|c|c|c} -\zeta_1 & \star & \cdots & \star \\ \hline 0 & & & \\ \vdots & a_2^{(1)} & \cdots & a_n^{(1)} \\ 0 & & & \end{array}\right).$$

Die erste Spalte hat sich reduziert, alle anderen wurden umgerechnet.

Schritt 2: Wende die Householdertransformation der Dimension $m-1$ auf die 1. Spalte der Restmatrix an.

Dieser Schritt lässt sich als Transformation mit einer Householdertransformation $T_2 \in \mathbb{R}^{m\times m}$ schreiben, wobei die erste Komponente von u gleich null ist. Daher läßt sich T_2 auch schreiben als

$$T_2 = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & T_2' & \\ 0 & & & \end{array}\right) \qquad \text{mit} \quad T_2' \in \mathbb{R}^{(m-1)\times(m-1)}$$

und T_2' beschreibt eine Housholdertransformation in $\mathbb{R}^{m-1}$. Die Anwendung von T_2 lässt dann die erste Zeile und die erste Spalte von $T_1 A$ unverändert:

$$T_2 T_1 A = \left(\begin{array}{c|c|c|c|c} -\zeta_1 & \star & \star & \cdots & \star \\ \hline 0 & -\zeta_2 & \star & \cdots & \star \\ \hline 0 & 0 & & & \\ \vdots & \vdots & a_3^{(2)} & \cdots & a_n^{(2)} \\ 0 & 0 & & & \end{array}\right).$$

Nach n Schritten sind dann alle Spalten reduziert, und A ist in eine obere Dreiecksmatrix und einen Nullblock überführt worden:

$$T_n \cdot \ldots \cdot T_1 \cdot A = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \text{bzw.} \quad A = T_1 \cdot \ldots \cdot T_n \begin{pmatrix} R \\ 0 \end{pmatrix} = Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Mit $Q := T_1 \cdot \ldots \cdot T_n$ hat man so die *QR-Zerlegung* der Matrix A gewonnen.

Das Produkt $Q = T_1 \cdot \ldots \cdot T_n$ der Householdertransformationen ist ebenfalls eine orthogonale Matrix, d.h. $Q^\top Q = I$.

Damit ist das oben formulierte Ziel erreicht, das lineare Ausgleichsproblem kann mittels der QR-Zerlegung äquivalent umformuliert und gelöst werden (beachte $Q^\top$ statt Q):

$$\begin{aligned} \|b - Ax\|_2^2 &= \|Q^\top b - Q^\top Ax\|_2^2 \\ &= \left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2^2 \\ &= \|c - R \cdot x\|_2^2 + \|d\|_2^2 \end{aligned}$$

Die Lösung $\hat{x}$ des Ausgleichsproblems folgt dann durch Rücksubstitution $R\hat{x} = c$. Es ist R regulär genau dann, wenn $\operatorname{rang}(A) = n$ gilt. Desweiteren ist die Norm des Residuums gerade $\|r(\hat{x})\|_2 = \|d\|_2$, welche man aus der Transformation des Vektors b sofort erhält.

Bei der Implementierung ist darauf zu achten, dass die Matrizen T_j nicht explizit aufgestellt werden, sondern nur ihre Wirkung auf A berechnet wird. Im j-ten Schritt wird T_j auf $A^{(j-1)} = T_{j-1} \ldots T_1 A$ und $b^{(j-1)} = T_{j-1} \ldots T_1 b$ angewendet. Da $T_j = I - u_j u_j^\top / \kappa_j$, ergibt sich als Rechenschritt

$$A^{(j-1)} \rightsquigarrow A^{(j)} = T_j A^{(j-1)} = A^{(j-1)} - u_j y_j^\top \,, \quad y_j^\top := \tfrac{1}{\kappa_j} u_j^\top A^{(j-1)} \,.$$

Aufwand: Im j-ten Schritt benötigt man

$(n - j + 1)(m - j + 1)$ Additionen/Multiplikationen für $u_j^\top A^{(j-1)}$
$(n - j + 1)(m - j + 1)$ Additionen/Multiplikationen für $A^{(j-1)} - u_j y_j^\top$

(die ersten $j - 1$ Komponenten von u_j sind null!).

Insgesamt ergibt sich als Aufwand

$$\begin{aligned}\sum_{j=1}^{n} 2(n-j+1)(m-j+1) &\doteq 2\sum_{j=1}^{n} mn-(m+n)j+j^2 \\ &\doteq 2\left(mn^2-(m+n)\frac{n^2}{2}+\frac{n^3}{3}\right) \\ &= mn^2-\frac{n^3}{3}\end{aligned}$$

Operationen. Für $m \gg n$ ist das etwa doppelt soviel wie bei dem Weg über die Cholesky-Zerlegung und die Normalgleichungen (Aufwand für Matrix-Matrix-Mult. $A^\top A$ ist $\frac{1}{2}mn^2$ wegen Symmetrie, Cholesky-Zerlegung erfordert $\frac{1}{6}n^3$). Im Spezialfall $m = n$ folgt ein Aufwand proportional zu $\frac{2}{3}n^3$, d.h. doppelt so hoch wie im Gauß-Algorithmus.

Beispiel: Gegeben ist die Matrix

$$A = \begin{pmatrix} -4 & 0 \\ 0 & 1 \\ 3 & 1 \end{pmatrix}.$$

Für den ersten Transformationsschritt folgt

$$\zeta_1 = -5, \quad u_1 = \begin{pmatrix} -4 \\ 0 \\ 3 \end{pmatrix} + (-5)\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -9 \\ 0 \\ 3 \end{pmatrix}, \quad \kappa_1 = 25+20 = 45,$$

$$T_1 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} - \frac{1}{45}\cdot 3 \begin{pmatrix} -9 \\ 0 \\ 3 \end{pmatrix} = \begin{pmatrix} \frac{3}{5} \\ 1 \\ \frac{4}{5} \end{pmatrix}.$$

Die transformierte Matrix ist daher

$$A^{(1)} = \begin{pmatrix} 5 & \frac{3}{5} \\ 0 & 1 \\ 0 & \frac{4}{5} \end{pmatrix}.$$

Die zweite Transformation operiert auf dem Teilvektor $\tilde{x} = (1, \frac{4}{5})^\top$. Es folgt

$$\zeta_2 = \frac{\sqrt{41}}{5}, \quad u_2 = \begin{pmatrix} 0 \\ 1 \\ \frac{4}{5} \end{pmatrix} + \frac{\sqrt{41}}{5}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1+\frac{\sqrt{41}}{5} \\ \frac{4}{5} \end{pmatrix}, \quad \kappa_2 = \frac{41}{25}+\frac{\sqrt{41}}{5}.$$

Das Ergebnis lautet daher

$$\begin{pmatrix} R \\ 0 \end{pmatrix} = A^{(2)} = \begin{pmatrix} 5 & \frac{3}{5} \\ 0 & -\frac{\sqrt{41}}{5} \\ 0 & 0 \end{pmatrix}.$$

Bemerkungen:

- Im Sonderfall $m = n$ ist die QR-Zerlegung eine Alternative zur LR-Zerlegung (der Nullblock verschwindet dann). Falls A dann Vollrang hat, liefert die QR-Zerlegung die Lösung des linearen Gleichungssystems $Ax = b$. Der Aufwand ist mit $\frac{2}{3}n^3$ Operationen (Multiplikationen) ungefähr doppelt so gross wie bei der LR-Zerlegung. Aus Stabilitätsgründen benutzt man dennoch manchmal die QR-Zerlegung.

- Für positive Diagonalelemente $r_{ii} > 0, i = 1, \ldots, n$, ist die QR-Zerlegung eindeutig.

- Pivotsuche: Businger/Golub haben 1965 eine Technik für den rangdefizienten Fall eingeführt, d.h. bei Auftreten einer Nullspalte in der ersten Spalte der Restmatrix. Man sucht dann nach der Spalte mit der größten 2-Norm und bringt sie durch Spaltenvertauschung an die aktuelle Stelle (wird missverständlich auch als Spaltenpivoting bezeichnet). Statt $A = Q \cdot R$ hat man nun $A \cdot P = Q \cdot R$ mit einer Permutationsmatrix P. Durch dieses Vorgehen sind die Diagonalelemente r_{ii} der Größe nach absteigend geordnet und ermöglichen eine Konditionsabschätzung über das Verhältnis r_{11}/r_{nn} (siehe Golub/van Loan, Matrix Computations, John Hopkins University Press 1989).

- Statt über Householdertransformationen kann man die QR-Zerlegung auch über ebene Rotationen (Givens-Rotationen) berechnen.

4.4 Kondition des linearen Ausgleichsproblems

Wir betrachten ein lineares Ausgleichsproblem $\|b-Ax\|_2 \to \min$. mit Matrix $A \in \mathbb{R}^{m\times n}$ ($\operatorname{rang}(A) = n$) und Vektor $b \in \mathbb{R}^n$. Diese Eingangsdaten ändern wir in $A + \Delta A$, $b + \Delta b$ ab und möchten die resultierende Abweichung Δx abschätzen.

Die Lösung x des linearen Ausgleichsproblems ist die eindeutige Lösung der Normalgleichungen $A^\top Ax = A^\top b$. Die Lösung $x + \Delta x$ des gestörten Problems erfüllt daher die Normalgleichungen

$$(A + \Delta A)^\top (A + \Delta A)(x + \Delta x) = (A + \Delta A)^\top (b + \Delta b).$$

Dabei nehmen wir ΔA als hinreichend klein an, so dass $\operatorname{rang}(A + \Delta A) = n$ gilt. Dadurch können wir umformen zu

$$x + \Delta x = ((A + \Delta A)^\top (A + \Delta A))^{-1} (A + \Delta A)^\top (b + \Delta b).$$

Für die inverse Matrix erhalten wir in erster Näherung (Linearsierung)

$$\begin{aligned}
& ((A + \Delta A)^\top (A + \Delta A))^{-1} \\
= \;& (A^\top A + \Delta A^\top A + A^\top \Delta A + \Delta A^\top \Delta A)^{-1} \\
\doteq \;& (A^\top A + \Delta A^\top A + A^\top \Delta A)^{-1} \\
= \;& (A^\top A(I + (A^\top A)^{-1}(\Delta A^\top A + A^\top \Delta A)))^{-1} \\
= \;& (I + (A^\top A)^{-1}(\Delta A^\top A + A^\top \Delta A))^{-1} (A^\top A)^{-1} \\
\doteq \;& (I - (A^\top A)^{-1}(\Delta A^\top A + A^\top \Delta A))(A^\top A)^{-1}.
\end{aligned}$$

Es gilt nämlich $(I + F)^{-1} \doteq I - F$ für kleine Matrizen F, da

$$(I + F)(I - F) = I - F^2 \doteq I.$$

Desweiteren formen wir um

$$(A + \Delta A)^\top (b + \Delta b) = A^\top b + \Delta A^\top b + A\Delta b + \Delta A \Delta b \doteq A^\top b + \Delta A^\top b + A\Delta b.$$

Somit erhalten wir

$$\begin{aligned}
x + \Delta x \;\doteq\; & (A^\top A)^{-1} A^\top b - (A^\top A)^{-1}(\Delta A^\top A + A^\top \Delta A)(A^\top A)^{-1} A^\top b \\
& + (A^\top A)^{-1} \Delta A^\top b + (A^\top A)^{-1} A^\top \Delta b.
\end{aligned}$$

Ersetzen von $x = (A^\top A)^{-1}A^\top b$ und $r := b - Ax$ liefert

$$\Delta x \doteq -(A^\top A)^{-1}A^\top \Delta A x + (A^\top A)^{-1}\Delta A^\top r + (A^\top A)^{-1}A^\top \Delta b.$$

Die von der Euklidischen Vektornorm induzierte Matrixnorm ist die Spektralnorm. Wir schätzen diesbezüglich ab

$$\begin{aligned}\|\Delta x\| \;\dot{\leq}\; & \|(A^\top A)^{-1}A^\top\| \cdot \|A\| \cdot \frac{\|\Delta A\|}{\|A\|} \cdot \|x\| \\ & + \|(A^\top A)^{-1}\| \cdot \|A^\top\| \cdot \|A\| \cdot \frac{\|\Delta A^\top\|}{\|A^\top\|} \cdot \frac{\|r\|}{\|Ax\|} \cdot \|x\| \\ & + \|(A^\top A)^{-1}A^\top\| \cdot \|A\| \cdot \frac{\|b\|}{\|Ax\|} \cdot \frac{\|\Delta b\|}{\|b\|} \cdot \|x\|.\end{aligned}$$

Man beachte, dass aus $\operatorname{rang}(A) = n$ folgt $A \neq 0$. Wir fordern $b \neq 0$, da sich sonst sofort die triviale Lösung $x = 0$ ergibt. Wegen $\operatorname{rang}(A) = n$ folgt auch $Ax \neq 0$ für $x \neq 0$. Wir nehmen $x \neq 0$ an, obwohl $x = 0$ auftreten könnte.

Aus einer QR-Zerlegung von A

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad Q^\top A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

erhalten wir mit $A^\top = (R^\top\ 0)Q^\top$ folgende Aussagen

$$\begin{aligned} A^\top A &= R^\top R, \\ (A^\top A)^{-1} &= R^{-1}(R^\top)^{-1}, \\ (A^\top A)^{-1}A^\top &= (R^{-1}\ 0)Q^\top. \end{aligned}$$

Dadurch gilt insbesondere $\|A\|_2 = \|R\|_2$.

Folgende Eigenschaften der Spektralnorm werden (ohne Beweis) verwendet:

(i) $\|C\|_2 = \|C^\top\|_2$ für jede Matrix $C \in \mathbb{R}^{m\times n}$,

(ii) $\|QC\|_2 = \|C\|_2$ für $C \in \mathbb{R}^{m\times n}$ und orthogonales $Q \in \mathbb{R}^{m\times m}$,

(iii) $\|CP\|_2 = \|C\|_2$ für $C \in \mathbb{R}^{m\times n}$ und orthogonales $P \in \mathbb{R}^{n\times n}$.

Für die Störungen in den Eingangsdaten nehmen wir an

$$\frac{\|\Delta A\|}{\|A\|} \leq \varepsilon_A, \qquad \frac{\|\Delta b\|}{\|b\|} \leq \varepsilon_b.$$

Insgesamt erhalten wir

$$\frac{\|\Delta x\|}{\|x\|} \dot{\leq} \kappa(R) \cdot \varepsilon_A + \kappa(R)^2 \cdot \frac{\|r\|}{\|Ax\|} \cdot \varepsilon_A + \kappa(R) \cdot \frac{\|b\|}{\|Ax\|} \cdot \varepsilon_b.$$

Wir können einen Winkel einführen durch

$$\tan\varphi = \frac{\|r\|}{\|Ax\|} \qquad \text{für } 0 \leq \varphi < \tfrac{\pi}{2}.$$

Dann gilt $\cos\varphi = \|Ax\|/\|b\|$. Also folgt

$$\frac{\|\Delta x\|}{\|x\|} \dot{\leq} \left[\kappa(R) + \kappa(R)^2 \tan\varphi\right] \varepsilon_A + \kappa(R)\sqrt{1+\tan^2\varphi}\,\varepsilon_b.$$

Dieses Ergebnis stellt eine obere Schranke der relativen Abweichung für kleine Störungen dar. Jedoch kann diese obere Grenze in den Anwendungen durchaus erreicht werden.

Wir erkennen, dass ein Verstärkungsfaktor $\kappa(R)^2$ auftritt. Zudem kann im Fall $\|Ax\| \ll \|r\|$ die relative Abweichung sehr hoch werden. Somit verhalten sich lineare Ausgleichsprobleme kritischer als lineare Gleichungssysteme. Die Kondition eines linearen Gleichungssystems hängt nur von der Matrix ab. Die Kondition eines linearen Ausgleichproblems ist abhängig von sowohl der Matrix A als auch dem Vektor b und der daraus resultierenden Lösung x, da diese $\tan\varphi$ bestimmen.

Die Kondition bezieht sich auf die Lösung des Problems und nicht auf einen bestimmten Algorithmus für die Aufgabenstellung. Als Algorithmen haben wir zum einen die Lösung der Normalgleichungen über Cholesky-Zerlegung und zum anderen die QR-Zerlegung kennengelernt. Zur Beurteilung der Stabilität der beiden Algorithmen ist eine Rundungsfehleranalyse notwendig. Dabei gehen Ergebnisse aus der Kondition des Problems in die Untersuchung ein. Es zeigt sich, dass im allgemeinen die QR-Zerlegung sich

ungefähr gleich gut oder deutlich besser als der Weg über die Normalgleichungen verhält. Daher verwenden wir die QR-Zerlegung in der Praxis.

Eine weitere Möglichkeit zur stabilen Lösung des linearen Ausgleichsproblems bietet die *Pseudoinverse*. Diese wird aus der Singulärwertzerlegung der Matrix A konstruiert und ist daher aufwändiger als eine QR-Zerlegung.

Bemerkung zu Linearen Gleichungssystemen:

Ist ein LGS $Ax = b$ mit regulärer Matrix $A \in \mathbb{R}^{n\times n}$ gegeben, dann erhält man durch Multiplikation mit $A^\top$ sofort die Normalgleichungen

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

Die Matrix $A^\top A \in \mathbb{R}^{n\times n}$ ist symmetrisch und positiv definit. Man könnte daher die Lösung x durch Cholesky-Zerlegung bestimmen, was nur einen Aufwand von ca. $\frac{n^3}{6}$ Operationen erfordert (Gauß-Algorithmus für $Ax = b$ benötigt $\frac{n^3}{3}$ Operationen). Jedoch ist die Matrix $A^\top A$ a priori zu bestimmen. Diese Matrix-Matrix Multiplikationen erfordert $\frac{n^3}{2}$ Operationen wegen der Symmetrie. Insgesamt ergibt sich der Aufwand

$$\frac{n^3}{2} + \frac{n^3}{6} = \frac{2}{3}n^3,$$

d.h. doppelt so hoch wie in der Gauß-Elimination des ursprünglichen Problems. Desweiteren kann man zeigen, dass in der Spektralnorm gilt

$$\kappa_2(A^\top A) = \|A^\top A\|_2 \cdot \|(A^\top A)^{-1}\|_2 = \|A\|_2^2 \cdot \|A^{-1}\|_2^2 = \kappa_2(A)^2.$$

Daher ist zu erwarten, dass sich in einem Algorithmus zur Lösung der Normalgleichungen die Rundungsfehler mehr verstärken.

Kapitel 5

Polynominterpolation

Dieses Kapitel behandelt numerische Algorithmen zur *Interpolation* von gegebenen Daten mittels *Polynomen*. Es zeigt sich jedoch, dass die Polynominterpolation nur bei kleinen Datenmengen sinnvoll ist.

5.1 Interpolation und Approximation

Gegeben seien Daten

$$(x_i, y_i) \quad \text{für} \quad i = 0, 1, \ldots, n$$

mit *Stützstellen* $x_i \in [a, b] \subset \mathbb{R}$ und *Stützwerten* $y_i \in \mathbb{R}$. Wir nehmen $x_i \neq x_j$ für $i \neq j$ an.

Interpolationsaufgabe: Gesucht ist eine Funktion $f : [a, b] \to \mathbb{R}$, die durch alle Stützpunkte (x_i, y_i) verläuft, sie also interpoliert:

$$f(x_i) = y_i \qquad \text{für alle} \quad i = 0, 1, \ldots, n.$$

Diese Aufgabe ist noch unscharf formuliert. Eindeutig wird sie erst dann, wenn man den Raum der als Interpolanten zugelassenen Funktionen f geeignet einschränkt. Die Wahl dieses Funktionenraums definiert auch die Eigenschaften des Interpolanten. Gewünschte Eigenschaften sind etwa:

- *Shape preserving (formerhaltend)*: Eigenschaften der Daten wie
  - Positivität (0. Ableitung),
  - Monotonie (1. Ableitung),
  - Konvexität (2. Ableitung) bzw. Krümmung bei Kurven

  sollen sich im Interpolanten widerspiegeln. Insbesondere sollen keine zusätzlichen Extremwerte oder Oszillationen erzeugt werden. Die interpolierende Funktion (der *Interpolant*) soll die Daten dann möglichst "schön" (*visually pleasing*) wiedergeben. Eng damit verbunden ist die Forderung nach
- *Glattheit des Interpolanten*: In der Praxis fordert man global $f \in C^1$ oder $f \in C^2$, selten sogar $f \in C^3$.

Derartige Interpolationsaufgaben finden sich in so praktischen Anwendungen wie der Textverarbeitung (skalierbare Buchstaben in TEX) und Nähmaschinen (Parallelnähte, Stickmuster). Hier haben sich stückweise Polynome geringen Grades mit globaler C^1- sowie C^2-Eigenschaft bestens bewährt. Näheres hierzu in in Kapitel 6.

Eng verwandt mit der Interpolationsaufgabe ist die Approximationsaufgabe.

Approximationsaufgabe: Hier nimmt man an, dass die Daten Wertepaare einer *unbekannten* (genügend glatten!) Funktion repräsentieren, also z.B. aus diskreten Auswertungen einer gegebenen Funktion g stammen, d.h. $y_i = g(x_i)$. Ziel ist es nun, eine Funktion (Kurve) f zu konstruieren, die diese in den Stützstellen interpoliert

$$f(x_i) = y_i \, (= g(x_i) \,) \quad \text{für} \quad i = 0, 1, \ldots, n$$

und überall sonst möglichst gut approximiert:

$$|g(x) - f(x)| \quad \text{"klein"} \quad \text{für alle} \quad a \leq x \leq b.$$

Auch diese Aufgabenstellung ist mit einer gewissen Wilkür behaftet: Welche Klasse von Funktionen wollen wir als Approximationsfunktion f zulassen? In welcher Norm soll f die Funktion g bestmöglich approximieren?

Beide Aufgabenstellungen (Interpolation und Approximation) können zu eng gefasst sein, da die interpolierende/approximierende Funktion nicht genau durch die Stützwerte gehen soll. Denn häufig stammen die Werte y_i aus mit Messfehlern behafteten Messungen. Abhilfe kann hier das Glätten von Daten (*smoothing*) liefern: Die Kurve soll in einem (zu definierenden!) Sinne die Messfehler in den Datenpunkten berücksichtigen.

Techniken der Interpolation und Approximation sind auch von zentraler Bedeuting im Bereich des CAD (*Computer Aided Design*). Beispiele hierfür sind Geometriedesign (z.B. Konstruktion optimaler Turbinenschaufeln) und das Zeichnen von Kurven (z.B. Corel Draw).

Für uns wichtig ist die einfache Realisierung der Interpolation bzw. Approximation auf dem Rechner. Die Berechnung/Auswertung des Interpolanten soll dabei möglichst mit elementaren Grundoperationen und schnell erfolgen.

5.2 Grundlagen der Polynominterpolation

Bei der Polynominterpolation soll ein geeignetes Polynom die Interpolationsaufgabe lösen. Motivation hierfür ist:

- Polynome stellen Terme in den Grundrechenarten dar (d.h. es sind keine Auswertungen von höheren Funktionen notwendig).
- Polynome können numerisch stabil ausgewertet werden (z.B. Horner-Schema falls Koeffizienten gegeben sind).
- Polynome sind beliebig oft differenzierbar (Funktionenraum C^∞).

Die Polynome vom Grad (höchstens) n bilden einen Vektorraum $\mathbb{P}_n$ der Dimension $n+1$, der zum Beispiel von der *Taylorbasis* (Monome)

$$\mathbb{P}_n := \text{span}\,\{1, x, x^2, \ldots, x^{n-1}, x^n\}$$

aufgespannt wird. Jedes Element $p \in \mathbb{P}_n$ wird durch $n+1$ Koeffizienten festgelegt, und damit wird man bei $n+1$ Datenpunkten annehmen, dass

ein eindeutiger Interpolant aus $\mathbb{P}_n$ existiert.

Etwas genauer: Seien x_i, $i = 0, \ldots, n$, die *verschiedenen* Stützstellen und y_i die zugehörigen Stützwerte. Ein $p \in \mathbb{P}_n$ hat die allgemeine Darstellung

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \,,$$

woraus nach Einsetzen der Interpolationsforderung $p(x_i) = y_i$ die $n+1$ Gleichungen

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1 \\ \vdots \qquad\qquad & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n \end{aligned} \tag{5.1}$$

folgen. Die Matrix zu diesem LGS heißt *Vandermonde Matrix*. Das LGS in den Unbekannten $a_0, a_1, \ldots, a_n$ ist eindeutig lösbar, falls die *Vandermondsche Determinante* ungleich null ist, welche lautet

$$\begin{vmatrix} 1 & x_0 & \ldots & x_0^n \\ 1 & x_1 & \ldots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \ldots & x_n^n \end{vmatrix} = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_j - x_i).$$

Für paarweise verschiedene Stützstellen $x_i \neq x_j$ ist die Determinante somit ungleich null. Wir erhalten folgendes Resultat.

Satz 5.1: *Zu den $n+1$ Stützpunkten (x_i, y_i) mit paarweise verschiedenen Stützstellen (Knoten) x_i existiert genau ein Interpolationspolynom $p \in \mathbb{P}_n$, d.h. $p(x_i) = y_i$ für $i = 0, 1, \ldots, n$.*

Das LGS (5.1) liefert eine Berechnungsvorschrift für die Koeffizienten a_i, allerdings mit einem Aufwand von $(n+1)^3/3$ Operationen. In der Praxis wird man daher andere Methoden zur Konstruktion/Auswertung des Interpolanten vorziehen, siehe Abschnitte 5.3 und 5.4.

Paarweise verschiedene Stützstellen nennt man auch *einfach*. Falls dagegen $x_{i-1} < x_i = \cdots = x_{i+m-1} < x_{i+m}$, so heisst x_i *m-fache Stützstelle*. In einer

m-fachen Stützstelle fordert man die Interpolation der Ableitungen

$$y_{i+k} := \frac{\partial^k}{\partial x^k} f(x_i), \quad k = 0, \ldots, m-1$$

bis zum Grad $m-1$. Auch für diese Interpolationsaufgabe kann man die Existenz eines eindeutigen Polynominterpolanten zeigen.

5.3 Interpolationsformel nach Lagrange

Der Ansatz über die Vandermondsche Matrix basiert auf der Taylorbasis. Die Taylorbasis ist unabhängig von der Wahl der Stützstellen x_i. Die Idee ist daher, die Basispolynome in Abhängigkeit der Stützstellen zu wählen um eine einfachere Darstellung zu erhalten.

Die *Lagrangepolynome* $L_i \in \mathbb{P}_n$ werden definiert als die eindeutig bestimmten Interpolationspolynome zu den Einheits-Stützwerten, d.h.

$$L_i(x_j) = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{für } i \neq j \end{cases} \tag{5.2}$$

für $i, j = 0, 1, \ldots, n$ Die Lagrangepolynome besitzen die Formel

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{5.3}$$

für $i = 0, 1, \ldots, n$. Die Eigenschaft (5.2) erkennt man an der Struktur

$$L_i(x) = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}.$$

Es gilt $\mathbb{P}_n = \text{span}\{L_0, L_1, \ldots, L_n\}$. Mithilfe der L_i kann dann das Interpolationspolynom explizit angegeben werden:

Satz 5.2: *Das Interpolationspolynom zu den $n+1$ Stützpunkten (x_i, y_i) hat die Darstellung*

$$p(x) = \sum_{i=0}^{n} y_i L_i(x)\,. \tag{5.4}$$

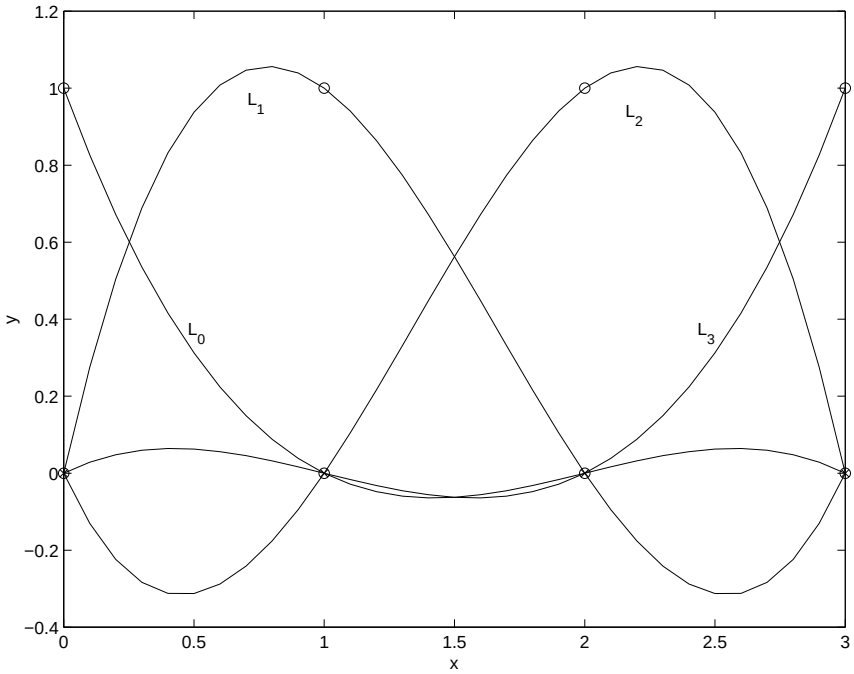


Abbildung 14: Lagrange-Polynome für $n = 3$.

Beweis: Mit der Eigenschaft der Lagrange-Polynome folgt sofort

$$p(x_j) = \sum_{i=0}^{n} y_i L_i(x_j) = y_j \qquad \text{für } \; j = 0, 1, \ldots, n$$

und damit ist p das eindeutige Interpolationspolynom. □

Die Darstellung (5.4) kann im übrigen auch für mehrfache Stützstellen hergeleitet werden, mit allerdings leicht veränderten Basispolynomen L_i.

Für praktische Zwecke ist der Zugang nach Lagrange zu aufwendig. Für theoretische Analysen ist er jedoch oft vorteilhaft. Ein Beispiel ist die Frage nach der Kondition der Interpolationsaufgabe, siehe Abschnitt 5.6.

Ein Nachteil der Interpolation mit Lagrange-Polynomen ist, dass die Polynombasis nicht direkt erweitert werden kann: Sind die Lagrange-Polynome zu den Stützstellen (x_i, y_i) für $i = 0, 1, \ldots, n$ gegeben, so ändern sich bei Hinzufügen einer weiteren Stützstelle (x_{n+1}, y_{n+1}) alle Basispolynome.

5.4 Aitken-Neville-Schema und Dividierte Differenzen

In diesem Abschnitt werden die beiden gebräuchlichsten Algorithmen zur Polynominterpolation vorgestellt. Ist man nur an der Auswertung an einer Stelle x interessiert, so ist das Schema nach Aitken-Neville die Methode der Wahl. Bei mehreren Auswertungen ist es dagegen sinnvoll, den Interpolanten nach Newton über sogenannte dividierte Differenzen darzustellen. Beide Vorgehensweisen sind rekursiv.

Aitken-Neville-Schema

Das Interpolationspolynom p zu den Daten $(x_0, y_0), \ldots, (x_n, y_n)$ bezeichnen wir im folgenden etwas konkreter als $p_{0,n}$. Es gilt $p_{0,n} \in \mathbb{P}_n$.

Das Polynom $p_{0,n-1}$ interpoliert dann die Punkte $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ und $p_{1,n}$ die Punkte $(x_1, y_1), \ldots, (x_n, y_n)$. Es gilt $p_{0,n-1}, p_{1,n} \in \mathbb{P}_n$.

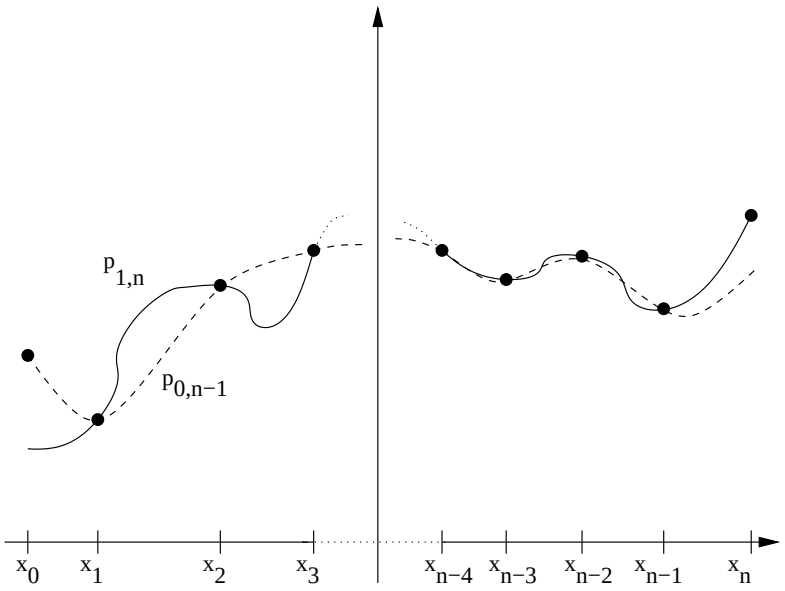


Abbildung 15: Skizze zu $p_{0,n-1}$ ($\cdots$) und $p_{1,n}$ (—).

Zwischen $p_{0,n}$, $p_{0,n-1}$ und $p_{1,n}$ besteht ein wichtiger Zusammenhang. Sei

$$\phi(x) := \frac{(x - x_0)p_{1,n}(x) - (x - x_n)p_{0,n-1}(x)}{x_n - x_0}\,.$$

Das Polynom ϕ interpoliert die Stützpunkte $(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$, d.h.

$$\phi(x_i) = y_i\,, \quad i = 1, \ldots, n-1.$$

Darüberhinaus gilt $\phi(x_0) = y_0$ und $\phi(x_n) = y_n$.

Insgesamt interpoliert ϕ die Punkte $(x_0, y_0), \ldots, (x_n, y_n)$ und stimmt somit aufgrund der Eindeutigkeit der Polynominterpolation mit $p_{0,n}$ überein.

Das ergibt

Lemma 5.3: *Lemma von Aitken*

Für das Interpolationspolynom $p_{0,n}$ gilt die Rekursionsformel

$$p_{0,n}(x) = \frac{(x - x_0)p_{1,n}(x) - (x - x_n)p_{0,n-1}(x)}{x_n - x_0}\,. \tag{5.5}$$

Mit (5.5) steht eine rekursive Vorschrift zur Auswertung von $p_{0,n}(x)$ zur Verfügung (gilt auch für mehrfache Stützstellen).

Notationswechsel: Man setzt

$$P_{ik} := p_{i-k,i}(x)\,,$$

damit ist z.B. $P_{nn} = p_{0,n}(x)$ und $P_{n,0} = p_{nn}(x)$. Man beachte, dass das Polynom $p_{i-k,i}$ bei den $k+1$ Stützstellen $x_{i-k}, x_{i-k+1}, \ldots, x_i$ interpoliert.

Die diskreten Werte $P_{i,k}$ berechnet man nach dem Schema von Neville gemäß folgender rekursiver Vorschrift:

Algorithmus 5.1: *Aitken-Neville-Schema*

für $i = 0 : n$
    $P_{i,0} := y_i$
für $k = 1 : n$
    für $i = k : n$

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

Der Algorithmus baut Spalte für Spalte ein dreieckiges Schema der Zwischenwerte $P_{i,k}$ auf:

$$\begin{array}{ccccccc}
P_{00} & & & & & & \\
 & \searrow & & & & & \\
P_{10} & \to & P_{11} & & & & \\
\vdots & & & \ddots & & & \\
P_{n-1,0} & \to & \cdots & \to & P_{n-1,n-1} & & \\
 & & & & & \searrow & \\
P_{n0} & \to & \cdots & \to & P_{n,n-1} & \to & P_{n,n}
\end{array}$$

Das Ergebnis $P_{nn} = p_{0,n}(x)$ steht schliesslich rechts unten.

Beispiel: Gegeben sind die Stützstellen $(0, 1), (1, 3), (3, 2)$. Wir werten das Interpolationspolynom an der Stelle $x = 2$ aus. Das Schema liefert:

$$\begin{array}{llll}
x_0 = 0 & y_0 = P_{00} = 1 & & \\
x_1 = 1 & y_1 = P_{10} = 3 & P_{11} = 3 + \frac{2-1}{1-0}(3-1) = 5 & \\
x_2 = 3 & y_2 = P_{20} = 2 & P_{21} = 2 + \frac{2-3}{3-1}(2-3) = \frac{5}{2} & P_{22} = \frac{5}{2} + \frac{2-3}{3-0}(\frac{5}{2} - 5) = \frac{10}{3}
\end{array}$$

Das Aitken-Neville-Schema benötigt einen Rechenaufwand an Additionen und Multiplikationen, der proportional zu n^2 ist. Bei geschickter Programmierung ist nur ein Speicher der Länge n notwendig.

Bei der numerischen Quadratur spielt eine Variante des Schemas eine grosse Rolle. Man spricht dort von sogenannten *Extrapolationsverfahren*, die ein gewisses Polynom an der Stelle 0 ausserhalb des Intervalls der Stützstellen auswerten.

Dividierte Differenzen

Das Lemma 5.3 stellt eine Rekursion für *Polynome* dar, wird aber im Aitken-Neville-Schema nur *punktweise* verwendet. Eine analoge Vorgehensweise für das Interpolationspolynom geht auf Newton zurück.

Als Basispolynome von $\mathbb{P}_n$ führen wir ein

$$\omega_i(x) := \prod_{j=0}^{i-1}(x - x_j)\,. \tag{5.6}$$

Offensichtlich ist $\omega_0(x) = 1$ ein Polynom vom Grad null und
$\omega_n(x) = (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1})$ ein Polynom vom Grad n.

Insgesamt: $\mathbb{P}_n = \text{span}\,\{\omega_0, \ldots, \omega_n\}$. Man bezeichnet $\{\omega_0, \ldots, \omega_n\}$ auch als sogenannte *Newton-Basis*.

In der Newton-Basis schreibt sich das Interpolationspolynom als

$$p_{0,n}(x) = a_0\omega_0(x) + a_1\omega_1(x) + \cdots + a_n\omega_n(x)$$

mit führendem Koeffizienten a_n, der aufgrund der Eindeutigkeit der Interpolationsaufgabe eindeutig festgelegt ist.

Man definiert nun die *n-te dividierte Differenz* zu

$$[x_0, x_1, \ldots, x_n]y := a_n\,. \tag{5.7}$$

Die Berechnung von $[x_0, x_1, \ldots, x_n]y$ kann rekursiv erfolgen, dazu kommen wir gleich. Zunächst halten wir fest, dass sich die Koeffizienten des gesamten Interpolanten aus dividierten Differenzen aufbauen lassen.

Satz 5.4: *Zu den Stützstellen* $(x_0, y_0), \ldots, (x_n, y_n)$ *besitzt das Interpolationspolynom die Darstellung*

$$p_{0,n}(x) = \sum_{i=0}^{n}[x_0, \ldots, x_i]y \cdot \omega_i(x)\,. \tag{5.8}$$

Beweis:

Mit Induktion über n: Für $n = 0$ ist die Aussage trivial. Sei also $n > 0$ und

$$p_{0,n-1} = \sum_{i=0}^{n-1} [x_0, \ldots, x_i] y \cdot \omega_i(x)$$

das Interpolationspolynom von y an $x_0, \ldots, x_{n-1}$. Dann gilt für das Interpolationspolynom $p_{0,n}$ von y an $x_0, \ldots, x_n$, dass

$$p_{0,n}(x) = [x_0, \ldots, x_n] y \cdot \omega_n(x) + Q_{n-1}(x)$$

mit einem Polynom $Q_{n-1} \in \mathbb{P}_{n-1}$. Nun erfüllt aber

$$Q_{n-1}(x) = p_{0,n}(x) - [x_0, \ldots, x_n] y \cdot \omega_n(x)$$

offensichtlich die Interpolationsbedingung für $x_0, \ldots, x_{n-1}$ wegen den Nullstellen $\omega_n(x_i) = 0$ für $i = 0, 1, \ldots, n-1$, so dass

$$Q_{n-1}(x) = p_{0,n-1}(x) = \sum_{i=0}^{n-1} [x_0, \ldots, x_i] y \cdot \omega_i(x).$$

Daraus folgt die Behauptung. □

Die Darstellung (5.8) eignet sich damit hervorragend, um den Interpolanten um zusätzliche Stützpunkte zu erweitern. Man startet mit (x_0, y_0) und fügt dann immer weitere Stützpunkte hinzu.

Die Berechnung der dividierten Differenzen selbst kann auf ähnliche Weise wie in Lemma 5.3 erfolgen. Dazu betrachte

$$p_{0,n}(x) = [x_0, \ldots, x_n] y \cdot \omega_n(x) + [x_0, \ldots, x_{n-1}] y \cdot \omega_{n-1}(x) + \cdots .$$

Nach (5.5) ist aber

$$\begin{aligned} p_{0,n}(x) &= \frac{(x - x_0) p_{1,n}(x) - (x - x_n) p_{0,n-1}(x)}{x_n - x_0} \\ &= \frac{[x_1, \ldots, x_n] y \cdot x^n + \cdots - [x_0, \ldots, x_{n-1}] y \cdot x^n + \cdots}{x_n - x_0} . \end{aligned}$$

Koeffizientenvergleich zum führenden Term x^n links und rechts ergibt die *Rekursionsformel*

$$[x_0, \ldots, x_n] y = \frac{[x_1, \ldots, x_n] y - [x_0, \ldots, x_{n-1}] y}{x_n - x_0} . \tag{5.9}$$

Man kann (5.9) sogar noch etwas allgemeiner formulieren (ohne Beweis, vgl. Lemmata 7.8 und 7.11 in Deuflhard/Hohmann):

Satz 5.5: *Dividierte Differenzen*

Die dividierte Differenz $[x_0, \ldots, x_n]y$ *genügt für* $x_i \neq x_j$ *der Rekursionsformel*

$$[x_0, \ldots, x_n]y = \frac{[x_0, \ldots, \hat{x}_i, \ldots, x_n]y - [x_0, \ldots, \hat{x}_j, \ldots, x_n]y}{x_j - x_i}\,, \qquad (5.10)$$

wobei das Symbol $\widehat{\ }$ *anzeigt, dass der entspechende Knoten weggelassen wird (seinen Hut nehmen muss).*

Zur Bestimmung der dividierten Differenzen kann man damit wie beim Schema von Aitken-Neville vorgehen.

Algorithmus 5.2: *Dividierte Differenzen*

für $i = 0 : n$
    $[x_i]y := y_i$
für $k = 1 : n$
    für $i = k : n$
    $$[x_{i-k}, \ldots, x_i]y := \frac{[x_{i-k+1}, \ldots, x_i]y - [x_{i-k}, \ldots, x_{i-1}]y}{x_i - x_{i-k}}$$

Schema:

$$\begin{array}{rcccccc}
y_0 = [x_0]y & & & & & & \\
 & \searrow & & & & & \\
y_1 = [x_1]y & \to & [x_0, x_1]y & & & & \\
\vdots & & & \ddots & & & \\
y_{n-1} = [x_{n-1}]y & \to & \cdots & \to & [x_0, \ldots, x_{n-1}]y & & \\
 & & & & & \searrow & \\
y_n = [x_n] & \to & \cdots & \to & [x_1, \ldots, x_n]y & \to & [x_0, \ldots, x_n]y
\end{array}$$

Beispiel: Wir berechnen das Interpolationspolynom zu den Werten

x_i	0	1	2	3
y_i	1	2	0	1

mit Hilfe der Newtonschen dividierten Differenzen:

$$\begin{array}{llll} [x_0]y = \underline{1} & & & \\ [x_1]y = 2 & [x_0,x_1]y = \underline{1} & & \\ [x_2]y = 0 & [x_1,x_2]y = -2 & [x_0,x_1,x_2]y = \underline{-\frac{3}{2}} & \\ [x_3]y = 1 & [x_2,x_3]y = 1 & [x_1,x_2,x_3]y = \underline{\frac{3}{2}} & [x_0,x_1,x_2,x_3]y = \underline{1} \end{array}$$

Das Interpolationspolynom ist daher

$$\begin{aligned} p_{0,3}(x) &= 1 + 1(x-0) + (-\tfrac{3}{2})(x-0)(x-1) + 1(x-0)(x-1)(x-2) \\ &= x^3 - 4.5x^2 + 4.5x + 1. \end{aligned}$$

Die Auswertung des Interpolanten erfolgt zweckmäßigerweise mit einer Modifikation des Hornerschemas (Algorithmus 2.1 aus Kapitel 2):

$$p_{0,n}(x) = a_0 + (x-x_0)\big(a_1 + (x-x_1)\big(a_2 + (x-x_2)\big(\cdots(x-x_{n-1})a_n\big)\big)\big).$$

Komplexität Berechnung dividierte Differenzen: const. $\cdot\, n^2$

Auswertung mit Horner: const. $\cdot\, n$

Die dividierten Differenzen haben eine Reihe wichtiger Eigenschaften:

- $[x_0,\ldots,x_n]y$ ist symmetrisch in allen Stützstellen, die Reihenfolge in der Liste $x_0,\ldots,x_n$ spielt keine Rolle.
- $[x_0,\ldots,x_n]y$ ist eine Linearkombination der Stützwerte und kann als Differenzenquotient interpretiert werden (bei einfachen Stützstellen), etwa mit einer Funktion $y : \mathbb{R} \to \mathbb{R}$ und $x_1 = x_0 + h$:

  $$[x_0,x_1]y = \frac{y(x_1)-y(x_0)}{x_1-x_0} = \frac{y(x_0+h)-y(x_0)}{h}$$

  und somit (falls y differenzierbar ist)

  $$\lim_{h\to 0}[x_0,x_0+h]y = y'(x_0).$$

- Produktregel analog zur Leibnizregel für Ableitungen:

$$[x_0, \ldots, x_n](y \cdot z) = \sum_{k=0}^{n} [x_0, \ldots, x_k]y \cdot [x_k, \ldots, x_n]z.$$

5.5 Erweiterter Mittelwertsatz und Restgliedformel

Gehen wir nun zur Diskussion der Approximationsaufgabe über. Hierzu verknüpfen wir die Datenpunkte (x_i, y_i) mit einer Funktion f, d.h. es gilt $y_i := f(x_i)$ oder kurz $y_i = f_i$.

Wie gut approximiert der Interpolant die Funktion f? Diese Frage wird im folgenden über die Darstellung mit dividierten Differenzen beantwortet.

Zunächst sei $f \in C^n[x_0, x_n]$ und $p = p_{0,n}$ dessen Interpolant in den Stützpunkten $(x_0, f_0), \ldots, (x_n, f_n)$, wobei $x_0 < x_1 < \cdots < x_n$.

Das *Restglied* ist die Differenz $f - p$, eine Funktion, die ebenfalls n mal stetig differenzierbar ist. Es gilt mit dem Satz von Rolle:

$f - p$ hat mindestens $n + 1$ Nullstellen;
$D(f - p) = f' - p'$ hat mindestens n Nullstellen;
$\vdots$
$D^n(f - p)$ hat mindestens eine Nullstelle, genannt ξ.

In der Stelle ξ ist

$$\begin{aligned} D^n f(\xi) &= D^n p(\xi) \\ &= D^n([x_0]f \cdot \omega_0(\xi) + \cdots + [x_0, \ldots, x_n]f \cdot \omega_n(\xi)) \\ &= [x_0, \ldots, x_n]f \cdot D^n \omega_n(\xi) \\ &= [x_0, \ldots, x_n]f \cdot n! \,. \end{aligned}$$

Die letzte Umformung folgt dabei aus

$$\omega_n(x) = (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1}) = x^n + \cdots .$$

Insgesamt erhält man damit den folgenden Satz.

Satz 5.6: *Erweiterter Mittelwertsatz*

Für $f \in C^n[x_0, x_n]$ existiert ein $\xi \in [x_0, x_n]$ mit

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

Der Satz gilt natürlich auch für eine Teilmenge $x_{i-k}, \ldots, x_i$ von Stützstellen mit entsprechend geringerer Differenzierbarkeitsordnung k, d.h.

$$[x_{i-k}, \ldots, x_i]f = \frac{D^k f(\xi)}{k!}.$$

Der Fall $k = 1$ liefert den üblichen Mittelwertsatz, denn mit $x := x_{i-1}$ und $h := x_i - x_{i-1}$ ergibt sich

$$\frac{f(x+h) - f(x)}{h} = f'(\xi) \quad \text{für ein} \quad \xi \in [x, x+h].$$

Im folgenden sei f sogar $n+1$ mal stetig differenzierbar und $\bar{x}$ eine beliebige zusätzliche Stützstelle. Das Polynom $q \in \mathbb{P}_{n+1}$ interpoliere f in den Stützstellen $x_0, x_1, \ldots, x_n, \bar{x}$.

Nach der Newtonschen Interpolationsformel (5.8) gilt

$$q(x) = p(x) + [x_0, \ldots, x_n, \bar{x}]f \cdot \omega_{n+1}(x)$$

und damit an der Stelle $\bar{x}$

$$f(\bar{x}) - p(\bar{x}) = q(\bar{x}) - p(\bar{x}) = [x_0, \ldots, x_n, \bar{x}]f \cdot \omega_{n+1}(\bar{x})\,.$$

Die dividierte Differenz $[x_0, \ldots, x_n, \bar{x}]f$ kann nach dem erweiterten Mittelwertsatz (Satz 5.6) durch die $(n+1)$-te Ableitung von f ausgedrückt werden.

Mit Umbenennung $\bar{x} \mapsto x$ folgt schliesslich die Restgliedformel.

Satz 5.7: *Restgliedformel*

Für $f \in C^{n+1}[a,b]$ existiert ein $\xi \in [a,b]$ mit

$$f(x) - p(x) = \omega_{n+1}(x)\,\frac{D^{n+1}f(\xi)}{(n+1)!}\,,$$

wobei $a := \min(x_0,\ldots,x_n,x)$ und $b := \max(x_0,\ldots,x_n,x)$. Die Zwischenstelle ξ hängt von x ab.

Mit diesem Satz hat man eine aussagekräftige Darstellung des Approximationsfehlers bei der Polynominterpolation an der Hand (siehe Übung).

Das übliche Vorgehen für eine globale Abschätzung des Approximationsfehlers ist wie folgt: Mit der Restgliedformel hat man

$$\max_{x\in[a,b]} |f(x)-p(x)| \le \frac{1}{(n+1)!}\left(\max_{x\in[a,b]}|\omega_{n+1}(x)|\right)\cdot\left(\max_{\xi\in[a,b]}\left|D^{n+1}f(\xi)\right|\right).$$

Um diese obere Schranke zu berechnen, ist $|D^{n+1}f|$ zu bestimmen und auf $[a,b]$ abzuschätzen. Desweiteren ist $|\omega_{n+1}|$ abzuschätzen. Für $n \le 2$ kann dies explizit durch eine Kurvendiskussion von ω_{n+1} erfolgen. Allgemein gilt die (sehr) grobe Abschätzung

$$|\omega_{n+1}(x)| = \prod_{i=0}^{n}|x-x_i| \le (b-a)^{n+1}$$

falls $a = x_0 < x_1 < \cdots < x_n = b$ und $x \in [a,b]$. Feinere Abschätzungen sind möglich, wenn x auf ein bestimmtes Teilintervall von $[a,b]$ eingeschränkt wird.

Es stellt sich die Frage nach einer optimalen Wahl der Stützstellen in einem Intervall $[a,b]$ um den Approximationsfehler klein zu halten. Für allgemeine Funktion f führt dies auf das MinMax-Problem

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

Dies ist äquivalent zur Bestimmung eines Polynoms in $\mathbb{P}_n$ mit führendem Koeffizient eins und minimalem Absolutbetrag. O.B.d.A. betrachten wir

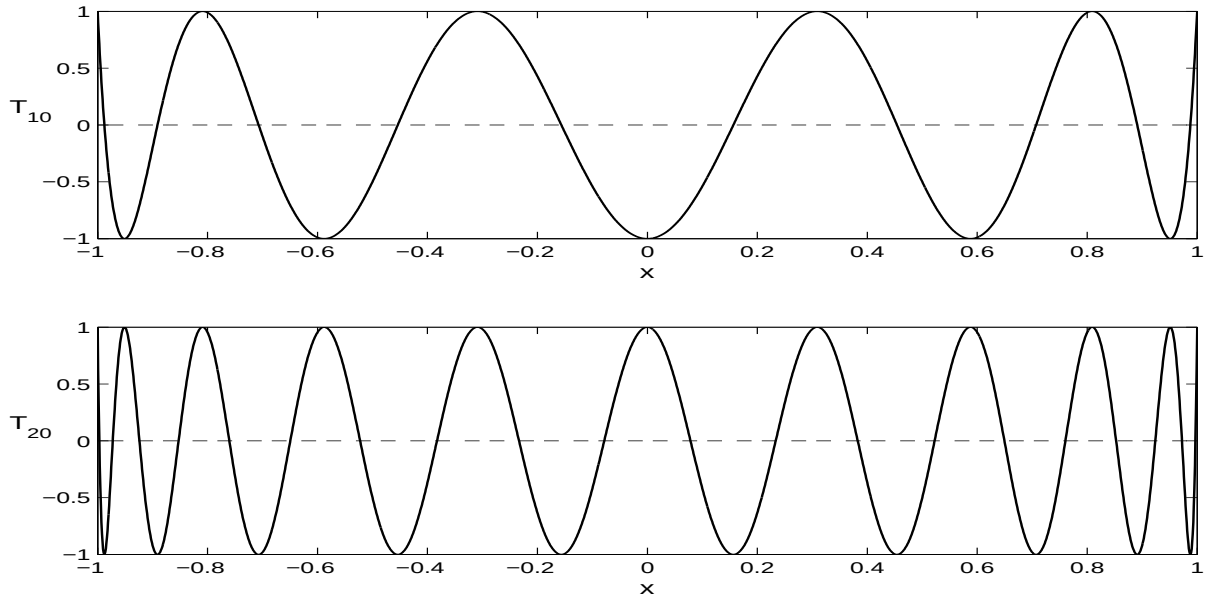


Abbildung 16: Tschebyscheff-Polynome T_{10} und T_{20} als Beispiel.

das Intervall $[-1, 1]$ (affin-lineare Transformation des Intervalls $[a, b]$). Das MinMax-Problem wird von dem Tschebyscheff-Polynom T_{n+1} gelöst (bis auf ein skalares Vielfaches). Die Definition der Tschebyscheff-Polynome lautet

$$T_n(x) = \cos(n \arccos(x)) \qquad \text{für } n = 0, 1, 2, \ldots$$

mit $x \in [-1, 1]$ und sie genügen der Rekursion

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad \text{für } n \geq 2$$

mit Startwerten $T_0(x) = 1$ und $T_1(x) = x$. Diese Formel gilt für beliebiges $x \in \mathbb{R}$. Das Polynom T_{n+1} besitzt $n + 1$ Nullstellen

$$x_i = \cos\left(\frac{2i+1}{n+1} \cdot \frac{\pi}{2}\right) \qquad \text{für } i = 0, 1, \ldots, n.$$

Die Wahl dieser Stützstellen liefert das gesuchte Minimum. Diese Stützstellen nennt man auch *Tschebyscheff-Knoten* oder *Tschebyscheff-Abszissen*.

Satz 5.7 gilt, wie auch die übrigen Aussagen zu den dividierten Differenzen, ebenfalls im Fall mehrfacher Stützstellen.

Der Approximationssatz von Weierstraß besagt, dass jede stetige Funktion auf einem kompakten Intervall gleichmäßig beliebig genau durch ein Polynom approximiert werden kann. Genauer: Zu $f \in C[a,b]$ und $\varepsilon > 0$ existiert ein Polynom p (Grad nicht spezifiziert) mit

$$\max_{x\in[a,b]} |f(x) - p(x)| < \varepsilon.$$

Um f durch Interpolationspolynome zu approximieren sei eine Folge von Zerlegungen

$$\Delta_m = \left\{ x_j^{(m)} : a \le x_0^{(m)} < x_1^{(m)} < \cdots < x_{n_m-1}^{(m)} < x_{n_m}^{(m)} \le b \right\}$$

gegeben. Diese Folge wird als immer feiner werdend vorausgesetzt, d.h.

$$\lim_{m\to\infty} \max\{|a - x_0^{(m)}|, |x_1^{(m)} - x_0^{(m)}|, \ldots, |x_{n_m}^{(m)} - x_{n_m-1}^{(m)}|, |b - x_{n_m}^{(m)}|\} = 0.$$

Man könnte vermuten, dass dann das korrespondierende Interpolationspolynom gleichmäßig gegen f konvergiert, zumindest für eine gute Wahl der Stützstellen (z.B. Tschebyscheff-Knoten). Der Satz von Faber zeigt jedoch, dass dies nicht der Fall ist. Zu jeder Folge von immer feiner werdenden Stützstellen Δ_m existiert eine Funktion $f \in C[a,b]$, so dass die Interpolationspolynome nicht gleichmäßig gegen f konvergieren.

Zu jeder gegebenen Funktion $f \in C[a,b]$ existieren individuelle Folgen von Zerlegungen Δ_m, so dass die Interpolationspolynome gleichmäßig gegen f konvergieren. Jedoch gibt es kein Konstruktionsprinzip für diese individuellen Zerlegungen. Eine universelle Zerlegung Δ_m, welche die Konvergenz für beliebiges $f \in C[a,b]$ garantiert, existiert nach dem Satz von Faber nicht. Daher sind andere Interpolationsmethoden zur Approximation vorzuziehen, siehe Kapitel 7.

5.6 Kondition der Interpolationsaufgabe

Das Interpolationspolynom soll an der Stelle x ausgewertet werden, d.h.

$$y := p(x)\,.$$

Wie hängt y von den Daten (x_i, y_i), $i = 0, 1, \ldots, n$, sowie x ab? Als Konditionszahlen treten auf, vergleiche Lagrange-Interpolation (5.4),

$$\frac{\partial y}{\partial x} = p'(x), \qquad \frac{\partial y}{\partial y_i} = L_i(x), \qquad \frac{\partial y}{\partial x_i} = -p'(x_i) \cdot L_i(x).$$

Die ersten beiden Ableitungen sind klar. Um die dritte einzusehen verschiebt man (x_i, y_i) *auf* p nach $(\widetilde{x}_i, \widetilde{y}_i) := (x_i + \varepsilon, p(x_i + \varepsilon))$. Es sei p_ε das Interpolationspolynom zu $(\widetilde{x}_i, \widetilde{y}_i)$ und (x_j, y_j) für $j \neq i$ sonst. Wegen der Eindeutigkeit des Interpolationspolynoms gilt somit $p_\varepsilon = p$ für alle $\varepsilon \in \mathbb{R}$, d.h. p_ε ist bezüglich ε eine konstante Funktion. Mit der Kettenregel der Differentiation folgt

$$0 = \frac{\mathrm{d}p_\varepsilon}{\mathrm{d}\varepsilon} = \frac{\partial p_\varepsilon}{\partial \widetilde{x}_i} \cdot \frac{\partial \widetilde{x}_i}{\partial \varepsilon} + \frac{\partial p_\varepsilon}{\partial \widetilde{y}_i} \cdot \frac{\partial \widetilde{y}_i}{\partial \varepsilon}.$$

An der Stelle $\varepsilon = 0$ erhalten wir dann

$$0 = \frac{\mathrm{d}p}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} = \underbrace{\frac{\partial p}{\partial \widetilde{x}_i}\Big|_{\varepsilon=0}}_{= \partial y/\partial x_i} \cdot \underbrace{\frac{\partial \widetilde{x}_i}{\partial \varepsilon}\Big|_{\varepsilon=0}}_{=1} + \underbrace{\frac{\partial p}{\partial \widetilde{y}_i}\Big|_{\varepsilon=0}}_{= L_i(x)} \cdot \underbrace{\frac{\partial \widetilde{y}_i}{\partial \varepsilon}\Big|_{\varepsilon=0}}_{= p'(x_i)}.$$

Dadurch folgt die dritte Ableitung.

Am wichtigsten ist der Einfluss von Fehlern δy_i in den Stützwerten auf den resultierenden Wert y, also die Konditionszahlen $\partial y/\partial y_i = L_i(x)$. Fasst man diese Konditionszahlen durch

$$\left\| \left(\frac{\partial y}{\partial y_0}, \frac{\partial y}{\partial y_1}, \ldots, \frac{\partial y}{\partial y_n} \right) \right\|_1 = \sum_{i=0}^{n} |L_i(x)|$$

zu einer Funktion von x zusammen, so ergibt sich mit Maximumbildung die absolute Konditionszahl

$$\Lambda_n := \max_{x \in [a,b]} \sum_{i=0}^{n} |L_i(x)|.$$

Man bezeichnet Λ_n als *Lebesgue-Konstante*. Sie hängt von der relativen Lage der Knoten zueinander ab (Übung). Für äquidistante Knoten wächst Λ_n schnell über alle Grenzen (Intervall $[-1, 1]$ betrachtet):

n	Λ_n, äquidistant	Λ_n, Tschebyscheff-Knoten
5	3.106	2.104
10	29.891	2.489
20	10986.534	2.901

Genauer zeigt sich, dass Λ_n exponentiell mit n bei äquidistanten Stützstellen anwächst, während für die Tschebyscheff-Knoten nur ein Wachstum von $\log(n)$ vorliegt.

Man beachte, dass $\sum_{i=0}^{n} |L_i(x)|$ bei äquidistanten Knoten sein Maximum erreicht, falls x ganz links aussen oder ganz rechts aussen liegt, d.h. x zwischen x_0 und x_1 oder x zwischen x_{n-1} und x_n. Einen Interpolanten zu äquidistanten Stützstellen sollte man deswegen nur zur Approximation im Zentrum einsetzen und nicht in ganz $[a, b]$.

Wählt man dagegen die Knoten nicht äquidistant sondern nach Tschebyscheff (siehe Beispiel unten), so werden die Stützstellen in den Randzonen verdichtet. Dadurch wird die Kondition entscheidend verbessert, d.h. die Konstante Λ_n relativ klein gehalten.

Beispiel: Es sei $n = 12$, $x_i = i$ für $i = 0, \ldots, n$ und $y_i = 1$ für $i \neq 7$. An der Stelle $x_7 = 7$ sei $y_7 = 2$. Abb. 17 zeigt das Ergebnis. Der Interpolant sieht in der Mitte relativ "gut" aus, am Rand neigt er zu starken Oszillationen. Stört man y_7 zu $\tilde{y}_7 = 2.2$, ändert sich der Interpolant vor allem am Rande schon relativ stark. Abhilfe schafft die Verwendung der Tschebyscheff-Knoten, da sie am Rand verdichtet auftreten und für sie die die Polynominterpolation besonders günstige Eigenschaften aufweist. Transformation auf $[0, 12]$ liefert den in Abb. 18 dargestellten Interpolanten, wobei $y_7 = 2$ und $y_i = 1$ sonst gesetzt wurde.

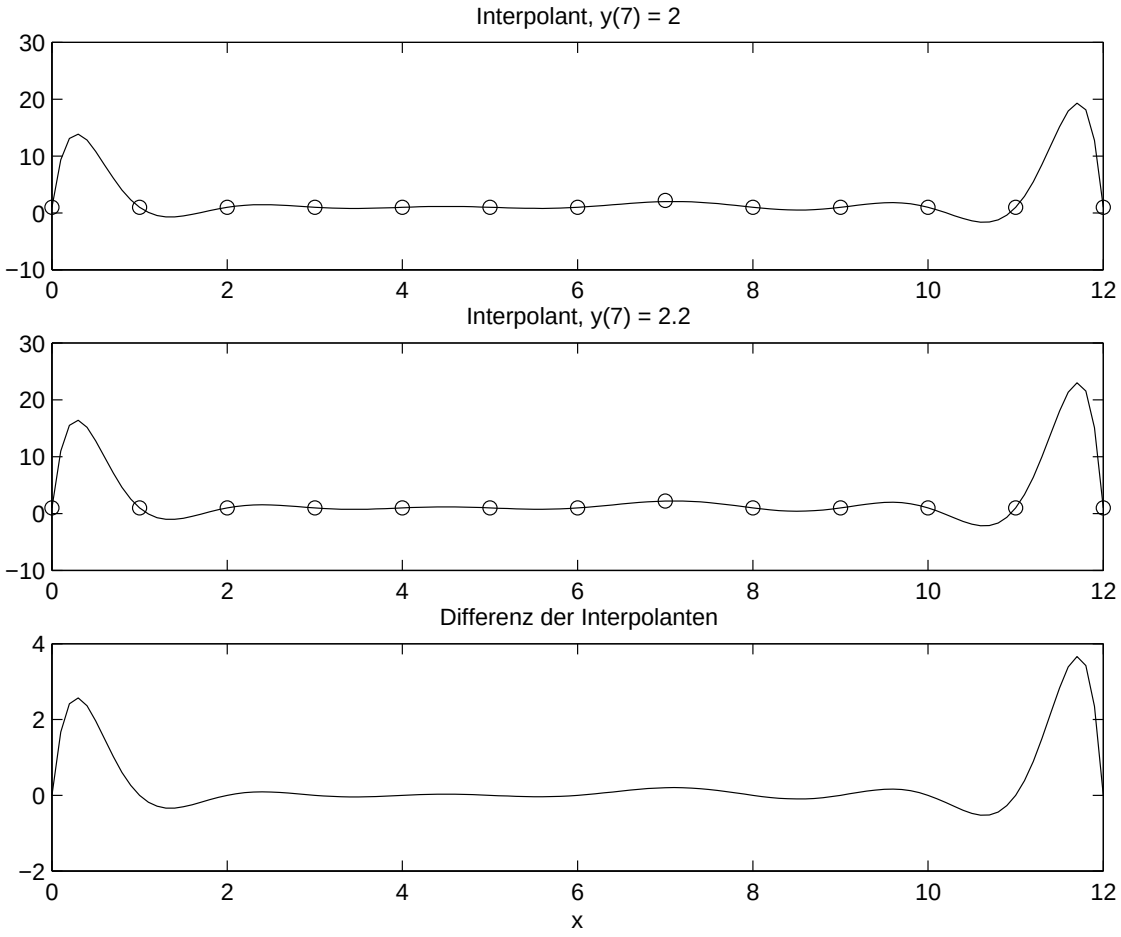


Abbildung 17: Interpolation mit äquidistantem Gitter.

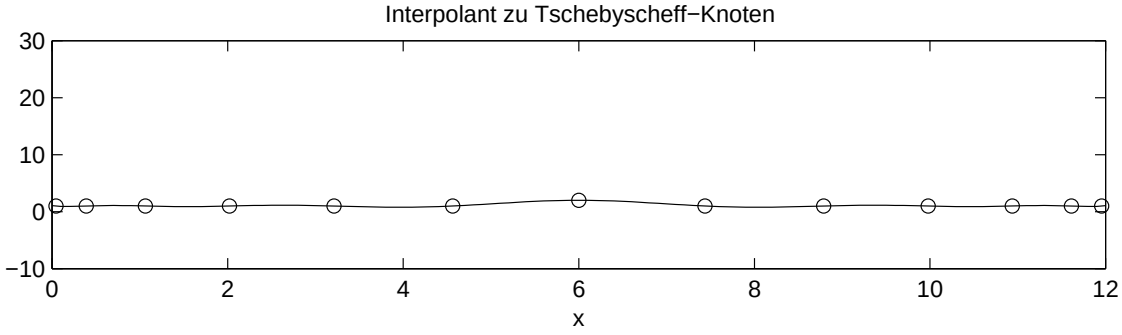


Abbildung 18: Interpolation mit Tschebyscheff-Knoten.

Abschliessende Bemerkungen zur Polynominterpolation

- Der Approximationsfehler bei der Polynominterpolation hängt entscheidend von der Wahl der Stützstellen ab. Sie sollten so gewählt werden, dass $\omega_{n+1}(x)$ klein bleibt. Die Tschebyscheff-Knoten erfüllen diese Forderung, siehe Deuflhard/Hohmann, Abschnitt 7.1.4.
- Nach Weierstraß gibt es zu jeder auf einem Kompaktum stetigen Funktion f ein Polynom p, das diese beliebig genau approximiert. Dieser positiven Aussage steht aber der Satz von Faber (Stoer, Abschnitt 2.1.4) gegenüber: Zu jeder feiner werdenden Folge von Stützstellenverteilungen gibt es eine stetige Funktion f, so dass die Folge der Interpolanten nicht gleichmäßig gegen f konvergiert.

Wie ist nun die Polynominterpolation zu werten, wenn wir die zu Beginn dieses Kapitels diskutierten Anforderungen an die Interpolation bzw. Approximation betrachten?

Positiv ist, dass für die Polynominterpolation eine abgeschlossene Theorie vorliegt: Existenz und Eindeutigkeit der Interpolationsaufgabe ist gesichert, und für die Approximationsaufgabe liegt mit der Restgliedformel eine aussagekräftige Fehlerformel vor. Die Berechung des Interpolanten kann einfach (rekursiv!) über die Dividierten Differenzen erfolgen, und die Funktionsauswertung über das Lemma von Aitken.

Jedoch überwiegen die negativen Eigenschaften: Die Berechnung des Interpolationspolynoms ist durch eine hohe Komplexität ($O(n^2)$, und nicht $O(n)$) gekennzeichnet. Weder liegt Formerhaltung vor, noch lässt die Analyse der Kondition der Interpolationsaufgabe auf einen "schönen" Interpolanten hoffen. Vielmehr erwarten wir für großes n immer stärkere Oszillationen am Rande.

Wieso dann überhaupt Polynominterpolation?

Die Polynominterpolation ist in der Numerik meist ein Hilfsmittel, um kontinuierliche Information bzw. Funktionen in *diskrete* zu transformieren. So ist

sie etwa zentrales Hilfsmittel, um Quadraturformeln zu gewinnen. Dabei beschränkt man sich auf einen Polynomgrad kleiner gleich 7 (Newton–Cotes–Formeln) oder ist insbesondere an einer Auswertung in der Intervallmitte interessiert (Extrapolationsverfahren). Mehr hierzu in Kapitel 7.

In Kapitel 6 werden wir dann die Frage beantworten, wie die Interpolationsaufgabe mit *in n linearer Komplexität* erfüllt werden kann, die den beiden Forderung nach *shape preserving* und *visually pleasing* genügt.

Kapitel 6

Splineinterpolation

Mit wachsendem Grad verliert die Polynominterpolation rasch ihre guten Eigenschaften: Der Aufwand zur Konstruktion des Interpolanten wächst quadratisch mit der Anzahl der Stützstellen, und der Interpolant wird sehr empfindlich gegenüber kleinen Datenänderungen. Als Alternative bietet sich eine *stückweise* Konstruktion mit Polynomen von niedrigem Grad an.

6.1 Motivation

In vielen Anwendungen ist eine große Zahl an Datenpunkten zu verarbeiten bzw. zu interpolieren. Ein natürlicher Zugang ist dann die Aufspaltung der Daten und damit auch des Interpolanten. Im folgenden wird der Interpolant nur noch stückweise definiert und aus *Polynomsegmenten* zusammengesetzt. Die darauf basierenden *Polynom-Splines* haben sich als das ideale Werkzeug für viele Interpolations- und Approximationsaufgaben erwiesen.

Wie bei der Polynominterpolation gehen wir von $n+1$ Stützpunkten

$$(x_0, y_0), \ldots, (x_n, y_n)$$

aus. Die Unterteilung bzw. das Gitter sei monoton und bestehe aus paarweise verschiedenen Knoten, d.h.

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b\,.$$

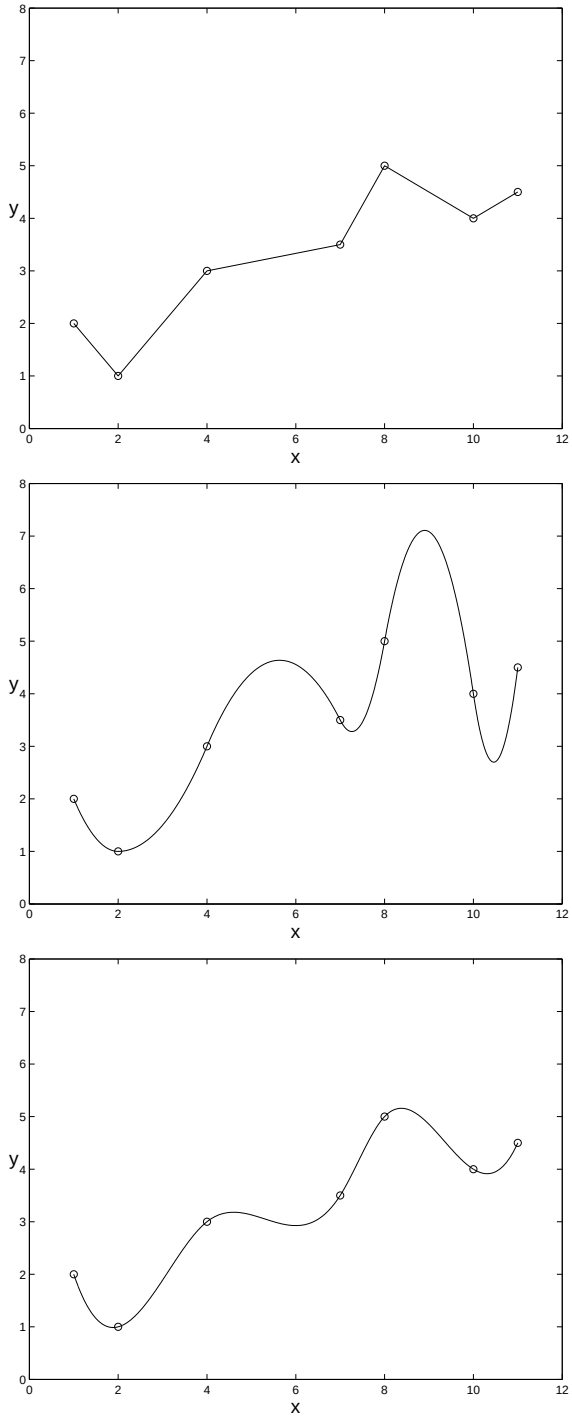


Abbildung 19: Interpolation der gleichen Stützpunkte mit linearem Spline/Polygonzug (oben), quadratischem Spline (mitte) und kubischem Spline (unten).

Der Interpolant $s : [a, b] \to \mathbb{R}$ soll nun erfüllen

$$s(x_i) = y_i \qquad \text{für } i = 0, 1, \ldots, n.$$

Der einfachste Fall eines stückweise zusammengesetzten Interpolanten ist der *Polygonzug*. Ein stückweise linearer Ansatz führt auf

$$s(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i} \qquad \text{für} \quad x_i \le x \le x_{i+1}\,. \tag{6.1}$$

Abb. 19 zeigt ein Beispiel. Ein Vorteil ist, dass dieser Interpolant sofort aufgestellt und einfach ausgewertet werden kann. Der Polygonzug ist stetig, jedoch nicht stetig differenzierbar.

Wir möchten zumindest einen überall (global) glatten Interpolanten, d.h. $s \in C^1[x_0, x_n]$, erhalten. Um dies zu erreichen, definieren wir stückweise quadratische Polynome

$$s(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \qquad \text{für} \quad x_i \le x \le x_{i+1}\,.$$

Die unbekannten Koeffizienten a_i, b_i, c_i können derart bestimmt werden, dass die Interpolationsbedingungen erfüllt sind und der Interpolant an den Stützstellen stetig differenzierbar ist. Abb. 19 enthält ebenfalls diesen Fall für den Beispieldatensatz. Wir erkennen einen "welligen" Verlauf des Interpolanten, d.h. ein Überschwingverhalten tritt auf.

Oft möchte man einen global zweimal stetig differenzierbaren Interpolanten, d.h. $s \in C^2[x_0, x_n]$, benutzen. Der Ansatz

$$s(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \qquad \text{für} \quad x_i \le x \le x_{i+1}\,. \tag{6.2}$$

mit stückweise kubischen Polynomen erlaubt diese Konstruktion. Das Beispiel aus Abb. 19 verdeutlicht, dass der resultierende Interpolant auch eine insgesamt "schönere" Form (weniger wellig) aufweist.

Diese Vorgehensweise führt auf die folgende Definition von Splines.

Definition 6.1: *Splines von Grad k*

Sei $x_0 < x_1 < \ldots < x_n$ und $k \in \mathbb{N}$. Dann heißt $s : [x_0, x_n] \to \mathbb{R}$ Splinefunktion vom Grad k, wenn $s \in C^{k-1}[x_0, x_n]$ und

$$s(x) = p_i(x) \quad \textit{für} \quad x_i \leq x < x_{i+1}$$

mit Polynomen $p_i \in \mathbb{P}_k$ für $i = 0, \ldots, n-1$. Die x_i sind die Knoten, und die Menge aller solchen Splines wird mit $\mathcal{S}_k(x_0, \ldots, x_n)$ bezeichnet.

Man beachte, dass die Definition des Splines nur die Knoten $x_0, \ldots, x_n$ aber nicht die Stützwerte $y_0, \ldots, y_n$, d.h. die Interpolationsbedingungen, benutzt.

Für $k = 0$ hat man Treppenfunktionen, für $k = 1$ Polygone, für $k = 2$ quadratische Splines und für $k = 3$ kubische Splines.

Die Summe zweier Splinefunktionen und ihr skalares Vielfaches sind selbst Splinefunktionen des gleichen Grades und mit den gleichen Knoten. Deshalb bildet $\mathcal{S}_k(x_0, \ldots, x_n)$ einen Vektorraum.

Der englische Name *Spline* steht für die im Schiffsbau verwendeten Latten oder Leisten aus astfreiem Holz oder Stahl, die mit Stiften oder Gewichten an bestimmten Stellen fixiert werden. Der Zusammenhang mit der Interpolationsaufgabe wird weiter unten erläutert.

Von besonderer Bedeutung sind die kubischen Splines ($k = 3$). Spricht man nur von Splines ohne Angabe des Grads, so sind kubische Splines gemeint.

Wir erläutern kurz eine rekursive Konstruktion für einen kubischen interpolierenden Spline. Sei $s_i(x)$ der Interpolant im i-ten Teilintervall für $i = 0, 1, \ldots, n-1$. Nun wird s_0 derart gewählt, dass die Bedingungen $s_0(x_0) = y_0$ und $s_0(x_1) = y_1$ erfüllt sind (z.B. interpolierende Gerade). Sei s_{i-1} bereits gegeben. Wir bestimmen s_i derart, dass $s_i(x_i) = y_i$ und $s_i(x_{i+1}) = y_{i+1}$ gilt. Zudem soll der globale Interpolant an der Stelle x_i zweimal stetig differenzierbar sein. Es folgen die Koeffizienten (vgl. (6.2))

$$a_i = y_i, \quad b_i = s'_{i-1}(x_i), \quad c_i = \tfrac{1}{2} s''_{i-1}(x_i),$$

$$d_i = \tfrac{1}{h_i^3} \left(y_{i+1} - y_i - h_i s'_{i-1}(x_i) - \tfrac{1}{2} h_i^2 s''_{i-1}(x_i) \right),$$

mit $h_i := x_{i+1} - x_i$. Eine geeignete Wahl der beiden Freiheitsgrade in s_0 (nur zwei statt vier Bedingungen) muss noch stattfinden. Es zeigt sich, dass Randbedingungen von Vorteil sind, d.h. eine Bedingung an s_0 und eine Bedingung an s_{n-1}. Somit entsteht statt der Rekursion dann ein lineares Gleichungssystem für die Koeffizienten.

6.2 Hermite-Interpolation

Bei der Hermite-Interpolation möchte man einen global stetig differenzierbaren Interpolanten p erhalten. Zudem sind neben den üblichen Interpolationsbedingungen auch Ableitungswerte vorgegeben:

$$(x_0, y_0, y_0'), \ldots, (x_n, y_n, y_n').$$

Der Hermite-Interpolant p soll die Bedingungen

$$p(x_i) = y_i, \quad p'(x_i) = y_i' \quad \text{für } \; i = 0, 1, \ldots, n$$

erfüllen. Diese Aufgabe kann mit stückweise kubischen Polynomen gelöst werden. Sei $s_i \in \mathbb{P}_3$ der Interpolant im i-ten Teilintervall. Die resultierenden Forderungen sind

$$s_i(x_i) = y_i, \quad s_i(x_{i+1}) = y_{i+1}, \quad s_i'(x_i) = y_i', \quad s_i'(x_{i+1}) = y_{i+1}' \,.$$

Wir erhalten somit vier Bedingungen für die vier unbekannten Koeffizienten in (6.2). Zudem sind die Forderungen an s_i unabhängig von s_j für $j \neq i$. Es ergibt sich folgender Interpolant (Übungsaufgabe)

$$s_i(x) = y_i \Phi_1(t) + y_{i+1} \Phi_2(t) + y_i' h_i \Phi_3(t) + y_{i+1}' h_i \Phi_4(t) \tag{6.3}$$

mit $h_i := x_{i+1} - x_i$ und $t = (x - x_i)/h_i$. Die beteiligten Basisfunktionen lauten

$$\begin{aligned} \Phi_1(t) &= 1 - 3t^2 + 2t^3, \\ \Phi_2(t) &= 3t^2 - 2t^3, \\ \Phi_3(t) &= t - 2t^2 + t^3, \\ \Phi_4(t) &= -t^2 + t^3. \end{aligned}$$

Man beachte, dass der konstruierte Hermite-Interpolant hier kein Spline laut Def. 6.1 ist, da Polynome vom Grad 3 verwendet werden jedoch global nur eine einmal stetig differenzierbare Funktion vorliegt.

Man kann $y'_0, \ldots, y'_n$ so vorgeben, dass ein Hermite-Interpolant folgt, der formerhaltend ist. Die Methode nach Fritsch/Carlson erhält z.B. die Monotonie bei Daten $y_0 \leq y_1 \leq \cdots \leq y_n$ bzw. $y_0 \geq y_1 \geq \cdots \geq y_n$.

Für das Auge sind Sprünge in der Krümmung einer Kurve (also Sprünge in der 2. Ableitung) noch wahrnehmbar. Ein Interpolant aus $C^2[a, b]$ ist daher der Hermite-Interpolation oft vorzuziehen. Wir verwenden im folgenden den kubischen Hermite-Interpolanten zur Konstruktion eines kubischen Spline-Interpolanten, indem die Ableitungswerte $y'_0, \ldots, y'_n$ derart gewählt werden, dass die Funktion global zweimal stetig differenzierbar wird und Randbedingungen erfüllt sind.

6.3 Kubische Spline-Interpolation

Für die zwei Randbedingungen des interpolierenden kubischen Splines wählen wir eine der drei folgenden Möglichkeiten:

(i) *natürliche Randbedingungen*: $s''(x_0) = 0,\ s''(x_n) = 0.$

(ii) *vollständige Randbedingungen*: $s'(x_0) = y'_0,\ s'(x_n) = y'_n,$
wobei y'_0 und y'_n vorzugeben sind.

(iii) *periodische Randbedingungen*: $s'(x_0) = s'(x_n),\ s''(x_0) = s''(x_n),$
welche nur im Fall $y_0 = y_n$ (periodische Daten) sinnvoll sind.

Eine weitere Möglichkeit, die wir nicht näher behandeln, ist die *not-a-knot*-Bedingung. Hierbei betrachtet man Splines laut Def. 6.1 zum reduzierten Knotensatz $x_0, x_2, \ldots, x_{n-2}, x_n$. Dafür werden bei den Randfunktionen nun die Interpolationsbedingungen $s_0(x_1) = y_1$ und $s_{\text{end}}(x_{n-1}) = y_{n-1}$ gefordert.

Berechnung des Spline-Interpolanten

Nach dem oben gesagten ist der Interpolant s intervallweise ein Polynom dritten Grades, d.h.

$$s(x) = s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x_i \le x \le x_{i+1}\,.$$

s_i genügt den Forderungen

$$s_i(x_i) = y_i, \quad s_i(x_{i+1}) = y_{i+1}, \quad s_i'(x_i) = b_i, \quad s_i'(x_{i+1}) = b_{i+1},$$

mit $a_i = y_i$ und weiteren, noch unbekannten Koeffizienten b_i, c_i, d_i.

Über die Formeln der Hermite-Interpolation erhält man folgende Darstellung

$$s_i(x) = y_i(1-3t^2+2t^3)+b_ih_i(t-2t^2+t^3)+y_{i+1}(3t^2-2t^3)+b_{i+1}h_i(-t^2+t^3)$$

sowie

$$s_i'(x) = z_i(6t - 6t^2) + b_i(1 - 4t + 3t^2) + b_{i+1}(-2t + 3t^2)\,.$$

Dabei ist $h_i := x_{i+1} - x_i$, $t := (x - x_i)/h_i$ und $z_i = (y_{i+1} - y_i)/h_i$. Die Koeffizienten c_i und d_i sind durch diese Konstruktion ebenfalls festgelegt. Nur die Steigungen b_i, b_{i+1} sind noch frei.

Als Hermite-Interpolant ist s aus $C^1[a, b]$. Die zweite Ableitung lautet

$$s_i''(x) = \left(z_i(6 - 12t) + b_i(-4 + 6t) + b_{i+1}(-2 + 6t)\right)/h_i\,.$$

Im Knoten x_i soll $s''(x_i)$ existieren, d.h.

$$\begin{aligned} 0 &= s''(x_i + 0) - s''(x_i - 0) \\ &= (6z_i - 4b_i - 2b_{i+1})/h_i - (-6z_{i-1} + 2b_{i-1} + 4b_i)/h_{i-1}\,. \end{aligned}$$

Die unbekannten Steigungen b_i genügen also den Bedingungen

$$\frac{b_{i-1}}{h_{i-1}} + \left(\frac{2}{h_{i-1}} + \frac{2}{h_i}\right) b_i + \frac{b_{i+1}}{h_i} = 3\left(\frac{z_{i-1}}{h_{i-1}} + \frac{z_i}{h_i}\right) \tag{6.4}$$

für $i = 1, \ldots, n - 1$.

Insgesamt liegen $n-1$ lineare Gleichungen für $n+1$ Unbekannte vor. Die verbleibenden 2 Unbekannten werden über die Randbedingungen bestimmt.

Am einfachsten ist der Fall der vollständigen Randbedingungen. Hier ist $b_0 = y_0'$ und $b_n = y_n'$ vorgegeben. Zu lösen ist ein LGS

$$A \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} 3(z_0/h_0 + z_1/h_1) - b_0/h_0 \\ 3(z_1/h_1 + z_2/h_2) \\ \vdots \\ 3(z_{n-3}/h_{n-3} + z_{n-2}/h_{n-2}) \\ 3(z_{n-2}/h_{n-2} + z_{n-1}/h_{n-1}) - b_n/h_{n-1} \end{pmatrix}$$

mit Tridiagonalmatrix

$$A = \begin{pmatrix} \left(\frac{2}{h_0} + \frac{2}{h_1}\right) & \frac{1}{h_1} & & & & \\ \frac{1}{h_1} & \left(\frac{2}{h_1} + \frac{2}{h_2}\right) & \frac{1}{h_2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \frac{1}{h_{n-3}} & \left(\frac{2}{h_{n-3}} + \frac{2}{h_{n-2}}\right) & \frac{1}{h_{n-2}} & \\ & & & \frac{1}{h_{n-2}} & \left(\frac{2}{h_{n-2}} + \frac{2}{h_{n-1}}\right) & \end{pmatrix}.$$

Die Matrix A ist strikt diagonaldominant (sowohl starkes Zeilensummenkriterium als auch starkes Spaltensummenkriterium sind erfüllt). Somit ist Gaußelimination ohne Pivotsuche möglich. Insbesondere ist die Matrix regulär wegen der Diagonaldominanz. Der Rechenaufwand ist proportional zu n aufgrund der tridiagonalen Struktur! Die Matrix A ist sogar symmetrisch und positiv definit. Die Cholesky-Zerlegung bei Tridiagonalmatrizen ist jedoch teurer als der Gauß-Algorithmus.

Im Fall natürlicher Randbedingungen ($s''(x_0) = s''(x_n) = 0$) sind zusätzlich zu (6.4) die beiden Gleichungen

$$\begin{aligned} 2\frac{b_0}{h_0} + \frac{b_1}{h_0} &= 3\frac{z_0}{h_0} \\ \frac{b_{n-1}}{h_{n-1}} + 2\frac{b_n}{h_{n-1}} &= 3\frac{z_{n-1}}{h_{n-1}} \end{aligned}$$

für b_0 und b_n mit zu lösen. Wiederum ist das LGS tridiagonal und strikt

diagonaldominant, diesmal aber von der Dimension $n+1$

$$A \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} = \begin{pmatrix} 3z_0/h_0 \\ 3(z_0/h_0 + z_1/h_1) \\ \vdots \\ 3(z_{n-2}/h_{n-2} + z_{n-1}/h_{n-1}) \\ 3z_{n-1}/h_{n-1} \end{pmatrix}$$

mit Koeffizientenmatrix

$$A = \begin{pmatrix} \frac{2}{h_0} & \frac{1}{h_0} & & & & \\ \frac{1}{h_0} & \left(\frac{2}{h_0} + \frac{2}{h_1}\right) & \frac{1}{h_1} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \frac{1}{h_{n-2}} & \left(\frac{2}{h_{n-2}} + \frac{2}{h_{n-1}}\right) & \frac{1}{h_{n-1}} \\ & & & \frac{1}{h_{n-1}} & \frac{2}{h_{n-1}} \end{pmatrix}.$$

Periodische Randbedingungen ergeben wegen $b_0 = s'(x_0) = s'(x_n) = b_n$ nur eine weitere Unbekannte. Aus $s''(x_0) = s''(x_n)$ erhält man als Gleichung für b_0 (oder betrachte (6.4) im Fall $i = 0$ und ersetze Index -1 durch $n-1$)

$$\frac{b_{n-1}}{h_{n-1}} + \left(\frac{2}{h_0} + \frac{2}{h_{n-1}}\right) b_0 + \frac{b_1}{h_0} = 3\left(\frac{z_0}{h_0} + \frac{z_{n-1}}{h_{n-1}}\right).$$

Insgesamt folgt ein LGS der Ordnung n für $b_0, b_1, \ldots, b_{n-1}$. Die Koeffizientenmatrix ist nur beinahe tridiagonal. In der rechten oberen und in der linken unteren Ecke hat man das Extraelement $1/h_{n-1}$:

$$A = \begin{pmatrix} \left(\frac{2}{h_{n-1}} + \frac{2}{h_0}\right) & \frac{1}{h_0} & & & & \frac{1}{h_{n-1}} \\ \frac{1}{h_0} & \left(\frac{2}{h_0} + \frac{2}{h_1}\right) & \frac{1}{h_1} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \frac{1}{h_{n-3}} & \left(\frac{2}{h_{n-3}} + \frac{2}{h_{n-2}}\right) & \frac{1}{h_{n-2}} & \\ \frac{1}{h_{n-1}} & & & \frac{1}{h_{n-2}} & \left(\frac{2}{h_{n-2}} + \frac{2}{h_{n-1}}\right) & \end{pmatrix}.$$

In allen drei Fällen ist das Ziel erreicht, den Interpolanten mit der Komplexität const. $\cdot\, n$ zu berechnen. Die Formulierung eines Algorithmus ergibt

sich von selbst, man stellt die Matrix A und rechte Seite auf und löst nach den Steigungen auf. Zur Auswertung des Interpolanten greift man dann auf die intervallweise Darstellung (6.3) über die Hermite-Interpolation zurück.

Eigenschaften des Spline-Interpolanten

Der kubische Spline-Interpolant hat gewisse Optimalitätseigenschaften bezüglich der Gesamtkrümmung von interpolierenden Funktionen. Dabei ist die (lokale) Krümmung einer Kurve/Funktion $f \in C^2[a,b]$ definiert als

$$\kappa(x) = \frac{f''(x)}{(1+f'(x)^2)^{3/2}}\,.$$

Zur Vereinfachung nehmen wir an, dass $f'(x) \approx 0$ gilt, d.h. näherungsweise $\kappa(x) \approx f''(x)$. Die Gesamtkrümmung definieren wir als quadratisches Mittel der Krümmung durch Aufintegrieren

$$K(f) := \left(\int_a^b f''(x)^2\,\mathrm{d}x\right)^{1/2}\,. \tag{6.5}$$

Wir werden nachweisen, dass der interpolierende kubische Spline minimal bezüglich der Gesamtkrümmung ist und somit ein Überschwingverhalten vermeidet. Sei s der interpoliernde Spline. Im Fall von natürlichen Randbedingungen ist daher zu zeigen

$$K(s) \le K(f)$$

für jede Funktion $f \in C^2[a,b]$ mit $f(x_i) = y_i$ für $i = 0, 1, \ldots, n$. Im Fall von vollständigen Randbedingungen lassen wir nur Funktionen f zu mit $f'(x_0) = y_0'$ und $f'(x_n) = y_n'$. Bei periodischen Randbedingungen beziehen wir uns auf periodische Funktionen f mit Periode $x_n - x_0$.

Lemma 6.2: *Integralrelation*

Sei s der kubische Spline-Interpolant zu Daten (x_i, y_i) für $i = 0, 1, \ldots, n$ mit natürlichen / vollständigen / periodischen Randbedingungen. Ist nun $f \in C^2[x_0, x_n]$ ein anderer Interpolant mit analogen Eigenschaften, dann gilt

$$\int_{x_0}^{x_n} \big(f''(x) - s''(x)\big) s''(x)\,\mathrm{d}x = 0\,.$$

Beweis:

Wir spalten das Integral auf in

$$\int_{x_0}^{x_n} (f'' - s'')s''\,\mathrm{d}x = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f'' - s'')s''\,\mathrm{d}x\,.$$

Partielle Integration in den Teilintervallen liefert

$$\int_{x_i}^{x_{i+1}} (f'' - s'')s''\,\mathrm{d}x = (f' - s')s''\big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} (f' - s')s^{(3)}\,\mathrm{d}x.$$

Eine weitere partielle Integration zeigt

$$\int_{x_i}^{x_{i+1}} (f' - s')s^{(3)}\,\mathrm{d}x = (f - s)s^{(3)}\big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} (f - s)s^{(4)}\,\mathrm{d}x = 0$$

wegen der Interpolationsbedingung $s(x_i) = f(x_i)$ für alle i und $s^{(4)} \equiv 0$. Es folgt mit der globalen Stetigkeit von f', s', s'' nun

$$\int_{x_0}^{x_n} (f'' - s'')s''\,\mathrm{d}x = (f'(b) - s'(b))s''(b) - (f'(a) - s'(a))s''(a).$$

Die Behauptung folgt nun in den drei Fällen:

i) natürliche Randbedingungen: $s''(a) = s''(b) = 0$.
ii) vollständige Randbedingungen: $f'(a) = s'(a)$, $f'(b) = s'(b)$.
iii) periodische Randbedingungen: $f'(a) = f'(b)$, $s'(a) = s'(b)$, $s''(a) = s''(b)$. □

Die Aussage aus Lemma 6.2 gilt auch, wenn nur $f \in C^1[a, b]$ vorliegt aber f stückweise (bezüglich der Unterteilung $x_0 < x_1 < \cdots < x_n$) zweimal stetig differenzierbar ist. Nun folgt die Minimaleigenschaft.

Satz 6.3 *Optimalität des kubischen Splines*

Sei s der kubische Spline-Interpolant zu Daten (x_i, y_i) für $i = 0, 1, \ldots, n$ mit natürlichen / vollständigen / periodischen Randbedingungen. Dann gilt die Relation

$$\int_{x_0}^{x_n} (s''(x))^2 \, \mathrm{d}x \leq \int_{x_0}^{x_n} (f''(x))^2 \, \mathrm{d}x$$

bei jeder interpolierenden Funktion $f \in C^2[a, b]$ im Fall (i), bei entsprechenden Funktionen mit $f'(a) = y_0'$ und $f'(b) = y_n'$ im Fall (ii) und bei periodischen Funktionen im Fall (iii).

Beweis:

Es gilt allgemein

$$(f'')^2 = (f'' - s'')^2 + 2(f'' - s'')s'' + (s'')^2 \geq 2(f'' - s'')s'' + (s'')^2.$$

Mit der Monotonie des Integrals und Lemma 6.2 hat man

$$\int_{x_0}^{x_n} (f'')^2 \, \mathrm{d}x \geq 2 \int_{x_0}^{x_n} (f'' - s'')s'' \, \mathrm{d}x + \int_{x_0}^{x_n} (s'')^2 \mathrm{d}x = \int_{x_0}^{x_n} (s'')^2 \, \mathrm{d}x.$$

Damit hat man bereits die Behauptung gezeigt. □

Dieser Satz besagt, dass unter allen interpolierenden Funktionen der kubische Spline mit entsprechenden Randbedingungen eine minimale Gesamtkrümmung besitzt. Wir erwarten daher, dass der Interpolant wenig "wellig" verläuft und ein Überschwingverhalten vermeidet, vgl. Abb. 19. (Die not-a-knot-Bedingung führt nicht auf einen optimalen Spline.)

Wie Lemma 6.2 so gilt auch Satz 6.3 für Funktionen $f \in C^1[a, b]$, die stückweise zweimal stetig differenzierbar sind.

Ohne Beweis sei noch angegeben (siehe Stoer, Abschnitt 2.4.3):

Satz 6.4: *Fehlerformel*

Die kubische Splinefunktion s mit Knoten $x_0 < x_1 < \ldots < x_n$ interpoliere dort $f \in C^4[x_0, x_n]$ mit vollständigen / periodischen Randbedingungen. Dann gilt im Intervall $[x_i, x_{i+1}]$ für $i = 0, \ldots, n-1$

$$\begin{aligned} |f - s| &\leq \tfrac{3}{64} L h_i^2 H^2 \\ |f' - s'| &\leq \tfrac{3}{16} L h_i H^2 \\ |f'' - s''| &\leq \tfrac{3}{8} L H^2 \\ |f''' - s'''| &\leq \tfrac{1}{2} L \left(\tfrac{H^2}{h_i} + h_i \right) \end{aligned}$$

mit $L := \max |f^{(4)}(x)|,\ h_i := x_{i+1} - x_i,\ H := \max(h_0, \ldots, h_{n-1})$.

Dieser Satz gibt eine Aussage jeweils über den maximalen Fehler.

Im Spezialfall von äquidistanten Stützstellen mit $a = x_0$ und $b = x_n$ gilt $h_i = H = (b-a)/n$ für alle i. Mit Satz 6.4 folgt

$$\max_{x \in [a,b]} |f(x) - s(x)| \leq \tfrac{3}{64} L \frac{(b-a)^4}{n^4}.$$

Insbesondere ist gleichmäßige Konvergenz ist im Fall von äquidistanten Stützstellen somit gegeben. Zudem ist die Konvergenz schnell.

6.4 B-Splines

Als Abschluss der Splineinterpolation stellen wir uns noch die Frage nach einer geeigneten Basis des Vektorraums $\mathcal{S}_k(x_0, \ldots, x_n)$, siehe Def. 6.1. Man kann sich überlegen, dass dieser Vektorraum die Dimension $n + k$ besitzt.

Eine für theoretische Zwecke günstige Basis erhält man über sogenannte *abgeschnittene Potenzfunktionen*

$$x_+^j := \begin{cases} x^j & \text{falls } x > 0, \\ 0 & \text{sonst.} \end{cases} \tag{6.6}$$

Abgeschnittene Potenzfunktionen sind also Splines mit einem Knoten. Für $j = 0$ erhält man die Sprungfunktion. Für $j = 1$ entsteht eine stetige Funktion. Bei $j \geq 2$ ist die Funktion (6.6) dann $(j-1)$-mal stetig differenzierbar.

Es gilt

$$\frac{\mathrm{d}}{\mathrm{d}x}\, x_+^j = j \cdot x_+^{j-1}\,, \qquad \int_{-\infty}^{x} t_+^j \,\mathrm{d}t = \frac{x_+^{j+1}}{j+1}$$

(mit Vorsicht bei $j = 1$ und $x = 0$, dort nur einseitige Differentiation).

Eine Basis von $\mathcal{S}_k(x_0, \ldots, x_n)$ wird nun gebildet von

$$\{1,\, x,\, x^2, \ldots,\, x^k,\, (x - x_1)_+^k, \ldots,\, (x - x_{n-1})_+^k \}\,. \tag{6.7}$$

Ein $s \in \mathcal{S}_k(x_0, \ldots, x_n)$ hat die Darstellung

$$s(x) = a_0 + a_1 x + \ldots + a_k x^k + \sum_{i=1}^{n-1} c_i (x - x_i)_+^k\,.$$

Diese Basis ist für die Praxis jedoch unbrauchbar:

1. Für eine Auswertung $s(x)$ mit x nahe bei x_n sind nahezu alle Basisfunktionen beteiligt.
2. Für Gitter mit stark unterschiedlichen Abständen $h_i := x_{i+1} - x_i$ sind die Basisfunktionen beinahe linear abhängig.

Man sucht daher nach Basisfunktionen mit möglichst kleinem *Träger*

$$\operatorname{supp} f := \{x :\ f(x) \neq 0 \}\,.$$

Die *B-Splines* (Basis-Splines) erfüllen diese Forderung. Auch sind diese Funktionen nach Konstruktion dann nichtnegativ.

Nach de Boor können die B-Splines rekursiv definiert werden. Für $k = 0$ hat man die elementaren Treppenfunktionen

$$M_{i,0} = \begin{cases} 1 & \text{für } \ x_i \leq x \leq x_{i+1} \\ 0 & \text{sonst} \end{cases}$$

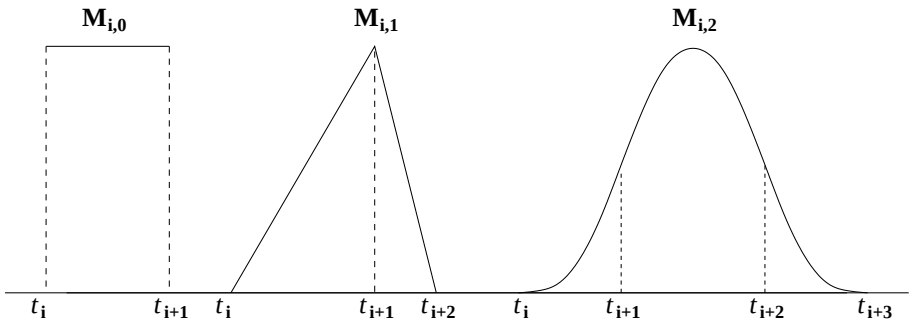


Abbildung 20: B-Splines vom Grad $k = 0, 1, 2$ (von links nach rechts).

mit $i = 0, 1, \ldots, n-1$. Bei $k > 0$ gibt man willkürlich weitere k Stützstellen $(x_n <) x_{n+1} < \cdots < x_{n+k}$ vor. Für $k > 0$ lautet die Rekursion

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

Eine Definition über dividierte Differenzen existiert ebenfalls. Abb. 20 zeigt Beispiele.

Im äquidistanten Fall $x_i = x_0 + ih$ mit $h = (x_n - x_0)/n$ für $i = 0, 1, \ldots, n$ sind alle B-Splines vom Grak k Transformierte einer einzigen Basisfunktion φ_k mit dem Träger supp $\varphi_k = [0, k+1]$. Man erhält die B-Splines über

$$M_{i,k}(x) = \varphi_k \left(\frac{x - x_i}{h} \right).$$

Im nichtäquidistanten Fall ist die Konstruktion der Basisfunktionen aufwändiger.

Wir verwenden die B-Splines nicht für die Interpolationsaufgabe, sondern lösen das korrespondierende LGS und werten den Hermite-Interpolanten aus. Die B-Splines sind jedoch hervorragend bei Aufgaben des *Computer-Aided-Design* (*CAD*), d.h. in der geometrischen Datenverarbeitung.

Für weitere Details zu B-Splines muss auf die Literatur verwiesen werden, siehe Stoer oder Deuflhard/Hohmann. In letzterem Buch findet man auch einen Abschnitt über *Bezier-Techniken*.

Kapitel 7

Numerische Quadratur

Unter numerischer Quadratur versteht man die Berechnung des bestimmten Integrals

$$I(f) := I_a^b(f) := \int_a^b f(x)\,\mathrm{d}x\,.$$

Der Name stammt von der "Quadratur des Kreises", d.h. dem Versuch, zum Einheitskreis mit Fläche π ein Quadrat mit gleicher Fläche zu konstruieren. Falls für eine gegebene Funktion f keine geschlossene (analytische) Integration möglich ist, benötigt man numerische Verfahren. Die Verfahren liefern im allgemeinen nur eine Approximation des exakten Integrals.

7.1 Quadraturformeln

Numerische Verfahren verlangen Integranden, die hinreichend oft differenzierbar ("glatt") sind, d.h. $f \in C^k[a,b]$ mit k möglichst groß.

Die Verfahren basieren auf *Quadraturformeln*

$$J(f) := \sum_{i=0}^{n} g_i f(x_i)$$

mit *Gewichten* $g_i \in \mathbb{R}$ und *Stützstellen/Knoten* $x_i \in [a,b]$. Man möchte $J(f) \approx I(f)$ erhalten. Die Quadraturformel hängt vom Integrationsintervall ab: $J(f) = J_a^b(f)$.

Anforderungen an Quadraturformeln

Das Integral I ist ein lineares, monotones Funktional auf $C[a,b]$:

- $I(\alpha f + \beta g) = \alpha I(f) + \beta I(g)$ für $f, g \in C[a,b]$ und $\alpha, \beta \in \mathbb{R}$,
- $f(x) \geq g(x)$ f.a. $x \in [a,b] \;\Rightarrow\; I(f) \geq I(g) \quad$ für $f, g \in C[a,b]$.
  Äquivalent: $f(x) \geq 0$ f.a. $x \in [a,b] \;\Rightarrow\; I(f) \geq 0$ (Positivität)

Zusätzlich ist I additiv bezüglich der Zerlegung $[a,b] = [a,\tau] \cup [\tau,b]$ mit $a < \tau < b$, d.h.: $I(f) = I_a^b(f) = I_a^\tau(f) + I_\tau^b(f)$.

Die Quadraturformel J soll analoge Eigenschaften haben! Bis auf die Positivität sind alle Forderungen durch den Ansatz bereits erfüllt. Positivität erhält man bei Wahl nichtnegativer Gewichte g_i. Denn dann gilt:

$$f(x) \geq 0 \quad \text{für alle } x \in [a,b] \quad \Rightarrow \quad J(f) \geq 0\,.$$

Fordert man noch, dass zumindest konstante Integranden exakt integriert werden sollen, so muss wegen

$$I(c) = \int_a^b c \,\mathrm{d}x = c(b-a), \qquad J(c) = \sum_{i=0}^{n} g_i \cdot c = c \sum_{i=0}^{n} g_i$$

mit der Konstanten $c \in \mathbb{R}$ gelten

$$\sum_{i=0}^{n} g_i = b - a,$$

d.h. die Summe der Gewichte ist die Intervallbreite.

In diesem Fall lässt sich die Wahl nichtnegativer Gewichte auch durch die Forderung minimaler Fehlerverstärkung motivieren.

Denn für negative Gewichte g_i ergibt sich eine Fehlerverstärkung bezüglich der Auswertung der Funktion f: Ist $\tilde{f}(x)$ die fehlerbehaftete Auswertung mit $|\tilde{f}(x) - f(x)| \leq \varepsilon$ für alle x, dann folgt

$$\left|J(\tilde{f}) - J(f)\right| = \left|J(\tilde{f} - f)\right| = \left|\sum_{i=0}^{n} g_i(\tilde{f}(x_i) - f(x_i))\right| \leq \varepsilon \sum_{i=0}^{n} |g_i|.$$

Die Wahl nichtnegativer Gewichte g_i ist optimal: $\sum |g_i| = \sum g_i = b - a$, d.h. minimal!
Negative Gewichte jedoch führen zu einer größeren Fehlerschranke: $\sum |g_i| > b - a$.

Übersicht zu Quadratformeln

a) *Newton–Cotes–Formeln und Summenformeln*, Abschnitt 7.2 und 7.3:
Es sei $p \in \mathbb{P}_n$ ein Polynom, das $f(x_i)$ für $i = 0, 1, \ldots, n$ interpoliert. Durch die Wahl äquidistanter Stützstellen x_i legt der Ansatz

$$J(f) := I(p) = \int_a^b p(x)\, \mathrm{d}x$$

die Gewichte g_i eindeutig fest.

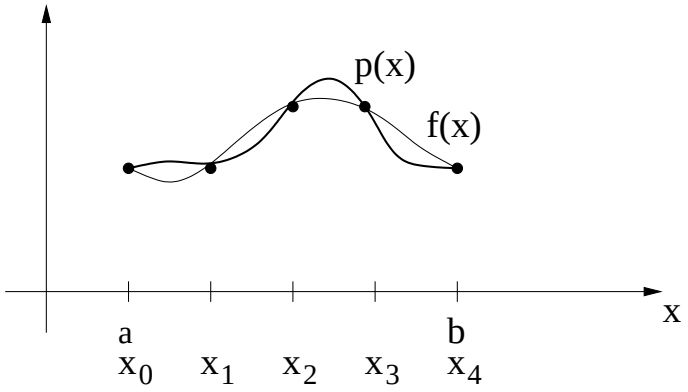


Abbildung 21: Idee der Newton–Cotes–Formeln.

Durch Aufteilung des Integrationsintervalls kann man aufgrund der Additivität von J in jedem Teilintervall $[x_i, x_{i+1}]$ eine elementare Quadraturformel anwenden und die Teilresultate aufsummieren. Die sich so ergebenden Summenformeln sind die Basis für zwei wichtige Klassen von Verfahren, die durch Adaptivität vorgegebene Fehlerschranken (Toleranzen) einhalten können: *adaptiver Simpson* und *Extrapolationsverfahren* (Abschnitt 7.4).

b) *Gauß-Quadratur*, Abschnitt 7.5:
Wähle x_i, g_i für $i = 1, \ldots, n$, so dass

$$\int_a^b p(x)\, \mathrm{d}x = \sum_{i=1}^{n} g_i p(x_i)$$

für alle Polynome p *möglichst hohen Grades*.

7.2 Newton–Cotes–Formeln

Die Idee einer sogenannten *interpolatorischen Quadraturformel* ist wie folgt: Wähle Knoten $a \leq x_0 < x_1 < \cdots < x_n \leq b$. Sei $p \in \mathbb{P}_n$ das Interpolationspolynom zu Stützpunkten $(x_i, f(x_i))$ für $i = 0, 1, \ldots, n$. Setze $J(f) := I(p)$. Es gilt

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

mit den Lagrange-Polynomen L_i. Somit erhalten wir

$$J(f) = I(p) = \sum_{i=0}^{n} f(x_i) \int_a^b L_i(x)\, \mathrm{d}x \quad \Rightarrow \quad g_i = \int_a^b L_i(x)\, \mathrm{d}x.$$

Wir erhalten eine Formel für die Gewichte. Die Integrale über die Polynome können analytisch bestimmt werden. Der Fehler der Approximation kann über die Restgliedformel untersucht werden. Insbesondere ist die Quadraturformel exakt für alle $q \in \mathbb{P}_n$.

Die *Newton-Cotes-Formeln* (N.C.F.) sind spezielle interpolatorische Quadraturen, bei denen äquidistanten Stützstellen verwendet werden. Die abgeschlossenen N.C.F. beziehen die Intervallränder mit ein, während offene N.C.F. diese auslassen, siehe Abb. 22. Im folgenden betrachten wir nur abgeschlossene Formeln.

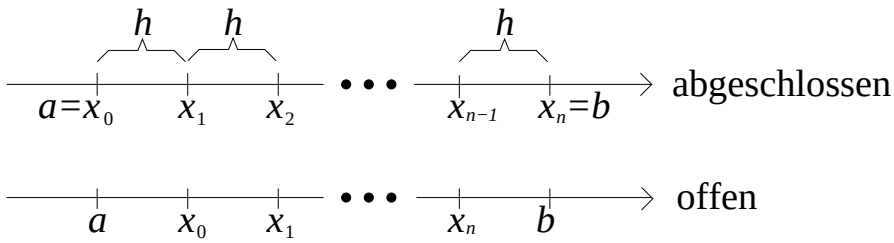


Abbildung 22: Knoten in Newton-Cotes-Formeln.

• $n = 1$: **Trapezregel**

Die beiden Stützstellen ergeben sich zu $x_0 = a$, $x_1 = b$, d.h. p ist linearer Interpolant zu $f(a), f(b)$. Somit

$$J(f) = I(p) = \frac{b-a}{2}(f(a) + f(b)) \tag{7.1}$$

mit Gewichten $g_0 = g_1 = \frac{b-a}{2}$.

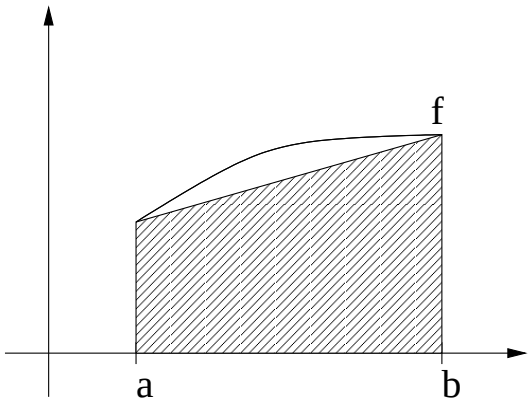


Abbildung 23: Trapezregel.

Die Analyse des Fehlers $R(f) := I(f) - J(f)$ kann über die Restgliedformel der Polynominterpolation erfolgen:

$$\begin{aligned} R(f) &= \int_a^b (f(x) - p(x))\,\mathrm{d}x \quad \text{mit} \quad f(x) - p(x) = \frac{1}{2}(x-x_0)(x-x_1)f''(\xi) \\ &= \frac{1}{2}\int_a^b (x-x_0)(x-x_1)f''(\xi(x))\,\mathrm{d}x \\ &= \frac{1}{2}\int_0^h t(t-h)f''(\xi(t+a))\,\mathrm{d}t \quad \text{für} \quad t = x-a,\ h = b-a. \end{aligned}$$

Da $t(t-h) \leq 0$ in $[0,h]$: Verallgemeinerter Mittelwertsatz der Integralrechnung liefert

$$R(f) = \frac{1}{2}f''(\xi^*)\int_0^h t(t-h)\,\mathrm{d}t = -\frac{1}{12}h^3 f''(\xi^*)$$

mit einer Zwischenstelle $\xi^* \in [a,b]$. Also ist der Fehler der Trapezregel

$$R(f) = I(f) - J(f) = -\tfrac{1}{12}h^3 f''(\xi^*)\,.$$

• $n = 2$**: Fassregel** (nach Kepler)

Die drei Stützstellen sind nun $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$. Damit ist p quadratischer Interpolant zu $f(x_0),\ f(x_1),\ f(x_2)$. Somit

$$J(f) = I(p) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{7.2}$$

mit den Gewichten $(g_0, g_1, g_2) = \frac{b-a}{6} \cdot (1,\ 4,\ 1)$.

Als Fehler erhält man für $h = \frac{b-a}{2}$ analog zur Trapezregel

$$R(f) = -\frac{h^5}{90} f^{(4)}(\xi)$$

mit einer Zwischenstelle $\xi \in [a, b]$. (Übungsaufgabe)

Die Fassregel ist sogar exakt für $f \in \mathbb{P}_3$, weil dann $f^{(4)} \equiv 0$ gilt.

Alternativer Beweis: Betrachte $q(x) = (x - \frac{a+b}{2})^3$. Dann ist $I(q) = 0$ wegen der Symmetrie der Funktion. Ebenso folgt $J(q) = 0$ wegen der Symmetrie $q(a) = -q(b)$. Ein beliebiges $p \in \mathbb{P}_3$ kann dargestellt werden als $p = \alpha q + r$ mit $\alpha \in \mathbb{R}$ und $r \in \mathbb{P}_2$. Es folgt

$$I(p) = \alpha \int_a^b q(x)\,\mathrm{d}x + \int_a^b r(x)\,\mathrm{d}x = \int_a^b r(x)\,\mathrm{d}x = J(r) = J(r) + \alpha J(q) = J(p).$$

Entscheidende Eigenschaft hier ist, dass die Knoten und Gewichte symmetrisch bezüglich der Intervallmitte liegen.

Die Kepler'sche Fassregel wird auch als *Simpson-Regel* bezeichnet.

Die Newton-Cotes-Formeln besitzen jedoch einen entscheidenden Nachteil: Ab $n = 8$ ergeben sich negative Gewichte bei den abgeschlossenen Formeln und ab $n = 2$ bei den offenen Formeln — daher wenig empfehlenswert!

Darin spiegeln sich die schlechten Eigenschaften der Polynominterpolation bei hoher Anzahl von Stützstellen wider. Der Ausweg war die Spline-Interpolation, d.h. eine stückweise polynomiale Konstruktion. Dieses Prinzip wenden wir im folgenden wieder an.

7.3 Summenformeln

Die Idee der Summenformeln besteht in der Einführung eines äquidistanten Gitters $a = x_0 < x_1 < \cdots < x_n = b$ mit $x_k := a + kh$, welches das Intervall $[a, b]$ in n Teilintervalle der Länge $h = \frac{b-a}{n}$ mit den entsprechenden Funktionswerten $f_k := f(x_k)$ zerlegt.

In jedem Teilintervall $[x_k, x_{k+1}]$ wende man nun eine elementare Quadraturformel an und summiere auf.

Einfachster Fall ist die *Rechtecksumme*

$$J(f) = h\,(f_0 + f_1 + f_2 + \cdots + f_{n-1})\,,$$

welche einer Diskretisierung des Integrals als Grenzwert von derartigen Rechtecksummen entspricht.

Die *Trapezsumme*

$$J(f) = h\left(\tfrac{1}{2}f_0 + f_1 + f_2 + \cdots + f_{n-1} + \tfrac{1}{2}f_n\right)$$

ergibt sich aus der Anwendung der Trapezregel $\frac{1}{2}h(f_k + f_{k+1})$ für das Teilintervall $[x_k, x_{k+1}]$ und anschließende Aufsummation.

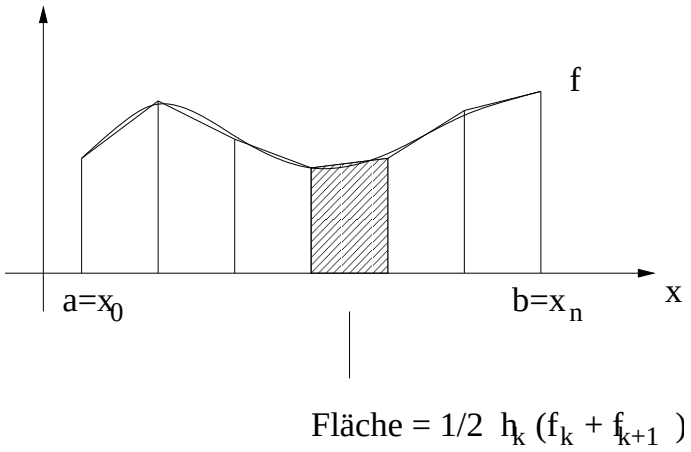


Abbildung 24: Idee der Trapezsumme.

Als Fehler erhält man aus der Fehlerformel für die Trapezregel

(mit $\xi_j \in [x_j, x_{j+1}]$ und $\xi \in [a,b]$):

$$R(f) = I(f) - J(f) = -\sum_{j=0}^{n-1} \frac{h^3}{12} f''(\xi_j) = -\frac{nh^3}{12} f''(\xi) = -\frac{b-a}{12} h^2 f''(\xi)\,,$$

wobei wir den Summenmittelwertsatz in der vorletzte Gleichung verwendet haben. Einen genaueren Fehlerausdruck kann man mit der *Euler–Mac-Laurinschen Summenformel* erhalten (vgl. nächster Abschnitt und Stoer).

Der Summenmittelwertsatz besagt $\sum g(\xi_i) = ng(\xi^*)$ für stetige Funktion $g \in C[a,b]$ und $\xi_i, \xi^* \in [a,b]$. Da eine stetige Funktion auf einem kompakten Intervall Maximum und Minimum annimmt, gilt $g(x_m) \le g(x) \le g(x_M)$ für alle $x \in [a,b]$. Aufsummation liefert

$$ng(x_m) \le \sum_{i=1}^{n} g(\xi_i) \le ng(x_M) \quad \Leftrightarrow \quad g(x_m) \le \frac{1}{n} \sum_{i=1}^{n} g(\xi_i) \le g(x_M).$$

Der Zwischenwertsatz besagt, dass eine Stelle $\xi^* \in [a,b]$ existiert, an der die stetige Funktion den Zwischenwert annimmt.

Die (zusammengesetzte) *Simpson-Regel* im Fall von geradem n

$$J(f) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n)$$

(mit $h = \frac{b-a}{2n}$) ergibt sich mit einem analogen Ansatz über die Fassregel statt Trapezregel. Als Fehler findet man (wieder mit $h = \frac{b-a}{n}$)

$$\begin{aligned} R(f) = I(f) - J(f) &= -\sum_{k=0}^{\frac{n}{2}-1} \frac{h^5}{90} f^{(4)}(\xi_k) = \frac{n}{2} f^{(4)}(\xi) \left(-\frac{b-a}{n} \cdot \frac{h^4}{90} \right) \\ &= -\frac{b-a}{180} h^4 f^{(4)}(\xi)\,. \end{aligned}$$

Man beachte, dass die Anzahl der Auswertungen von f in den verschiedenen Summenformeln (fast) gleich ist. Unterschiedliche Linearkombinationen bewirken unterschiedliche Genauigkeiten.

Adaptives Simpson-Verfahren

Adaptivität ist eine wichtige Eigenschaft numerischer Verfahren. Man will eine Berechnung nur bis zu einer gewissen Genauigkeitsforderung wie etwa "5 Stellen" durchführen und dann abbrechen, um nicht unnötig viel Rechenaufwand zu investieren.

Ein äquidistantes Gitter verlangt einen unverhältnismäßig hohen Aufwand, um solch eine Genauigkeitsforderung zu erfüllen, wenn der Integrand einen lokal sehr unterschiedlichen Verlauf hat.

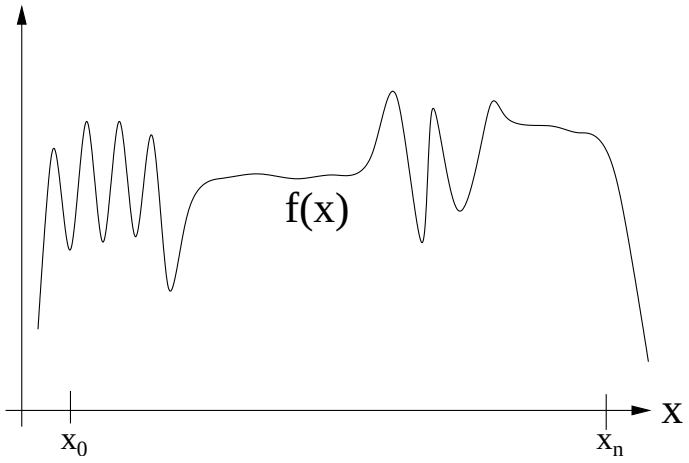


Abbildung 25: Beispiel für Integranden mit stark unterschiedlichem Verlauf.

Von einem guten adaptiven Verfahren wird man erwarten, dass es dort ein verfeinertes Gitter einsetzt, wo der Integrand f starke Nichtlinearitäten aufweist.

Ein Beispiel für ein robustes und effizientes adaptives Quadraturverfahren ist das *adaptive Simpson-Verfahren*, welches auf der Fassregel basiert.

Idee: Sei $[a, b]$ das aktuelle Intervall. Berechne $I(f)$ über die Fassregel und schätze den Fehler. Falls der Fehler klein genug ist, breche ab. Andernfalls unterteile $[a, b]$ in zwei Hälften $[a, \frac{a+b}{2}]$ und $[\frac{a+b}{2}, b]$ und fahre rekursiv fort.

Zur Durchführung eines adaptiven Algorithmus braucht man einen *Fehlerschätzer*. In unserem Fall eignet sich dazu die Information, die aus Fassregel und Simpsonsumme folgt.

Wir haben:

- Fassregel: $J(f)[a,b] = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$
- Simpsonsumme (2-fache Anwendung der Fassregel, Aufsummation):

$$\widehat{J}(f)[a,b] \;=\; J(f)[a,\tfrac{a+b}{2}] + J(f)[\tfrac{a+b}{2},b]$$

Aus den beiden Fehlern für die

- Fassregel ($h = \frac{b-a}{2}$): $J(f) - I(f) = \frac{1}{90}h^5 f^{(4)}(\xi_F)$
- Simpsonsumme: $\widehat{J}(f) - I(f) = \frac{1}{90\cdot 16}h^5 f^{(4)}(\xi_S)$

ergibt sich mit der Annahme $f^{(4)}(\xi_F) \approx f^{(4)}(\xi_S)$ ein Fehlerschätzer für $\widehat{J}(f)$:

$$J(f) - \widehat{J}(f) \approx \tfrac{1}{90}h^5 f^{(4)}(\xi_S)(1 - \tfrac{1}{16}) = 15(\widehat{J}(f) - I(f)),$$

d.h.

$$\widehat{J}(f) - I(f) \approx \tfrac{1}{15}(J(f) - \widehat{J}(f))$$

aus dem Vergleich der beiden Approximationen $\widehat{J}(f)$ und $J(f)$. Damit haben wir das folgende Verfahren.

Algorithmus 7.1: Adaptiver Simpson

Gegeben `TOL`;

Berechne $J := J(f)[a,b],\ \widehat{J} := \widehat{J}(f)[a,b]$;

$$I(f)[a,b] := \begin{cases} \widehat{J} & \text{falls } |J - \widehat{J}| < 15 \cdot \texttt{TOL} \\ I(f)[a,\frac{a+b}{2}] + I(f)[\frac{a+b}{2},b] & \text{sonst} \end{cases}$$

Das Abbruchkriterium ist dabei in den Verfeinerungen mit der aktuellen Intervalllänge zu skalieren

$$|J(f)[a_{\text{lokal}},b_{\text{lokal}}] - \widehat{J}(f)[a_{\text{lokal}},b_{\text{lokal}}]| < 15 \cdot \texttt{TOL} \cdot \frac{b_{\text{lokal}} - a_{\text{lokal}}}{b_{\text{global}} - a_{\text{global}}}.$$

Ist nämlich $R(f)$ der Gesamtfehler bzgl. des globalen Intervalls $[a,b]$ und $R_k(f)$ der Fehler im k-ten Teilintervall $[a_k,b_k]$, dann folgt

$$\begin{aligned} |R(f)| &= \left|\sum_{k=1}^{K} R_k(f)\right| \quad \leq \quad \sum_{k=1}^{K} |R_k(f)| \\ &< \sum_{k=1}^{K} \texttt{TOL}\frac{b_k - a_k}{b-a} \quad = \quad \frac{\texttt{TOL}}{b-a}\sum_{k=1}^{K} b_k - a_k \quad = \quad \texttt{TOL} \end{aligned}$$

sofern das Abbruchkriterium gilt.

Bemerkungen:

- Bei der Berechnung von $\widehat{J}(f)[a,b]$ kann man die drei Funktionsauswertungen von $J(f)[a,b]$ wieder verwenden, so dass nur zwei zusätzliche Funktionsauswertungen entstehen.
- Noch besser ist eine Fehlerabfrage mit *absoluten* und *relativen* Toleranzen `ATOL` und `RTOL`.
- In der Praxis wird man zusätzliche *Heuristiken* einführen, um den Algorithmus robust zu machen.
- Statt $\widehat{J}$ kann man auch die bessere Approximation $\widetilde{J} = \frac{1}{15}(16\widehat{J} - J)$, den sogenannten extrapolierten Wert, als Lösung hernehmen. Der Fehlerschätzer bezieht sich aber auf $\widehat{J}$ und nicht auf $\widetilde{J}$.
- Das gleichzeitige Unterteilen nach links und rechts bezeichnet man auch als *Randwertmethode*, im Gegensatz zur *Anfangswertmethode*, bei der man bei a startet und sich sukzessive bis nach b vorarbeitet. Die Namensgebung stammt von den Aufgabenstellungen bei der numerischen Simulation von Differentialgleichungen.

Analog kann man ein adaptives Verfahren auf der Basis der Trapezregel und Trapezsumme konstruieren.

7.4 Extrapolationsverfahren

Extrapolationsverfahren repräsentieren einen allgemeinen Ansatz, der auch bei der numerischen Differentiation oder der Lösung von gewöhnlichen Differentialgleichungen verwendet wird. Dabei wird ausgehend von einem Basisverfahren eine verbesserte Approximation konstruiert.

Idee der Extrapolation

Es bezeichne $T(h)$ die Näherung des exakten Integrals $I(f)$ aus einem numerischen Verfahren mit der Schrittweite h, z.B. aus der Rechtecksumme oder der Trapezsumme. Das Verfahren ist als konvergent, vorausgesetzt, d.h. es gilt

$$\lim_{h \to 0} T(h) = I(f).$$

Die Auswertung von $T(h)$ an der Stelle $h = 0$ ist nicht definiert. Je kleiner h gewählt wird, desto mehr Funktionsauswertungen von f sind erforderlich und umso höher der Rechenaufwand.

Es seien Näherungen $T(h_i)$ für Schrittweiten $h_1 > h_2 > \cdots > h_m > 0$ bereits bestimmt. Wir konstruieren eine (hoffentlich) bessere Approximation ohne wesentlichen Rechenaufwand wie folgt: Es sei $p \in \mathbb{P}_{m-1}$ das eindeutige Interpolationspolynom zu den m Stützpunkten $(h_i, T(h_i))$ für $i = 1, \ldots, m$. Dann setzen wir die Approximation $J(f) = p(0)$, siehe Abb. 26. Da das Interpolationspolynom außerhalb der Stützstellen ausgewertet wird, liegt eine *Extrapolation* vor. Die Auswertung des Polynoms erfolgt mit dem Aitken-Neville-Schema, siehe Abschnitt 5.4.

Die Polynominterpolation besitzt bei äquidistanten Stützstellen schlechte Approximationseigenschaften nahe den Rändern. Wir erwarten hier jedoch brauchbare Ergebnisse, sofern die Stützstellen h_i zum Punkt $h = 0$ hin verdichtet werden.

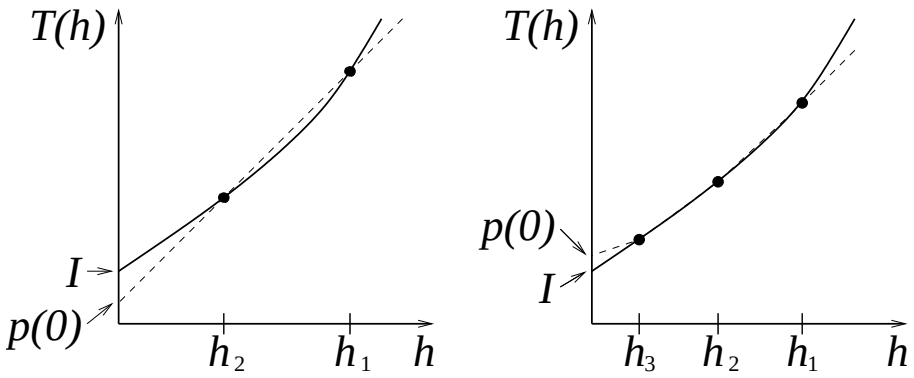


Abbildung 26: Idee der Extrapolation.

Asymptotische Entwicklung

Es bezeichne $T(h)$ die Näherung des exakten Integrals $I(f)$ aus einem numerischen Verfahren mit Schrittweite h. Häufig besitzt die Näherung eine *asymptotische Entwicklung* in Potenzen von h der Form

$$T(h) = \tau_0 + \tau_1 h + \tau_2 h^2 + \cdots + \tau_{m-1} h^{m-1} + \tau_m(h) h^m. \tag{7.3}$$

Dabei sind $\tau_0, \ldots, \tau_{m-1} \in \mathbb{R}$ Koeffizienten und die Funktion τ_m erfüllt

$$|\tau_m(h)| < C_m \qquad \text{für alle } \; h < H \tag{7.4}$$

mit einem konstanten $H > 0$. Es gilt dann

$$\tau_0 = \lim_{h \to 0} T(h) = I(f) \tag{7.5}$$

für ein konvergentes Verfahren, d.h. τ_0 ist der gesuchte Integralwert.

Für die Trapezsumme $T(h) = h(\frac{1}{2}f_0 + f_1 + \cdots + f_{n-1} + \frac{1}{2}f_n)$ gilt bei genügend glatten Integranden $f \in C^{2m}[a,b]$ die *Euler-MacLaurinsche Summenformel*

$$\begin{aligned} T(h) \;=\; & \int_a^b f(x)\,\mathrm{d}x + \sum_{k=1}^{m-1} h^{2k} \frac{B_{2k}}{(2k)!} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right) \\ & + h^{2m} \frac{B_{2m}}{(2m)!} (b-a) f^{(2m)}(\xi) \end{aligned} \tag{7.6}$$

mit den Bernoulli-Zahlen B_i ($B_2 = \frac{1}{6}, B_4 = -\frac{1}{30}, B_6 = \frac{1}{42}, B_8 = -\frac{1}{30}, \ldots$). In diesem Fall sind die Koeffizienten

$$\tau_{2k} = \frac{B_{2k}}{(2k)!} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right)$$

und $\tau_{2k+1} = 0$. Zulässige Konstanten (7.4) lauten

$$C_{2m} = \frac{B_{2m}}{(2m)!}(b-a) \max_{\xi \in [a,b]} \left| f^{(2m)}(\xi) \right| .$$

Somit ist dies eine Entwicklung in Potenzen von h^2.

Bemerkungen:

- Bei gewöhnlichen Reihen kann das Restglied durch den Übergang $m \to \infty$ beliebig klein gemacht werden, d.h. dort wäre $\tau_m(h) h^{2m} \to 0$ für eine (genügend kleine) feste Schrittweite h. Bei asymptotischen Entwicklungen braucht das Restglied dagegen für $m \to \infty$ weder zu existieren noch muss es gegen 0 streben.
- Stattdessen verlangt man die Beschränktheit von $\tau_m(h)$ für alle hinreichend kleinen h, insbesondere $h \to 0$, wobei m fest ist. Dann gilt (7.5).
- Oft ist f nicht unbegrenzt differenzierbar. Dann ist der Grenzübergang $m \to \infty$ nicht sinnvoll, der Übergang $h \to 0$ dagegen schon (und konvergiert).
- Für periodische Funktionen $f \in C^{2m}$ mit der Periode $b - a$ gilt zudem $f^{(l)}(a) = f^{(l)}(b)$ für alle $l = 0, 1, \ldots, 2m$. Somit entfallen laut (7.6) alle h^{2k}-Terme ($k < m$). In diesem Fall wendet man keine Extrapolation an, da die Trapezsumme allein schon die optimale Approximation liefert.

Die asymptotische Entwicklung (7.3) kann man schreiben als

$$T(h) = \tilde{p}(h) + \tau_m(h)h^m$$

mit einem Polynom $\tilde{p} \in \mathbb{P}_{m-1}$. Es gilt $\tilde{p}(0) = \tau_0$, d.h. der gesuchte Wert. Das Interpolationspolynom $p \in \mathbb{P}_{m-1}$ ist definiert durch

$$p(h_i) = T(h_i) \qquad \text{für } \; i = 1, \ldots, m. \tag{7.7}$$

Das Restglied wurde vernachläßigt. Bei kleinem Restglied erwarten wir unter geeigneten Voraussetzungen $p(0) \approx \tilde{p}(0) = \tau_0$.

Mit der asymptotischen Entwicklung (7.3) kann man die führenden Fehlerterme eliminieren:

Ansatz mit 2 Schrittweiten h_1 und h_2:

$$\begin{aligned} T(h_1) &= \tau_0 + \tau_1 h_1 + \tau_2 h_1^2 + \cdots \qquad | \cdot -h_2/h_1 \\ T(h_2) &= \tau_0 + \tau_1 h_2 + \tau_2 h_2^2 + \cdots \qquad | \\ \Rightarrow \qquad \frac{h_1 T(h_2) - h_2 T(h_1)}{h_1 - h_2} &= \tau_0 + \tau_2 \frac{h_1 h_2^2 - h_1^2 h_2}{h_1 - h_2} + \cdots . \end{aligned}$$

Für $h_1 = h$, $h_2 = h/2$ ergibt sich z.B.

$$2T(\tfrac{h}{2}) - T(h) = \tau_0 - \tau_2 \tfrac{1}{2} h^2 + \mathcal{O}(h^3).$$

Geschickte Kombination von $T(h_1)$ und $T(h_2)$ eliminiert den führenden Fehlerterm und liefert eine bessere Approximation!

Aus m Approximationen

$$\begin{aligned} T(h_1) &= \tau_0 + \tau_1 h_1 + \tau_2 h_1 + \cdots + \tau_{m-1} h_1^{m-1} + \tau_m(h_1) h_1^m \\ T(h_2) &= \tau_0 + \tau_1 h_2 + \tau_2 h_2 + \cdots + \tau_{m-1} h_2^{m-1} + \tau_m(h_2) h_2^m \\ &\vdots \\ T(h_m) &= \tau_0 + \tau_1 h_m + \tau_2 h_m + \cdots + \tau_{m-1} h_m^{m-1} + \tau_m(h_m) h_m^m \end{aligned}$$

ergibt sich schließlich

$$\text{Kombination} = \tau_0 + 0 + \cdots + 0 + \mathcal{O}((\max\{h_1, \ldots, h_m\})^m).$$

Dieses auf der asymptotischen Entwicklung basierende Vorgehen der sukzessiven Elimination der führenden Fehlerterme stammt von Richardson. Man spricht auch von *Richardson–Extrapolation.*

Wie lässt sich dies nun praktisch durchführen? Man könnte die Fehlerelimination durch den Gauß-Algorithmus von LGSen realisieren (bzw. programmieren). Bequemer ist jedoch das Interpolationspolynom (7.7) an der Stelle $h = 0$ auszuwerten, d.h. nach 0 zu extrapolieren. Man kann zeigen, dass der resultierende Näherungswert identisch ist zur Kombination aus der Elimination der führenden Fehlerterme.

Das Interpolationspolynom ist

$$p(h) = \gamma_0 + \gamma_1 h + \gamma_2 h^2 + \cdots + \gamma_{m-1} h^{m-1}$$

und somit $p(0) = \gamma_0$. Die Interpolationsbedingungen liefern

$$\begin{aligned} T(h_1) &= \gamma_0 + \gamma_1 h_1 + \gamma_2 h_1^2 + \cdots + \gamma_{m-1} h_1^{m-1} \\ T(h_2) &= \gamma_0 + \gamma_1 h_2 + \gamma_2 h_2^2 + \cdots + \gamma_{m-1} h_2^{m-1} \\ &\vdots \\ T(h_m) &= \gamma_0 + \gamma_1 h_m + \gamma_2 h_m^2 + \cdots + \gamma_{m-1} h_m^{m-1}. \end{aligned}$$

Die Elimination der führenden Fehlerterme mit $\tau_1, \ldots, \tau_{m-1}$ in der asymptotischen Entwicklung entspricht somit der Elimination der Terme mit $\gamma_1, \ldots, \gamma_{m-1}$ des Polynom p. Zurück bleibt die gesuchte Kombination.

Romberg-Quadratur

Die Romberg-Quadratur ist gerade die Richardson-Extrapolation auf Basis der Trapezsumme. Laut (7.6) besitzt die Trapezsumme $T(h)$ die asymptotische Entwicklung

$$T(h) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \cdots + \tau_{m-1} h^{2m-2} + \tau_m(h) h^{2m}$$

in Potenzen von h^2. $T(h)$ kann numerisch berechnet werden, $\tau_0 = I(f)$ ist gesucht. Nun sei $q \in \mathbb{P}_{m-1}$ das Interpolationspolynom zu den Daten $(h_i^2, T(h_i))$, d.h.

$$T(h_i) = q(h_i^2) \qquad \text{für } i = 1, \ldots, m.$$

In der Praxis verwendet man häufig geometrische Schrittweitenfolgen

$$h_i = c^{i-1} h \qquad \text{mit einem } 0 < c < 1$$

für ein vorgegebenes h. Günstig ist die Wahl $c = \frac{1}{2}$, da dann alle Funktionsauswertungen $f(x_k)$ in der Trapezsumme $T(h_i)$ auch in $T(h_{i+1})$ auftreten und somit nicht neu berechnet werden müssen.

Zu bestimmen ist die Approximation $q(0) \approx \tau_0$. Dazu verwenden wir das Aitken-Neville-Schema, siehe Abschnitt 5.4. Das Rekursionsschema lautet

$$P_{i,j} = P_{i,j-1} + \frac{x - x_i}{x_i - x_{i-j}} \left(P_{i,j-1} - P_{i-1,j-1} \right).$$

In unserem Fall ist $x_i = h_i^2$ und dadurch

$$\frac{x - x_i}{x_i - x_{i-j}} = \frac{-h_i^2}{h_i^2 - h_{i-j}^2} = \frac{1}{\frac{h_{i-j}^2}{h_i^2} - 1}.$$

Wir erhalten das folgenden Verfahren.

Algorithmus 7.2: *Romberg-Quadratur*

$\underline{\text{für } k = 1, \ldots, m}$:
  wähle n_k
  $h_k := (b-a)/n_k$ [Schrittweite]
  $T_{k,1} := T(h_k)$ [Trapezsumme]
  $\underline{\text{für } l = 2, \ldots, k}$:
    $T_{k,l} = T_{k,l-1} + \frac{T_{k,l-1} - T_{k-1,l-1}}{h_{k-l+1}^2 / h_k^2 - 1}$ [Aitken–Neville]
$q(0) := T_{m,m}$

Der Algorithmus liefert das *Extrapolationstableau* :

$$\begin{array}{lccccccccc}
T(h_1) = T_{1,1} & & & & & & & & & \\
 & \searrow & & & & & & & & \\
T(h_2) = T_{2,1} & \rightarrow & T_{2,2} & & & & & & & \\
 & \searrow & & \searrow & & & & & & \\
T(h_3) = T_{3,1} & \rightarrow & T_{3,2} & \rightarrow & T_{3,3} & & & & & \\
 & \searrow & & \searrow & & \searrow & & & & \\
 & \vdots & & \vdots & & & \ddots & & & \\
 & \searrow & & \searrow & & \searrow & & \searrow & & \\
T(h_m) = T_{m,1} & \rightarrow & T_{m,2} & \rightarrow & T_{m,3} & \rightarrow & \cdots & \rightarrow & T_{m,m}
\end{array}$$

Alle Werte im Extrapolationstableau sind Approximationen des gesuchten Integrals $I(f)$. Bezüglich der Güte der Näherungen gilt der folgende Satz.

Satz 7.1 *Fehler in der Romberg-Quadratur*
Es sei $f \in C^{2m}[a,b]$ *und* $h_i = c^{i-1}h$ *mit* $0 < c < 1$. *Dann gilt*

$$\left| T_{i,k} - \int_a^b f(x)\ dx \right| \leq h^{2k} c^{2ik} c^{-k^2-3k} \frac{|B_{2k}|}{(2k)!}(b-a) \max_{\xi\in[a,b]} \left| f^{(2k)}(\xi) \right|$$

mit den Bernoulli-Zahlen B_{2k}.

Für festes k konvergiert der Fehler mit $i \to \infty$ gegen null, d.h. es liegt Konvergenz in jeder Spalte des Extrapolationstableaus vor. Die Ordnung der Konvergenz in der k-ten Spalte beträgt $2k$. Die Konvergenz entlang der Diagonalen, d.h. $T_{i+j,k+j}$ für $j \to \infty$ läßt sich aus diesem Satz nicht folgern. Sie gilt jedoch für z.B. $c = \frac{1}{2}$, kann aber langsam sein.

Man kann eine adaptive Romberg-Quadratur durchführen, bei der das Extrapolationstableau zeilenweise aufgebaut wird. Als Fehlerschätzer wird die Differenz $|T_{k,k-1} - T_{k,k}|$ verwendet. Es werden so viele Zeilen berechnet, bis ein Abbruchkriterium erfüllt ist.

7.5 Gauß-Quadratur

O.E.d.A. betrachten wir als Integrationsintervall $[-1, 1]$, da ansonsten eine affin-lineare Transformation das Intervall $[a, b]$ in $[-1, 1]$ überführt. Bei der Gauß-Quadratur wählt man Knoten x_i und Gewichte g_i für $i = 0, 1, \ldots, n$ so, dass

$$I(p) = \int_{-1}^{+1} p(x)\,\mathrm{d}x = \sum_{i=0}^{n} g_i p(x_i) = J(p) \tag{7.8}$$

für alle Polynome p *möglichst hohen Grades* gilt.

Also $I(p) = J(p)$ für alle $p \in \mathbb{P}_K$ mit K größtmöglich. Bei den abgeschlossenen Newton-Cotes-Formeln mit $n + 1$ Knoten gilt $K = n$ für n ungerade und $K = n + 1$ für n gerade. Bei den Newton-Cotes-Formeln werden die Knoten äquidistant gewählt. Die Idee ist nun, ein höheres K zu erzielen, indem die Knoten und Gewichte optimal gewählt werden.

Welches K kann noch erreicht werden? Sei eine beliebige Quadraturformel gegeben. Dann betrachte das Polynom

$$q(x) = (x - x_0)^2 (x - x_1)^2 \cdots (x - x_n)^2 \in \mathbb{P}_{2n+2}.$$

Es gilt $q(x) \geq 0$ für alle x und insbesondere $I(q) > 0$. Jedoch haben wir

$$J(q) = \sum_{i=0}^{n} g_i \cdot \underbrace{q(x_i)}_{=0} = 0,$$

d.h. $I(q) \neq J(q)$. Somit wissen wir $K \leq 2n + 1$.

Wir haben wegen der Linearität der Funktionale

$$I(p) = J(p) \;\text{ f.a. } p \in \mathbb{P}_K \quad \Leftrightarrow \quad I(x^l) = J(x^l) \;\text{ für }\; l = 0, 1, \ldots, K.$$

Für $K = 2n + 1$ erhalten wir $2n + 2$ Gleichungen. Gleichzeitig hat die Quadraturformel $n + 1$ Knoten und Gewichte, d.h. $2n + 2$ Freiheitsgrade. Wir erhalten ein System mit genau so vielen Gleichungen wie Unbekannten. Die Frage ist, ob dieses nichtlineare Gleichungssystem für jedes n (eindeutig) lösbar ist.

Beispiel:

Fall $n = 1$: Wie sind x_0, x_1 und g_0, g_1 zu wählen, so dass

$$\int_{-1}^{+1} f(x)\,\mathrm{d}x = g_0 f(x_0) + g_1 f(x_1)$$

für alle Polynome vom Grad kleiner gleich 3 gilt?

Lösung: Setze die Polynome 1, x, x^2 und x^3 ein, dann ergibt sich das Gleichungssystem

$$g_0 + g_1 = \int_{-1}^{+1} 1\,\mathrm{d}x = 2, \qquad g_0 x_0 + g_1 x_1 = \int_{-1}^{+1} x\,\mathrm{d}x = 0,$$
$$g_0 x_0^2 + g_1 x_1^2 = \int_{-1}^{+1} x^2\,\mathrm{d}x = \frac{2}{3}, \qquad g_0 x_0^3 + g_1 x_1^3 = \int_{-1}^{+1} x^3\,\mathrm{d}x = 0.$$

Diese 4 Gleichungen mit 4 Unbekannten sind eindeutig lösbar: Multipliziere die Gleichung rechts oben mit x_0^2 und subtrahiere die Gleichung rechts unten:

$\Rightarrow \quad g_1 x_0^2 x_1 - g_1 x_1^3 = 0 \quad \Rightarrow \quad x_0^2 = x_1^2$

$\Rightarrow \quad x_1 = -x_0$ (keine doppelte Stützstelle!) $\quad \Rightarrow \quad g_0 = g_1$

$$\Rightarrow \quad g_0 = g_1 = 1 \quad \Rightarrow \quad x_0 = -\sqrt{\frac{1}{3}},\ x_1 = \sqrt{\frac{1}{3}}$$

Ergebnis: 2 Stützstellen x_0, x_1 und 2 Gewichte g_0, g_1 reichen aus!

Die Faßregel zum Vergleich ist ebenfalls exakt für $p \in \mathbb{P}_3$, benötigt aber 3 Stützstellen. Die eben hergeleitete Quadraturformel heißt *2-Punkt Gauß-Legendre-Regel*.

Herleitung der n-Punkt Gauß-Legendre-Regel

Angenommen es existieren Knoten und Gewichte, so dass $I(p) = J(p)$ für alle $p \in \mathbb{P}_{2n+1}$. Wir betrachten dann bei beliebigem $q \in \mathbb{P}_n$ das Polynom

$$q(x) \cdot \underbrace{(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_n)}_{=:L_{n+1}} \in \mathbb{P}_{2n+1}.$$

Der Grad von L_{n+1} ist $n + 1$. Es folgt $J(q \cdot L_{n+1}) = 0$. Somit muss gelten

$$\int_{-1}^{+1} q(x) \cdot L_{n+1}(x) \, \mathrm{d}x = 0 \qquad \text{für alle } \; q \in \mathbb{P}_n. \tag{7.9}$$

Wir definieren in $C[-1, 1]$ das *Skalarprodukt*

$$\langle f, g \rangle := \int_{-1}^{+1} f(x) \cdot g(x) \, \mathrm{d}x$$

(vergleiche Hilbertraum $\mathcal{L}_2$). Dann bedeutet (7.9), dass $L_{n+1}(x)$ bzgl. $\langle \cdot, \cdot \rangle$ *orthogonal* zu allen Polynomen $q \in \mathbb{P}_n$ ist, d.h. $\langle q, L_{n+1} \rangle = 0$.

Als Hilfsmittel konstruieren wir ein System aus orthogonalen Polynomen $(L_n)_{n \in \mathbb{N}}$, also

$$\langle L_n, L_m \rangle \begin{cases} = 0 & \text{für } \; n \neq m, \\ \neq 0 & \text{für } \; n = m. \end{cases}$$

Das System kann aus der Monom-Basis $(x^n)_{n \in \mathbb{N}}$ mit dem Gram-Schmidt-Orthogonalisierungsverfahren konstruiert werden. Setze $L_0(x) = 1$, dann lautet die Rekursion

$$L_n(x) := x^n - \sum_{i=0}^{n-1} \frac{\langle x^n, L_i(x) \rangle}{\langle L_i(x), L_i(x) \rangle} L_i(x).$$

Das gesuchte L_n ist eindeutig gegeben (bis auf Normierungsfaktor) in Form des *Legendre-Polynoms* vom Grad n

$$L_n(x) = \frac{n!}{(2n)!} D^n \left((x^2 - 1)^n \right). \tag{7.10}$$

Die ersten Legendre-Polynome sind:

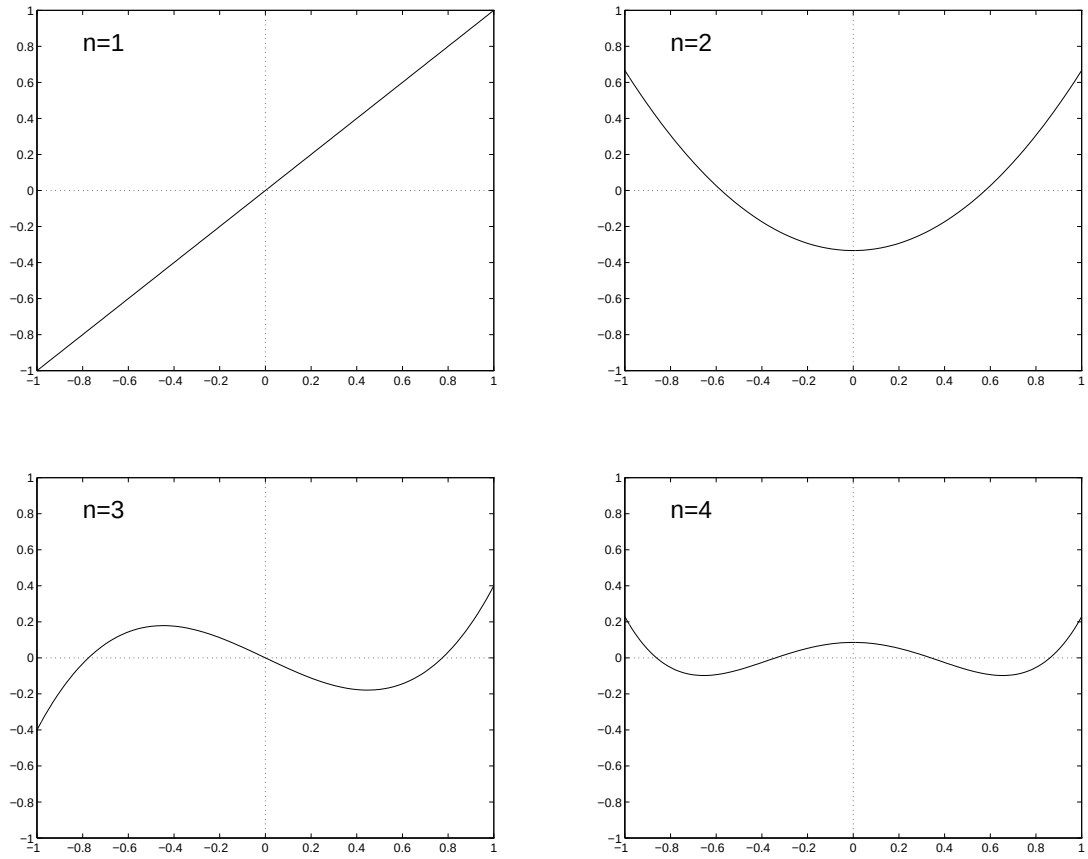


Abbildung 27: Skizze der Legendre-Polynome L_n für $n = 1, 2, 3, 4$.

$$n = 1: \;\; L_1(x) = x, \qquad\qquad n = 2: \;\; L_2(x) = x^2 - \tfrac{1}{3},$$

$$n = 3: \;\; L_3(x) = x^3 - \tfrac{3}{5}x, \qquad\qquad n = 4: \;\; L_4(x) = x^4 - \tfrac{6}{7}x^2 + \tfrac{3}{35}.$$

Die Orthogonalität lässt sich auch explizit mit der Formel (7.10) beweisen:

$$\begin{aligned}
&\int_{-1}^{+1} q(x) D^n (x^2-1)^n \,\mathrm{d}x \\
&= \underbrace{q(x) D^{n-1}(x^2-1)^n \big|_{-1}^{1}}_{=0} - \int_{-1}^{+1} q'(x) D^{n-1}(x^2-1)^n \,\mathrm{d}x = \cdots = \\
&= \underbrace{(-1)^{n-1}\, q^{(n-1)}(x) D (x^2-1)^n \big|_{-1}^{1}}_{=0} + (-1)^n \int_{-1}^{+1} \underbrace{q^{(n)}}_{\equiv 0}(x) D(x^2-1)^n \,\mathrm{d}x \quad = \quad 0,
\end{aligned}$$

da $(x^2-1)^n = (x+1)^n (x-1)^n$ n-fache Nullstellen bei $x = \pm 1$ hat und $D^k (x^2-1)^n$ damit $(n-k)$-fache Nullstellen.

Satz 7.2 *Nullstellen des Orthogonalpolynoms*

Das Legendre-Polynom L_n besitzt n paarweise verschiedene Nullstellen x_i im offenen Intervall $(-1, 1)$.

Beweis:

Es seien $-1 < x_1 < x_2 < \cdots < x_l < 1$ die paarweise verschiedenen Nullstellen von L_n im Intervall $(-1, 1)$, an denen L_n einen Vorzeichenwechsel hat. Da $\operatorname{grad}(L_n) = n$ gilt, folgt sofort $l \leq n$. Wir zeigen $l = n$. Angenommen es gelte $l < n$. Wir definieren das Polynom

$$q(x) = (x - x_1)(x - x_2) \cdots (x - x_l) \in \mathbb{P}_l$$

(falls keine Nullstelle mit Vorzeichenwechsel in $(-1, 1)$ existiert, setze $q(x) = 1$). Die Vorzeichenwechsel von q erfolgen genau in $x_1, \ldots, x_l$. Folglich gilt $q \cdot L_n \geq 0$ oder $q \cdot L_n \leq 0$ in ganz $(-1, 1)$. Da nur endlich viele Nullstellen auftreten ist $\langle q \cdot L_n \rangle \neq 0$. Dies ist ein Widerspruch zur Orthogonalität $\langle p \cdot L_n \rangle = 0$ für alle $p \in \mathbb{P}_{n-1}$. □

Für gegebenes n wählen wir nun als Knoten die $n + 1$ paarweise verschiedenen Nullstellen des Orthogonalpolynoms L_{n+1}. Für beliebiges $p \in \mathbb{P}_{2n+1}$ liefert Polynomdivision die Darstellung

$$p(x) = q(x)L_{n+1}(x) + r(x)$$

mit einem $q \in \mathbb{P}_n$ und einem Rest $r \in \mathbb{P}_n$. Die Orthogonalität führt auf

$$\begin{aligned} I(p) &= \int_{-1}^{+1} p(x)\,\mathrm{d}x = \underbrace{\int_{-1}^{+1} q(x) \cdot L_{n+1}(x)\,\mathrm{d}x}_{=0} + \int_{-1}^{+1} r(x)\,\mathrm{d}x \\ &= \int_{-1}^{+1} r(x)\,\mathrm{d}x = I(r). \end{aligned}$$

Die Quadraturformel (7.8) zeigt uns

$$J(p) = J(q \cdot L_{n+1}) + J(r) = \left(\sum_{i=0}^{n} g_i \cdot q(x_i) \cdot \underbrace{L_{n+1}(x_i)}_{=0} \right) + J(r) = J(r)$$

für beliebige Wahl der Gewichte. Wir erhalten somit $I(p) = J(p)$ für alle $p \in \mathbb{P}_{2n+1}$ falls nur noch $I(r) = J(r)$ für alle $r \in \mathbb{P}_n$ gilt. Dies erreichen wir einfach durch die Wahl der Gewichte gemäß einer interpolatorischen

Quadraturformel, siehe Abschnitt 7.2. Da r identisch zu seinem eigenen Interpolationspolynom zu den Knoten ist, folgt

$$I(r) = I\left(\sum_{i=0}^{n} r(x_i)\ell_i(x)\,\mathrm{d}x\right) = \sum_{i=0}^{n}\left(\int_{-1}^{+1}\ell_i(x)\,\mathrm{d}x\right) r(x_i)$$

mit den Lagrange-Polynomen $\ell_0, \ell_1, \ldots, \ell_n$. Die Gewichte sind somit

$$g_i = \int_{-1}^{+1} \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,\mathrm{d}x \quad \text{für } i = 0, 1, \ldots, n \tag{7.11}$$

und können aus den Knoten $x_0, x_1, \ldots, x_n$ berechnet werden. Für die Bestimmung der Nullstellen der Orthogonalpolynome existieren eigene Techniken über Eigenwertprobleme.

Satz 7.3: *Gauß-Legendre-Quadratur*

Seien die Stützstellen x_i für $i = 0, 1, \ldots, n$ die Nullstellen des Legendre-Polynoms L_{n+1} aus (7.10) und die Gewichte g_i für $i = 0, 1, \ldots, n$ nach (7.11) bestimmt. Dann gilt

$$\int_{-1}^{+1} p(x)\,\mathrm{d}x = \sum_{i=0}^{n} g_i p(x_i) \qquad \textit{für alle } p \in \mathbb{P}_{2n+1},$$

und für $f \in C^{2n+2}[-1, 1]$ gilt die Fehlerformel

$$\int_{-1}^{+1} f(x)\,\mathrm{d}x - \sum_{i=0}^{n} g_i f(x_i) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\int_{-1}^{1} L_{n+1}(x)^2\,\mathrm{d}x.$$

Die Aussage für die Polynome folgt durch unsere Konstruktion von oben. Die Fehlerformel kann man über die Restgliedformel der Polynominterpolation zeigen, wobei in den Knoten sowohl die Funktion als auch die erste Ableitung interpoliert wird.

Für die Qualität der Quadraturformel gibt der folgende Satz noch eine vorteilhafte Aussage.

Satz 7.4: *Gewichte in Gauß-Legendre-Quadratur*

Die Gewichte g_i für $i = 0, 1, \ldots, n$ der $(n+1)$-Punkt Gauß-Legendre Quadratur sind stets positiv.

Beweis:

Betrachte die Polynome

$$q_k(x) = \prod_{j=0, j\neq k}^{n} (x - x_j)^2 \in \mathbb{P}_{2n}$$

für $k = 0, 1, \ldots, n$. Dann ist $q_k(x) \geq 0$ für alle x. Insbesondere folgt $I(q_k) > 0$, da q_k nur an endlich vielen Stellen null ist. Wir erhalten wegen $I(p) = J(p)$ für alle $p \in \mathbb{P}_{2n+1}$

$$0 < I(q_k) = J(q_k) = \sum_{i=0}^{n} g_i q_k(x_i) = 0 + \cdots + 0 + g_k q_k(x_k) + 0 + \cdots + 0 = g_k q_k(x_k).$$

Da $q_k(x_k) > 0$ ist, muss $g_k > 0$ gelten. □

Die Tabelle 3 zeigt die Knoten und Gewichte der $(n+1)$-Punkt Gauß-Legendre Quadratur in den Fällen $n = 0, 1, 2, 3$. Die Knoten und Gewichte liegen stets symmetrisch zur Intervallmitte $x = 0$. Die Summe der Gewichte entspricht der Intervalllänge 2.

Falls ein Integral über einem Intervall $[a, b]$ vorliegt, so erhält man mit der Substitution $t = \frac{b-a}{2}x + \frac{a+b}{2}$

$$\int_a^b f(t)\, \mathrm{d}t = \tfrac{b-a}{2} \int_{-1}^{+1} f\left(\tfrac{b-a}{2}x + \tfrac{a+b}{2}\right)\, \mathrm{d}x.$$

Dadurch kann die Gauss-Legendre-Quadratur in $[-1, 1]$ angewendet werden. Alternativ kann man auch Knoten und Gewichte transformieren über

$$\tilde{x}_i = \tfrac{b-a}{2}x_i + \tfrac{a+b}{2}, \qquad \tilde{g}_i = \tfrac{b-a}{2}g_i$$

und so die Quadraturformel auf das Intervall $[a, b]$ anpassen.

Tabelle 3: Knoten und Gewichte in Gauß-Legendre-Quadratur.

	x_i	g_i
$n=0$		
1	0	2
$n=1$		
1	$-\sqrt{\frac{1}{3}} \approx -0.57735026919$	1
2	$+\sqrt{\frac{1}{3}} \approx 0.57735026919$	1
$n=2$		
1	$-\sqrt{\frac{3}{5}} \approx -0.774596669241$	$\frac{5}{9} \approx 0.555555555556$
2	0	$\frac{8}{9} \approx 0.888888888889$
3	$+\sqrt{\frac{3}{5}} \approx 0.774596669241$	$\frac{5}{9} \approx 0.555555555556$
$n=3$		
1	$-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}} \approx -0.861136311594053$	$\frac{18-\sqrt{30}}{36} \approx 0.347854845137454$
2	$-\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}} \approx -0.339981043584856$	$\frac{18+\sqrt{30}}{36} \approx 0.652145154862546$
3	$+\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}} \approx 0.339981043584856$	$\frac{18+\sqrt{30}}{36} \approx 0.652145154862546$
4	$+\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}} \approx 0.861136311594053$	$\frac{18-\sqrt{30}}{36} \approx 0.347854845137454$

Gauß-Quadratur mit Gewichtsfunktionen

Die oben gezeigte Herleitung der n-Punkt Gauß-Legendre-Regel läßt sich verallgemeinern auf Integrale

$$I(f) := \int_a^b \omega(x) f(x)\, \mathrm{d}x,$$

wobei $\omega \in C[a,b]$ eine positive *Gewichtsfunktion* ist. Dabei sind sogar uneigentliche Integrale zugelassen.

Ist beispielsweise X eine normalverteilte Zufallsvariable und $f : \mathbb{R} \to \mathbb{R}$ eine stetige Funktion, dann ist der Erwartungswert gerade

$$\mathbb{E}(f(X)) = \int_{-\infty}^{+\infty} \tfrac{1}{\sqrt{2\pi}}\, \mathrm{e}^{-\frac{1}{2}x^2} f(x)\, \mathrm{d}x.$$

Die Gewichtsfunktion ist somit die Dichtefunktion

$$\omega(x) = \tfrac{1}{\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{1}{2}x^2} > 0.$$

Bei einer allgemeinen Verteilung setzt man als Integrationsintervall gerade den Träger ($\operatorname{supp}\omega$) der Dichtefunktion.

Wieder ist eine Quadraturformel

$$J(f) := \sum_{i=0}^{n} w_i f(x_i)$$

gesucht, die für Polynome möglichst hohen Grades exakt sein soll. Folgende Aussagen gelten (hier ohne Beweis, siehe dazu Stoer):

1. Es existieren eindeutig bestimmte normierte Polynome Q_n für jedes $n \in \mathbb{N}$ (mit $Q_0 = 1$), die bezüglich des auf $C[a,b]$ bzw. $C(\mathbb{R})$ definierten Skalarproduktes

   $$\langle f, g\rangle := \int_a^b \omega(x) f(x) g(x)\, \mathrm{d}x$$

   orthogonal sind:

   $$\langle Q_i, Q_j\rangle = 0 \qquad \text{für } i \neq j.$$

   Man kann sie z.B. mit dem Gram-Schmidt-Verfahren erhalten.

2. Die Nullstellen $x_0, x_1, \ldots, x_n$ von Q_{n+1} sind alle reell, paarweise verschieden und im Inneren von $[a, b]$ bzw. $(-\infty, +\infty)$. Die Nullstellen lassen sich aus einem Eigenwertproblem bestimmen.

3. Die Gewichte w_i ergeben sich aus dem linearen Gleichungssystem

$$\underbrace{\begin{pmatrix} Q_0(x_0) & \cdots & Q_0(x_n) \\ \vdots & \ddots & \vdots \\ Q_n(x_0) & \cdots & Q_n(x_n) \end{pmatrix}}_{\text{invertierbar !}} \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \langle Q_0, Q_0 \rangle \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

   denn

$$\sum_{i=0}^{n} w_i Q_k(x_i) = \int_a^b \omega(x) Q_k(x) \, \mathrm{d}x = \langle Q_k, Q_0 \rangle$$

   für beliebiges k.

4. Die so bestimmte Quadraturformel ist exakt für alle $p \in \mathbb{P}_{2n+1}$, Exaktheit für $\mathbb{P}_{2n+2}$ ist nicht erreichbar.

5. Fehlerformel für $f \in C^{2n+2}[a, b]$:

$$\int_a^b \omega(x) f(x) \, \mathrm{d}x - \sum_{i=0}^{n} w_i f(x_i) = C_n \, \frac{f^{(2n+2)}(\xi)}{(2n+2)!}.$$

Im Fall einer Wahrscheinlichkeitsdichte $\omega(x)$ sind die korrespondierenden Orthogonalpolynome im Fall von Standardverteilungen wohlbekannt und tragen eigene Namen.

Verteilung	Dichtefkt.	Träger	Orthogonalpolynome
Gleichvert.	$\omega(x) = \frac{1}{2}$	$[-1, 1]$	Legendre-Polynome
Normalvert.	$\omega(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-x^2/2}$	$(-\infty, +\infty)$	Hermite-Polynome
Exponentialvert.	$\omega(x) = C\mathrm{e}^{-\lambda x}$	$[0, +\infty)$	Laguerre-Polynome
Beta-Vert.	$\omega(x) = Cx^{\mu-1}(1-x)^{\nu-1}$	$[0, 1]$	Jacobi-Polynome

Im Fall der Normalverteilung nennt man die korrespondierende Quadratur dann Gauß-Hermite-Quadratur, etc.

Vor- und Nachteile der Gauß-Quadratur

+ sehr effizient (für n Funktionsauswertungen das genaueste Ergebnis)

− aber nicht adaptiv (Fehlerkontrolle?)

In der Praxis ist die Gauß-Quadratur vor allem bei 2- und 3-dimensionalen Integralen wichtig, z. B. in der *Methode der finiten Elemente.*

Kapitel 8

Iterative Lösung großer linearer Gleichungssysteme

In diesem Kapitel wird die iterative Lösung von linearen Gleichungssystemen

$$Ax = b, \qquad A \in \mathbb{R}^{n\times n}, \quad \det A \neq 0, \quad x, b \in \mathbb{R}^n \tag{8.1}$$

behandelt. Da eine LR-Zerlegung const. $\cdot\, n^3$ Rechenoperationen erfordert, wird eine direkte Lösung für großes n extrem aufwendig. In den Anwendungen sind große Matrizen A typischerweise schwach besetzt (englisch *sparse*), d.h. nur wenige Elemente sind ungleich Null. Ein Weg, diese Besetzungsstruktur auszunutzen, führt auf Varianten der LR-Zerlegung, bei denen der Eliminationsprozess die Anzahl der entstehenden, von Null verschiedenen Einträge (englisch *fill-in*) minimiert, meist mit Hilfe der Graphentheorie und eingeschränkter Pivotsuche. Da nur Rechenoperationen mit von Null verschiendenen Einträgen auch real durchgeführt werden, sind diese *direct sparse solver* für den Bereich $500 \leq n \leq 50000$ eine bewährte Wahl, siehe in MATLAB den `sparse`-Befehl. Schwerpunkt im folgenden ist aber eine Einführung in iterative Verfahren, die auch bei deutlich größeren Dimensionen zum Erfolg führen. Dabei soll jeder Iterationsschritt möglichst wenig aufwendig sein, d.h. der Aufwand soll den einer Matrix-Vektormultiplikation nicht übersteigen (wobei die Besetzungsstruktur selbstverständlich ausgenutzt wird und keine unnötigen Operationen mit Nullen durchgeführt werden).

8.1 Stationäre Iterationsverfahren

Ein stationäres Iterationsverfahren zur näherungsweisen Lösung von (8.1) besitzt die Form

$$x^{k+1} = \Phi(x^k), \quad k = 0, 1, 2, \ldots \tag{8.2}$$

bei vorgebenen Startvektor x^0. Stationär bedeutet, dass die Funktion Φ in jedem Iterationsschritt identisch gewählt wird. Die exakte Lösung x vonn (8.1) soll der einzige Fixpunkt der Iteration sein.

Eine Iterationsvorschrift entsteht durch die Wahl einer regulären Matrix $B \in \mathbb{R}^{n\times n}$ aus der Zerlegung

$$Bx + (A - B)x = b \quad \Rightarrow \quad Bx^{k+1} + (A - B)x^k = b, \tag{8.3}$$

welche ein lineares Gleichungssystem für x^{k+1} darstellt. Aufgelöst erhält man

$$x^{k+1} = x^k - B^{-1}(Ax^k - b) = (I - B^{-1}A)x^k + B^{-1}b. \tag{8.4}$$

Zur Berechnung von x^{k+1} ist ein Matrix-Vektor-Produkt bezüglich A und die Lösung eines linearen Gleichungssystems bezüglich B notwendig. Die Iterationsmatrix $I - B^{-1}A$ dient nur zur Untersuchung der Konvergenz.

Die Matrix B soll in diesem Zusammenhang zwei Eigenschaften besitzen:

1. Lineare Gleichungssysteme mit der Matrix B sollen leicht auflösbar sein, damit die Iteration mit wenig Aufwand durchführbar ist.
2. Die Matrix B soll A gut approximieren, d.h. wesentliche Informationen aus A enthalten, damit die Konvergenz der Iteration gesichert ist.

Zur Untersuchung der Konvergenz ist der Begriff des *Spektralradius* von zentraler Bedeutung. Für $A \in \mathbb{R}^{n\times n}$ ist der *Spektralradius* definiert durch

$$\rho(A) := \max_{i=1,\ldots,n} |\lambda_i|, \tag{8.5}$$

wobei $\lambda_i \in \mathbb{C}$ die Eigenwerte von A sind.

Satz 8.1: *Konvergenz stationärer Verfahren*

Das Iterationsverfahren (8.4) ist für beliebigen Startwert genau dann konvergent, wenn gilt

$$\rho(I - B^{-1}A) < 1. \tag{8.6}$$

Hinreichend für die Konvergenz von (8.4) bei beliebigem Startwert ist die Bedingung

$$||I - B^{-1}A|| < 1, \tag{8.7}$$

wobei $||\cdot||$ eine beliebige Matrixnorm ist, die konsistent zu einer bestimmten Vektornorm.

Beweis:

Für den Fehler $f^k := x^k - x$ hat man mit

$$\begin{aligned} x^{k+1} &= (I - B^{-1}A)x^k + B^{-1}b \\ x &= (I - B^{-1}A)x + B^{-1}b \end{aligned}$$

durch Subtraktion die Rekursionsformel

$$f^{k+1} = (I - B^{-1}A)f^k$$

und damit

$$f^k = (I - B^{-1}A)^k f^0, \quad k = 0, 1, 2, \ldots .$$

Sei (8.4) konvergent. Ist λ ein Eigenwert von $I - B^{-1}A$, so wähle man f^0 als zugehörigen Eigenvektor. Damit gilt $f^k = \lambda^k f^0$. Durch die Konvergenz ist $\lim f^k = 0$ und somit notwendigerweise $|\lambda| < 1$.

Sei umgekehrt $\rho(I - B^{-1}A) < 1$. Zur Iterationsmatrix und gegebenem $\varepsilon > 0$ existiert eine Vektornorm, so dass in der korrespondierenden Matrixnorm

$$||I - B^{-1}A|| < \rho(I - B^{-1}A) + \varepsilon$$

gilt (ohne Beweis). Damit folgt

$$||(I - B^{-1}A)^k|| \leq ||I - B^{-1}A||^k \leq (\rho(I - B^{-1}A) + \varepsilon)^k = \mu^k$$

mit $\mu < 1$ für ε hinreichend klein. Also ist $\lim(I - B^{-1}A)^k = 0$ und somit $\lim f^k = 0$ für alle f^0.

Sei schließlich $||I - B^{-1}A|| < 1$ in einer Matrixnorm, die konsistent zu einer festen Vektornorm ist. Für eine solche Matrixnorm gilt stets $\rho(C) \leq \|C\|$ (ohne Beweis). Damit ist das hinreichende Kriterium aus dem zweiten Teil des Satzes erfüllt. □

Dem Beweis entnimmt man insbesondere die Abschätzung

$$||x^{k+1} - x|| \leq ||I - B^{-1}A|| \cdot ||x^k - x|| \tag{8.8}$$

in einer beliebigen Vektornorm und der induzierten Matrixnorm. Damit gilt für $||I - B^{-1}A|| < 1$ dann globale lineare Konvergenz in dieser Norm. Dies kann man hier auch aus dem *Banachschen Fixpunktsatz* folgern.

Die Konvergenzgeschwindigkeit des Verfahrens (8.4) ist umso größer, je kleiner der Spektralradius der Iterationsmatrix ist. Der folgende Satz gibt eine Aussage, die von Matrixnormen unabhängig ist.

Satz 8.2: *Konvergenzgeschwindigkeit*

Beim Iterationsverfahren (8.4) gilt für den Fehler $f^k = x^k - x$

$$\sup_{f^0 \neq 0} \limsup_{k \to \infty} \left(\frac{||f^k||}{||f^0||} \right)^{1/k} = \rho(I - B^{-1}A) \tag{8.9}$$

in einer beliebigen Vektornorm $|| \cdot ||$.

Beweis siehe z.B. Stoer, Bulirsch: Numerische Mathematik 2.

8.2 Klassische Iterationsverfahren

Im folgenden werden gängige stationäre Iterationsverfahren vorgestellt. Sie entstehen durch die Zerlegung der Gestalt

$$A = D + L + R, \tag{8.10}$$

wobei D Diagonalmatrix mit der Diagonalen von A und L, R den unteren bzw. oberen Dreiecksanteil (ohne Diagonale) von A enthalten.

Jacobi-Verfahren

Ein einfaches Iterationsverfahren entsteht, wenn man in (8.4) $B = D$ wählt, also nur die Diagonale von A. Gilt stets $a_{ii} \neq 0$, so kann die Invertierung sofort vorgenommen werden. Wir erhalten als Iteration

$$x^{k+1} = x^k - D^{-1}((D + L + R)x^k - b) = -D^{-1}((L + R)x^k - b).$$

Es ergibt sich für jede Komponente $i = 1, \ldots, n$ die Formel

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^k \right). \tag{8.11}$$

Da jede Komponente sofort separat berechnet werden kann, nennt man diese Iteration auch *Gesamtschrittverfahren*.

Das Jacobi-Verfahren ist konvergent, wenn die Diagonale von A den Hauptanteil der gesamten Matrix bildet. Die Matrix A erfüllt das *starke Zeilensummenkriterium*, wenn gilt

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{für} \quad i = 1, \ldots, n. \tag{8.12}$$

Analog erfüllt A das *starke Spaltensummenkriterium*, falls

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}| \quad \text{für} \quad j = 1, \ldots, n. \tag{8.13}$$

Sowohl (8.12) als auch (8.13) ist nach Satz 8.1 hinreichend für die Konvergenz des Jacobi-Verfahrens. Aus (8.12) folgt nämlich sofort

$$||I - B^{-1}A||_\infty = \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} \sum_{j \neq i} |a_{ij}| < 1 \tag{8.14}$$

in der Maximumnorm und aus (8.13)

$$||I - B^{-1}A||_1 = \max_{j=1,\ldots,n} \frac{1}{|a_{jj}|} \sum_{i \neq j} |a_{ij}| < 1 \tag{8.15}$$

in der 1-Norm.

Unter zusätzlichen Voraussetzungen kann die Konvergenz des Jacobi-Verfahrens noch garantiert werden, wenn in (8.12) oder (8.13) nur eine kleinergleich Beziehung gilt, also schwache Summenkriterien. Dies ist bei vielen Matrizen, die aus Diskretisierungen entstehen, gegeben (siehe Beispiel in Abschnitt 8.3).

Gauß-Seidel-Verfahren

Mehr Information über die Matrix A wird einbezogen, wenn man die Dreiecksmatrix $B = D + L$ ansetzt. Die Invertierung entspricht dann gerade einer Vorwärtssubstitution, wozu ebenfalls $a_{ii} \neq 0$ notwendig ist. Die Iteration lautet

$$x^{k+1} = x^k - (D+L)^{-1}((D+L+R)x^k - b) = -(D+L)^{-1}(Rx^k - b).$$

Es entsteht für die einzelnen Komponenten $i = 1, \ldots, n$ die Formel

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{k+1} - \sum_{j>i} a_{ij} x_j^k \right). \tag{8.16}$$

Da die Komponenten nur sukzessive berechnet werden können, wird diese Iteration auch als *Einzelschrittverfahren* bezeichnet.

Für die Konvergenz des Gauß-Seidel-Verfahrens sind ebenfalls die Bedingungen (8.12) oder (8.13) hinreichend. Bei bestimmten Matrix-Klassen lässt sich zeigen, dass das Gauß-Seidel-Verfahren schneller konvergiert als das Jacobi-Verfahren.

Im Gegensatz zum Jacobi-Verfahren konvergiert das Gauß-Seidel-Verfahren auch für symmetrisch positiv-definite Matrizen:

Zu zeigen ist, dass die Eigenwerte von $K = -(D+L)^{-1}L^{\mathrm{T}}$ im Einheitskreis liegen. Mit A ist auch D positiv definit, und damit haben K und

$$K' := D^{1/2} K D^{-1/2} = -(I + \tilde{L})^{-1} \tilde{L}^{\mathrm{T}} \quad \text{mit} \quad \tilde{L} = D^{-1/2} L D^{-1/2}$$

das gleiche Spektrum. Zu zeigen ist daher nur $\rho(K') < 1$.

Aus $K'x = \lambda x$ mit $x^{\mathrm{H}}x = 1$ ergibt sich

$$-\tilde{L}^{\mathrm{T}} x = \lambda (I + \tilde{L}) x \quad \Rightarrow \quad -x^{\mathrm{H}} \tilde{L}^{\mathrm{T}} x = \lambda (1 + x^{\mathrm{H}} \tilde{L} x)\,,$$

und mit $\alpha + i\beta := x^{\mathrm{H}} \tilde{L}^{\mathrm{T}} x$ daraus

$$|\lambda|^2 = \left| \frac{-\alpha - i\beta}{1 + \alpha + i\beta} \right|^2 = \frac{(-\alpha - i\beta)(-\alpha + i\beta)}{(1 + \alpha + i\beta)(1 + \alpha - i\beta)} = \frac{\alpha^2 + \beta^2}{1 + 2\alpha + \alpha^2 + \beta^2} < 1\,,$$

falls $1 + 2\alpha > 0$.

Wegen $x^{\mathrm{H}}D^{-1/2}AD^{-1/2}x = (D^{-1/2}x)^{\mathrm{H}}A(D^{-1/2}x)$ ist $D^{-1/2}AD^{-1/2}$ positiv definit. Aus $D^{-1/2}AD^{-1/2} = I + \tilde{L} + \tilde{L}^{\mathrm{T}}$ folgt sofort

$$0 < x^{\mathrm{H}}(I + \tilde{L} + \tilde{L}^{\mathrm{T}})x = 1 + x^{\mathrm{H}}\tilde{L}x + x^{\mathrm{H}}\tilde{L}^{\mathrm{T}}x = 1 + x^{\mathrm{H}}\tilde{L}x + \overline{x^{\mathrm{H}}\tilde{L}x} = 1 + 2\alpha.$$

Das Gauß-Seidel-Verfahren kann analog auch über $B = D + R$ angesetzt werden. Desweiteren existieren symmetrische Varianten, die abwechselnd je einen Schritt mit dem unteren und dann einen Schritt mit dem oberen Dreiecksanteil von B durchführen.

Relaxationsverfahren

In der Iteration (8.4) kann man allgemeiner die Matrix B in Abhängigkeit von einem Parameter $\omega \in \mathbb{R}$ wählen. Ziel ist es dann, ω optimal zu wählen dahingehend, dass $\rho(I - B(\omega)^{-1}A)$ minimal wird und somit die Konvergenz des Verfahrens zu beschleunigen. Bei den Relaxationsverfahren wird zum Relaxationsparameter $\omega > 0$ über die Zerlegung (8.10) die Matrix

$$B(\omega) = \frac{1}{\omega}(D + \omega L) \tag{8.17}$$

gewählt. Man beachte, dass für $\omega = 1$ das Einzelschrittverfahren (8.16) entsteht. Für die Berechnung der einzelnen Komponenten erhalten wir die Formel

$$\begin{aligned} z_i^{k+1} &= \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k\right), \\ x_i^{k+1} &= (1-\omega)x_i^k + \omega z_i^{k+1}. \end{aligned}$$

Die Hilfsgröße z_i^{k+1} wird dabei wie im Einzelschrittverfahren (8.16) bestimmt. Als neue Näherung der i-ten Komponente wird jedoch eine Linearkombination aus alter Näherung und der Hilfsgröße gebildet. Für $0 < \omega \leq 1$ ist dies eine Konvexkombination. Bei $\omega > 1$ spricht man von Überrelaxation, wodurch der Name *SOR-Verfahren* (successive overrelaxation) entstand.

Für bestimmte Matrizenklassen lassen sich optimale Relaxationsparameter $1 \leq \omega_{\mathrm{opt}} < 2$ explizit berechnen. Die Konvergenz ist dann erheblich schneller als für das Gauß-Seidel-Verfahren.

Iterative Nachbesserung

In den Kontext der iterativen Verfahren (8.4) – jedoch nicht auf der Zerlegung (8.10) basierend – fällt auch die sogenannte *Nachiteration*. Wird das lineare Gleichungssystem $Ax = b$ über LR-Zerlegung auf dem Rechner direkt gelöst, so erhält man eine durch Rundungsfehler beeinträchtigte Lösung $\tilde{x} \approx A^{-1}b$. Diese kann, falls A nicht zu schlecht konditioniert ist, durch diese Nachiteration bis auf Maschinengenauigkeit verbessert werden.

Der Gauß-Algorithmus liefert auf dem Rechner eine durch Rundungsfehler verfälschte LR-Zerlegung $A \approx \hat{L}\hat{R}$, d.h. es gilt nur

$$A = \hat{L}\hat{R} + E, \tag{8.18}$$

wobei die unbekannte Matrix E Fehleranteile enthält. Für die Matrix B in (8.4) wählt man dann $B = \hat{L}\hat{R}$ ($\hat{L}, \hat{R}$ haben hier nicht die Bedeutung von L, R aus (8.10)) und es entsteht die Vorschrift

$$x^{k+1} = x^k - (\hat{L}\hat{R})^{-1}(Ax^k - b) = x^k - (\hat{L}\hat{R})^{-1}r^k \tag{8.19}$$

mit dem Residuum r der k-ten Näherung. Als Startwert bietet sich die direkte Lösung $x^0 = (\hat{L}\hat{R})^{-1}b$ an.

Die Iteration (8.19) lässt sich mit wenig Aufwand durchführen, da die Zerlegung $\hat{L}\hat{R}$ bereits berechnet ist und somit nur Vorwärts- und Rückwärtssubstitution erforderlich sind. Da nur Rundungsfehler in $\hat{L}, \hat{R}$ auftreten, gilt $(\hat{L}\hat{R})^{-1}A \approx I$. Folglich ist der Spektralradius der Iterationsmatrix klein und die Konvergenz der Iteration sehr schnell. Zudem liegt über x^0 bereits ein guter Startwert für die Iteration vor. Dementsprechend erhält man im allgemeinen nach etwa zwei Iterationsschritten die Lösung auf Maschinengenauigkeit.

Bei der Berechnung des Residuums r^k tritt durch Subtraktion Auslöschung wegen $Ax^k \approx b$ auf. Dieser Wert muss daher mit höherer Genauigkeit als der Maschinengenauigkeit bei den anderen Operationen berechnet werden. Nur dann kann das Ergebnis auf die ursprüngliche Maschinengenauigkeit erhalten werden. Ist x^0 nur eine grobe Näherung, so kann die Nachbesserung mit stets gleicher Rechengenauigkeit die Anzahl der Stellen erhöhen, jedoch nicht auf die volle Stellenzahl. Die Problematik der Auslösung in (8.4) tritt sonst bei den Iterationsverfahren nicht auf, da nur eine relativ geringe Genauigkeit erzielt werden soll (d.h. $Ax^k \approx b$ gilt weniger stark).

8.3 Anwendungsbeispiel

Wir betrachten das Dirichletsche Randwertproblem für eine reellwertige Funktion $u(x,y)$ im Einheitsquadrat $\Omega := \{(x,y) : 0 < x, y < 1\}$

$$\begin{array}{rcll} -\Delta u = -u_{xx} - u_{yy} & = & f(x,y) & (x,y) \in \Omega \\ u(x,y) & = & 0, & (x,y) \in \partial\Omega \end{array} \tag{8.20}$$

bei vorgegebener stetiger Funktion f. Die Differenzenquotienten werden auf einem uniformen Gitter der Schrittweite $h := \frac{1}{M+1}$ für $M \in \mathbb{N}$ diskretisiert

$$\Omega_h := \{(x_i, y_j) = (ih, jh) : i, j = 1, \ldots, M\}. \tag{8.21}$$

Der symmetrische Differenzenquotient zweiter Ordnung liefert als Näherungsformel für $u_{i,j} := u(x_i, y_j)$

$$-\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} - \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} \doteq f(x_i, y_j) \tag{8.22}$$

und äquivalent mit $f_{i,j} := f(x_i, y_j)$

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} \doteq h^2 f_{i,j} \tag{8.23}$$

bei $i, j = 1, \ldots, M$. Die Unbekannten und die rechten Seiten seien in der Form

$$\begin{array}{rcl} u & = & (u_{1,1}, u_{2,1}, \ldots, u_{M,1}, u_{1,2}, \ldots, u_{M,2}, \ldots, u_{1,M}, \ldots, u_{M,M})^{\mathrm{T}} \\ b & = & h^2 (f_{1,1}, f_{2,1}, \ldots, f_{M,1}, f_{1,2}, \ldots, f_{M,2}, \ldots, f_{1,M}, \ldots, f_{M,M})^{\mathrm{T}} \end{array} \tag{8.24}$$

angeordnet, wodurch ein lineares Gleichungssystem $Au = b$ der Dimension $n = M^2$ entsteht. Die Matrix A besitzt die folgende Bandstruktur

$$A = \begin{pmatrix} C & -I & & \\ -I & C & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & C \end{pmatrix} \quad \text{mit} \quad C = \begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}. \tag{8.25}$$

Die Matrix A ist nur schwach besetzt, da jede Zeile maximal 5 Elemente ungleich null enthält. Desweiteren ist A symmetrisch und positiv definit. Eine Cholesky-Zerlegung $A = LL^{\mathrm{T}}$ würde jedoch zahlreiche Einträge ungleich

null in L erzeugen, d.h. L ist nicht schwach besetzt. Dagegen erfordern Matrix-Vektor-Produkte bezüglich A nur vergleichsweise wenig Operationen. (Nur Elemente ungleich null müssen abgearbeitet werden.)

Für die Matrix (8.25) kann der Spektralradius der jeweiligen Iterationsmatrix $I - B^{-1}A$ für die obigen drei Verfahren explizit berechnet werden. Zur Wahl des optimalen Relaxationsparameters im SOR-Verfahren siehe auch Abb. 28. Im einzelnen ergeben sich die folgenden Werte:

$$\begin{aligned}
\text{Jacobi-Verfahren:} \qquad \rho(I - D^{-1}A) &= \cos\left(\tfrac{\pi}{M+1}\right) \\
\text{Gauß-Seidel-Verfahren:} \qquad \rho(I - (D+L)^{-1}A) &= \cos^2\left(\tfrac{\pi}{M+1}\right) \\
\text{SOR-Verfahren:} \qquad \rho(I - B(\omega_{\text{opt}})^{-1}A) &= \frac{\cos^2\left(\frac{\pi}{M+1}\right)}{\left(1+\sin\left(\frac{\pi}{M+1}\right)\right)^2}
\end{aligned}$$

Wir erkennen die Relation

$$1 > \rho_{\text{J}} > \rho_{\text{GS}} > \rho_{\text{SOR}} > 0 \qquad \text{für festes } \; M \geq 2.$$

Nach Satz 8.1 ist damit die Konvergenz in allen drei Iterationsverfahren garantiert. Nach Satz 8.2 ist die Größenordnung des Spektralradius entscheidend für die Konvergenzgeschwindigkeit, d.h. wieviele Schritte für eine bestimmte Genauigkeit benötigt werden. Mit steigender Problemgröße $M \to \infty$ folgt $\rho \to 1$, wodurch die Konvergenz in allen drei Verfahren immer langsamer wird. Für festes M ist jedoch das Gauß-Seidel-Verfahren etwa doppelt so schnell wie das Jacobi-Verfahren wegen $\rho_{\text{GS}} \approx \rho_{\text{J}}^2$. Es lässt sich desweiteren für großes M annähern

$$\rho_{\text{J}}^{\kappa} = \rho_{\text{SOR}} \quad \Rightarrow \quad \kappa = \frac{\ln \rho_{\text{SOR}}}{\ln \rho_{\text{J}}} \approx \frac{4(M+1)}{\pi},$$

wodurch das SOR-Verfahren mit optimalem Relaxationsparameter mehr als M mal so schnell wie das Jacobi-Verfahren konvergiert.

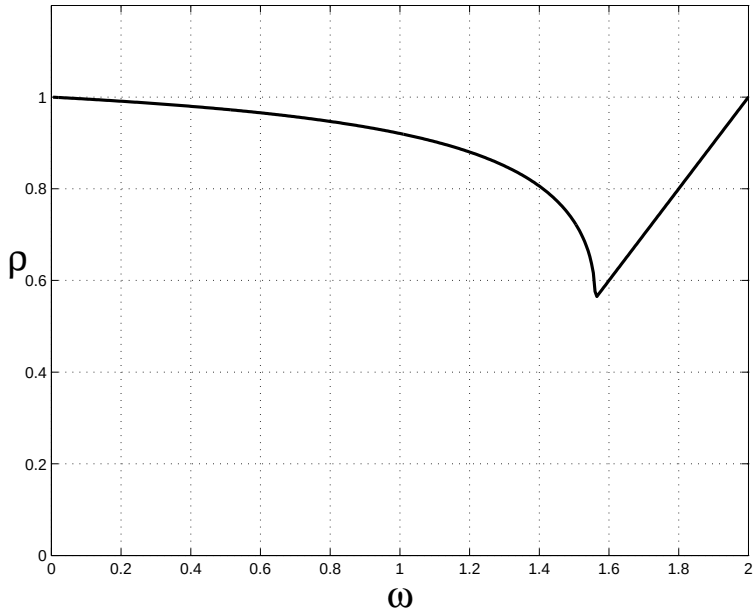


Abbildung 28: Spektralradius der Iterationsmatrix im Relaxationsverfahren für verschiedene Parameter $\omega \in [0, 2]$ bei Matrix (8.25) mit $M = 10$.

Ausblick:

Es gibt neben den stationären Verfahren noch weitere fortgeschrittene Iterationsmethoden für große lineare Gleichungssysteme.

- *Verfahren der Konjugierten Gradienten* (*CG-Verfahren*)
  nur für symmetrische positiv definite Matrizen; basiert auf sukzessiver Minimierung des Fehlers entlang von eindimensionalen Suchrichtungen; Verallgemeinerungen auf allgemeine Matrizen und nichtlineare Gleichungssysteme existieren.
- *Generalised Minimal Residual method* (*GMRES*)
  für allgemeine Matrizen; basiert auf Minimierung des Residuums; Rechenaufwand steigt in der Iteration jedoch mit der Schrittzahl an.
- *Mehrgitterverfahren*
  bei Matrizen die aus Diskretisierungen von partiellen Differentialgleichungen entstehen; Wechsel zwischen groben und feinen Gittern findet in der Iteration statt.

Kapitel 9

Nichtlineare Gleichungssysteme

In diesem Kapitel behandeln wir die numerische Lösung von nichtlinearen Gleichungssystemen bzw. Nullstellenprobleme. Am Fall einer Funktion einer Veränderlichen kann man bereits die wesentlichen Begriffe und Verfahren studieren. Die Verallgemeinerung auf n-dimensionale nichtlineare Gleichungssysteme ergibt sich direkt, erfordert bei der Analyse jedoch einen gewissen technischen Aufwand. Wir betrachten hauptsächlich das Newton-Verfahren und seine Varianten, welche die zentralen Methoden zur Lösung nichtlinearer Gleichungssysteme darstellen.

Isaac Newton verfasste von 1664-1671 seine Arbeit 'Methodus fluxionum et serierum infinitarum' (Von der Methode der Fluxionen und unendlichen Folgen). Darin beschrieb er eine Idee zur approximativen Bestimmung der Nullstelle eines Polynoms dritten Grades. Joseph Raphson formalisierte 1690 in seiner Arbeit 'Analysis aequationum universalis' diese Berechnung auf die allgemeine Gleichung 3. Grades, wodurch das bekannte Iterationsverfahren folgte. Daher wird diese Iteration als Newton-Verfahren oder Newton-Raphson-Verfahren bezeichnet.

9.1 Der eindimensionale Fall

Gegeben sei eine hinreichend oft stetig differenzierbare Funktion $f : D \to \mathbb{R}$ mit $D \subseteq \mathbb{R}$. Wir nehmen an, dass f eine eindeutige Nullstelle $\hat{x} \in D$ besitzt, d.h. eine Lösung der Gleichung

$$f(x) = 0. \tag{9.1}$$

Das Nullstellenproblem (9.1) bedeutet, die Lösung $\hat{x}$ durch numerische Verfahren näherungsweise zu bestimmen. Ist f eine nichtlineare Funktion, so liegt eine nichtlineare Gleichung vor.

Für eine allgemeine nichtlineare Funktion f existiert kein direktes Verfahren zur Lösung der Gleichung (9.1). Wir sind daher auf Iterationsverfahren angewiesen, die ausgehend von einem Startwert eine Folge von Näherungen liefern

$$x^0 \quad \longrightarrow \quad x^1 \quad \longrightarrow \quad x^2 \quad \longrightarrow \quad x^3 \quad \longrightarrow \quad \cdots ,$$

wobei das Verfahren derart konstruiert ist, dass

$$\lim_{k\to\infty} x^k = \hat{x}$$

gilt. Für hinreichend hohes k ist die Approximation genau genug und man kann die Iteration abbrechen. Die Konvergenz soll möglichst schnell sein, wodurch eine kleine Schrittzahl k dann bereits ausreichend ist.

Bisektion

Ein einfacher und robuster Ansatz zur numerischen Lösung der nichtlinearen Gleichung (9.1) ist die *Bisektion*, d.h. eine Intervallschachtelung basierend auf Vorzeichenbetrachtungen wird vorgenommen. Dabei ist lediglich f als stetig vorauszusetzen, wodurch der Zwischenwertsatz gilt. Abb. 29 veranschaulicht diese Strategie. Befindet sich im Anfangsintervall eine eindeutige Nullstelle, so ist die Konvergenz dieser Methode offensichtlich.

Algorithmus 9.1: *Bisektion*

wähle a, b mit $a < b$ und $f(a) \cdot f(b) < 0$
$A := f(a),\ B := f(b)$
while $(b - a) >$ TOL
    $t = \frac{a+b}{2},\ T = f(t)$
    if $A \cdot T > 0$: $\quad a := t,\ A := T$
    else $\quad b := t,\ B := T$
end(while)
$\hat{x} := \frac{a+b}{2}$

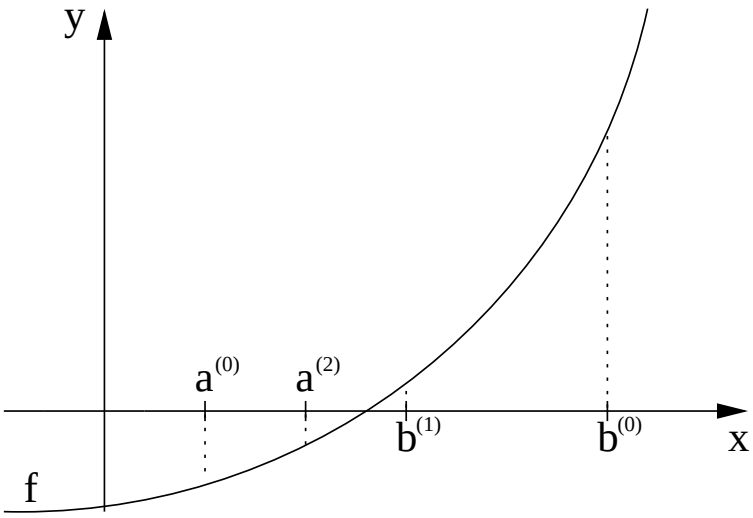


Abbildung 29: Prinzip der Bisektion.

Es sei $[a^k, b^k]$ das Intervall aus dem k-ten Schritt der Bisektion. Die korrespondierende Näherung ist daher $x^k := \frac{a^k+b^k}{2}$. Man überlegt sich

$$|x^k - \hat{x}| \leq \frac{b^k - a^k}{2} = \frac{b^0 - a^0}{2^{k+1}} \qquad \text{für } k = 0, 1, 2, \ldots . \tag{9.2}$$

Dadurch liegt lineare Konvergenz vor mit dem Faktor $C = \frac{1}{2}$.

Newton-Verfahren

Das *Newton-Verfahren* dagegen verwendet eine Linearisierung der Funktion f aus (9.1), d.h. Tangenten an die Funktion werden gebildet. Daher ist $f \in C^1(D)$ notwendig zur Durchführung des Ansatzes. Abb. 30 illustriert die entstehende Iteration geometrisch.

Die Tangente durch $(x^k, f(x^k))$ ist gegeben über

$$t(x) = f(x^k) + (x - x^k) f'(x^k).$$

Die Nullstelle der Tangente ist die neue Approximation, d.h. $t(x^{k+1}) = 0$. Es folgt die Iteration

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}.$$

Daher ist $f'(x) \neq 0$ für alle $x \in (\hat{x} - \delta, \hat{x} + \delta)$ mit einem $\delta > 0$ zu fordern.

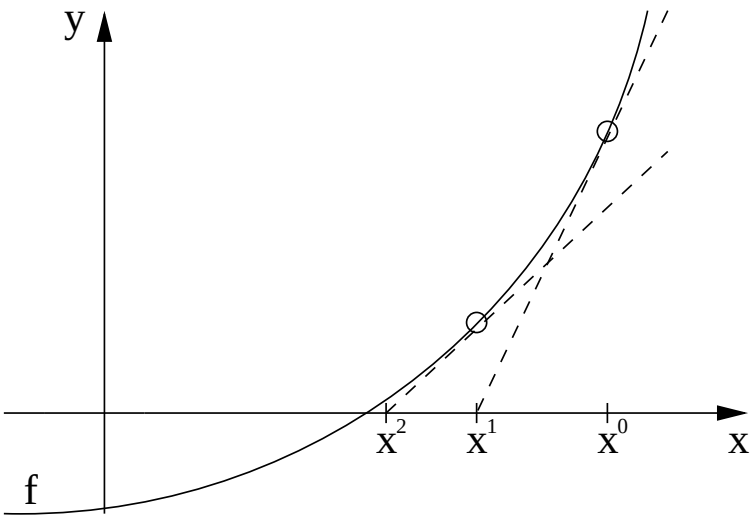


Abbildung 30: Prinzip des Newton-Verfahrens.

Algorithmus 9.2: *Newton-Raphson-Verfahren*

wähle x^0
for $k = 0, 1, 2, \ldots, k_{\max}$
    $x^{k+1} := x^k - f(x^k)/f'(x^k)$
    if $|x^{k+1} - x^k| < \text{TOL}$: exit
end(for)
$\hat{x} := x^{k+1}$

Das Newton-Raphson-Verfahren ist im allgemeinen nicht global konvergent. Betrachte beispielsweise das Nullstellenproblem

$$f(x) := \arctan(x) = 0. \tag{9.3}$$

Für Startwerte x^0 nahe der Nullstelle $\hat{x} = 0$ konvergiert die Newton-Iteration. Für zu hohes $|x^0|$ divergiert die Iteration alternierend gegen unendlich.

Dies ist ein qualitativer Unterschied zu den bisherigen Verfahren. Bei einem linearen Gleichungssystem liefert der Gauß-Algorithmus nach endlich vielen Schritten die exakte Lösung (bis auf Rundungsfehler). Die stationären Iterationsverfahren für lineare Gleichungssysteme sind unter geeigneten Voraussetzungen global konvergent.

Konvergenzuntersuchung

Um die Konvergenz und Konvergenzgeschwindigkeit der Iterationsverfahren zu diskutieren, betrachten wir eine allgemeine *Fixpunktiteration*. Ein Einschritt-Verfahren kann in der Form

$$x^{k+1} = \Phi(x^k) \tag{9.4}$$

geschrieben werden mit einer Iterationsfunktion $\Phi : D \to D$ ($D \subset \mathbb{R}$). Falls die Iteration gegen ein $\hat{x} \in D$ konvergiert und Φ stetig ist, dann ist $\hat{x}$ ein *Fixpunkt* von Φ, denn es gilt

$$\hat{x} = \lim_{k\to\infty} x^{k+1} = \lim_{k\to\infty} \Phi(x^k) = \Phi\left(\lim_{k\to\infty} x^k\right) = \Phi(\hat{x}).$$

Definition 9.1: Sei $\hat{x} \in D$ ein Fixpunkt von $\Phi : D \to D$. Die Iteration (9.4) ist *lokal konvergent*, falls eine Umgebung $U \subset D$ von $\hat{x}$ existiert, so dass $\lim_{k\to\infty} x^k = \hat{x}$ für alle Startwerte $x^0 \in U$ gilt. Das Verfahren (9.4) heißt *global konvergent*, wenn $\lim_{k\to\infty} x^k = \hat{x}$ für alle Startwerte $x^0 \in D$ vorliegt.

Bemerkung: Lokale und globale Konvergenz implizieren, dass der Fixpunkt eindeutig in U bzw. D ist.

Definition 9.2: Sei $\hat{x} \in D$ ein Fixpunkt von $\Phi : D \to D$. Die Iteration (9.4) ist lokal konvergent von (mindestens) der *Ordnung* $p \geq 1$, wenn eine Umbegbung $V \subset D$ von $\hat{x}$ existiert, so dass

$$|x^{k+1} - \hat{x}| \leq C|x^k - \hat{x}|^p$$

für alle Startwerte $x^0 \in V$ gilt mit einer Konstanten $C \geq 0$. Im Fall $p = 1$ muss zudem $C < 1$ gelten.

Bemerkung: Die Definitionen können verallgemeinert werden aus Funktionen $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ mit einer Vektornorm $\|\cdot\|$ anstelle des Absolutbetrags $|\cdot|$. In diesem Kapitel verwenden wir die Euklidische Norm, sofern nicht anders gekennzeichnet.

Faustregel: Ist die Iteration konvergent mit Ordnung $p \geq 2$, dann steigt die Anzahl der korrekten Dezimalstellen in der Näherung mit jedem Iterationsschritt etwa um den Faktor p. (Also $p = 2$ bewirkt eine Verdoppelung der korrekten Stellen pro Schritt.)

Die Bisektion läßt sich laut (9.2) als linear konvergent ($p = 1$) mit der Konstante $C = \frac{1}{2}$ interpretieren. Im wichtigen Spezialfall $p = 2$ zeigt man über vollständige Induktion

$$|x^k - \hat{x}| \leq C^{2^k-1}|x^0 - \hat{x}|^{2^k}.$$

Die Konvergenzordnung wird üblicherweise durch Taylor-Entwicklung um den Fixpunkt ermittelt:

$$\begin{aligned} \Phi(x) &= \Phi(\hat{x}) + \Phi'(\hat{x})(x - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x})(x - \hat{x})^2 + \cdots \\ \Rightarrow x^{k+1} = \Phi(x^k) &= \Phi(\hat{x}) + \Phi'(\hat{x})(x^k - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x})(x^k - \hat{x})^2 + \cdots . \end{aligned}$$

Für die Newton-Iteration erhalten wir die Formel (sofern $f'(x) \neq 0$)

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} \quad \Rightarrow \quad \Phi(x) = x - \frac{f(x)}{f'(x)}$$

$$\Phi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = f(x)\frac{f''(x)}{f'(x)^2}.$$

Wegen $f(\hat{x}) = 0$ gilt $\Phi(\hat{x}) = \hat{x}$ und $\Phi'(\hat{x}) = 0$, wodurch folgt

$$x^{k+1} = \Phi(x^k) = \underbrace{\Phi(\hat{x})}_{=\hat{x}} + \underbrace{\Phi'(\hat{x})}_{=0}(x^k - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x} + \vartheta^k(x^k - \hat{x}))(x^k - \hat{x})^2$$

$$\Rightarrow \quad x^{k+1} - \hat{x} = \tfrac{1}{2}\Phi''(\hat{x} + \vartheta^k(x^k - \hat{x}))(x^k - \hat{x})^2$$

mit $\vartheta^k \in (0, 1)$. Ist Φ'' stetig, dann wählen wir eine kompakte Umbegung $V \subset D$ von $\hat{x}$. Es existiert ein $C > 0$ mit $|\Phi''(x)| < 2C$ für alle $x \in V$ und daher

$$|x^{k+1} - \hat{x}| < C|x^k - \hat{x}|^2 \qquad \text{für } x^k \in V.$$

Das Newton-Raphson-Verfahren ist somit lokal konvergent von (mindestens) der Ordnung 2, d.h. quadratische Konvergenz liegt vor. Im allgemeinen ist

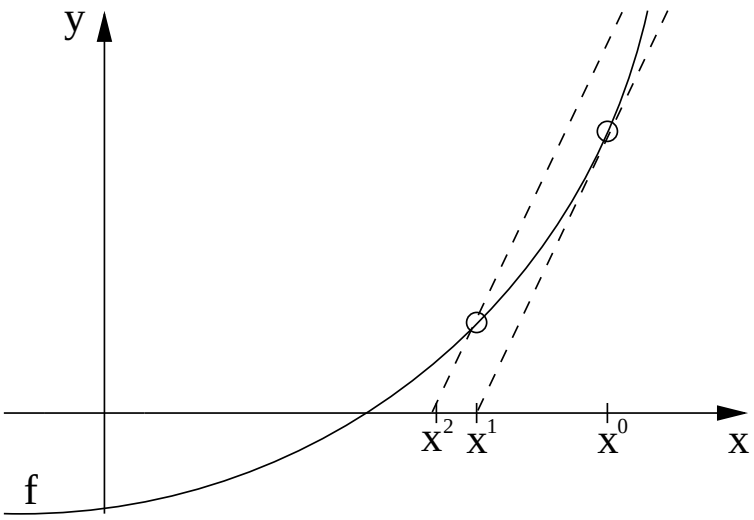


Abbildung 31: Prinzip des vereinfachten Newton-Verfahrens.

die Konvergenzordnung genau 2, da meistens $\Phi''(\hat{x}) \neq 0$ vorliegt. Es ist dann $\frac{1}{2}\Phi''(\hat{x})$ der dominierende Faktor in der Taylorentwickulung der Iterationsfunktion. Die Newton-Iteration ist jedoch nicht global konvergent, siehe das Gegenbeispiel (9.3).

Für Fixpunktiterationen gibt der Satz von Banach ein Kriterium zur Konvergenz. Allgemein gilt für $\Phi \in C^1$ die Aussage

$$|\Phi(x) - \Phi(y)| \leq L \cdot |x - y| \qquad \text{für alle } x, y \in U$$

mit $L : \sup\{|\Phi'(x)| : x \in U\}$, welche man mit dem Mittelwertsatz der Differentialrechnung folgt. Voraussetzung für den Fixpunktsatz von Banach ist Kontraktivität, d.h. $L < 1$. Wegen $\Phi'(\hat{x}) = 0$ gilt $L < 1$ falls $U = [\hat{x} - \delta, \hat{x} + \delta]$ mit $\delta > 0$ hinreichend klein wegen der Stetigkeit von Φ'. Der Satz von Banach garantiert dann die Konvergenz gegen $\hat{x}$ für alle $x^0 \in U$ mit der Konvergenzordnung $p \geq 1$.

An dieser Stelle diskutieren wir noch das *vereinfachte Newton-Verfahren*, obwohl dieses erst im mehrdimensionalen Fall von Bedeutung ist. Im eindimensionalen Fall lautet die Iteration

$$x^{k+1} = x^k - \frac{f(x)}{f'(x^0)} \qquad \text{für } k = 0, 1, 2, \dots ,$$

d.h. der Ableitungswert wird konstant gehalten und keine weitere Berechnung der Ableitung ist erforderlich. Abb. 31 zeigt eine geometrische Inter-

pretation. In diesem Fall lautet die Iterationsfunktion

$$\Phi(x) = x - \frac{f(x)}{f'(x^0)}.$$

Folglich haben wir wieder $\Phi(\hat{x}) = \hat{x}$. Die erste Ableitung ist

$$\Phi'(x) = 1 - \frac{f'(x)}{f'(x^0)}.$$

Es gilt im allgemeinen $\Phi'(\hat{x}) \neq 0$, d.h. $\Phi'(\hat{x})$ ist der dominierende Faktor in der Taylorentwicklung der Iterationsfunktion. Falls $x^0 \approx \hat{x}$ und $x \approx \hat{x}$ ist, dann erhalten wir $f'(x) \approx f'(x^0)$ und somit $|\Phi'(x)| \leq C < 1$. Das vereinfachte Newton-Verfahren ist damit konvergent von der Ordnung 1, d.h. lineare Konvergenz liegt vor, sofern x^0 hinreichend nahe an $\hat{x}$ ist.

9.2 Der mehrdimensionale Fall

Wir betrachten nun ein nichtlineares Gleichungssystem der Form

$$F(x) = 0 \tag{9.5}$$

mit $F : D \to \mathbb{R}^n$ ($D \subseteq \mathbb{R}^n$). Wieder nehmen wir die Existenz einer eindeutigen Nullstelle $\hat{x} \in D$ an. Der Fall $n = 1$ wurde im vorhergehenden Abschnitt behandelt.

In der mehrdimensionalen Situation wird eine geometrische Interpretation des *Newton-Verfahrens* wie im eindimensionalen Fall (siehe Abb. 30) unanschaulich. Alternativ motivieren wir das Newton-Verfahren nur über eine Linearisierung des Systems (9.5). Für $F \in C^2$ führen wir eine Taylor-Entwicklung der Nullstelle $\hat{x}$ um den Startwert x^0 durch

$$0 = F(\hat{x}) = F(x^0) + DF(x^0)(\hat{x} - x^0) + \mathcal{O}(\|\hat{x} - x^0\|^2).$$

Durch Vernachlässigung des Restterms erhalten wir eine im allgemeinen bessere Näherung für die Nullstelle

$$0 = F(x^0) + DF(x^0)(x^1 - x^0).$$

Diese Formel stellt ein lineares Gleichungssystem für die neue Approximation dar. Sukzessiv folgt somit die Newton-Iteration im mehrdimensionalen Fall

$$x^{k+1} = x^k - DF(x^k)^{-1}F(x^k) \qquad \text{für } k = 0, 1, 2, \dots , \tag{9.6}$$

wobei in jedem Iterationsschritt ein lineares Gleichungssystem zu lösen ist. Die Auswertung der Funktionalmatrix $DF(x) \in \mathbb{R}^{n\times n}$ erfolgt meist über numerische Differentiation

$$DF(x) = (\Delta_1 F, \dots, \Delta_n F), \qquad \Delta_i F := \tfrac{1}{h_i}\left(F(x + h_i e_i) - F(x)\right)$$

mit Schrittweiten $h_i > 0$ und den Einheitsvektoren $e_i \in \mathbb{R}^n$. Somit sind n zusätzliche Funktionsauswertungen von F notwendig.

Gilt lediglich $F \in C^1$, so kann man die Taylorentwicklung nur bis zum ersten Glied durchführen

$$0 = F(\hat{x}) = F(x^0) + DF(\xi x^0 + (1-\xi)\hat{x})(\hat{x} - x^0) \quad \text{mit } \xi \in [0, 1].$$

Da DF stetig ist, wird die Approximation $DF(\xi x^0 + (1-\xi)\hat{x}) \doteq DF(x^0)$ bei hinreichend gutem Startwert sinnvoll. Es entsteht also wieder das obige mehrdimensionale Newton-Verfahren.

Algorithmus 9.3: *mehrdim. Newton-Verfahren*

wähle x^0
for $k = 0, 1, 2, \dots, k_{\max}$
    bestimme $F(x^k)$ und $DF(x^k)$
    löse $DF(x^k)\Delta x^k = -F(x^k)$
    $x^{k+1} := x^k + \Delta x^k$
    if $\|\Delta x^k\| <$ TOL: exit
end(for)
$\hat{x} := x^{k+1}$

Der Aufwand in jedem Iterationsschritt (9.6) setzt sich damit zusammen aus:

1. Auswertung der nichtlinearen Funktion F.
2. Bestimmung der Funktionalmatrix DF.
   (Bei numerischer Differentiation bedeutet dies n zusätzliche Auswertungen von F.)
3. Lösung des linearen Gleichungssystems.
   (Über LR-Zerlegung erfordert dies einen Aufwand $\mathcal{O}(n^3)$ bei vollbesetzten Matrizen.)

Je nachdem aus welcher Anwendung die Funktion F entsteht, kann der Aufwand in den drei Teilbereichen sehr unterschiedlich ausfallen.

Um den Rechenaufwand in den Anteilen 2 und 3 zu reduzieren, kann man zum *vereinfachten Newton-Verfahren* übergehen. Dabei verwendet man die Funktionalmatrix aus dem ersten Iterationsschritt in allen weiteren Schritten. Somit ist nur eine einmalige Auswertung der Funktionalmatrix notwendig. Eine LR-Zerlegung dieser Matrix kann in allen weiteren Iterationsschritten zur Lösung der linearen Gleichungssysteme verwendet werden. Es sind lediglich Vorwärts- und Rückwärts-Substitutionen durchzuführen.

Algorithmus 9.4: *vereinfachtes Newton-Verfahren*

wähle x^0
bestimme $DF(x^0)$
zerlege $DF(x^0) = L \cdot R$
for $k = 0, 1, 2, \ldots, k_{\max}$
    bestimme $F(x^k)$
    löse $\quad Ly = -F(x^k) \quad$ und $\quad R\Delta x^k = y$
    $x^{k+1} := x^k + \Delta x^k$
    if $\|\Delta x^k\| <$ TOL: $\quad$ exit
end(for)
$\hat{x} := x^{k+1}$

Im vereinfachten Newton-Verfahren liegt jedoch nur lineare Konvergenz vor, wodurch eventuell viele Iterationsschritte notwendig sind. Zudem ist der Konvergenzbereich im allgemeinen deutlich kleiner als beim gewöhnlichen Newton-Verfahren, d.h. man benötigt bereits gute Startwerte.

9.3 Konvergenz des gewöhnlichen Newton-Verfahrens

Zu der Konvergenz des gewöhnlichen Newton-Verfahrens bei nichtlinearen Gleichungssystemen seien hier zwei Sätze zitiert und diskutiert. Die Aussagen gelten auch im Spezialfall $n = 1$.

Satz 9.3: *Sei $\hat{x}$ eine Nullstelle von $F : D \to \mathbb{R}^n$ $(D \subset \mathbb{R}^n)$,*

$$F(x) = (f_1(x), \ldots, f_n(x))^\top, \quad x = (x_1, \ldots, x_n)^\top$$

und $F \in C^2(D)$. Wir definieren bezüglich der Maximumnorm

$$\mathcal{K} := \{x \in \mathbb{R}^n : \|x - \hat{x}\|_\infty \leq r\} \subset D$$

und

$$M := \max \left\{ \left| \frac{\partial^2 f_l}{\partial x_i \partial x_j}(x) \right| : 1 \leq l, i, j \leq n, \;\; x \in \mathcal{K} \right\}.$$

Wenn $\det DF(\hat{x}) \neq 0$ *und*

$$\beta r \leq \tfrac{1}{2} \quad \textit{mit} \quad \beta := n^2 M \|DF(\hat{x})^{-1}\|_\infty$$

gilt, dann existiert die Folge $(x^k)_{k\in\mathbb{N}}$ definiert durch das Newton-Verfahren für beliebiges $x^0 \in \mathcal{K}$, d.h. alle Matrizen $DF(x^k)$ sind regulär. Die Folge konvergiert gegen $\hat{x}$ und es gilt

$$\|x^{k+1} - \hat{x}\|_\infty \leq \beta \|x^k - \hat{x}\|_\infty^2 \leq \tfrac{1}{2} \|x^k - \hat{x}\|_\infty.$$

Beweis:

Zuerst zeigen wir

$$\|DF(x) - DF(y)\|_\infty \leq n^2 M \|x - y\|_\infty \quad \text{für alle } \; x, y \in \mathcal{K}.$$

Über den Mittelwertsatz folgt mit $\vartheta_l \in (0,1)$

$$\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y) = \sum_{q=1}^{n} \frac{\partial^2 f_l}{\partial x_q \partial x_j}(y + \vartheta_l(x-y))(x_q - y_q)$$

$$\left|\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y)\right| \leq nM \max_{i=1,\ldots,n} |x_i - y_i| = nM\|x-y\|_\infty$$

$$\sum_{j=1}^{n} \left|\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y)\right| \leq n^2 M\|x-y\|_\infty$$

für jedes l, wodurch die Behauptung gezeigt ist. Wir verwenden Induktion. Es sei daher $x^k \in \mathcal{K}$ ($x^0 \in \mathcal{K}$ ist vorausgesetzt). Taylor-Entwicklung liefert

$$0 = F(\hat{x}) = F(x^k) + DF(x^k)(\hat{x} - x^k) + R^k,$$

wobei die l-te Komponente von $R^k \in \mathbb{R}^n$ die Gestalt

$$R_l^k = \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f_l}{\partial x_i \partial x_j}(x^k + \vartheta_{l,k}(\hat{x} - x^k))(\hat{x}_i - x_i^k)(\hat{x}_j - x_j^k).$$

hat. Damit erhalten wir direkt

$$\|R^k\|_\infty \leq \tfrac{1}{2} n^2 M \|x^k - \hat{x}\|_\infty^2.$$

Um die Newton-Iteration anwenden zu können, benötigen wir $\det DF(x^k) \neq 0$. Wir definieren die Hilfsmatrix $H^k := DF(x^k) - DF(\hat{x})$. Weil $\hat{x}, x^k \in \mathcal{K}$ gilt, folgt nun

$$\|H^k\|_\infty = \|DF(x^k) - DF(\hat{x})\|_\infty \leq n^2 M \|x^k - \hat{x}\|_\infty \leq n^2 M r.$$

Desweiteren haben wir

$$DF(x^k) = DF(\hat{x}) + DF(x^k) - DF(\hat{x}) = DF(\hat{x})(I + DF(\hat{x})^{-1} H^k).$$

Somit gilt $\det DF(x^k) \neq 0$ genau dann, wenn $\det(I + DF(\hat{x})^{-1} H^k) \neq 0$. Dies wird garantiert durch die Bedingung

$$\|DF(\hat{x})^{-1} H^k\|_\infty \leq \|DF(\hat{x})^{-1}\|_\infty \cdot \|H^k\|_\infty \leq \tfrac{\beta}{n^2 M} \cdot n^2 M r = \beta r \leq \tfrac{1}{2} < 1.$$

Die Formel für die Newton-Iteration und die obige Taylor-Entwicklung liefern

$$x^{k+1} - \hat{x} = x^k - \hat{x} - DF(x^k)^{-1} F(x^k) = DF(x^k)^{-1} R^k.$$

Die allgemeine Aussage $\|(I+B)^{-1}\| \leq 1/(1 - \|B\|)$ für $\|B\| < 1$ impliziert

$$\|DF(x^k)^{-1}\|_\infty = \|(I + DF(\hat{x})^{-1} H^k)^{-1} DF(\hat{x})^{-1}\|_\infty \leq 2\|DF(\hat{x})^{-1}\|_\infty.$$

Schließlich erhalten wir

$$\begin{aligned}\|x^{k+1}-\hat{x}\|_\infty &\leq \|DF(x^k)^{-1}\|_\infty \cdot \|R^k\|_\infty \\ &\leq 2\|DF(\hat{x})^{-1}\|_\infty \tfrac{1}{2}n^2 M\|x^k-\hat{x}\|_\infty^2 \\ &= \beta\|x^k-\hat{x}\|_\infty^2 \\ &\leq \beta r\|x^k-\hat{x}\|_\infty \\ &\leq \tfrac{1}{2}\|x^k-\hat{x}\|_\infty\end{aligned}$$

und damit gelten die behaupteten Konvergenzaussagen. □

Das Newton-Verfahren im mehrdimensionalen Fall ist somit lokal quadratisch konvergent. In diesem Satz wird $F \in C^2$ vorausgesetzt. Ebenfalls ist bereits die Existenz einer Nullstelle gefordert. Ein Resultat, das auf diese Forderung verzichtet und sich stattdessen am Startwert orientiert, ist der folgende Satz von Newton-Kantorovich.

Satz 9.4: (*Newton-Kantorovich*)
Sei $D \subset \mathbb{R}^n$ offen und konvex sowie $F : D \to \mathbb{R}^n$ eine glatte Funktion ($F \in C^1$). Für einen Startwert $x^0 \in D$ sei $\det DF(x^0) \neq 0$. Konstanten $\alpha, \beta, \gamma \geq 0$ sollen existieren mit

$$\begin{aligned}&\text{(i)} && \|DF(x^0)^{-1}F(x^0)\| \leq \alpha \\ &\text{(ii)} && \|DF(x^0)^{-1}\| \leq \beta \\ &\text{(iii)} && \|DF(x)-DF(y)\| \leq \gamma\|x-y\| \quad \textit{für alle } x, y \in D.\end{aligned}$$

in einer beliebigen Vektornorm und korrespondierender Matrixnorm. Wir definieren die Werte

$$h := \alpha\beta\gamma, \quad \rho_{1,2} := \frac{\alpha}{h}\left(1 \mp \sqrt{1-2h}\right)$$

und Mengen

$$S_{\rho_{1/2}}(x^0) := \{x \in \mathbb{R}^n : \|x-x^0\| < \rho_{1/2}\}.$$

Wenn $h \leq \frac{1}{2}$ und $\overline{S_{\rho_1}(x^0)} \subset D$ gilt, dann existiert die Folge $(x^k)_{k\in\mathbb{N}}$ aus dem Newton-Verfahren (alle Matrizen $DF(x^k)$ sind regulär). Die Folge ist in $S_{\rho_1}(x^0)$ enthalten und konvergiert gegen eine Nullstelle von F. Diese Nullstelle ist eindeutig in der Menge $D \cap S_{\rho_2}(x^0)$.

Zum Beweis siehe: J. M. Ortega, W. C. Rheinboldt: Iterative Solution of Nonlinear Equations in Several Variables. SIAM, Philadelphia, 2000.

Der Satz ist eher von theoretischem Interesse, da die Voraussetzungen in den Anwendungen im allgemeinen nicht nachprüfbar sind. Im Satz von Newton-Kantorovich wird nur $F \in C^1$ verlangt. Zudem stellt die Forderung (iii) jedoch eine Lipschitz-Bedingung an DF. Diese ist für $F \in C^2$ lokal erfüllbar.

Als Beispiele betrachten wir die Gleichungssysteme

$$G(x_1, x_2) \equiv \begin{pmatrix} x_2 - 10 \arctan(x_1) \\ x_1 + 10 \arctan(x_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{9.7}$$

und

$$H(x_1, x_2) \equiv \begin{pmatrix} x_2 - \sin^3\left(\frac{\pi}{2} x_1\right) \\ x_1 + \sin^3\left(\frac{\pi}{2} x_2\right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{9.8}$$

Beide Systeme sind so konstruiert, dass die eindeutige Nullstelle $x_1 = x_2 = 0$ ist. Abbildung 32 und Abbildung 33 zeigen das Konvergenzverhalten beim gewöhnlichen und beim vereinfachten Newton-Verfahren. Die Grafiken verdeutlichen Konvergenz bzw. Divergenz für verschiedene Startwerte. In allen Darstellungen erkennt man einen konvexen Bereich um die Nullstelle, in dem für alle Startwerte Konvergenz eintritt. Für Beispiel (9.7) liegt außerhalb dieses Gebiets (abgesehen von einem kleinen Grenzbereich) stets Divergenz vor. In Beispiel (9.8) konvergieren bei weiter entfernten Startwerten die Iterationen teilweise. Man erkennt, dass der Konvergenzbereich des vereinfachten Newton-Verfahrens kleiner als beim gewöhnlichen Newton-Verfahren ist.

Es sei jedoch betont, dass diese Systeme hier Modellprobleme sind. Das Verhalten bei hochdimensionalen Gleichungssystemen aus Anwendungen kann mit diesen Beispielen nicht beurteilt werden.

Als ein praktisches Beispiel betrachten wir eine Kette aus $2n$ Segmenten der Länge 1, die an zwei gegenüberliegenden Punkten mit Abstand 6 befestigt ist. Unter der Schwerkraft nimmt die Kette eine bestimmte Form an, die eindeutig durch die Winkel zwischen den Segmenten und der Horizontalen festgelegt ist. Aus Symmetriegründen sind nur n Winkel zu bestimmen. Es folgt ein nichtlineares System aus $n + 1$ Gleichungen für die n Winkel $x_1, \ldots, x_n$ und die Horizontalkomponente x_{n+1} einer Kraft. Das System

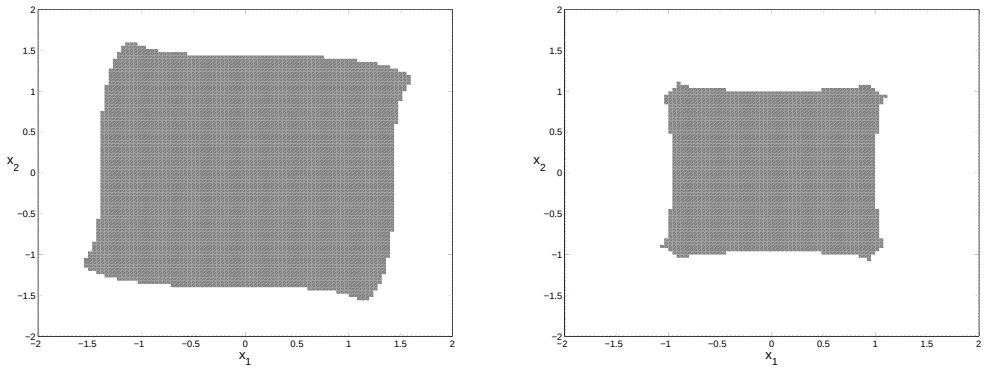


Abbildung 32: Konvergenz des gewöhnl. Newton-Verf. (links) und des vereinf. Newton-Verf. (rechts) bei Beispiel (9.7). (grau: Konvergenz, weiß: Divergenz)

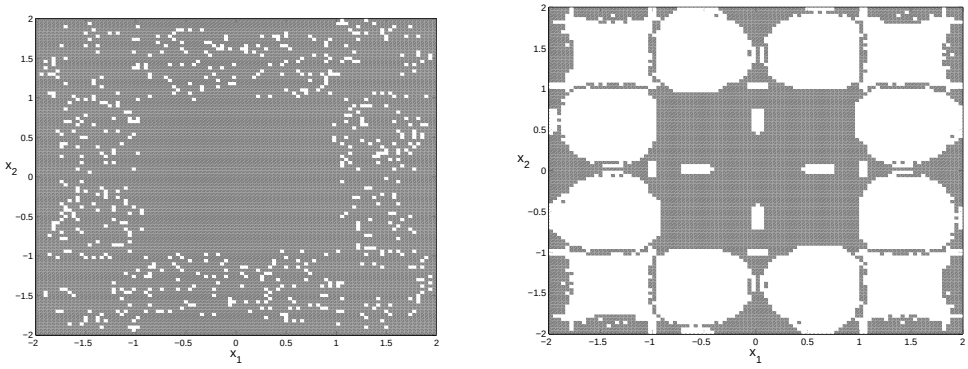


Abbildung 33: Konvergenz des gewöhnl. Newton-Verf. (links) und des vereinf. Newton-Verf. (rechts) bei Beispiel (9.8). (grau: Konvergenz, weiß: Divergenz)

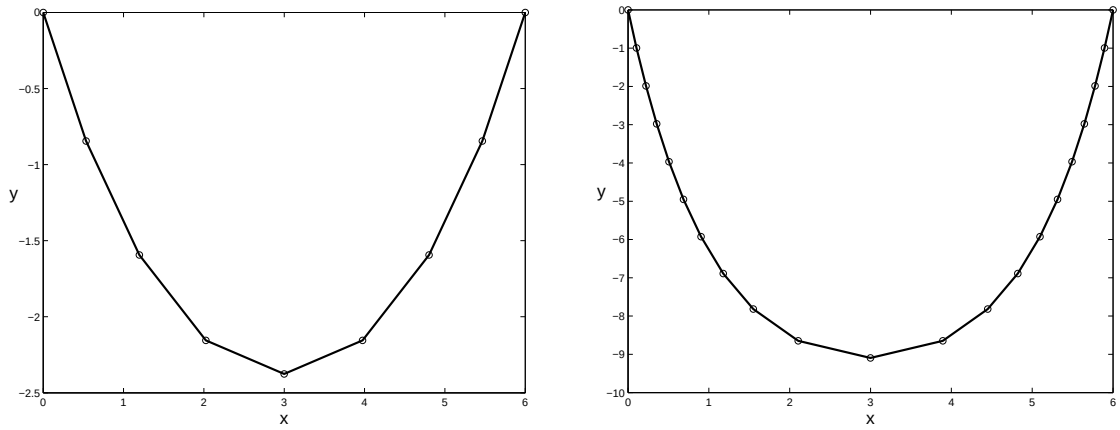


Abbildung 34: Kette mit 8 Segmenten (links) und 20 Segmenten (rechts).

$F(x) = 0$ mit $F = (f_1, \ldots, f_{n+1})^\top$ ist gegeben durch

$$\begin{aligned} f_i &= 2x_{n+1}\sin(x_j) - (2n - 1 - 2(j-1))\cos(x_j), \qquad i = 1, \ldots, n\ , \\ f_{n+1} &= -3 + \sum_{j=1}^{n} \cos(x_j). \end{aligned}$$

Wir lösen dieses nichtlineare Gleichungssystem mit dem (gewöhnlichen) Newton-Verfahren. Als Startwert verwenden wir $x_j^0 = \frac{\pi}{4}$ für $j = 1, \ldots, n$, da die Winkel im Intervall $(0, \frac{\pi}{2})$ liegen, und $x_{n+1}^0 = 1$. Abb. 34 zeigt die resultierende Kette in den Fällen $n = 4$ bzw. $n = 10$. Wählt man für die Winkel zu kleine Startwerte (z.B. $x_j^0 = \frac{\pi}{16}$), dann divergiert die Newton-Iteration.

Allgemein gilt: Divergiert bei gegebenem Startwert x^0 die Newton-Iteration und sind keine besseren Startwerte vorhanden, dann können folgende Alternativen versucht werden:

- *Modifiziertes Newton-Verfahren*
  Kombination aus dem gewöhnlichen Newton-Verfahren und einer Minimierung des Residuums $r(x) = \|F(x)\|_2^2$.
- *Einbettungsverfahren* (auch: *Fortsetzungsverfahren*)
  Einbettung des nichtlinearen Gleichungssystems in eine parameterabhängige Schar $\tilde{F}(x, \lambda) = 0$ aus Systemen mit $\tilde{F}(x, 1) = F(x)$, zu denen eine Anfangslösung bei $\tilde{F}(x, 0) = 0$ bereits bekannt ist.