Höhere Mathematik für Ingenieure IV
Vorlesungsmitschrift

Sebastian T. Hafner
sebastian@uni-saarland.info
Thomas Stahl
thommy@rids.de

12. Juli 2005

Inhaltsverzeichnis

0 Organisatorisches

0.1 Dozent

Prof. Dr. Volker John
Gebäude 27, Raum 301
Telefon: 0681 / 302 - xxxx
eMail: john@math.uni-sb.de

0.2 Übung

Mo A. Krebs	Di	Mi	Do C. Sucio	Fr C. Sucio
11-13				
14-16			14-16	14-16
16-18				

0.3 Übungsblätter

- wöchentlich Ausgabe montags (Download von der Homepage), Abgabe Woche danach
- Übungsgruppe Abgabe mit bis zu 3 Personen
- Programmieraufgaben Matlab, SciLab, Octave
- Abgabe der Programmieraufgaben per eMail an Annabel Krebs (annabelkrebs@aol.com, annabel@fs.math.uni-sb.de) und Oana Carina Sucio (oana_carina@yahoo.com)

0.4 Homepage

http://david.math.uni-magdeburg.de/~john/

0.5 Klausur

Montag, 27. Juli 2005, 9:00 Uhr, Länge 3-4h

Anmeldung bei Frau Leva

Zulassung: hinreichend: 50% der Übungsaufgaben richtig.

Bestanden: hinreichend: 50% richtig gelöst.

0.6 HMI3 Nachklausur

Anfrage an Hans Crauel

0.7 Literatur

- Liste auf Homepage

1 Einleitung

Diese Vorlesung beinhaltet Themen aus dem Gebiet der Numerischen Mathematik. Es werden numerische Verfahren für wichtige Probleme, die bei praktischen Aufgabenstellungen auftreten, präsentiert.

1.1 Aufgabenstellungen und Ziele der Numerischen Mathematik

Viele Probleme, die in der Mathematik auftreten, lassen sich mit „rein mathematischen“ Mitteln schwer oder gar nicht lösen. Einige Beispiele sind:

- Die Berechnung vieler bestimmter Integrale
- Die Berechnung von Nullstellen von Funktionen
- Die Lösung größerer linearer Gleichungsprobleme
- Die Lösung von Differentialgleichungen

Eine Aufgabe der Numerischen Mathematik besteht in der Entwicklung von Verfahren zur approximativen Lösung dieser Probleme, welche mit Hilfe eines Computers abgearbeitet werden können. Wichtige Fragestellungen, die im Rahmen der Numerischen Mathematik untersucht werden, sind:

- Wie teuer sind diese Verfahren?
  Beispiel: Ist die Lösung linearer Gleichungssysteme mit dem Gauß-Verfahren oder der Cramer'schen Regel schneller?
- Wie genau ist das berechnete Ergebnis?
- Unter welchen Bedingungen konvergieren Iterationsverfahren? Wie schnell ist die Konvergenz?
- Wie robust sind die Verfahren gegenüber kleinen Störungen in den Daten?

In dieser Vorlesung wird vor allem auf die Motivation für numerische Verfahren (Welche Idee steht dahinter?) und auf das Verhalten der Verfahren in der Praxis Wert gelegt, weniger auf die Analysis der Verfahren.

2 Computerzahlen

2.1 Darstellung von Zahlen

Auf einem Computer kann man nur eine endliche Menge von reellen ZTahlen darstellen. Deswegen ist jede Rechenoperation mit Computerzahlen potentiell mit Rundungsfehlern behaftet.
Es gibt unterschiedliche Möglichkeiten, reelle Zahlen darzustellen. Im Allgemeinen benutzt

man das *Positionssystem*. Sei $\beta \in \mathbb{N}$, $\beta \geq 2$, und $x \in \mathbb{R}$ mit einer endlichen Anzahl von Ziffern x_k mit $0 \leq x_k \leq \beta$, $k = -m \ldots n$ im Positionssystem zur Basis β hat x die Gestalt

$$x_\beta = (-1)^s [x_n \; x_{n-1} \; \ldots \; x_0 \; x_{-1} \; \ldots \; x_m] \tag{1}$$

$$x \neq 0$$

Im Fall $\beta = 10$ wir der Punkt Dezimalpunkt genannt. Die Potent s hat den Wert Null, falls x positiv ist und den Wert 1, falls x negativ ist. Die Darstellung (1) lässt sich wie folgt kurz hinschreiben:

$$x_\beta \; = (-1)^s \left(\sum_{k=-m}^{n} x_n \beta^k \right) .$$

Beispiel: $x_{10} = 123.75$

dezimal: $x_{10} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$

binär: $x_2 = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1\dot{2}^{-1} + 1 \cdot 2^{-2} = 1111011.11$

Die Ziffern des Binärsystems werden *Bits* genannt.
Hexadezimal:

$$\begin{aligned} x_{16} =& 7 \cdot 16^1 + \underbrace{B}_{\equiv 11} \cdot 16^0 + \underbrace{C}_{\equiv 12} \cdot 16^{-1} \\ =& 7B.C \end{aligned}$$

■

Eine rationale Zahl kann in einer Basis eine endliche Anzahl von Ziffern besitzen und in einer anderen eine unendliche Anzahl.

Beispiel:

$$x_{10} = \frac{1}{6}$$

dezimal:

$$\begin{aligned} x_{10} =& 1 \cdot 10^{-1} + 6 \cdot 10^{-2} + \ldots \\ =& 0.166\overline{6} \end{aligned}$$

sextal:

$$x_6 = 1 \cdot 6^{-1} = \quad 0.1$$

■

Auf einem Computer kann man nur Zahlen mit endlich vielen Ziffern darstellen. Man kann zeigen, dass jede reelle Zahl durch eine Folge von reelle Zahlen mit einer endlichen Anzahl von Ziffern angenähert werden kann. Dieser Umstand rechtfertigt die Verwednung von Zahlen mit endlich vielen Ziffern auf Computern.
Theoretisch sind zwar alle Basen äquivalent, jedoch verwenden fast alle modernen Computer $\beta = 2$.

Beispiel: Umrechnung von Zahlen verschiedener Basen

- β-System $\rightarrow$ Dezimalsystem

$$x_\beta = \sum_{i=-m}^{n} a_i \beta^i$$

einfach ausrechnen.

- Dezimalsystem $\rightarrow$ β-System
gegeben: $x_\beta \in \mathbb{Z}$ im Dezimalsystem
gesucht: x_β im β-System

$$x_\beta \quad = \sum_{i=0}^{n} a_i \beta^i = a_0 + a_1\beta + a_2\beta^2 + \ldots + a_n\beta^n \quad a_i \in [0 \ldots \beta - 1]$$

- 1. Schritt

$$\frac{x}{\beta} = \underbrace{a_1 + a_2\beta + \ldots + a_n\beta^{n-1}}_{x_1}$$

Rest $a_0 \;\rightarrow\; a_0$ bestimmt

- 2. Schritt

$$\frac{x_1}{\beta} = \underbrace{a_2 + a_3\beta + \ldots + a_n\beta^{n-2}}_{x_2}$$

Rest $a_1 \;\rightarrow\; a_1$ bestimmt

- ...

- n. Schritt

$$\frac{x_{n-1}}{\beta} = a_n$$

Rest $a_{n-1} \;\rightarrow\; a_{n-1}$ bestimmt

Beispiel: $x_{16} = 811$, gesucht x_b

$$\left.\begin{array}{llllll} 811:6 & = & 135 & R.1 & = & a_0 \\ 135:6 & = & 22 & R.3 & = & a_1 \\ 22:6 & = & 3 & R.4 & = & a_2 \end{array}\right\} x_6 = 3431$$

■

Als beste Form der Darstellung von Computerzahlen hat sich die *Fließkomma-Darstellung (floating point)* herausgestellt.

$$x_\beta = \quad (-1)^s (0\ a_1\ a_2\ a_t)\beta^e$$

$$= \quad (-1)^s \underbrace{(a_1\ a_2 \ldots a_t)}_{m} \beta^{e-t} \tag{2}$$

Hierbei ist $t \in \mathbb{N}$ die Anzahl der signifikanten Stellen, $0 \leq a_i \leq \beta - 1$. $m = a_1 a_2 \dots a_t$ ist die *Mantisse* und $e \in \mathbb{Z}$ wird *Exponent* genannt. Der Exponent variiert in einem Intervall zulässiger Zahlen

$$L \leq e \leq U \qquad L < 0,\ U > 0$$

Hat man zur Darstellung der Computerzahl N Speicherplätze, dann werden diese wie folgt aufgeteilt:

- Vorzeichen: ein Platz
- Mantisse: t Plätze
- Exponent: $N - t - 1$ Plätze

Die Zahl Null besitzt eine eigene Darstellung. Auf einem Computer sind üblicherweise zwei Formate zur Darstellung von Fließkommazahlen verfügbar: einfach und doppelt genau (*single and double precision*). Sei $\beta = 2$. Der Standard ist wie folgt:

- **einfach genau**, $N = 32$
  | s = 1 Bit | e = 8 Bit | m = 23 Bit |
- **doppelt genau**, $N = 64$
  | s = 1 Bit | e = 11 Bit | m = 52 Bit |

Die Menge der Fließkommazahlen wird mit

$$\mathbb{F}(\beta, t, L, U) = \{0\} \vee \left\{ x \ \in \mathbb{R} \ : \ x = (-1)^s \beta^e \sum_{\substack{i=1 \\ L \leq i \leq U}}^{t} a_i \beta^i \right\}$$

bezeichnet. Man kann nachrechnen, dass ihre Anzahl

$$\text{card } \mathbb{F} = 2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

Die Darstellung (2) der Fließkommazahlen wird erst durch die Bedingung $a_i \neq 0$ eindeutig.

Beispiel: $\beta = 10, T = 4, L = -1, U = 4$

$$x_{10} = 1$$
$$x_{10} = 0 \cdot 1000 \cdot 10^{-1} = 0 \cdot 0100 \cdot 10^{-2} = 0 \cdot 0010 \cdot 10^{-3} = 0 \cdot 0001 \cdot 10^{-4}$$

■

Man nennt die Zahlen in $\mathbb{F}(\beta, t, L, U)$ mit $a_1 \neq 0$ **normalisiert**.

Mathematische Fließkommazahlen:

$$a_1 \neq 0$$

Man sieht sofort, dass mit $x \in \mathbb{F}(\beta, t, U, L)$ auch $-x \in \mathbb{F}(\beta, t, U, L)$. Abgesehen von der Zahl Null findet man folgende Schranken für den Betrag einer Computerzahl:

$$\begin{aligned} x_{min} =& \beta^{L-1} \\ \leq& |x| \\ \leq& \beta^U(1-\beta^{-t}) = x_{max} \end{aligned}$$

Insbesondere enthält $\mathbb{F}(\beta, t, U, L)$ keine Zahl im Intervall $[0, x_{min}[$. Diese Lücke kann man aber auch einfach füllen, indem für den kleinsten Exponenten die Bedingung $a_1 \neq 0$ fallen lässt. Dann erhält man noch zusätzliche Zahlen, die vom Betrag her kleiner als x_{min} sind. In diesem Fall ist β^{L-t} die kleinste Zahl. Diese Zahl nennt man *denormalisiert*.
Die Fließkommazahlen sind auf der reellen Achse nicht mit gleichen Abständen geteilt. Sie sind desto dichter, je kleiner ihr Betrag ist. Man kann nachrechnen, dass der Abstand einer Zahl $x \in \mathbb{F}(\beta, t, U, L)$ zu seinem nächsten Nachbarn $y \in \mathbb{F}(\beta, t, U, L)$, $x \neq y$, sich wie fogt abschätzen lässt:

$$\begin{aligned} \beta^{-1}\epsilon_M x \leq& |x-y| \\ \leq& \epsilon_M |x| \end{aligned}$$

wobei

$$\epsilon_m = \beta^{1-t}$$

Maschinen Epsilon genannt wird. Das Maschinen Epsilon ϵ_M ist die kleinste Fließkommazahl, so dass

$$1 + \epsilon_m > 1$$

Zusatz: Die Systeme $\mathbb{F}(\beta, t, U, L)$, die man heute verwendet sind (weitestgehend) standardisiert:

- einfach genau:
$$\mathbb{F}(2, 24, -125, 128)$$
- doppelt genau:
$$\mathbb{F}(2, 53, -1021, 1024)$$

In beiden Systemen sind außerdem die denormalisierten Zahlen enthalten.
Der nun um Eins größere Wert von t erklärt sich dadurch, dass man im Fall $\beta = 2$ das Bit a_1 nicht speichern muss, da immer gilt $a_1 = 1$.

2.2 Runden von reellen Zahlen

Sei $x \in \mathbb{R}$ eine beliebige reelle Zahl. Im Allgemeinen wird $x \notin \mathbb{F}(\beta, t, U, L)$. auch das Ergebnis einer Operation von zwei Zahlen aus $\mathbb{F}(\beta, t, U, L)$ wird im allgemeinen nicht in $\mathbb{F}(\beta, t, U, L)$ liegen. Es muss die Frage geklärt werden, wie man Zahlen, die nicht zu $\mathbb{F}(\beta, t, U, L)$ gehören am besten in $\mathbb{F}(\beta, t, U, L)$ approximiert.

Der einfachste Weg ist das Runden. Diese Operation ist lässt sich als Abbildung beschreiben: Sei x in der Form (2) gegeben, wobei m mehr als t Stellen haben kann.

$$fl: \quad \mathbb{R} \longrightarrow \mathbb{F}(\beta, t, U, L)$$

$$fl(x) = (-1)^s(0, a_1 a_2 \ldots, \tilde{a}_t)\beta^l \tag{3}$$

$$\tilde{a}_t \begin{cases} a_t & \text{falls } a_{t+1} < \frac{\beta}{2} \\ a_t + 1 & \text{falls } a_{t+1} \geq \frac{\beta}{2} \end{cases}$$

Folgende Eigenschaften sind offensichtlich

- $x \in \mathbb{F}(\beta, t, U, L) \rightarrow f(x) \in \mathbb{F}(\beta, t, U, L)$
- $x \leq y \Rightarrow fl(x) \leq fl(y)$

Bemerkung Abbildung (3) ist nur wohldefiniert, falls $L \leq e \leq U$. Ist $x \in]-\infty, -x_{max}[\vee]x_{max}, \infty[$, so ist $fl(x)$ nicht definiert. Man spricht von einem *Overflow*, was normalerweise zum Abbruch des ausgeführten Programms führt. Ist $x \in]-x_{min}, x_{min}[$, spricht man von einem *Underflow*. In diesem Fall ist die Abbildung fl aber noch definiert. ■
Eine Alternative zum Runden ist das Abschneiden, bei welchem $\tilde{a}_t = a_t$ gesetzt wird. Bezüglich des Rundungsfehlers lässt sich folgender Satz beweisen:

Satz Sei $x \in \mathbb{R}$ mit $x_{min} \leq |x| \leq x_{max}$. Dann gilt:

$$fl(x) = x(1 + \delta) \quad |\delta| < u \tag{4}$$

wobei

$$u = \frac{1}{2}\underbrace{\beta^{1-t}}_{\epsilon_M} = \frac{1}{2}\epsilon_M$$

■

Die Zahl u wird *Rundungseinheit* oder *Maschinengenauigkeit* genannt. Aus (4) folgt für den relativen Fehler

$$E_{rel}(x) = \frac{|x - x - x\delta|}{|x|} = \frac{|x\delta|}{|x|} \tag{5}$$

2.3 Operationen mit Fließkommazahlen

Die Arithmetik von Fließkommazahlen sollte so weit wie möglich analog zur Arihtmetik in $\mathbb{R}$ sein.
Sei

$$\circ: \ \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

eine arithmetische Operation („+“„−“„·“„:“).
Die zugehörige Maschinenoperation wird mit $\boxed{\circ}$ bezeichnet. Die Anwendung von $\boxed{\circ}$ auf zwei reelle Zahlen x, y besteht aus drei Schritten:

1. Transformiere x, y auf Fließkommaformat
   $x \longrightarrow fl(x)$, $y \longrightarrow fl(y)$
2. Berechne $fl(x) \boxed{\circ} fl(y)$
3. Transformiere das Ergebnis auf Fließkommaformat
   $fl(x) \circ fl(y) \longrightarrow fl(fl(x) \circ fl(y))$

Es gibt Eigenschaften von „$\circ$“ die auch $\boxed{\circ}$ besitzt, wie das Kommutativgesetz der Addition oder der Multiplikation. Andere gehen jedoch verloren, zum Beispiel das Assoziativgesetz der Addition.

Beispiel

$$x = -1, \quad y = x_{max}, \quad z = -x_{max}$$

- $$x + y + z = 1$$
- $$(x \boxplus y) \boxplus z$$

$$\begin{aligned} x \boxplus y =& fl(fl(x) + fl(y)) \\ =& fl(-1 + x_{max}) \\ \overset{runden}{=}& x_{max} \end{aligned}$$

$$\begin{aligned} x_{max} \boxplus z =& fl(fl(x_{max}) + fl(-x_{max})) \\ =& fl(x_{max} - x_{max}) \\ =& 0 \end{aligned}$$

- $$x \boxplus (y \boxplus z)$$

$$\begin{aligned} y \boxplus z =& fl(fl(x_{max}) + fl(-x_{max})) \\ =& 0 \end{aligned}$$

$$\begin{aligned} x \boxplus 0 =& fl(-1 + 0) \\ =& -1 \end{aligned}$$

■

Man wird erwarten, dass für eine Fließkommaoperation, sofern sie wohldefiniert ist, gilt, dass für alle $x, y \in \mathbb{F}(\beta, t, U, L)$ ein $\delta \in \mathbb{R}$ existiert, so dass

$$x \boxed{\circ} y = (x \circ y)(1 + \delta) \text{mit } |\delta| < u. \tag{6}$$

Bei der Addition erhält man für den relativen Fehler

$$\frac{|(x \boxplus y) - (x + y)|}{|x + y|} \leq (2n + n^2)\frac{|x| + |y|}{|x + y|}$$

Der relative Fehler ist also klein, d.i. im Bereich von wenigen Vielfachen der Maschinengenauigkeit, falls $|x + y|$ nicht klein ist. Man erhält jedoch einen großen Fehler, falls man zwei fast gleichgroße Zahlen mit unterschiedlichen Vorzeichen addiert.
Die Subtraktion erfüllt die Eigenschaft (6) nur, falls die Struktur der Zahlen in $\mathbb{F}(\beta, t, U, L)$ noch um die sogenannte Rundungsziffer erweitert wird.

Beispiel:

$$x = 1, \quad y = 0.99$$

Berechne $x \boxminus y$ in $\mathbb{F}(10, 2, L, U)$

- $x - y = 0.01$
- $$\begin{array}{rlll} fl(x) & = 0.1 \cdot 10^1 & \longrightarrow & 0.10 \cdot 10^1 \\ fl(x) & = 0.99 \cdot 10^0 & \longrightarrow & 0.09 \cdot 10^1 \\ \hline & & x \boxminus y & 0.01 \cdot 10^1 \quad = \underline{0.1} \end{array}$$

Das Ergebnis ist um den Faktor 10 falsch!
Der Grund dafür ist, das beim Ausgleichen der Exponenten Stellen verloren gehen. Die Rundungsziffer vermindert diesen Effekt:

$$\begin{array}{rlll} fl(x) & = 0.10 \cdot 10^1 & \longrightarrow & 0.10\boxed{0} \cdot 10^1 \\ fl(x) & = 0.99 \cdot 10^0 & \longrightarrow & 0.09\boxed{9} \cdot 10^1 \\ \hline & & x \boxminus y & 0.001 \cdot 10^1 \quad = \underline{0.01} \end{array}$$

$\square$*Rundungsziffer*

In diesem Fall ist Ergebnis exakt!
Unterscheiden sich die Exponenten um mehr als Eins, trifft dies nicht mehr zu. Aber man kann zeigen, dass (6) gilt.
Die Rundungsziffer erhöht die Genauigkeit, verlangsamt aber die Berechnung. Der Trend ist jedoch, dass man sogar zwei Rundungsziffern verwendet. ■
Ein weiteres Problem bei der Subtraktion zweier fast gleichgroßer Zahlen, wird *Auslöschung* genannt. Wenn nämlich führende Stellen in der Mantisse übereinstimmen, gehen sie bei dieser Operation „verloren“.

Beispiel:

$$x = 0.123456, \quad y = 0.122123, \quad \mathbb{F}(10, 6, L, U)$$

Die letzten 3 Stellen von x und y seien fehlerbehaftet, d.i. man hat jeweils 3 sichere Stellen.

$$\begin{aligned} fl(x) &= 0.123\boxed{456} \cdot 10^0 \\ fl(y) &= 0.122\boxed{123} \cdot 10^0 \\ fl(x) - fl(y) &= 0.001\boxed{333} \cdot 10^0 \\ fl(fl(x) - fl(y)) &= 0.1\underline{33300} \cdot 10^{-2} \end{aligned}$$

Im Ergebnis gibt es nur eine sichere Stelle. D.i. die Fehler in den Ausgangsdaten werden extrem verstärkt.

Fazit: Man sollte nach Möglichkeit die Subtraktion zweier fast gleichgroßer Zahlen vermeiden!

Eine elementare Fließkommaoperation wird *Flop* genannt (in mehreren Büchern wird eine Operation der Gestalt $a + b \cdot c$ Flop genannt, Vorsicht!)

3 Klassifizierung von Problemen und numerischen Verfahren

3.1 Klassifizierung von Problemen

Wir betrachten das Problem: Finde x, so dass

$$F(x, d) = 0 \tag{1}$$

wobei d eine Datenmenge ist und F die funktionale Beziehung zwischen x und d beschreibt. Wir werden in dieser Vorlesung nur *direkte Probleme* betrachten, d.h. F und d sind gegeben, x ist unbekannt.
(sonst inverseses Problem $\longrightarrow$ Vorlesung Professour Louis)

Beispiel Gesucht ist die Lösung von $Ax = b$, $A \in \mathbb{R}^{n\times n}$ reguläre Matrix, $b \in \mathbb{R}^n$. Dann ist b die Datenmenge d und A beschreibt die funktionale Beziehung zwischen x und d. ■

Definition Das Problem (1) wird *korrekt gestellt* oder *stabil* genannt, wenn

- es eine eindeutige Lösung besitzt,
- die Lösung stetig von den Daten abhängt.

Andernfalls heißt das Problem *schlecht gestellt.* ■
In dieser Vorlesung nur gut gestellt Probleme.
Beispiel für ein schlecht gestelltes Problem siehe (Serie 02).

Stetige Abhängigkeit von den Daten bedeutet, dass kleine Störungen in den Daten nur kleine Änderungen in der Lösung bewirken. Solche Störungen in den Daten können beispielweise schon beim Runden $x \longrightarrow fl(x)$ auftreten. Bezeichnet man mit δd zulässige Störungen in den Daten und mit δx die resultierende Veränderung der Lösung, d.i.

$$F(x + \delta x, d + \delta d) = 0$$

Dann bedeutet die stetige Abhängigkeit von den Daten

$$\forall \mu > 0 \; \exists K(\mu, d) \; : \; \|\delta d\| < \mu \; \Rightarrow \; \|\delta x\| \; \leq \; K(\mu, d)\|\delta d\|$$

wobei $\| \bullet \|$ geeignete Normen sind. Natürlich ist es günstig, wenn $K(\mu, d)$ klein ist.

Definition Die *relative Konditionszahl* für Problem (1) definiert durch

$$K(d) = \sup_{\delta d \in D} \frac{\frac{\delta x\|}{\|x\|}}{\frac{\delta d}{\|d\|}}, \quad x \neq 0, \; d \neq 0,$$

wobei D eine Umgebung des Ursprungs ist, welche die zulässigen Störungen enthält, für die das gestörte Problem noch sinnvoll ist. Die *absolute Konditionszahl* ist gegeben durch

$$K_{\text{abs}} = \sup_{\delta d \in D} \frac{\|\delta x\|}{\|\delta d\|}$$

■

Problem (1) wird *schlecht konditioniert* genannt, falls $K(d)$ „groß“ ist für irgendwelche zulässigen Daten d. Was „groß“ genau bedeutet, hängt vom Problem ab, Beispiele später.
Da (1) eine eindeutige Lösung besitzt, gibt es notwendigerweise eine Abbildung G von der Menge der Daten auf die Menge der Lösungen mit

$$x = G(d).$$

Für das gestörte Problem gilt

$$x + \delta x = G(d + \delta d)$$

Nimmt man an, dass G differenzierbar ist, so verhält man aus der Taylorentwicklung.

$$\delta x = G(d + \delta d) - x = G(d + \delta d) - G(d) \overset{T.E.}{=} \underline{G(d)} + G'(d)\delta d - \underline{G(d)} + \sout{\text{Terme höherer Ordnung in } \delta d}$$

Hierbei ist $G(\delta)$ die Jacobi-Matrix von G in d. Vernachlässigt man die Terme höherer Ordnung erhält man

$$\begin{aligned} K(d) =& \sup_{\delta d} \frac{\|\delta x\|}{\|x\|} \frac{\|d\|}{\|\delta d\|} \\ \approx& \sup_{\delta d} \frac{\overbrace{\|G'(d)\delta d\|}^{\approx \delta x}}{\underbrace{G(d)}_{=x}} \frac{\|d\|}{\|\delta d\|} \\ =& \underline{\sup_{\delta d}} \frac{\|G'(d)\| \underline{\|\delta d\|}}{\|G(d)\|} \frac{\|d\|}{\underline{\|\delta d\|}} = \quad \|G(d)\| \frac{\|d\|}{\|\delta d\|} \end{aligned}$$

wobei $\| \bullet \|$ geeignete Normen sind.
Analog zeigt man

$$K_{\text{abs}}(d) \leq \|G(d)\|$$

Anmerkung: Selbst wenn die Konditionszahl formal ∞ ist, folgt nicht notwendigerweise, dass das Problem schlecht gestellt ist. Manchmal ist es möglich, das gegebene Problem äquivalent so umzuformen, dass man ein korrekt gestelltes Problem mit endlicher Konditionszahl erhält. Es gilt also (1) schlecht gestellt $\Rightarrow K(d) = \infty$, aber die Umkehrung gilt im Allgemeinen nicht. ■

3.2 Klassifizierung von numerischen Verfahren

Ein numerisches Verfahren zur Approximation der Lösung von (1) löst im Allgemeinen eine Folge von Näherungsproblemen

$$F_n(x_n, d_n) = 0 \ , \quad n \geq 1 \tag{2}$$

wobei n ein Parameter ist, der vom konkreten Problem abhäng. Für ein bracuhbares Verfahren erwartet man, dass für $n \longrightarrow \infty$ gilt $x_n \longrightarrow x$, d.h. die Folge der numerischen Lösungen konvergiert gegen die Lösung von (1).

Definition: Die Folge der Näherungsprobleme heißt *konsisten*, falls die Daten d für alle F_n zulässig sind und

$$F_n(x, d) = F_n(x, d) - \underbrace{F(x, d)}_{\longrightarrow 0}$$

für $n \longrightarrow \infty$.
Sie heißt *stark konsisten*, falls $F_n(x, d) = 0$ für jedes n. ■
Analog zu jedem Problem (1) definiert man auch für die Folge der Näherungsprobleme (2) den Begriff „korrekt gestellt“ oder „stabil“ und die relative und absolute Konditionszahl $K_n(d_n)$, $K_{\text{abs},n}(d)$.

Definition: Die *relative asymptotische Konditionszahl* der numerischen Methode (2) ist gegeben durch

$$K^{\text{num}}(d_n) = \lim_{k \longrightarrow \infty} \sup_{n \geq K} K_n(d_n) \ .$$

Das numerische Verfahren wird *gut konditioniert* genannt, falls K^{num} „klein“ ist, ansonsten *schlecht konditioniert*.

Beispiel: Summe zweier Zahlen: $d = (a, b)$

$$x = a + b \quad \Rightarrow \quad G(a, b) = a + b$$

Analog bei der Konditionszahl für (1) leitet man her

$$K_n(d_n) \approx \|G_n'(d_n)\| \frac{\|d_n\|}{\|G_n'(d_n)\|} \ .$$

Wir nehmen die l_1-Vektornorm. Dann ist mit $d = d_n,\ G = G_n\ \forall n$.

- $G'(a,b) = (1,1),\ \|G'(a,b)\|_1 = 1 + 1 = 2$
- $\|d\|_1 = |a| + |b|$
- $\|G(a,b)\|_1 = |a + b|$

Damit ist die Konditionszahl

$$K(a,b) \approx 2\frac{|a| + |b|}{|a + b|} \ .$$

Das heißt, die Additions zweier Zahlen ist gut konditioniert, falls diese das gleiche Vorzeichen besitzen, $K(a,b) \approx 2$.
Sie ist schlecht konditioniert, falls man zwei Zahlen subtrahiert, die etwa den gleichen Betrag besitzen. ■

Definition: Das numerische Verfahren wird *konvergent* genannt, falls

$$\forall \epsilon > 0 \exists n_0(\epsilon),\ \exists \delta(n_0, \epsilon) > 0 \ .$$

$$\forall n > n_0(\epsilon)\ \forall \|\delta d_n\| < \delta(n_0, \epsilon) \Rightarrow \|x(d) - x_n(d + \delta d_n)\| \leq \epsilon \ ,$$

wobei d zulässige Daten für (1) sind, $x(d)$ die zugehörige Lösung und $x_n(d + \delta d_n)$ ist die Lösung von (2) mit den Daten $d + \delta d_n$. ■

Satz (von Lax-Richtmeyer): *P. Lax (1965), R. Richtmeyer, K. Morton (1967);* Für ein konsistentes numerisches Verfahren sind Stabilität und Konvergenz äquivalent.

Beweis: Literatur.

3.3 Analyse numerischer Verfahren

Man unterscheidet zwei Vorgehensweisen.

Vorwärtsanalyse: Es wir der Fehler in der Kösung $\|\delta x_n\|$ abgeschätzt in der Abhängigkeit von den Störungen $\|\delta d_n\|$ und den Verfahrensfehlern „$F - F_n$“.

Rückwärtsanalyse: Sei eine Lösung $\widehat{x_n}$ gegeben. Dann sucht man bei der Rückwartsanalyse Datenstörungen δd_n, so dass

$$F_n(\widehat{x_n}, d + \delta d_n) = 0$$

Die numerische Lösung $\widehat{x_n}$ braucht nicht mit dem Verfahren F_n berechnet worden sein. Das brechnete Ergebnis $\widehat{x_n}$ kann als exaktes Ergebnis mit modifizierten Daten angesehen werden.

Die Vorwärtsanalyse ist oft komplizierter als die Rückwärtsanalyse, insbesondere bei komplexen Verfahren.

Summe von n Zahlen

$$s = \sum_{i=1}^{n} x_i$$

```
s = 0
for i = 1:n
  s = s + x(i)
end
```

Vorwärtsanalyse: Der i-te Schrit in der Schleife sieht im Detail wie folgt aus

$$\begin{aligned} s = &\quad fl(fl(s) + fl(x_i)) \\ \overset{(refkap2.5)}{=} &\quad (s + x_i)(1 + \delta_i) \quad |\delta_i| < \underbrace{u}_{\text{Maschinengenauigkeit}} \end{aligned} \tag{3}$$

Man erhält der Reihe nach

$$\begin{aligned} s_1 =& (s_0 + x_1)(1 + \delta_1) \\ =& x_1(1 + \delta_1) \\ =& 0 \\ s_2 =& (s_1 + x_2)(1 + \delta_2) \\ =& (x_1 + (1 + \delta_1) + x_2)(1 + \delta_2) \\ =& x_1(1 + \delta_1)(1 + \delta_2) + x_2(1 + \delta_2) \\ &\vdots \\ s_n =& \sum_{i=1}^{n} \left(\prod_{j=i}^{n} (1 + \delta_j) \right) x_i \\ =& fl(s) \end{aligned}$$

Wenn man Terme höherer Ordnung in δ_i vernachlässigt, ist

$$\prod_{j=i}^{n} (1 + \delta_j) \approx 1 + \sum_{j=i}^{n} \delta_j$$

Damit erhält man folgende Abschätzung für den relativen Fehler:

$$\begin{aligned}
\frac{|s - fl(s)|}{|s|} &< \frac{\left|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \delta_j\right) x_i\right|}{\left|\sum_{i=1}^{n} x_i\right|} \\
&\overset{\triangle \text{ Ungl.}}{\leq} \frac{\left|\sum_{i=1}^{n} \left(\sum_{j=1}^{n} |\delta_j|\right) |x_i|\right|}{\left|\sum_{i=1}^{n} x_i\right|} \\
&\overset{|\delta_j|<u}{\leq} \frac{\sum_{i=1}^{n} (n-i+1) u |x_i|}{\left|\sum_{i=1}^{n} x_i\right|} \\
&\overset{(n-i+1)<n}{<} \frac{n\ u\ \sum_{i=1}^{n} |x_i|}{\left|\sum_{i=1}^{n} x_i\right|}
\end{aligned}$$

Rückwärtsanalyse: Aus (3) folgt

$$\begin{aligned}
s_i &= (s_{i-1} + x_i)(1 + \delta_i) \\
&= s_{i-1} + x_i + (s_{i-1} + x_i)\delta_i \\
&\overset{(3)}{=} s_{i-1} + x_i + s_i \frac{\delta_i}{1 + \delta_i} \\
&\overset{(\delta \text{ klein}}{\approx} s_{i-1} + x_i + s_i \delta_i
\end{aligned}$$

Mit $s_0 = 0$ erhält man für den berechneten Wert

$$\begin{aligned}
fl(s) &= s_n \\
&= s_n - s_0 \\
&\overset{\text{Teleskopsumme}}{=} \sum_{i=1}^{n} n(s_i - s_i - 1) \\
&\approx \sum_{i=1}^{n} (x_i + s_i - \delta_i) \\
&= s + \sum_{i=1}^{n} s_i \delta_i
\end{aligned}$$

Für den relativen Fehler ergibt sich somit

$$\begin{aligned}\frac{|s - fl(s)|}{|s|} &\approx \frac{\left|\sum_{i=1}^{n} s_i \delta_i\right|}{|s|} \\ &\leq \frac{\sum_{i=1}^{n} |s_i| \overbrace{|\delta_i|}^{<u}}{|s|} \\ &\leq \frac{u \sum_{i=1}^{n} |s_i|}{\left|\sum_{i=1}^{n} x_i\right|}\end{aligned}$$

Diese Schranke ist im Allgemeinen realistischer als bei der Vorwärtsanalyse, insbesondere bei kleinen Zwischensummen ist die Abschätzung mit der Rückwärtsanalyse deutlich besser.

4 Numerische Lösung linearer Gleichungsprobleme

Dieses Kapitel behandelt numerische Verfahren zur Lösung linearer Gleichungssysteme (GS) der Gestalt

$$a_x = b \quad A \in \mathbb{R}^{n\times n},\ x, b\ \in \mathbb{R}^n \tag{1}$$

(Im letzten Abschnitt auch $A \in \mathbb{R}^{m\times n}$, $n \neq m$.)
Viele methoden zur numerischen Lösung von Problemen erfordern im Kern die Lösung solcher linearer Systeme. Deshalb sind effiziente und zuverlässige Verfahren zur numerischen Lösung von (1) von hoher Bedeutung.
In der Praxis unterscheidet man folgende Typen von linearen Systemen:

- die Dimension n ist klein, $n \leq 1000$,
- die Dimension n ist groß, $1000 \leq n \leq 10^8$, und die Elemente der Matrix sind überwiegend Null (A ist *schwach besetzt*),
- die Dimension n ist groß und A ist nicht schwach besetzt.

Es werden vorwiegend Verfahren für Matrizen vom Typ 1 behandelt. Außerdem werden Verfahren kurz vorgestellt, die man prinzipiell für alle Typen verwenden kann, die jedoch im Allgemeinen nicht sehr effizient sind.

4.1 Theorie

Aus der linearen Algebra ist folgender Sachverhalt bekannt:

Satz: Die folgenden Aussagen sind äquivalent:

- $A \in \mathbb{R}^{n\times n}$ ist eine reguläre Matrix, d.i. $Rg(A) = n$ (Rg = Matrixrand),
- alle Eigenwerte von A sind ungleich Null,
- Das System (1) hat genau eine Lösung,
- Das System (1) mit der rechten Seite $b = 0$ besitzt, nur die triviale Lösung $x = 0$,
- $\det(A) \neq 0$,
- A ist invertierbar.

Damit (1) korrekt gestellt ist, muss die Lösung eindeutig sein, also mus A regulär sein. Außerdem gilt für eine reguläre Matrix A und eine Datenstörung δb

$$Ax = b\delta b \Longleftrightarrow x = A^{-1}b + a^{-1}(\delta b)$$

Da die Matrixmultiplikation stetig ist, folgt

$$\lim_{\delta b \longrightarrow 0} A^{-1}(\delta b) = A^{-1}0 = 0$$

Also hängt die Änderung in der Lösung stetig von den Datenstörungen ab.
Damit ist gezeigt:

Satz: Problem (1) ist genau dann korrekt gestellt, wenn A regulär ist.
Zur Beschreibung der Kondition von (1) benötigt man Vektor- und Matrixnormen.

Vektornormen

$$x \in \mathbb{R}^n,\ l^p \text{ Norm},\ p \in [1, \infty[$$

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

p	Norm
$p = 1$	Summennorm
$p = 2$	Euklidische Norm
$p = \infty$	Maximum Norm

$$\|x\|_\infty = \max_{i=1..n} |x_i|$$

Matrixnormen

$$A \in \mathbb{R}^{m \times n}$$

induzierte Matrizen

$$\|A\|_p := \max_{x \in \mathbb{R}^n,\ x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p \tag{2}$$

Man findet

$$\|A\|_1 = \max_{j=1..n} \sum_{i=1}^{n} |a_{ij}| \quad \text{Spaltensummennorm}$$

$$\|A\|_2 = \sqrt{\underbrace{\lambda \max}_{\text{maximaler Eigenwert}} (A^T\ A)} \quad \text{Spektralnorm}$$

$$\|A\|_\infty = \max_{i=1..n} \sum_{j=1}^{n} |a_{ij}| \quad \text{Zeilensummennorm}$$

Außerdem

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}| \right) \quad \text{Frobenius Norm}$$

Aus (2) folgt sofort für alle $x \in \mathbb{R}^n$, $x \neq 0$

$$\|A\|_p \geq \frac{\|Ax\|_p}{\|x\|_p} \quad \Longleftrightarrow \quad \underbrace{\|Ax\|_p \leq \|A\|_p \|x\|_p}_{\text{gilt auch für } x=0}$$

Falls eine Vektor- und Matrixnorm solch eine Ungleichung erfüllen, nennt man sie *verträglich*. Induktiv folgt für $B \in \mathbb{R}^{n\times l}$:

$$\|ABx\|_p \le \|A\|_p\|Bx\|_p \le \|A\|_p\|B\|_p\|x\|_p \quad \Longleftrightarrow \quad \|AB\|_p = \max_{x\in\mathbb{R}^l\setminus\{0\}} \frac{\|ABx\|_p}{\|x\|_p} \le \|A\|_p\|B\|_p$$

Sei jetzt wieder $A \in \mathbb{R}^{n\times n}$, A regulär. Für die relative Konditionszahl hatten wir in Kapitel 3 eine Näherungsformel hergeleitet:

$$K(b) \approx \frac{\|G'(b)\|\|b\|}{\|G(b)\|},$$

wobei G die Abbildung von den Daten b auf die Lösung x ist. Für das lineare System (1) gilt $x = A^{-1}b$, also $G(b) = A^{-1}b$, und es folgt

$$G'(b) = \frac{dG(b)}{db} = A^{-1}$$

Durch Einsetzen erhält man beliebige verträgliche Normen

$$\begin{aligned} K(b) \approx & \frac{\|A^{-1}\|\|b\|}{\|A^{-1}b\|} \\ = & \frac{\|A^{-1}\|\|Ax\|}{\|x\|} \\ \le & \frac{\|A^{-1}\|\|A\|\underline{\|x\|}}{\underline{\|x\|}} \\ = & \|A^{-1}\|\|A\| \end{aligned}$$

Definition Die Konditionszahl einer Matrix $A \in \mathbb{R}^{n\times m}$ ist definiert als

$$\kappa(A) = \|A\|\|A^{-1}\|,$$

wobei $\|\cdot\|$ eine der oben angegebenen Normen ist. ■

Verschiedene Normen ergeben unterschiedliche Konditionszahlen. Diese werden durch einen Index unterschieden, z.B.:

$$\kappa_\infty = \|A\|_\infty\|A^{-1}\|_\infty$$

Eigenschaften der Konditionszahl

- Es gilt: $\kappa(A) > 1$
$$1 = \|\mathtt{I}\| = \|A\ A^{-1}\| \le \|A\|\|A^{-1}\| = \kappa(A)$$
- $\kappa(A^{-1}) = \kappa(A)$

- Sei $\alpha \in \mathbb{R}\backslash\{0\} \Rightarrow$

$$\begin{aligned}\kappa(\alpha A) =& \|\alpha A\|\|(\alpha A)^{-1}\| \\ =& \alpha\|A\|\|\alpha^{-1}A^{-1}\| \\ =& \underbrace{\alpha\alpha^{-1}}_{=1}\|A\|\|A^{-1}\| \\ =& \kappa(A)\end{aligned}$$

- Ist A eine orthogonale (unitäre) Matrix, d.i. $A^T = A^{-1}$, dann ist $\kappa_2(A) = 1$

$$\begin{aligned}\|A\|_2 =& \|A^T\|_2 \\ =& \|A^{-1}\|_2 \\ \|A\|_2 =& \sqrt{\lambda\max\underbrace{A^T A}_{\mathtt{I}}} \\ =& \sqrt{1} \\ =& 1\end{aligned}$$

- Sei A symmetrisch und positiv definit (alle Eigenwerte von A positiv). Dann gilt:

$$\kappa_2(A) = \frac{\lambda\max(A)}{\lambda\min(A)} \qquad \text{Spektral-Konditionszahl}$$

Definiert man den relativen Abstand einer regulären Matrix A zur Menge der singulären Matrizen wie folgt:

$$\mathrm{dist}_p(A) := \min\left\{\frac{\|\delta A\|_p}{\|A\|_p} \;:\; A + \delta A \text{ ist regulär}\right\}$$

So kann man zeigen

$$\mathrm{dist}_p(A) = \frac{1}{\kappa_p(A)}$$

Daraus folgt, daß bei Matrizen mit großer Konditionszahl schon kleine Störungen dazu führen können, daß die gestörte Matrix singulär ist. Die Matrix $A + \delta A$ ist auf jeden Fall regulär, falls

$$\frac{\|\delta A\|_p}{\|A\|_p} < \kappa_p^{-1}(A) \quad \Longleftrightarrow \quad \|\delta A\|_p\|A^{-1}\|_p < 1$$

$$Ax = b\,, \quad A \in \mathbb{R}^{n\times n}$$

- korrekt festellt $\leftrightarrow$ A invertierbar.
- Konditionszahl

$$u(A) = \|A\| \cdot \|A^{-1}\|$$

- Abstand von zwei Matrizen A und B .... $\frac{1}{k(A)}$ ... „nächstliegende reguläre Matrix“

Wegen der Rundungsfehler wird ein numerisches Verfahren zur Lösung von (4.1) nur eine Näherungslösung berechnen, die jedoch Lösung eines gestörten linearen Systems ist.

$$(A + \delta A)(x + \delta x) = b + \delta b \ . \tag{3}$$

Es gilt folgende Abschätzung

Satz: Seien $A \in \mathbb{R}1n \times n$ reguläre Matrix, $\| \cdot \|$ eine Matrixnorm und $\delta A \in \mathbb{R}^{n \times n}$, so dass $\|\delta A\| \cdot \|A^{-1}\| < 1$. Sei x die Lösung von $Ax = b \ , \quad b \in \mathbb{R}^n \ , \quad b \neq 0$, dann gilt für δx aus (3):

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{k(A)}{1 - k(A)\frac{\|\delta A\|}{\|A\|}} \cdot \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right)$$

wobei $k(A)$ die zu $\| \cdot \|$ gehörige Konditionszahl ist.

Beweis: Literatur.

■

4.2 Numerische Lösung linearer Systeme mit Dreiecksmatrix

Die Matrix A habe eine der beiden Formen

$$L = \begin{pmatrix} l_{11} & & & 0 \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \quad \text{untere Dreiecksmatrix („lower“)}$$

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ & u_{22} & \dots & \vdots \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{pmatrix} \quad \text{obere Dreiecksmatrix („upper“)}$$

Da A regulär sein soll, folgt $l_{il} \neq 0$, $n_{ii} \neq 0$, $i = 1..n$. Die Komponenten des Lösungsvektors können in diesen Fällen nacheinander berechnet werden, z.B. für $Lx = b$

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}} \\ x_2 &= \frac{b_2 - l_{21}x_1}{l_{22}} \\ x_i &= \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} - x_j \right) \quad i = 1..n \end{aligned}$$

Die Vorgehensweise wird in dieser Form interpretiert. Die Gesamtanzahl der benötigten Flops ist n^2.
Für $l_x = b$ ist die Reihenfolge der berechneten Lösungskomponenten $x_1 \longrightarrow x_2 \longrightarrow \ldots \longrightarrow$

x_n und man spricht von *Vorwärtssubstitution*. Löst man $Ux = b$, so erhält man $x_n \longrightarrow x_{n-1} \longrightarrow \ldots \longrightarrow x_1$, dies wird *Rückwärtssubstitution* genannt.

Benötigt man den Vektor b nicht mehr, so kann im Verfahren die berechnete Komponente x_i sofort auf den Platz von b speichern.
Führt man eine Rundunksfehleranalyse für die Vorwärts- (Rückwärts-) Substitution durch, erhält man folgendes Ergebnis:

Satz: Die mit der Vorwärtssubstitution berechnete Lösung x genügt der Gleichung

$$(l + \delta L)x = b$$

mit einer unteren Dreiecksmatrix δL, die komponentenweise klein ist im Sinne von

$$|(\delta L)_{ij}| \leq uj \ \ |l_{ij}| \quad i, j = i.n, \ i \geq j$$

u Maschinengenauigkeit

Beweis: Literatur. ■

4.3 Das Gauß-Verfahren und die LU-Zerlegung

Das bekannte Gauß-Verfahren ist, bei richtiger Anwendung, ein stabiles Verfahren zur Lösung linearer Gleichungssysteme.
Gegeben sind:

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & \vdots & \ldots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ \ldots \\ b_n \end{pmatrix}$$

und gesucht ist die Lösung von $Ax = b$

$$x = \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix}$$

Komponentenschreibweise

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

Vorgehen: Man formt das System $Ax = b$, in ein äquivalentes System $Ux = \vec{b}$ um, U - Obere Dreiecksmatrix, welches sich durch Rückwärtssubstitution leicht lösen lässt. Bei den Uformungen nutzt man aus, dass sich die Lösung des Systems $Ax = b$ nicht ändert, wenn:

- ein Vielfaches einer Gleichung zu einer anderen Gleichung addiert wird,
- zwei Gleichungen vertauschen,
- eine Gleichung mit einer reellen Zahl $\neq 0$ multipliziert wird.

4.3.1 Vorwärtselimination

(Erzeugung von $U_x = \widetilde{G}$).

- Falls $a_{11} \neq 0 \rightarrow$ Elimination x_1 aus den letzten $(n-1)$ Gleichungen, indem man von der i-ten gleichung das
$$m_{i1} = \frac{a_{i1}}{a_{11}}$$
-fache der 1. Gleichung subtrahiert, $i = 2, \ldots, n$. Man erhält das modifizierte System
$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$
mit
$$a_{1i}^{(1)} = a_{1i} \ , \quad a_{ij}^{(2)} = a_{ij} - m_{i1} a_{1j}$$
$$b_1^{(1)} = b_1 \ , \quad b_2^{(2)}) b_i^{(1)} - m_{i1} b_1^{(1)}$$

Bezeichnung:
$$A^{(2)} x = b^{(2)}$$

- Falls $_{22}^{(2)} \neq 0$, wendet man das gleiche Vorgehen auf $A^{(2)} x = b^{(2)}$ an und eleminiert x_2. Dieses Vorgehen wird festgesetzt, falls $a_{ii}^{(i)} \neq 0 \ , \quad i = 3 \ldots (n-1)$. Als Ergebnis erhält man ein oberes Dreiecksystem
$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \ldots & a_{21}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix} \tag{4}$$

4.3.2 Rückwärtssubstitution

Die Lösung x wird nun durch Rückwärtssubstitution aus (4.4) berechnet.

■

Dieses Vorgehen funktioniert, solange die sogenannten *Pivotelemente* $a_{ii}^{(i)}$, $i = 1, \ldots, n$ ungleich Null sind. Das ist für reguläre Matrizen nicht sebstverständlich, z.B.

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \rightarrow a_{11}^{(1)} = 0$$

Sei beim k-ten Schritt das Pivotelement $a_{kk}^{(k)} = 0$. Eines der Elemente $a_{ik}^{(k)}$, $i = k, \ldots, n$ der k-ten Spalte muss $\neq 0$ sein, etwa $a1(k)_{k'k}$. Sonst wäre die k-te Spalte von den vorhergehenden linear abhängig und A ist singulär.
Man vertauscht die Zeilen k und k', auch in der rechten Seite, und setzt die Vorwärtselimination fort.
Zur Fehlerdämpfung stellt es sich als günsitg heraus, wenn die Multiplikatoren $m_{ik} = \frac{a_{ik}^{(i)}}{a_{kk}^{(k)}}$ von Betrage her möglichst klein sind. Das bedeutet, $|a_{kk}^{(k)}|$ sollte möglichst groß sein. Zwei gebräuchliche Strategien, um dieses Ziel zu erreichen, sind:

Spaltenpivotsuche: Suche betragsmäßig größtes Element in der k-ten Spalte: $|a_{kk}^{(k)}| \geq |a_{ik}^{(k)}|$; $i \geq k$. Tausche dann die Zeilen k und k', auch in b.

vollständige oder totale Pivotsuche: Wähle als $a_{kk}^{(k)}$ das betragsgrößte Element der ganzen Restmatrix $A^{(k)}$.

$$|a_{k'k''}^{(k)}| > \max_{i,j} |a_{ij}^{(k)}| \ , \quad i, j \geq k \ .$$

Dann vertauscht man die Zeilen k und k''. Man beachte, dass ein Spaltentausch die Reihenfolge der Unbekannten ändert. Nach der Rückwärtssubstitution muss man zurück tauschen.

Bei beiden Pivotstrategien erhält man $|m_{ik}| \leq 1$. Erhält man zum Schluss der Vorwärtselimination auch mit Pivotsuche kleine Pivotelemente, so ist das ein Hinweis darauf, dass A schlicht konditioniert ist. Die vollständige Pivotsuche ist wesentlich teurer. In der Praxis reicht i.A. die Spaltenpivotsuche.
Sei $n = 2$ und sei das Gauß-Verfahren ohne Pivotsuche durchgeführt. Wir definieren mit $m_{21} = \frac{a_{21}}{a_{11}}$

$$L = \begin{pmatrix} 1 & 0 \\ m_{21} & 1 \end{pmatrix} \ , \quad U = \begin{pmatrix} a_{11} & a_{12} \\ & a_{22}^{(2)} \end{pmatrix} \ ,$$

wobei

$$a_{22}^{(2)} = a_{22} - m_{21} a_{12} \ .$$

Es gilt

$$\begin{aligned} LU &= \begin{pmatrix} a_{11} & a_{12} \\ m_{21}a_{11} & m_{21}a_{12} + a_{22}^{(2)} \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21}\frac{\underline{a_{11}}}{\underline{a_{11}}} & \underline{m_{21}a_{12}} + a_{22} - \underline{m_{21}a_{12}} \end{pmatrix} \\ &= A \end{aligned}$$

Diese Beobachtung lässt sich für $A \in \mathbb{R}^{n\times n}$ verallgemeinern. Sei

$$L = \begin{pmatrix} 1 & & & 0 \\ m_{21} & 1 & & \\ \vdots & & \ddots & \\ m_{n,1} & \dots & m_{n,n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ 0 & & & a_{nn}^{(n)} \end{pmatrix}$$

Dann gilt

$$A = LU \quad \text{bzw.} \quad PA = LU \ ,$$

wobei P eine Permutationsmatrix ist, welche die Zeilen- und Spaltenvertauschungen beschreibt, die bei der Vorwärtselimination vorgenommen werden. Diese Matrix muss nicht gespeichert werden, dalls alle erforderlichen Vertauschungen in der rechten Seite gleich ausgeführt werden. $b \to Pb$. Man spricht von der *LU-Zerlegung von A*.

$$Ax = b$$

- Gauß-Verfahren
- Pivotsuche

$$L = \begin{pmatrix} 1 & 0 & \dots & & \\ m_{21} & 1 & 0 & \dots & \\ m_{31} & m_{32} & 1 & 0 & \dots \\ \dots & & & \vdots & \\ m_{n1} & \dots & & & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \\ 0 & a_{22}^{(1)} & \dots & \\ & & & a_{nn}^{(1)} \end{pmatrix}$$

Die allgemeine praktische Herangehensweise zur Lösung von $Ax = b$ besteht aus drei Schritten:

1. Berechne die LU-Zerlegung A: $PA = LU$
2. Berechne y aus

   $$L(\underbrace{Ux}_{y}) = Ly = Pb$$

   durch Vorwärtssubstitution

3. Berechne x aus

$$Ux = y$$

durch Rückwärtssubstitution

■

Wenn die Matrix A nicht mehr explizit benötigt wird, können L und U auf dem Platz von A gespeichert werden. Beachte, die Diagonale von L braucht man nicht speichern, daderen Einträge ($= 1$) sowieso bekannt sind. Ebenso kann der Vektor b zunächst mit y und zum Schluss mit der Lösung x überschrieben werden.
Die Kosten zur Lösung von $Ax = b$ sind wie folgt:

1. $\frac{2(n-1)n(n+1)}{3} - n$ Flops
2. n^2 Flops
3. n^2 Flops

$$\frac{2}{3}n^3 + n^2 - \frac{1}{3}n \text{ Flops } \approx \underline{\frac{2}{3}n^3} \text{ Flops}$$

Führt man eine Rundungsfehleranalyse für die LU-Zerlegung durch, erhält man folgendes Ergebnis:

Satz Führt man die LU-Zerlegung von A mittels Gauß-Elimination mit Pivotsuche durch (Spalten- oder vollständig), so genügen die berechneten Dreiecksmatrizen L und U der Gleichung

$$LU = A + \epsilon$$

und für die Fehlermatrix ϵ gilt

$$\|\epsilon\|_\infty \leq n^2 u \underbrace{\left(\max_{i,j,k} \frac{|a_{ij}^{(k)}}{\|A\|_\infty} \right)}_{=:\rho} \|A\|_\infty$$

Beweis Literatur ■

Bemerkungen 1

Der Satz besagt, dass die berchneten Dreiecksfaktoren L und U die exakten Dreiecksfaktoren einer gestörten Matrix $A + \epsilon$ sind. Wenn n^2 und ρ nicht zu groß sind, liegen die Störungen in der Größenordnung des Rundungsfehlers der a_{ij} (Datenfehler).

Bemerkungen 2

Für die Gauß-Elimination mit Spaltenpivotsuche kann man Beispiele konstruieren, bei denen ρ sehr groß ist ($\rho = 2^{n+1}$). Für diese Beispiele ist ρ bei der Gauß-Elimination mit vollständiger Pivotsuche wesentlich kleiner. Diese Beispiele sind jedoch eher pathologisch und in den allermeisten praktischen Fällen ist ρ auch bei der Spaltenpivotsuche klein. Wegen dieser Erfahrungen wird Gauß-Elimination mit Spaltenpivotsuche als pratisch stabil angesehen und im Allgemeinen verwendet.

Bemerkungen 3

Die LU-Zerlegung ohne Pivotsuche ist im Allgemeinen instabil. Es gibt allerdings Klassen von Matrizen, für die man zeigen kann, dass die LU-Zerlegung auch ohne Pivotsuche stabil ist, beispielsweise für symmetrisch (positiv oder negativ) definite Matrizen.

Analysiert man das Gesamtverfahren 1) - 3) zur Lösung von $Ax = b$, erhält man folgende Aussagen:

Satz Löst man das lineare Gleichungssystem $Ax = b$ mittels Gauß-Elimination mit Spalten- oder vollständiger Pivotsuche, so ist die berechnete Lösung x die exakte Lösung des gestörten Systems

$$(A + \delta A)\, x = b$$

und für die Fehlermatrix δA gilt die Abschätzung

$$\|\delta A\|_\infty \leq \left(n^3 u + 3n^2\right) \rho u \|A\|_\infty$$

wobei ρ im vorherigen Satz definiert wurde. Sei x^* die Lösung von $Ax = b$, dann gilt für den Fehler $\delta x = x - x^*$:

$$\|\delta x\|_\infty \leq k_\infty(A)(n^3 u + 3n^2)\rho u \|x\|_\infty.$$

Beweis Kombiniere die Aussagen zur LU-Zerlegung und zur Vorwärts-/Rückwärtssubstitution. ■

Bemerkungen 1

Die Abschätzung von $\|\delta A\|_\infty$ im Satz stellt insbesondere für große n im Allgemeinen eine starke Überschätzung dar.
Inder Praxis ist $\|\delta A\|_\infty$ kaum größer als $nu\|A\|_\infty$. Ebenso ist die Abschätzung von $\|\delta x\|_\infty$ im Allgemeinen pessimistisch.
Man erkennt jedoch den Einfluß der Konditionszahl.

Bemerkungen 2

Hat man mehrere Systeme mit der Matrix A und unterschiedlichen rechten Seiten $b_1, \ldots, b_k$ zu lösen. und sind alle rechten Seite ngleichzeitig bekannt, so führt man die Vorwärtssubstitution wie gehabt aus. (ca. $\frac{2}{3}n^3$ Flops). Danach wendet man Vorwärts- und Rückwärtssubstitution für alle rechten Seiten an ($2n^2$ Flops pro rechte Seite).
Werden die rechten Seiten nacheinander berechnet, hängt eine rechte Seite b_j vielleicht sogar von den vorausgegegangenen rechten Seiten und Lösungen ab, so führt man für b_1 das oben beschriebene Gauß-Verfahren durch (ca. $\frac{2}{3}n^3$ Flops). Die LU-Zerlegung von A und gegebenenfalls die Permutationsmatrix merkt man such (Spaltenpivotsuche: P ist ein Vektor der Dimension n). Für alle nachfolgenden rechten Seiten hat man

$$\begin{aligned} Ax = b_{ij} &\Leftrightarrow PAx = Pb_j \\ &\Leftrightarrow LUx = Pb_j \end{aligned}$$

zu lösen. D.i. man hat nur die Vorwärts- und Rückwärtssubstitution auszuführen. ($2n^2$ Flops pro rechte Seite).

Bemerkungen 3

Sind die Matrizen L und U auf dem Speicherplatz von A gespeichert und brauchtman den Wert des Produktes $x = Ay$, so erhält man diesen durch $z = Uy$, $x = Lz$.
Beide Wege benötigen die gleiche Anzahl von Flops.

Bemerkungen 4

Die inverse Matrix erhält man, falls man die in Bemerkung 2 beschriebene Vorgehensweise mit $b_j = e_j - j$-ter Einheitsvektor, $j = 1..n$ durchführt. Die Berechnung der inversen Matrix ist pratisch fast nie von Interesse.
Hat man in irgendeiner Aufgabenstellung „$A^{-1}b$“ zu berechnen, so heißt das „Löse das System $Ax = C$
A^{-1} wird hierbei nicht gebraucht.

Bemerkungen 5

Es gilt $\det(P) = 1$ (Eigenschaft von Permutationsmatrizen)

$$\begin{aligned} \det(PA) &= \det(P)\det(A) \\ &= \det(A) \\ &= \det(LU) \\ &= \underbrace{\det(L)}_{=1}\det(U) \\ &= \prod_{i=1}^{n} a_{ii}^{(i)}. \end{aligned}$$

Bemerkungen 6

Ist A symmetrisch und (positiv) definit ($A^T = A\ , \quad \lambda(A) > 0$), so verwerndet man eine Variante des Gauß-Algorithmus, bei der man eine Zerlegung

$$A = C^T C\ , \quad C = \text{obere Dreiecksmatrix}$$

erhält - *Cholesky-Verfahren*. Wenn A symmetrisch ist, reicht es das obere Dreieck von A zu speichern. Dei der normalen *LU-Zerlegung* müsste man sowohl C als auch U speichern, also eine volle quadratische Matrix. Beim Cholesky-Verfahren kann man dem Platz von A speichern.

Bemerkungen 7

Falls sich die Einträge von A um viele Größenordnungen unterscheiden, ist es wahrscheinlich, dass während des Eliminationsprozesses betragsmäßig große Einträge und kleine Einträge summiert werden und große Rundungsfehler verursachen. Das kann man verhinder, in dem man $Ax = b$ vorher äquivalent in ein System $\bar{A}x = \bar{b}$ umformt. Eine gebräuchliche Herangehensweise ist die *Zeilenäquilibrierung*, bei welcher $\bar{A} = DA$ ist mit einer Diagonalmatrix D und

$$d_i = \frac{\overbrace{\max_k \sum_j |a_{kj}|}^{\text{maximale Zeilensumme}}}{\underbrace{\sum_j |a_{ij}|}_{i\text{-te Zeilensumme}}}$$

in diesem Fall kann man zeigen, dass

$$K_\infty(\bar{A}) \leq K_\infty(A)\ ,$$

so dass man i.A. eine Verbesserung der Abschätzung des Rundungsfehlers erhält.

4.4 Klassische Interationsverfahren zur Lösung linearer GS

Bei vielen Anwendungen, z.B. bei der numerischen Lösung partieller Differentialgleichungen, muss man letztlich große lineare Systeme mit schwach besetzten Matrizen lösen. Dazu sind direkte Verfahren, wie das Gauß-Verfahren, ungeeignet. Zum einen ist die Rechenzeit zu lang:

n	Computer mit 1 Gigaflop (10^9 Flops/s)
10	$7e-7s$
10^2	$6e-4s$
10^3	$0.6\bar{6}s$
10^4	$667s \approx 11min$
10^5	$666667s \approx 7.7d$
10^6	$6661e8s \approx 21.1a$

Für alle Rechenzeiten gilt $\frac{2}{3}n^3$.

Der Durchschnittscomputer besitzt heutzutage wesentlich weniger als 1 GFlop (effektive) Rechenleistung.

Wendet man die Gauß-Elimination auf eine schwach besetzte Matrix an, dann muss man damit rechnen, dass die Faktoren L und U nicht mejr schwach besetzt sind. Deshalb benötigt man beim Gauß-Verfahren im Allgemeinen $O(n^2)$ Speicherplätze.

$$\lim_{n\longrightarrow\infty} \frac{\text{Ausdruck}}{n^2} = C \ .$$

n	Speicherplatz n^2 (doppelt genau)
10	800 Byte
10^2	7.8 kByte
10^3	7.6 MByte
10^4	762.9 MByte
10^5	74.5 GByte

Dieser Speicherbedarf ist eine weitere Restriktion für die Größe des linearen Systems, auf die das Gaußsche Verfahren angewendet werden kann.
Für große Systeme nutzt man statt eines direkten Verfahrens, iterale Methoden. Dabei kann man die bekannte Idee der Fixpunktiteration nutzen. Man zerlegt di eMatrix A in die form

$$A = M - N \ , \quad M, N \in \mathbb{R}^{n\times n} \ ,$$

wobei M invertierbar ist. Aus

$$Ax = b$$

erhält man damit die äquivalente Fixpunktgleichung

$$Mx = b + Nx \Leftrightarrow x = M^{-1}(b + Nx) \ .$$

Ist eine Startnäherung $x^{(0)} \in \mathbb{R}^n$ gegeben, so lautet die Fixpunktiteration zur Lösung von $Ax = b$:

$$x^{(k+1)} = M^{-1}\left(b + Nx^{(k)}\right) \ , \quad k = 0, 1, 2, \ldots \tag{5}$$

Über die Konvergenz dieser Iteration gibt natürlich der Banachsche Fixpunktsatz Auskunft. Für den Spezialfall (5) erhält man

Satz: Das iterative Verfahren (5) konvergiert genau dann für alle $x^{(0)} \in \mathbb{R}^{n\times n}$ gegen die Lösung von $Ax = b$, wenn für die zugehörige Iterationsmatrix $M^{-1}N$ gilt:

$$\max\left\{|\lambda| : \ \lambda \text{ ist ein Eigenwert von } M^{-1}N\right\} < 1 \ .$$

■

In jedem Iteratinsschritt von (5) muss man ein lineares Gleichungssystem der Gestalt

$$My = b + Nx^{(k)}$$

lösen. Das soll natürlich einfach gehen. Im einfachsten Fall wählt man

$$M = \omega^{-1}B = \omega^{-1}\text{diag}\,(a_{ii})$$

wobei $\omega > 0$ ein Parameter ist und man voraussetzen muss, dass alle Diagonaleinträge von A ungleich Null sind. Es folgt:

$$N = M - A = (\omega^{-1}D - A)$$

und man erhält die Iteration

$$\begin{aligned} x^{(k+1)} &= M^{-1}\left(b - Nx^{(k)}\right) \\ &= \omega D^{-1}\left(b + \omega^{-1}Dx^{(k)} - Ax^{(k)}\right) \\ &= x^{(k)} + \omega D^{-1}\left(b - Ax^{(k)}\right) \ . \end{aligned}$$

Das ist das *gedämpfte Jacobi-Verfahren* bzw. für $\omega = 1$ das *Jacobi-Verfahren.*

Eine andere Möglichkeit besteht darin, M als Dreicksmatrix zu wählen. Zerlege

$$A = \underbrace{\underbrace{L}_{\text{untere DM}} + \underbrace{D}_{\text{Diagonalmatrix}} + \underbrace{U}_{\text{obere DM}}}_{\text{Dreiecksmatritzen mit Hauptdiagonale Nullen}}$$

Wählt man $M = L + \omega^{-1}D$, so erhält man folgende Iteration:

$$(\underline{L} + \omega^{-1}D)x^{(k+1)} = b + (L + \underline{\omega^{-1}D})x^{(k)} - (L + D + U)x^{(k)}$$

$$\Leftrightarrow$$

$$\omega^{-1}Dx^{(k+1)} = \underline{\omega^{-1}D^{(k)}} + \left[b - \underline{Lx^{(k+1)}} - \underline{(D+U)x^{(k)}}\right]$$

Schreibt man dieses Verfahren in Komponentenschreibweise, so sieht man, dass man die rechte Seite berechnen kann, obwohl sich darin die neue Iterierte $x^{(k+1)}$ befindet:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\cdot x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}\cdot x_j^{(k)}\right) \quad i = 1..n$$

Man nutzt die neu berecheten Komponenten von $x^{(k+1)}$ sofort zur Berechnung der weiteren Komponenten. dieses Verfahren heißt *SOR* (*successive overrelaxation*, einander folgende Entspannung) Im Fall $\omega = 1$ spricht man vom *Gauß-Seidel-Verfahren.*
Es gibt viele Untersuchungen zur Konvergenz dieser Iterationsverfahren. Ein bemerkenswertes Ergebnis ist wie folgt:

Satz Sei A symmetrisch und positiv definit. dann konvergiert das SOR-Verfahren für alle $x^{(0)} \in \mathbb{R}$ genau dann, wenn $\omega \in]0, 2[$. ■

Bemerkungen 1

Der Satz gibt keine Aussage über die Konvergenzgeschwindigkeit und über die Existenz (Wahrheit) eines optimalen Parameters ω. Es gibt Fälle, in denen man zeigen kann, dass ein optimaler Parameter ω exisiert. Die Berechnung dieses Parameters erfordert jedoch im Allgemeinen Informationen über die Systemmatrix A, die wiederum nur mit großem Aufwand zu beschaffen sind.
Es zeigt sich, dass die Lösung großer Gleichungssysteme mit den vorgestellten Verfahren zwar oft im Prinzip geht, jedoch im Allgemeinen sehr ineffizient ist (sehr viele Iteration → sehr lange Rechenzeiten). Man hat inzwischen wesentlich effizientere Iterationsverfahren zur Lösung linearer Systeme konstruiert → Spezialisierung, Literatur. Die hier vorgestellten einfachewn Iterationsverfahren sind jedoch oft wichtige Teilkomponenten der komplizierten Iterationsverfahren. ■

4.5 lineare Angleichsprobleme

In der Praxis stimmt die Zahl m der Bedingungen in einem Problem nicht immer mit der der Freiheitsgrade (unbekannte Größen) n überein. Oft will man etqwa den Verlauf einer abhängigen Größe $y = f(t)$ durch linearkombination spezieller Funktionen beschreiben, also

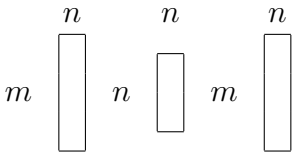
$$f(t) = \sum_{j=1}^{n} f_j(t) x_j \ ,$$

wobei die Funktionen $f_j(t)$ vorgegeben sind. Nun führt man zu gewissen Zeitpunkten t_i MEssungen aus und erhält die Messwerte b_i , $\quad i = 1 \dots m$. Aus dem Ansatz folgt:

$$b_i = \sum_{j=1}^{n} f_j(t_i) x_j \ , \quad i = 1 \dots m$$

In der Praxis ist es wegen der unvermeidlichen Mess- und Modellfehler sinnvoll, dass die Anzahl der Messungen m größer als die Anzahl der zu bestimmenden Parameter ist ($m > n$). Man erhält also ein lineares Gleichungssystem der Gestalt

$$Ax = b \ , \quad A \in \mathbb{R}^{m \times n} \ , \quad x \in \mathbb{R}^n \ , \quad b \in \mathbb{R}^m$$



(mehr Gleichungen als Unbekannte). Dieses ist in der Regel nicht lösbar.
Um dennoch „eine Art von Lösung“ $\widehat{x}$ zu erhalten, minimiert man in der Praxis die Euklidische Norm des Residuums (oder Defekts):

$$\|b - A\widehat{x}\|_2^2 = \min_{x \in \mathbb{R}^n} \|b - Ax\|_2^2 \tag{6}$$

Diese Vorgehensweise heißt *Methode der kleinesten Quadrate (least squares method).*
Eine wichtige Frag ist, ob $\widehat{x}$ eindeutig bestimmt ist. Allein durch Dedingung (6) ist dies nicht der Fall, z.B. wenn A nicht spaltenregulär ist, d.h.

$$\text{Rg } (A) < n \ .$$

Definition Ein Vektor $\widehat{x} \in \mathbb{R}^n$ heißt *kleinste-Quadrat-Lösung* von $Ax = b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, wenn er (6) erfüllt. Unter diesen Vektoren wird mit X^+ derjeinige mit kleinster euklidischer Norm bezeichnet

$$\|x^+\|$$

Bezeichne:

$$Ker(A) = \{x \in \mathbb{R}^n \ : \ Ax = 0\}$$

den Kern oder Nullraum von A und

$$Im(A) = \{y \in \mathbb{R}^m \ : \ \exists x \in \mathbb{R}^n \ : \ y = Ax\}$$

den Bildram von A. Aus der linearen Algebra ist bekannt, dass

$$Im(A)^\perp = Ker(A^T)$$

wobei $^\perp$ das orthogonale Komplement bezeichnet. Sei

$$P \ : \ \mathbb{R}^m \longrightarrow Im(A)$$

der Orthogonal-Projektor auf $Im(A)$

$$PICTURE$$

Er hat die Eigenschaften

$$P^2 = P \qquad P^T = P$$

Nun können die Lösungen $\widehat{x}$ und x^+ genauer characterisiert werden.

Satz

1. Es Gibt Minimallösungen von $\widehat{x}$ von (6). Die Bedingung (6) ist äquivalent zu

$$A\widehat{x} = Pb \tag{7}$$

   sowie zu

$$A^T A\widehat{x} = A^T \quad \Leftrightarrow A^T(A\widehat{x} - b) = 0 \tag{8}$$

2. Die allgemeine Lösung von (6) hat die Gestalt

$$\widehat{x} + Ker(A)$$

   Die Lösung x^+ mit kleinster Norm ist eindeutig. Es gilt

$$x^+ \in Ker(A)^\perp$$

■

Der Satz besagt, dass die Lösung eindeutig ist, wenn man den Definitionberech von A auf $Ker(A)^\perp$ einschränkt:

$$\tilde{A} \ : \ A\,|_{Ker(A)^\perp} \longrightarrow Im(A)$$

Die Abbildung $\widetilde{A}$ ist sogar bijektiv, d.h. es existiert eine Umkehrabbildung

$$\begin{aligned} A^+ : \quad & \mathbb{R}^n \longrightarrow \mathbb{R}^n \\ & A^+ b := \widetilde{A}^{-1} P b = x^+ \ . \end{aligned}$$

Die Abbildung A^+ wird *verallgemeinerte Inverse* oder *Pseudo-Inverse* genannt. Die Pseudo-Inverse A^+ existiert für jede Matrix A. Sie ist eindeutig bestimmt und stimmt bei regulären Matrizen mit A^{-1} überein. Zur Berechnung der Kleinste-Quadrate-Lösung wird sie nicht berechnet, genausowenig wie A^{-1} für reguläre lineare Systeme.
Die Gleichung (8) wird *Normalgleichung* genannt. Ist Rg $(A) = n$ so ist Rg $(A^T A) = n$ und (8) ist ein reguläres lineares Gleichungssystem mit symmetrischer Systemmatrix. Das kann im Prinzip mit den benannten Verfahren gelöst werden. Allerdings kann diese Vorghehensweise zu unnötig großen Fehlern führen, da die Konditionszahl von $A^T A$ sehr groß sein kann. Es gilt für $m = n \ , \quad A$ regulär

$$K_2(A^T A) = (K_2(A))^2 \ .$$

D.h., wenn $K_2(A)$ groß ist, dann ist $K_2(A^T A)$ sehr groß. Deshalb benötigt man Verfahren, die die ursprüngliche Kondition nicht erhöhen. Betrachtet man insbesondere die Spektralkonditionszahl, dann sollte man Verfahren mit orthogonalen (unitären) Umformungen verwenden, da diese die ursprüngliche Konditionszahl nicht verändern (Serie 3, Aufgabe 1).

$$\|QA\|_2 = \|A\|_2$$

Anstatt wie bei der LU-Zerlegung die Matrix A in ein Produkt aus zwei Dreiecksmatrizen zu zerlegen, verwendet man nun eine Zerlegung in ein Produkt aus einer orthogonalen Matrix Q und einer oberen Dreiecksmatrix R:

$$A = QR \ , \quad Q \in \mathbb{R}^{m \times m} \ , \quad Q^T Q = I$$

$$A = QR \qquad R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ & \ddots & & \vdots \\ & & \ddots & \\ 0 & & & r_{mm} \\ 0 & \dots & \dots & 0 \end{pmatrix} \quad \text{für } m > n$$

Das ist die songenannte *QR-Zerlegung*. Zunächst soll geklärt werden, wie man eine QR-Zerlegung von A berechnen kann. Die Spalten von Q sind orthonormale Vektoren, die man im Prinzip aus den Spalten von A mittels Gram-Schmidt-Orthogonalisierungsverfahren berechnen kann. Dieses Verfahren in seiner ursprünglichen Form ist jedoch numerisch instabil (Auslöschung). Man kann jedoch ein modifiziertes Gram-Schmidt-Verfahren konstruieren, dass numerisch gutartig ist. In der Praxis verwendet man oft einen anderen Zugang, bei dem die QR-Zerlegung mittels elementarer Spiegelungen konstruiert wird (*Householder-Transformationen*, 1950).

Lemma: Sei $u \in \mathbb{R}^m$, $\|u\|_2 = 1$. Dann ist die Matrix

$$H = I - Z_{nn}^T$$

symmetrisch ($H^T = H$), orthogonal ($H^T H = I$) und es gilt $H^2 = I$. ■

Beweis: Übungsaufgabe.

Geometrische Bedeutung

$$\begin{aligned} y = Hx =& x - 2u \underbrace{(u^T x)}_{\in \mathbb{R}} \\ =& x - \underbrace{2(u^T x)u}_{\text{in } u\text{-Richtung wird } 2u^T x \text{ abgezogen}} \end{aligned}$$

Hier fehlt noch eine Zeichnung...

Householder-Transformation

Im ersten Schritt ist ein Spiegelvektor u so zu bestimmen, dass die erste Spalte von A auf den ersten Einheitsvektor e_1 gespiegelt wird, damit die erste Spalte von R die Form

$$\begin{pmatrix} r_{11} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ besitzt.}$$

$$Hx = x - 2(u^T x)u \stackrel{!}{=} \beta e_1$$

Die Eigenschaften von H resultieren in folgender Nebenbedingung:

$$H^T H = I$$

$$\|\beta e_1\|_2 = |\beta| \underbrace{\|e_1\|_2}_{=1} \stackrel{\text{H orthog.}}{=} \|Hx\|_2 = \|x\|_2$$

also

$$|\beta| = \|x\|_2 \Rightarrow \beta = \pm \|x\|_2 \ .$$

Aus dem Ansatz folgt

$$2(u^T x)u = x - \beta e_1 \Rightarrow u \text{ ist Vielfaches von } x - \beta e_1$$

und da $\|u\|_2 = 1$ ist, ist somit

$$u = \frac{x - \beta e_1}{\|x - \beta e_1\|_2} \ .$$

Bei der Differenz $x-\beta e_1$ wir nur die erste Komponente von x verändert. Um dabei Auslöschung zu vermeiden, wählt man das Vorzeichen von β so, dass eine Addition der Beträge stattfindet. Sei $\sigma \in \{-1, 1\}$ das Vorzeichen von x_1, dann gilt

$$\begin{aligned} x_1 - \beta =& \sigma|x_1| \underbrace{\mp \|x\|_2}_{=p} \\ \stackrel{!}{=}& \sigma(|x_1| + \|x\|_2) \end{aligned}$$

$$\Rightarrow \beta = -\sigma\|x\|_2 \ .$$

Für den Nenner folgt

$$\begin{aligned} \|x - \beta e_1\|_2^2 &= \left[\underbrace{(|x_1| + \|x\|_2)^2}_{=1} + x_2^2 + \ldots + x_m^2 \right] \\ &= \left[\|x\|_2^2 + 2|x_1|\|x\|_2 + \underbrace{x_1^2 + \ldots + x_m^2}_{\|x\|_2^2} \right] \\ &= 2\|x\|_2(|x_1| + \|x\|_2) \end{aligned}$$

Lemma Sei $x \in \mathbb{R}^m$, $x \neq 0$, $x_1 = G|x_1|$. Dann wird x durch die Matrix

$$\begin{aligned} H =& \mathtt{I} - \frac{vv^T}{2\|x\|_2 \left(|x_1| + \|x\|_2\right)} \\ v =& x + \Sigma\|x\|_2 e_1 \end{aligned}$$

und $-\Sigma\|x\|_2 e_1$ abgebildet. ■

Die elimination in den folgenden Spalten kann analog ausgeführt werden. Dabei dürfen die bereits behandelten Spalten nicht wieder verändert werden. Sei

$$A^{(k)} = \left(\begin{array}{ccc|ccc} r_{11} & \ldots & r_{1\,k-1} & & & \\ & \ddots & \vdots & & * & \\ 0 & & r_{k-1\,k-1} & & & \\ \hline & & & a_{kk}^{(k)} & \ldots & a_{kn}^{(k)} \\ & 0 & & \vdots & \ddots & \vdots \\ & & & a_{mk}^{(k)} & \ldots & a_{mn}^{(k)} \end{array}\right) = \left(\begin{array}{c|c} R^k & * \\ \hline 0 & B^{(k)} \end{array}\right)$$

mit $R^{(L)} \in \mathbb{R}^{(l-1,k-1)}$, $B^{(L)} \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$ mit der Wahl

$$H^{(k)} = \left(\begin{array}{c|c} \mathtt{I}_{k-1} & 0 \\ \hline 0 & \tilde{H}^{(L)} \end{array}\right)$$

$$U^{(k)} = \left(\underbrace{0, \ldots, 0}_{k-1}, *, \ldots, *\right)^T$$

für die k-te Transformation bleiben beim Produkt

$$H^{(k)}A^{(k)} = A^{(k+1)}$$

die ersten $(k-1)$ Zeilen und Spalten unverändert. Wir wenden nun das letzte Lemma auf den Teilvektor

$$\widetilde{H}^k \begin{pmatrix} a_{kk}^{(k)} \\ \vdots \\ a_{mk}^{(k)} \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} \beta \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

an. In

$$l := \min\{m-1, n\}$$

Schritten erhält man auf diese Art und Weise die Zerlegung

$$A^{(l+1)} = R^{(l+1)} =: R = H^{(l)} \dots H^{(2)} H^{(1)} A \Leftrightarrow A = \underbrace{H^{(1)} H^{(2)} \dots H^{(l)}}_{Q} R =: QR$$

mit der orthogonalen Matrix Q (Produkt der orth. Matrizen $H^{(j)}$ ist eine orthogonale Matrix) und der oberen Dreiecksmatrix R. Man hat

$$R = \begin{pmatrix} r_{11} & \dots & r_{1m} & \dots & r_{1n} \\ & \ddots & \vdots & & \vdots \\ 0 & & r_{nm} & \dots & r_{mn} \end{pmatrix} \quad \text{für } l \le m,\ l = m-1$$

$$R = \begin{pmatrix} r_{11} & \dots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ 0 & \dots & 0 \end{pmatrix} \quad \text{für } m > n,\ l = n\ .$$

Praktische Durchführung Die Matrix Q wird nicht explizitr berechnet. Stattdessen merkt man sich die Vektoren $v^{(k)}$ und speichert diese im unteren Dreieck von R_k beis einschließlich Hauptdiagonale. Die Hauptdiagonale von R_k speichert man in einem Extravektor.
Der aufwendigste Schritt bei der Berechnung der QR-Zerlegung ist

$$\begin{aligned} A^{(k+1)} =& H^{(k)} A^{(k)} \\ =& A^{(k)} - \frac{v^{(k)} v^{(k)^T}}{\|v^{(k)}\|_2^2} A^{(k)} \\ =& A^{(k)} - \frac{v^{(k)}}{\|v^{(k)}\|_2^2} \left(v^{(k)^T} A^{(k)} \right)\ . \end{aligned}$$

Alle anderen Kosten sind für große $m,\ n$ vernachlässigbar.
Man hat

$$\begin{aligned} \left(z^{(k)}\right)^T :=& v^{(k)^T} A^{(k)} \\ =& (\underbrace{0, \dots, 0}_{k-1}, \underbrace{*, \dots, *}_{m-k+1}) \left(\begin{array}{c|c} * & * \\ \hline 0 & \underbrace{B^{(k)}}_{n-k+1} \end{array}\right) = (\underbrace{0, \dots, 0}_{k-1}, \bullet, \underbrace{*, \dots, *}_{n-k}) \end{aligned}$$

„•“ ist bekannt ($= \beta$). Somit benätigt man $(n-k)$ Skalarprodukte der Länge $(m-k+1)$, d.i. rund $2(n-k)(m-k+1)$ Flops.
Zur Berechnung von $A^{(k+1)}$ sind jetzt noch $(n-k)$ Sklaparprudkte der Länge $m-k+q$ und die Addition des jeweiligen Ergebnisses zur entsprechenden Komplonente von $B^{(k)}$ nötig. $\Rightarrow$ $2(n-k)(m-k+1)$ Flops (k-te Spalte bekannt $\begin{pmatrix} \beta \\ 0 \\ \vdots \\ 0 \end{pmatrix}$). Man erhält im Fall $m > n$:

$$4\sum_{k=1}^{m}(m-k+1)(n-k) = \frac{\not\nabla^{\not\Vdash}}{\not\propto} + 2(m-n)n^2 + \dots$$

Eine analoge Aussage erhält man für $m \leq n$. Der Rechenaufwand f+r die QR-Zerlegung ist

$$4\min\left\{m^3, n^3\right\} + 2\,|m-n|\min\left\{n^2, m^2\right\} + \dots \text{ Flops}$$

Im Fall $m = n$ ist die QR-Zerlegung somit ungefähr doppelt so teuer wie die LU-Zerlegung. Dafür wird die Spektralkonidtionszahl von A bei der QR-Zerlegung nicht vergrößert.

Berechnung der Kleinste-Quadrate-Läsung x^+

Habe $A \in \mathbb{R}^{m\times n}$ Vollrang, d.i. $Rg(A) = n$ für $m > n$, und sei

$$A = QR, \quad R = \begin{pmatrix} R_0 \\ 0 \end{pmatrix}, \quad R_0 \in \mathbb{R}^{n\times n}$$

Dann ist R_0 regulär. Mit Hilfe von Q^T wird die rechte Seite zerlegt.

$$Q^T b = \begin{pmatrix} \widehat{b} \\ \dots \\ \tilde{b} \end{pmatrix} \begin{matrix} n \\ \cdots \\ n+1 \\ \vdots \\ m \end{matrix}$$

Das Produkt $Q^T b$ lässt sich aus den gespeicherten Informationen von Q berechnen.
Nun ist:

$$\begin{aligned} \|Ax - G|_2^2 &= \|QRx - b\|_2^2 \\ &= \left\|Q\left(Rx - Q^T b\right)\right\|_2^2 \\ &\overset{m \text{ variant}}{=} \left\|Rx - Q^T b\right\|_2^2 \\ &= \left\| \begin{pmatrix} R_0 x - \widehat{b} \\ \tilde{b} \end{pmatrix} \right\|_2^2 \\ &= \left\|R_0 x - \widehat{G}\right\|_2^2 + \|\tilde{b}\|_2^2. \end{aligned}$$

Der zweite Summand ist von x unabhängig. Also erhält man das Minimum von $\|Ax - b\|_2^2$ durch Minimierung des ersten Summanden, d.h. durch Lösung von

$$R_o x = \widehat{x} \quad \Rightarrow \quad x^+ = R_0^{-1}\widehat{b} \ .$$

(genau dann ist der erste Summand Null).

Bemerkung: Aus für den Fall $R_g(A) < n$ kann man x^+ mit Hilfe der QR-Zerlegung von A berechnen.

5 Nullstellenberechnung bei nichtlinearen Funktionen

Der Einfachheit wegen werden zunächst reelle Funktionen einer reellen Variablen betrachtet:

$$f : \mathbb{R} \longrightarrow \mathbb{R} \ .$$

5.1 Theorie

Man unterscheidet zwei Normalformen von nichtlinearen Gleichungen, die *Nullstellengleichung*

$$f_1(x) = \ 0 \tag{1}$$

und die *Fixpunktgleichung*

$$f_2(x) = \ x \ . \tag{2}$$

Eine Zahl α wird *Fixpunkt* von f_2 genannt, falls

$$f_2(\alpha) = \alpha \ .$$

Jede Nullstellengleichung lässt sich in eine Fixpunktgleichung überführen. Sei $g(x)$ eine stetige Funktion mit $g(x) \neq 0$, dann wird (1) mit $g(x)$ multipliziert und auf beiden Seiten x addiert:

$$F_1(x) = g(x)f_1(x) + x = x \ .$$

Eine Nullstelle von f_1 ist ein Fixpunkt von F_1. Subtrahiert man andererseits in (2) x, so erhält man die Nullstellengleichung

$$F_2 = f_2(x) - x = 0 \ .$$

D.h. ein Fixpunkt von (2) ist eine Nullstelle von F_2 und umgekehrt.
Eine Frage, die beantwortet werden muss, ist ob (1) korrekt gestellt ist. Sei $f_1 \in C^1(\mathbb{R})$. Wir betrachten die nichtlineare Gleichung

$$F(x, d) = f_1(x) - d = 0 \ , \tag{3}$$

wobei $d \in \mathbb{R}$ ein Datum ist. Das Problem ist nur dann wohl definiert, wenn f_1 in einer Umgebung von d invertierbar ist. Dann ist $\alpha = f^{-1}(d)$, wobei α die gesuchte Wurzel (Nullstelle) ist und f^{-1} die inverse Funktion sind. Mit der Differentiationsregel für inverse Funktionen erhält man

$$(f^{-1})'(d) = \frac{1}{f_1'(\alpha)} \ .$$

Im Fall $d \neq 0$ erhält man als Approximation für die relative Konditionszahl

$$\begin{aligned} K(d) \approx & \left(\|G'(d)\| \frac{\|d\|}{\|G(d)\|} \right) \left| (f_1^{-1})'(d) \right| \frac{|d|}{|\underbrace{f_1^{-1}(d)}_{\alpha}|} \\ = & \frac{|d|}{|\alpha|} \frac{1}{|f_1'(\alpha)|} \end{aligned}$$

und im Fall $d = 0$ oder $\alpha = 0$ erhält man für die absolute Konditionszahl

$$K_{\text{abs}}(d) = \frac{1}{|f_1'(\alpha)|} \ .$$

Das Lösen der nichtlinearen Gleichung (3) ist also ein gut konditioniertes Problem, falls $|f_1'(\alpha)|$ groß ist. Die Konditions wird schlechter, falls $|f_1'(\alpha)|$ kleiner wird. Im Fall mehrfacher Wurzeln, d.i. $f_1'(\alpha) = 0$, kann man, mit höheren Differenzierbarkeitsvoraussetzungen an f_1 zeigen, dass

$$\begin{aligned} K_{abs}(d) \approx & \left| \frac{m! \, \delta d}{l_1^{(m)}(\alpha)} \right|^{\frac{1}{m}} \frac{1}{|\delta d|} \\ = & \left| \frac{m!}{f_1^{(m)}(\alpha)} \right|^{\frac{1}{m}} \cdot |\delta d|^{\frac{1}{m}-1} \ , \end{aligned}$$

wobei m die Vielfachheit der Wurzel ist. Das kann selbst bei kleinen Datenstörungen δd eine große Zahl sein.
Es folgt, dass das Wurzelfunden einer nichtlinearen Gleichung gut konditioniert ist, falls die Wurzel α einfach ist und $A_1'(\alpha)$ hinreichend weit von Null entfernt ist., Ansonsten istdas Problem schlecht konditioniert.
Numerische Verfahren zur Lösung nichtlinearer Gleichungen sind im allgemeinen iterativ. beginnend mit einem Startwert $x^{(0)}$ wird von Verfahren eine Folge $\left\{x^{(k)}\right\}$ generiert. Dass Ziel ist, dass

$$\lim_{k \longrightarrow \infty} x^{(k)} = \alpha$$

Definition Die Folge $x^{(k)}$ *konvergiert gegen* α*, mit Ordnung* $p \geq 1$, falls

$$\exists \, C \ : \ \frac{\left|x^{(k+1)} - \alpha\right|}{|x^{(k)} - \alpha|^p} \leq C$$

$\forall k \geq k_0$, wobei $k_0 \in \mathbb{N}$ ein geeigneter Laufindex ist. Man sagt, dass die *Methode von p-ter Ordnung* ist. Ist $p = 1$, so muss notwendigerweise $G < 1$ sein, damit $\left\{x^{(k)}\right\}$ gegen α konvergiert. In diesem Fall wird C *Konvergenzfaktor* der Methode genant. ■

Ob ein Iterationsverfahren in einem konkreten Fall konvergiert, kann uach vom Startwert $x^{(0)}$ abhängen.

Definition Sei `I` die Menge aller zulässigen Startwerte, Gibt es eine Umgebung $U(\alpha)$ der Lösung α, so dass ein Iterationsverfahren für alle $x^{(0)} \in U(\alpha)$ konvergiert, so nennt man das Verfahren *lokal konvergent*. Konvergiert das Verfahren für alle $x^{(0)} \in$`I`, dann wird es global konvergent genant. ■

Zur Lösung der Fixpunktglechung (2) kann man das Iterationsverfahren

$$x^{(k+1)} = f_2(x^{(k)}) \quad k = 0, 1, \ldots \tag{4}$$

verwenden (Fixpunktiteration).

Beispiel

$$x = \cos x \quad f_2 = \cos x$$

$$0 = \cos x - x$$

$$f_1(x) = 0$$

Äquivalente Aufgabenstellung

$$f_2(x) = x$$

Zur Existenz des Fixpunktes und zur Konvergenz von (4) gibt es folgenden berühmten Satz:

Satz: Fixpunktsatz von Banach (1922) Die Funktion $f_2(x)$ sei im abgeschlossenen Intervall $I = [a, b] \subset \mathbb{R}$ kontraktiv, d.i.

1. I wird durch f_2 in sich abgebildet

$$f_2(I) \subseteq I$$

2. f_2 ist Lipschitz-stetig mit einer Lipschitz-Konstanten $0 \leq L < 1$:

$$|f_2(x) - f_2(y)| \leq L|x - y| \quad \forall x, y \in I$$

Dann besitzt f_2 in I genau einen Fixpunkt α und die nach der Iterationsvorschrift (4) berechnete Folge konvergiert gegen α fur jeden Startwert $x^{(0)} \in I$. Es gelten die Fehlerabschätzungen

$$\left|\alpha - x^{(k)}\right| \leq \frac{L^k}{1 - L} \left|x^{(1)} x^{(0)}\right|$$

$$\left|\alpha - x^{(k)}\right| \leq \frac{L^k}{1 - L} \left|x^{(k)} x^{(k-1)}\right|$$

■

Bemerkung 1

Die Aussagen des Banachschen Fixpunktsatzes gelten allgemein in Banach-Räumen, also auch insbesondere für Funktionen $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $n \in \mathbb{N}$.

Bemerkung 2

Ist f_2 in I stetig differenzierbar bis zum Rand, $f_2 \in C^1([s,n])$, so gilt

$$L = \max_{x \in [a,b]} |f_2'(x)|$$

Bemerkung 3

Geometrische Veranschaulichung es Banachschen Fixpunktsatzes

PICTURE

Sei $f_2(x)$ differenzierbar. Der Betrag des Anstieges von f_2 ist kleiner als 1, siehe *Bemerkung 2*. Falls f_2 die Gerade $y = x$ einmal geschnitten hat, kann es keinen zweiten Schnittpunkt geben, weil f_2 dann schneller steigen müsste als $y = x$. Das geht nicht, da $y' = (x)' = 1$. Dass Ã$\frac{1}{4}$berhaupt ein Schnittpunkt von f_2 und $y = x$ existiert, folgt daraus, dass

1. f_2 bildet I in I ab
2. f_2 ist stetig
3. I ist abgeschlossen.

Bemerkung 4

Sei $L > 0$. Dann folgt aus der Fehlerabschätzung

$$\frac{\left|\alpha - x^{(k)}\right|}{|\alpha - x^{(k-1)}|} \leq L$$

Die Konvergenzordnung ist linear und der Konvergenzfaktor ist L. Die Konvergenzgeschwingkeit ist desto besser, je kleiner L ist. ■
Numerische Verfahren die auf dem Prinzip der Fixpunktiteration beruhen, werden in späteren Abschnitten behandelt.

5.2 Das Bisektions-Verfahren

Bei diesem Verfahren handelt es sich um das einfachste und sicherste Verfahren zur Berechnung einer Nullstelle einer stetigen Funktion im Intervall $[a,b] = I$. Dieses Verfahren lässt sich jedoch nicht auf höhere Domensionen erweitern. Es beruht auf dem Zwischenwertsatz: Jede stetige Funktion $f : [a,b] \longrightarrow \mathbb{R}$ nimmt jeden Funktionswert zwischen $f(a)$ und $f(b)$ an (falls $f(a) < f(b)$, sonst umgekehrt)

PICTURE

eine folgerung ist, dass aus $f(a) \cdot f(b) < 0$ folgt, dass f mindestens eine Nullstelle in $[a, b]$ besitzt. Beim Bisektions-Verfahren versucht man ein genügend kleines Intervall zu finden, in dem eine Nullstelle von f liegt.
Man muss zunächst Werte a und b finden, so dass $f(a)$ und $f(b)$ unterschiedliche Vorzeichen besitzen.
Setze $I_0 = [a, b]$. Dann werden Teilintervalle

$$I_k = \left[a^{(k)}, b^{(k)}\right], \quad k \geq 0,\ I_k < I_{k-1},\ k \geq 1$$

gesucht, so dass $f(a^{(k)})f(b^{(k)}) < 0$. Das geschieht mit folgendem Algorithmus:

1. Setze $a^{(0)} = a$, $b^{(0)} = b$, $\underbrace{x^{(0)} = \frac{a^{(0)} + b^{(0)}}{2}}_{\text{Mittelpunkt}}$
2. für $k > 0$
   Setze $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = x^{(k)}$, falls $f(x^{(k)})f(a^{(k)}) < 0$
   Setze $a^{(k+1)} = x^{(k)}$, $b^{(k+1)} = b^{(k)}$, falls $f(x^{(k)})f(b^{(k)}) < 0$
3. anschließend setze
   $$x^{(k+1)} = \frac{a^{(k+1)} + b^{(k+1)}}{2}$$

Man bricht das Bisektionsverfahren ab, falls

1. $f(x^{(k)}) = 0 \Rightarrow$ Nullstelle gefunden
2. oder: $|x^{(k)} - \alpha| \leq |I_k| < \epsilon$, wobei ϵ eine vorgegebene Toleranz ist.
   $|I_k|$ - Länge von I_k.

Veranschaulichung

PICTURE

Konvergenzgeschwindigkeit Es sind

$$|I_0| = b - a \quad |I_k| = \frac{|I_0|}{2^k} = \frac{b-a}{2^k}$$

Sowohl die k-te Iterierte $x^{(k)}$ als auch die Lösung α liegen im Intervall I_k. Deshalb gilt für den absoluten Fehler

$$\left|e^{(k)}\right| = \left|x^{(k)} - \alpha\right| \leq |I_k| = \frac{b-a}{2^k}$$

und

$$\lim_{k \longrightarrow \infty} \left|e^{(k)}\right| = 0,$$

Das gilt für beliebige Startwerte $a^{(0)}$, $b^{(0)}$ mit $f(a^{(0)}) \cdot f(b^{(0)}) < 0$. Das Bisektionsverfahren ist global konvergent.

Sei $\epsilon > 0$ gegeben. Gesucht ist die Anzahl m von Bisektionen, damit $|x^{(m)} - \alpha| < \epsilon$. Das ist sicher erfüllt falls $|I_m| < \epsilon$.

$$\begin{aligned} & |I_m| < \epsilon \\ \Leftrightarrow \quad & \frac{b-a}{2^m} < \epsilon \\ \Leftrightarrow \quad & \frac{b-a}{\epsilon} < 2^m \\ \Leftrightarrow \quad & m > \frac{\log\left(\frac{b-a}{\epsilon}\right)}{\log 2} \end{aligned}$$

Jede Bisektion verbessert die Genauigkeit um eine Dualstelle. Daraus folgt, dass man zur Verbesserung um eine Dezimalstelle $\log_2 10 \approx 3.32$ Bisektionen benötigt. Insgesamt ist die Konvergenz der Bisektion zwar sicher, aber langsam.
Die Konvergenz braucht nicht monoton zu sein, siehe Abbildung $x^{(1)}$ und $x^{(2)}$.
Aus diesem Grunde passt die Bisektion nicht in den Rahmen der Definition pber die Konvergenzordnung, man kann also nicht von einer Methode 1. Ordnung im Sinne dieser Definition sprechen. In der Literatur wird die Konvergenzordnung manchmal trotzdem als linear bezeichnet.
Die Bisektion eigenet sich vor allem als Annäherungsverfahren an die Nullstelle, um einen Startwert für schnelle Iterationsverfahren zu liefern, die nur lokal konvergent sind.

5.3 Das Sekanten-Verfahren und Varianten

Sei α eine Nullstelle von $f(x) \in C^1(I),\ I = [a, b]$. Sei x nahe genug an α, o.B.d.A $\alpha > x$, dann erhält man durch Taylorentwicklung

$$f(x) = f(\alpha) + (x - \alpha)f'(\xi), \quad \xi in]\alpha, x[$$

Diese Beziehung kann man in die Fixpunktgleichungen

$$\alpha = x - \frac{f(x) - \overbrace{f(\alpha)}^{=0}}{f'(\xi)} = x - \frac{f(x)}{f'(\xi)} \tag{5}$$

umformen, falls $f'(\xi) \neq 0$. diese Eigenschaft ist zum Beispiel in einer Umgebung von α gesichert, falls α eine einfache Nullstelle von f ist. Die Verfahren, die in diesem Anschnitt vorgestellt werden, approximieren $f'(\xi)$ in (5) unter Benutzung von Werten von $f(x)$.
Die Grundidee der Approximation einer Ableitung besteht in der Verwendung einer *finiten Differenz*

$$f'(x) \approx \frac{f(x_1) - f(x_2)}{x_1 - x_2}$$

wobei $x_1, x_2 \in U(x)$

$$PICTURE$$

Sind x_1 oder x_2 relativ weit von x entfernt, wird die Approximation relativ ungenau sein. Liegen andererseits x_1 und x_2 sehr dicht beieinander, kann es sowohl im Zähler als auch im

Nenner der finiten Differenz Auslöschungseffekte geben, was widerum zu ungenauen Ergebnissen führen kann. Beim *Sekante-Verfahren* verwendet man

$$f'(\xi) \approx \frac{f(x^{(k)} - f(x^{(k-1)}}{x^{(k)} - x^{(k-1)}} \ , \quad x \geq 0 \ ,$$

wobei $x^{(k-1)}$, $x^{(k)}$ die letzten beiden Iterierten sind. Man benötigt also zunöchstr zwei Startwerte $x^{(-1)}$, $x^{(0)}$. Diese sind gegeben, so lautet das Sekanten-Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} f(x^{(k)}) \ , \quad k \geq 0 \tag{6}$$

Über die Konvergenz dieses Verfahrens gibt es folgende Aussage:

Satz: Sei $U(\alpha)$ eine geeignete Umgebung von α, $f \in C^2(U(\alpha))$ und $f'(\alpha) \neq 0$, $f''(\alpha) \neq 0$. Falls die Startwerte $x^{(-1)}$, $x^{(0)} \in U(\alpha)$ hinreichend nahe an α gewählt werden, konvergiert die mit (6) berechnete Folge $\{x^{(k)}\}$ gegen α und die Konvergenzordnung ist

$$p = \frac{1 + \sqrt{5}}{2} \approx 1,63 \ .$$

Geometrische Veranschaulichung:

PICTURE

Bemerkung 1: Das Sekanten-Verfahren ist lokalö konvergent, aber im allgemeinen nicht global. In der Praxis bricht man im Allgemeinen ab, falls $\underline{|x^{(k)} - x^{(k-1)}| < \epsilon}$ oder $|f(x^{(k)}| < \epsilon$ für eine vorgegebene Toleranz ϵ.

Bemerkung 2: Man benötigt keine zusätzlichen Funktionsberechnungen im k-ten Schritt, sondern nur $f(x^{(k)})$.

Bemerkung 3: $x^{(k+1)}$ liegt nicht unbedingt im Intervall $[x^{(k)}, x^{(k-1)}]$. Das geschieht, falls $f(x^{(k)})$ und $f(x^{(k-1)})$ gleiches Vorzeichen besitzen.

Bemerkung 4: Verallgemeinerungen auf konvergente Nullstellen und Funktionen in Banach-Räumen sind möglich.

Bemerkung 5: Eine weitere Verallgemeinerung besteht darin, statt einer Sekante eine Parabel durch drei Punkte zu nehmen - Verfahren von Müller mit Konvergenzordnung $1,84$. ■

Falls $f(x^{(k)})$ und $f(x^{(k-1)})$ gleiches Vorzeichen besitzen und fast gleich groß sind, kann $x^{(k+1)}$ sehr weit entfernt von α liegen. Es kann sein, dass f für $x^{(k+1)}$ gar nicht definiert ist oder sehr große Werte annimmt, die zu einem Überlauf führen. Ein Verfahren, bei dem alle Iterierten $x^{(k)}$ im Intervall $[x^{(-1)}, x^{(0)}]$ liegen, und welches ebenfalls Sekanten benutzt, ist die *Regula*

falsi. Anstatt die Sekante durch die letzten beiden Iterierten zu nehmen, verwendet man die Sekante durch $(x^{(k)}, f(x^{(k)}))$ und $(x^{(k')}, f(x^{(k')}))$, wobei k' der größte Index kleiner als k ist, für den $f(x^{(k)})f(x^{(k')}) < 0$ gilt. Für die Regula falsi benötigt man zwei Startwerte $x^{(-1)}$, $x^{(0)}$ mit $f(x^{(-1)})f(x^{(0)}) < 0$.
Die Iterationsvorschrift lautet:

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k')}}{f(x^{(k)}) - f(x^{(k')})} \cdot f(x^{(k)}) \ , \quad k \geq 0 \ . \tag{7}$$

Satz Sei $f \in C([x^{(-1)}, x^{(0)}])$. Dann gilt fur die mit (7) erzeugte Folge $x^{(k)}$, dass $x^{(k)} \in [x^{(-1)}, x^{(0)}]$, $k \geq 1$, und $\lim\limits_{k \longrightarrow \infty} x^{(k)} = \alpha$. Ist $f \in C^2([x^{(-1)}, x^{(0)}])$ α eine einfache Nullstelle $(f'(\alpha) \neq 0)$ und $f''(\alpha) \neq 0$, dann ist die Konvergenzordnung der Regula falsi gleich Eins. ■

Geometrische Veranschaulichung

PICTURE

In diesem Beispiel ist $k' = 1$ für alle k. Das ist der schlechsteste Fall, der dafür sorgt, dass im Allgemeinen die Konvergenzordnung nicht größer als Eins ist.

Bemerkung 1

Die Regula falsi ist global konvergent. Da sie jedoch im Allgemeinen langsam ist, eigenet sie sich vor allem zur Beschaffung von Startwerten für lokal konvergente und schnelle Verfahren.

Bemerkung 2

Der Aufwand von Regula falsi und Sekanten-Verfahren ist gleich, gemessen in Funktionswertberechnungen.

Bemerkung 3

Es gibt Varianten der Regula falsi, die eine bessere Konvergenzordnung besitzen, indem sie vermeiden, dass k' fixiert wird: Illinois-Algorithmus: $p = \sqrt[3]{3} \approx 1.442$, Pegasus-Verfahren: $p \approx 1.642$

Bemerkung 4

Desweiteren gibt es Verfahren, die Bisektion (Sicherheit) und Sekanten-Verfahren (Schnelligkeit in der Nähe der Nullstelle) kombinieren, z.B. Verfahren von Dekker und Brent (1937).
■

5.4 Das Newton-Verfahren

Newton 1669, veröffentlicht von Wallis 1685, in der heutigen Form von Raphson 1697; deshalb auch manchmal als Newton-Raphson-Verfahren genannt.
Sei $f(x)$ stetig differenzierbar. Das Newton-Verfahren verwendet zur Approximation von

$f'(\xi)$ in (5) den Wert der Ableitung von f an der Stelle $x^{(k)}$. Sei also $x^{(0)}$ gegeben, dann lautet das Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}, \quad k \geq 0 \tag{8}$$

Satz Sei $f \in C^3([a,b])$, $\alpha \in [a,b]$, $f'(\alpha) \neq 0$ (einfache Nullstelle). Dann existiert eine Umgebung $U(\alpha)$, so dass für jeden Startwert $x^{(0)} \in U(\alpha)$ die mit (8) berechnete Folge $\{x^{(k)}\}$ gegem α konvergiert. Die Konvergenzordnung ist mindestens 2. ■

Geometrische Veranschaulichung

$$PICTURE$$

In $]x^{(k)}, f(x^{(k)})[$ wird die Tangente angelegt. Der Schnittpunkt der Tangente mit der x-Achse ist $x^{(k+1)}$.

Bemerkung 1

Das Newton-Verfahren ist lokal konvergent, im Allgemeinen jedoch nicht global.
Man bricht die Iteration ab, falls

$$\left|x^{(k)} - x^{(k-1)}\right| < \epsilon$$

oder

$$\left|f\left(x^{(k)}\right)\right| < \epsilon$$

für eine vorgegebene Toleranz ϵ.

Bemerkung 2

Bei einer l-fachen Nullstelle, $l \geq 2$, konvergiert das Newton-Verfahren (8) linear mit dem Konvergenzfaktor $C = \frac{-l1}{l}$. Ist die Vielfachheit der Nullstelle bekannt, so konvergiert das modifizierte Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - l \cdot \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} \quad k \geq 0$$

quadratisch.

Bemerkung 3

Das Newton-Verfahren benötigt zwei Funktionswertaufrufe pro Iteration $f\left(x^{(k)}\right)$, $f'\left(x^{(k)}\right)$. Diesbezüglich ist es doppelt so teuer wie das Sekanten-Verfahren. Zwei Schritte des Sekanten-Verfahrens haben die Konvergenzordnung

$$\left(\frac{1+\sqrt{5}}{2}\right)^2 \approx 2.62 > 2.$$

Damit konvergiert, gemessen am Aufwand, das Sekanten-Verfahren schneller.

Newton Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

Bemerkung 4

Im Abschnitt 5.1 wurde gezeigt, dass sich die Nullstellengleichung $f(x) = 0$ äquivalent in die Fixpunktgleichung

$$x = x + g(x)f(x), \quad g(x) \neq 0$$

umformen lässt. Desweiteren besagt der Banach'sche Fixpunktsatz, dass die Iteration

$$x^{(k+1)} = x^{(k)} + g(x^{(k)})f(x^{(k)})$$

desto schneller konvergiert, je kleiner die Lippschitz-Konstante der rechten Seite der Fixpunktgleichung ist.
Seien nun $g(x),\ f(x)$stetig differenzierbar in $U(\alpha)$. Dann folgt:

$$\begin{aligned} L &= \max_{x \in \overline{U(\alpha)}} |(x + g(x)f(x))'| \\ &= \max_{x \in \overline{U(\alpha)}} |1 + g(x)'f(x) + g'(x)f(x)| \end{aligned}$$

L ist in $\overline{U(\alpha)}$ wahrscheinlich klein, falls für $x = \alpha$ der Asudruck im Betrag Null ist. Da $f(\alpha) = 0$ folgt

$$|1 + g(\alpha)f'(\alpha)| = 0 \quad \Rightarrow \quad g(\alpha) = -\frac{1}{f'(x)}\ ,$$

woraus das Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

folgt $\rightarrow$ Newton-Verfahren.
Man erhält also das Newton-Verfahren, indem man die Funktion $g(x)$ in der Fixpunktgleichung so wählt, dass die Lipschitz-Konstante für $\overline{U(\alpha)}$ möglichst klein wird.
Ist $f \in C^2(\overline{U(\alpha)})$, dann erhält man folgende hinreichende Bedingung für die Konvergenz des Newton-Verfahrens.

$$\begin{aligned} 1 > C &= \max_{x \in \overline{U(\alpha)}} \left| \left(x - \frac{f(x)}{f'(x)} \right)' \right| \\ &= \max_{x \in \overline{U(\alpha)}} \left| 1 - \frac{(f')^2 - ff''}{(f')^2} \right| \\ &= \max_{x \in \overline{U(\alpha)}} \left| \frac{f(x)f''(x)}{(f'(x))^2} \right| \ . \end{aligned}$$

Bemerkung 6

Die Übertragung des Newton-Verfahrens auf komplexe Nullstellen und allgemein auf Banach-Räume ist möglich. Der Fall $f:\ \mathbb{R}^n \longrightarrow \mathbb{R}^n$ wird im Abschnitt 5.6 behandelt.

5.5 Nullstellen von Polynomen

Dieser Abschnitt stellt ein Verfahren vor, mit dem man alle Nullstellen eines Polynoms

$$p_n(x) = \sum_{i=0}^{n} a_i \cdot x^i a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \tag{9}$$

$a_i \in \mathbb{R}$, berechnen kann. $p_n(x)$ hat genau §n§ Nullstellen in $\mathbb{C}$, wobei mehrfache Nullstellen ihrer Vielfachheit gemäß gezählt werden. Aus der Algebra ist bekannt:

1. $n \in \{1, 2\}$ - es gibt einfache Berechnungsvorschriften für die Nullstellen
2. $n \in \{3, 4\}$ - es gibt relativ komplizierte Berechnungsvorschriften für die Nullstellen
3. $n \geq 5$ - es gibt im Allgemeinen keine expliziten Berechnungsvorschriften für die Nullstellen (Galois).

Das heißt, für Polynome mit $n \geq 3$ ist die Nutzung numerischer Verfahren zu empfeehlen für $n \geq 5$ ist sie im Allgemeinen notwendig.
Bevor wir zur Nullstellenuntersuchung kommen, wird zunächst ein Schema vorgestellt, mit dessen Hilfe man Funktionswerte eines Polynoms schnell berechnen kann. Nutzt man den numerischen Weg, jeden Summanden von (9) einzeln zu berechnen, so hat man fur den i-ten Summanden i Multiplikationen und zum Schluss n Additionen. Also ist die Anzahl der Flops:

$$\sum_{i=0}^{n} i + n = \frac{n}{2}(n+1) + n = \frac{n^2}{2} + \frac{3}{2}n$$

Günstiger ist es, wenn man sich die Potenzenvon x zwischenspeichert.

```
p = a_0 + a_1 x
t = x
for  i = 2 n
    t = t x
    p = p + a_i t
end
```

Der Weg benötigt n Additionen und $(2n-1)$ Multiplikationen, also $\mathbf{3n-1}$ Flops.
Die Darstellung (9) ist nicht die einzig mögliche für ein Polynom. Äquivalent dazu ist

$$p_n(x) = a_0 + x(a_1 + x(a_2 + x(\ldots + x(a_{n-1} + a_n x)\ldots)))) \tag{10}$$

Nutzt man (10), so kann man $p_n(x)$ mit n Additionen und n Multiplikationen, also $\mathbf{2n}$ Flops berechnen. Man nennt (10) auch *eingebettete Multiplikation* und (10) ist die Basis der sogenannten *Horler-Schemas* (synthetische Division) zur Berechnung von ${}_n(z)$:

```
b(n) = a(n)
for  i = n -1 :  -1  :  0
    b(i) = a(i)  +  b(i+1) z
end
```

$$p_3(n) = a_0 + z\,(a_1 + z(a_2 + \underbrace{a_3}_{b_3}\, z))$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{b_2}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{b_1}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{b_0}$$

Dieses Verfahren, veröffentlicht von Horner (1818), findet man schon bei Newton über 100 Jahre früher. Danach hat man das Verfahren handschriftlich durchgeführt und dabei folgendes Schema entwickelt

$$SOMETHINGSTUPID$$

Beispiel-Rechnung

$$POLYNOM$$

an der Stelle $x = 3$

$$SOMETHINGSTUPID$$

■

Man kann das Polynom $p_n(x)$ eindeutig zerlege in

$$p_n(x) = p_0 + (x - z)p_{n-1}(x)\ ,$$

wobei p_0 eine Konstante und $p_{n-1}(x)$ ein Polynom vom Grad $n - 1$ (Polynomdivision) ist. Setzt man $x = z$, dann folgt $p_0 = b_0$. Also

$$p_{n-1} = \frac{p_n(x) - b_0}{x - z}$$

Man findet durch Nachrechnen (Übungsaufgabe)

$$\begin{aligned} p_{n-1}(x) =& b_1 + b_2 x + \ldots + b_n x^{n-1} \\ =& \sum_{i=1}^{n} b_i x^{i-1} =: q_{\min}(x)\ . \end{aligned}$$

Somit gilt

$$p_n(x) = b_0 + (x - n)q_{\min}(x)\ .$$

Die restlichen $(n-1)$ Nullstellen von $p_n(x)$ sind gerade die Nullstellen von $q_{\min}(x)$. Man hat die Nullstelle z abgespalten. Diesen Vorgang nennt man *Deflation*. Auf dieser Basis kann man folgende verallgemeinerte Strategie zur Berechnung aller Nullstellen verwenden:

1. Finde Nullstelle z von $p_n(x)$ mit einem geeigneten Verfahren.
2. Berechne die Koeffizienten von $q_{\min}(x)$ mit dem Horner-Schema.
3. Setze $p_{n-1} = q_{\min})(x)\ , \quad n = n - 1$ und gehe zu 1.

Beim ersten Punkt kann man das Newton-Verfahren verwenden. Dabei benötigt man $p_n(z)$ und $p'_n(z)$ für die gewählte Iterierte z. Aus (**??**) folgt

$$p'_n(x) = q_{\min}(x) + q'_{\min}(x) - zq_{\min}(x) \ ,$$

also

$$p > n'(x) = q_{\min} \ .$$

Diesen Wert kann man gemeinsam mit $p_n(x)$ mit dem *erweiterten Horner-Schema* berechnen.

$$SOMETHINGSTUPID$$

Beispiel: (Erweitertes Horner-Schema) Berechne $p_4(z),\ p'_4(z)$ für

$$p_4(z) = 3x^4 - 5x^2 + 26x - 17, \quad z = 2$$

p_4	3	0	-5	26	-17	
$z = 2$		6	12	14	80	
p_{n-1}	3	6	7	40	64	$= p_4(2)$
$z = 2$		6	24	63		
	3	12	31	102		$= p'_4(2)$

Dieses Vorgehen kann man erweiteren, um alle Ableitungen von $p_n(\alpha)$ an der Stelle z zu bestimmen → *vollständiges Horner-Schema* → Literatur.

Bemerkung 1

Die Verbindung von Newton-Verfahren und Horner-Schema zur Berechnung einer Nullstelle von $p_n(x)$nennt man *Newton-Horner-Verfahren.* Ein Iterationsschritt benötigt $4n$ Flops ($p_n(z) : 2n$, $p_{n-1}(z) : 2(n-1)$, eine Division, eine Subtraktion).

Bemerkung 2

Beginnt man mit einer reellen Startnäherung, so erhält man beim Newton-Horner-Verfahren reelle Iterierte. Komplexe Nullstellen kann man so nicht finden. Sowohl das Newton-Verfahren als auch das Horner-Schema besitzen für komplexe Zahlen die gleiche Gestalt wie für reelle Zahlen. Verwendet man eine komplexe Startnäherung, so erhält man im Allgemeinen komplexe Iterierte. Dazu benötigt man Computerarithmetik für komplexe Zahlen (wird in `MATLAB`, `SCILAB` automatisch erledigt, falls komplexe Eignabedaten vorliegen).

Bemerkung 3

Eine forgesetzte Defraktion erhöht den Einfluss von Rundungsfehlern. Um diesen gering zu halten, sollte man zwei Dinge beachten:

1. Berechne zuerst die betragsmäßig kleinen Nullstellen, weil diese am anfälligsten gegen Rundungsfehler sind.

2. Hat man eine Approximation für für eine Nullstelle mit dem Newton-Horner-Verfahren gefunden, so nehme man diese Approximation als Startnäherung für das Newton-Verfahren angewandt auf das Originalpolynom $p_n(x)$ → *Newton-Horner-Verfahren mit Verfeinerung*. Damit erziehlt man oft eine wesentliche Verbesserung der Genauigkeit.

■

5.6 Numerische Lösung von Systemen nichtlinearer Gleichungen

Gegeben ist eine nichtlineare Abbildung

$$F \ : \ \mathbb{R}^n \longrightarrow \mathbb{R}^n, \quad n > 1$$

$$F = \begin{pmatrix} F_1 \\ \vdots \\ F_n \end{pmatrix}$$

und gesucht sind Vektoren $x \in \mathbb{R}^n$ mit

$$F(x) = 0 \tag{11}$$

In der Praxis verwendet man zur Lösung dieses Systems von nichtlinearen Gleichungen im Allgemeinen das Newton-Verfahren, Abwandlungen dieses Verfahrens oder (andere) Fixpunktiterationen.

Fixpunkiteration

Die Nullstellengleichung (11) muss in eine Fixpunktgleichung umgeformt werden.
Sei $\alpha \in \mathbb{R}^n$ eine Nullstelle von (11) und $G(x)$ eine Matrix, die in einer Umgebung von α invertierbar ist. Dann kann (11) äquivalent in die Fixpunktgleichung

$$x = x + G(x)F(x)$$

umgeformt werden. Ist ein Startvektor $x^{(0)}$ gegeben, so lautet die Fixxpunktiteration

$$x^{(k+1)} = x^{(k)} + G\left(x^{(k)}\right) F\left(x^{(k)}\right) \quad k = 0, 1, 2, \ldots$$

Beispiel

$$\begin{aligned} 2x_1 + x_2 - 1 &= 0 \\ x_1^2 + x_2^2 - 1 &= 0 \\ F : \mathbb{R}^n &\longrightarrow \mathbb{R}^n \end{aligned}$$

Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \mathbf{J}_F^{-1}\left(x^{(k)}\right) F\left(x^{(k)}\right)$$

vereinfachte Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \mathbf{J}_F^{-1}\left(x^{(k)}\right) F\left(x^{(k)}\right)$$

Unexakte Lösung der linearen Systeme Man nutzt ein iteratives Verfahren zur Lösung der linearen Systeme (SOR) und bricht dieses nach wenigen Interationen ab. Damit spart man Kosten, erhält aber nur eine Näherung an die Lösung der linearen Systeme und muss mit einer nicht mehr quadratischen Konvergenz des so modifizierten Newton-Verfahrens rechnen.

Approximation der Jacobi-Matrix durch finite Differenzen Falls die Berechnung der Jacobi-Matrix teuer oder nicht möglich ist, kann man diese spaltenweise wie folgt approximieren:

$$\left(\mathbf{J}_F\left(x^{(k)}\right)\right)_{kj} \approx \frac{F\left(x^{(k)} + h_j^{(k)} e_j\right) - F\left(x^{(k)}\right)}{h_j^{(k)}}$$

wobei e_j der j-te Einheitsvektor und $h_j^{(k)} > 0$ ist. Man kann zeigen, dass man mit dieser Approximation die quadratische Konvergenz erhält, falls die Größen $h_j^{(k)}$ geeignet gewählt werden. Datin liegt jedoch die Schwierigkeit in der Praxis *(siehe Newton-Verfahren in einer Dimension)*. In der Literatur gibt es Vorschläge zur Wahl der $h_j^{(k)}$. ■

Den Konvergenzbereich des Newton-Verfahren kann man oft durch eine geeignete Dämpfungsstrategie erhöhen. Das *gedämpfte Newton-Verfahren* hat die Gestalt

$$x^{(k+1)} = x^{(k)} - \underline{\lambda_k} \mathbf{J}_F^{-1}\left(x^{(k)}\right) F\left(x^{(k)}\right)$$

mit $\lambda_k \in]0, 1]$. Eine gängige Strategie zur Wahl des Dämpfungsparameters λ_k besteht darin, dass man ein λ_k sucht, s dass für die neue Iterierte gilt

$$\left\|F\left(x^{(k+1)}\right)\right\|_2 < \left\|F\left(x^{(k)}\right)\right\|_2 \ .$$

Man kann zeigen, dass es solch einen Wert $\lambda \in]0, 1]$ gibt. Man probiert zunächst $\lambda_k = 1$, wenn das nicht klappt, $\lambda_k = \frac{1}{2}$ usw. Nahe bei der Nullstelle wählt diese Methode automatisch $\lambda_k = 1$, so dass die asymptotisch quadratische Konvergenz des Newton-Verfahrens nicht gestört wird.

5.7 Abbruchkriterium für iterative Verfahren

Ein wesentliches Problem in der Praxis sind Kriterien, mit denen man die Iteration abbricht. Eine erste Möglichkeit ist die *Kontrolle des Residuums*.
Sei $\epsilon > 0$ vorgegeben, dann breche Iteration ab, falls $|f\left(x^{(k)}\right)| < \epsilon$.
Es gibt jedoch Situationen, bei denen dieses Kriterium entweder zu stark oder zu otimistisch ist. Für ein zu starkes Kriterium ergibt sich folgende Darstellung

PICTURE

und für ein zu optimistisches Kriterium

PICTURE

Aus diesem Grunde ist im allgemeinen besser, die *Änderung der Korrektur* zu betrachten. Sei $\epsilon > 0$ gegeben, dann breche Iteration ab, falls $\left|x^{(k+1)} - x^{(k)}\right| < \epsilon$.
Man sollte immer eine maximale Anzahl von Iterationen vorgeben.

6 Interpolation

6.1 Aufgabenstellung

Gegeben seien $(n+1)$ Paare (x_i, y_i), $i = 0..n$, wobei die x_i paarweise verschieden sind. Die Interpolationsaufgabe besteht nun darin, eine Funktion $\underline{\Phi}$ zu finden, so dass $\overline{\underline{\Phi}} = y_i$, $i = 1..n$.

PICTURE

Man bezeichnet $\{x_i\}$ als die Menge der *Stützstellen* und die zugehörigen $\{y_i\}$ als Menge der *Stützwerte*. Die Funktion $\overline{\underline{\Phi}}$ interpoliert $\{y_i\}$ an den Stützstellen $\{x_i\}$.
Natürlich gibt es unendlich viele Funktionen, die die Interpolationsaufgabe erfüllen. Deshalb muss man auch die Klasse der Funktionen festlegen, in der man die Interpolierte $\overline{\underline{\Phi}}(x)$ sucht z.B.:

1. $\overline{\underline{\Phi}}(x)$ ist ein Polynom - *Polynominterpolation*

2. $$\overline{\underline{\Phi}}(x) = a_0 + a_1 e^{ix} + \ldots + a_n e^{nix}$$

   trigonometrische Interpolation

3. $\overline{\underline{\Phi}}(x)$ it stückweise ein Polynom - *Spline-Interpolation* (*spline, engl. Kurvenlinie*)

4. $\overline{\underline{\Phi}}(x)$ ist eine rationale Funktion - *rationale Interpolation*

6.2 Polynominterpolation

Wir bezeichnen mit

$$P_n := \left\{p(x) \;:\; p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0,\ a_i \in \mathbb{R},\ i = 0..n\right\}$$

den Raum aller Polynome vom Höchstgrad n. wir betrachten $(n+1)$ Paare (x_i, y_i), $i = 0..n$, $x_i \neq y_i$ für $i \neq j$. die Aufgabe der Polynominterpolation besteht nun darin, ein $p \in P_n$ zu finden, so dass gilt

$$p(x_i) = y_i \quad i = 0..n$$

Als erstes stellt sich die Frage nach der Existenz un Eindeutigkeit eines solchen Polynoms. diese wird mit dem folgenden Satz beantwortet.

Satz Zu gegebenen, paarweise verschiedenen Stützstellen $x_0..x_n \in \mathbb{R}$ und zugehörigen Werten $y_0..y_n \in \mathbb{R}$ existiert genau ein Polynom $p \in P_n$ mit

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

■

Man kann zeigen, dass dieses Polynom sich wie folgt darstellen lässt:

$$p_n(x) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x-y_i)\omega'_{n+1}(x_i)} \tag{1}$$

wobei $\omega_{n+1}(x)$ das sogenannte *Knotenpolynom vom Grad* $n+1$ ist

$$p_n(x) = \prod_{j=0}^{n}(x - x_j) \tag{2}$$

Damit ist

$$\sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - y_i)\omega'_{n+1}(x_i)} = \prod_{j=0, j\neq i}^{n} (x - x_j) \tag{3}$$

woraus man für $x = x_i$ erhält

$$\prod_{j=0, j\neq i}^{n} \frac{(x_i - x_j)}{\prod_{j=0, j\neq i}^{n} (y_i - x_j)} = 1$$

also $p_n(x_i) = y_i$, $i = 0..n$. Mann hat damit auch eine zu (1) äquivalente Darstellung

$$p_n(x) = \sum_{i=0}^{n} \left(\prod_{j=0, j\neq i}^{n} \frac{x - x_j}{y_i - x_j} \right) y_i \tag{4}$$

Formel (1) oder (4) bezeichnet man als *Lagrange-Form* des Interpolationspolynoms.
Zur Untersuchung der Genauigkeit der Polynominterpolation geht man wie folgt vor: Man nimmt sich eine Funktion f gibt sich $(n+1)$ Paare von Stützstellen und Stützwerten (x_i, y_i) vor, berechnet das Interpolationspolynom $p_{n,f}(x)$ durch diese Punkte und vergleicht dieses mit der Funktion $f(x)$.

PICTURE

Man erhält folgende Fehlerabschätzung

Satz Seien $x_0..x_n$ paarweise verschiedene Stützstellen und sei x ein Element des Definitionsbereichs von f. Sei weiter $f \in C^{n+1}(I_x)$, wobei I_x das kleinste Intervall ist, mit $x_i \in I_x$, $i = 0..n$. Dann ist der Interpolationsfehler im Punkt x durch

$$\begin{aligned} E_n(x) =& f(x) - p_{n,f}(x) \\ =& \frac{f(\xi)}{(n+1)!}\omega_{n+1}(x) \end{aligned}$$

gegeben, wobei $\xi \in I_x$ ist und $\omega_{n+1}(x)$ in (2) definiert ist. ■

PICTURE

$$f(x) - p_{n,f}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

Die Aussage des Satzes impliziert nicht, dass für $n \to \infty$ $p_{n,f}(x) \longrightarrow f(x)$ $\forall x \in I_x$. In der Tat kann man Funktionen und zugehörige Mengen ovn Stützstellen finden, z.B. mit gleichem Abstand, so dass es Teilintervalle von I_x gibt, in denen $p_{n,f}(x) \not\longrightarrow f(x)$ für alle Argumente x aus diesen Teilintervallen.
Zur Untersuchung der Stabilität der Polynominterpolation betrachtet man neben den Paaren $(x_i, f(x_i))$ auh Paare mit gestörten Werten $(x_i, \tilde{f}(x_i))$. Dann gilt:

$$\begin{aligned}\left\|p_{n,f} - p_{n,\tilde{f}}\right\|_\infty &= \max_{x\in I_x}\left|\sum_{i=0}^{n}\left(\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}\right)(f(x_i-\tilde{f}(x_i)))\right| \\ &\overset{\text{Dreiecksungleichung}}{\leq} \max_{i=1..n}\left|f(x_i)-\tilde{f}(x_i)\right| \underbrace{\left\|\sum_{i=0}^{n}\left(\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}\right)\right\|}_{\mathbf{\Lambda_n(x)}}\end{aligned}$$

Man nennt $\Lambda_n(x)$ *Lebesgue-Konstante* und diese Konstante spielt die Rolle einer Konditionszahl fpr die Polynominterpolation. Es folgt, das kleine Önderungen in den Daten nur zu kleinen Önderungen im Ergebnis führen, falls $\Lambda_n(x)$ für alle x klein ist.
Man kann jedoch zeigen, dass $\Lambda_n(x) \longrightarrow \infty$ für $n \longrightarrow \infty$ zum Beispiel bei äquidistanten Stützstellen hat man

$$\Lambda_n(x) \approx \frac{2^{n+1}}{e\ n\ ln(n)}.$$

Daraus folgt, dass die Polynominterpolation für große n instabil ist.
$I_x = [0, 1]$
Zerlegung in (n) äquidistante Intervalle
$f(x_i) \longrightarrow$ Zufallswert
Im Prinzip haben wir mit den Darstellungen (1) und (4) die Aufgabe der Polynominterpolation gelöst. Diese Darstellungen sind aber aus numerischer Sicht unvorteilhaft:

- Es werden unnötig viele Flops zur Auswertung von $p(x)$ an einer bestimmten Stelle x benötigt.
- Es ist in diesen Darstellungen unmöglich, auf einfache Art und Weise zusätzliche Stützstellen zu verarbeiten.

Es gibt eine Modifikation der Lagrange-Interpolation, die diese Nachteile nicht besitzt und ein gutes numerisches Verfahren ist. Diese ist jedoch derzeit unpopulär.

6.3 Die Newton-Interpolation

In diesem Abschnitt wird eine Herangehensweise zur Berechnung des Interpolationspolynoms eingeführt, die die beiden Nachteile der Standard-Lagrange-Interpolation nicht besitzt - die *Newton-Interpolation*.
Seien $n+1$ Paare von Stützstellen und Stützwerten $(x_i, f(x_i))$, $i = 0..n$, gegeben und $p_{n,f}(x)$ sei das zugehörige Interpolationspolynom. Wir wollen $p_{n,f}(x)$ als eine Summe des Interpolationspolynoms $p_{n-1,f}(x)$ von $(x_i, f(x_i))$, $i = 0..n-1$ und einem Polynom n-ten Grades $q_n(x)$ darstellen:

$$p_{n,f}(x) = p_{n-1,f}(x) + q_n(x) \tag{5}$$

Das Polynom $q_n(x)$ soll nur von den Stützstellen x_i abhängen und nur einen unbekannten Koeffizienten besitzen. Falls es eine solche Darstellung (5) gibt, ist die Hinzunahme zusätzlicher Stützstellen kein Problem.
Aus (5) folgt

$$\begin{aligned} q_n(x_i) =& p_{n,f}(x_i) - p_{n-1,f}(x_i) \\ =& f(x_i) - f(x_i) \\ =& 0 \forall i = 0..n-1 \end{aligned}$$

Damit sind n Nullstellen von $q_n(x)$ bekannt. Mehr kann es nicht geben. Somit besitzt $q_n(x)$ die Darstellung

$$q_n(x) = a_n \underbrace{(x-x_0)(x-x_1)\dots(x-x_{n-1})}_{\omega_n(x)} = a_n \omega_n(x)$$

Damit hat $q_n(x)$ den einzigen unbekannten Koeffizienten a_n. Zur Bestimmung von a_n verwendet man die Stützstelle $(x_n, f(x_n))$ und erhält

$$q_n(x_n) = a_n \omega_n(x) = \underbrace{p_{n,f}(x)}_{f(x_n)} - p_{n-1,f}(x_n)$$

$$\Rightarrow a_n \frac{f(x_n) - p_{n-1,f}(x_n)}{\omega_n(x)} \tag{6}$$

Definition Seien $(x_i, f(x_i)) \in \mathbb{R} \times \mathbb{R}$, $i = 0..n$, mit paarweise verschiedenen x_i gegeben. Die *k-te dividierte Differenz* $f[x_i, x_{i+1}, \dots, x_{i+k}]$ wird rekursiv durch

$$\begin{aligned} f[x_i] &= f(x_i) && i = 0..n \\ f[x_i, x_{i+1}, \dots, x_{i+k}] &= \frac{f[x_{\mathbf{i+1}}, x_{i+2}, \dots, x_{\mathbf{i+k}}] - f[x_{\mathbf{i}}, x_{i+1}, \dots, x_{\mathbf{i+k-1}}]}{x_{\mathbf{i+k}} - x_{\mathbf{i}}} \end{aligned}$$

■

Um den Zusammenhang zwischen den dividierten Differenzen und der Interpolationsaufgabe herzustellen, betrachten wir zunächst den Fall $n = 1$. Dann sind

$$p_{0,f}(x) = f(x_0) \quad \text{(Konstante)}$$

und

$$\begin{aligned} a_1 &\overset{(6)}{=} \frac{f(x_n)p_{0,f}((x_1))}{\omega_1(x_1)} \\ &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ &\overset{\text{Def.}}{=} \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\ &= f[x_0, x_1] \end{aligned}$$

Damit ergibt sich aus (5)

$$p_{1,f}(x) = f[x_0] + f[x_0, x_1](x - x_0)$$

Für $n = 2$ erhält man

$$\begin{aligned} a_2 &\overset{(6)}{=} \frac{f(x_2)p_{1,f}((x_2))}{\omega_2(x_2)} \\ &\overset{\text{einsetzen}}{=} \frac{f[x_2] - f[x_0] - f[x_0, x_1](x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \\ &= \frac{1}{x_2 - x_0} \left[\frac{(f[x_2] - \overbrace{f[x_1])}^{=\mathbf{0}} + (f[x_1] - f[x_0]) - f[x_0, x_1](x_2 - x_0)}{x_2 - x_1} \right] \\ &\overset{\text{Def.}}{=} \frac{1}{x_2 - x_0} \left[\frac{f[x_1, x_2](x_2 - x_1) + f[x_0, x_1](x_1 - x_0) f[x_0, x_1](x_2 - x_0)}{x_2 - x_1} \right] \\ &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ &\overset{\text{Def.}}{=} f[x_0, x_1, x_2] \end{aligned}$$

also

$$p_{2,f}(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Durch Induktion erhält man für a_n nach (6)

$$a_n = f[x_0, x_1, \ldots, x_n]$$

und

$$p_{n,f}(x) = \sum_{k=0}^{n} \omega_k(x) f[x_0..x_k] \tag{7}$$

Diese Darstellung wird *Newtonsche Interpolationsformel* genannt. Die Eindeutigkeit der Polynominterpolation sichert, dass ma ndas gleiche Polynom erhält, wie mit der Lagrangeschen

Interpolationsformel.
Zur Berechnung der dividierten Differenzen nutzt man ein Dreiecksschema

$$\begin{array}{lllll} & k=0 & & k=1 & \\ x_0 & f[x_0]=f(x_0) & & & \\ & & \searrow & & \\ x_1 & f[x_1]=f(x_1) & \longrightarrow & f[x_0,x_1] & \\ & & \searrow & & \searrow \\ x_2 & f[x_2]=f(x_2) & \longrightarrow & f[x_0,x_1] & \longrightarrow \quad f[x_0,x_1,x_2] \\ \vdots & & & & \\ x_n & f[x_n]=f(x_n) & \longrightarrow & f[x_{n-1},x_n] & \longrightarrow \quad \ldots \qquad \longrightarrow \quad f[x_0,\ldots,x_n] \end{array}$$

Beispiel Man berechne das Newton-Interpolationspolynom durch die in der Tabelle angegebenen Paare von Stützstellen und Stützwerten

x_i	$f[x_i]$	$=$ $f(x_i)$		
-1	2			
0	4	2		
2	6	1	$\frac{1-2}{2-(-1)}=-\frac{1}{3}$	
3	12	6	$\frac{6-1}{3-0}=\frac{5}{3}$	$\frac{\frac{5}{3}-\left(-\frac{1}{3}\right)}{3-(-1)}=\frac{1}{2}$

$$\Rightarrow p_{3,f}(x)=2+2(x+1)-\frac{1}{3}(x+1)x+\frac{1}{2}(x+1)x(x-2)$$

■

Zur Berechnung der Koeffizienten im Newton-Interpolationspolynom (7) benötigt man nur die Werte in der Hauptdiagonalen. Deshalb benötigt man bei ihrer Berechnung nur einen Vektor der Länge $n+1$. Man geht spaltenweise vor. Zunächst belegt man den Vektor mit $f[x_0]..f[x_n]$. Im zweiten Schritt überschreibt man $f[x_1]..f[x_n]$ mit $f[x_0,x_1]..f[x_{n-1},x_n]$ usw. Der Rechenaufwand pro dividierter Differenz, $k>0$, beträgt 3 Flops. Zur Berechnung aller Koeffizienten des Newton-Interpolationspolynoms hat man $n+(n-1)+\ldots+1=\frac{n}{2}(n+1)$ dividierte Differenzen zu berechnen, so dass der Gesamtaufwand $\frac{3}{2}n^2+\frac{3}{2}n$ Flops beträgt.

6.4 Spline-Interpolation

spline, engl. längliches, dünnes Stück Holz oder Metal, Kurvenlineal

Wir haben in den vergangenen Abschnitten gesehen, dass Polynominterpolation

- für große Stützstellen, instabil werden kann, insbesondere bei äquidistanten Stützstellen.
- für nichtglatte Funktionen der Interpolationsfehler beliebig groß werden.

Die Situation, dass man viele äquidistante Stützstellen hat oder nichtglatte Funktionen zu interpolieren treten in Anwendungen jedoch häufig auf. Für solche Situationen verwendet

man häufig die sogenannte spline-Interpolation.

Hat jemand diese Formel? (8)

Hat jemand diese Formel? (9)

Definition Seien $x_0..x_n \in [a,b]$ paarweise verschiedene Punkte mit $a = x_0 < x_1 < \ldots x_n = b$ die Funktion $Sk(x) : [a,b] \to \mathbb{R}$ wird Spline von Grad k bezüglich der Stützstelle x_j genannt, falls $Sk(x) \mid_{[x_{i,j},x_{j+1}]}$ darstellen. Man hat $n(k+1)$ unbekannte Koeffizienten $s_{i,j}$. Aus (9) folgt $S^m_{k,j-1}(x_i = S^m_{k,j}(x_j$ für $j = 1..n-1$ und $m = 0..k-1$ also $m-k$-te Ableitung von links $= m-k$-te Ableitung von rechts.
Das heisst, man hat damit nur $k \cdot (n-1)$ Bedingungen. Man benötigt also noch $n \cdot (k+1) - k \cdot (n-1) = n + k$ weitere Bedingungen. Soll der Spline eine Funktion f approximieren, so hat man weitere $n+1$ Bedingungen

$$s_k(x_j) = f(x_j) \quad j = 0..n.$$

Damit benötigt man nur noch $k-1$ Bedingungen.
Für Interpolation nutzt man oft *kubische Splines*, d.i. $k = 3$. Das ist der minimale Grad, damit $s_k(x)$ zweimal stetig differenzierbar in $[a,b]$ ist und beispielweise die Krümmung noch wohl definiert ist.
die zwei fehlenden Bedingungen setzt man, z.B.

$$s_3''(a) = s_3''(b) = 0$$

Betrachten wir $s_3(x)$ zur Interpolation von $f(x)$ in $[a,b]$. Die zweite Ableitung von $s_3(x)$ muss stetig sein. Wir verwenden die Bezeichungen

$$f_i = s_3(x_i), m_i = s_3'(x_i),\ M_i = s_3''(x_i) \quad i = 0..n.$$

Die zweite Ableitung von $s_3(x)$ ist eine stückweise lineare Funktion (Polygonzug). Es gilt also

$$s_3''(x) = M_{i-1}\frac{x_i - x}{x_i - x_{i-1}} + M_i\frac{xd - x_{i-1}}{x_i - x_{i-1}}$$
$$s_3''(x_{i-1}) = M_{i-1}\ , s_3''(x_i) = M_i \quad x \in [x_{i-1}, x_i]$$

zweimaliges Integrieren liefert:

$$s_3(x) = \frac{M_i}{6}\frac{(x - x_i^2)}{x_i - x_{i-1}} + \frac{M_i}{6}\frac{(x - x_{i-1}^3)}{x - x_{i-1}} + c_{i-1}(x - x_{i-1}) + \tilde{c}_{i-1} \tag{10}$$

Die Konstanten c_{i-1} und $\tilde{c}_{i-1}$ kann man durch Einsetzen der Endpunkte des Teilintervalls bestimmen:

$$x_{i-1} \ : \underbrace{s_3(x_{i-1})}_{f_{i-1}} = \frac{M_{i-1}}{6}(x_i - x_{i-1})^2$$
$$+0 + 0 + \tilde{c}_{i-1}$$

$$\Rightarrow \tilde{c}_{i-1} = f_{i-1} - \frac{M_{i-1}}{6}(x_i - x_{i-1}) \tag{11}$$

analog

$$c_{i-1} = \frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{6}(M_i - M_{i-1}) \tag{12}$$

Betrachten wir nun die Stetigkeit der ersten Ableitung. Wir erhalten:

$$s_3' = \frac{-M_{i-1}}{2} \cdot \frac{(x_i - x)^2}{x_i - x_{i-1}} + \frac{M_i}{2} \cdot \frac{(x - x_{i-1})^2}{x_i - x_{i-1}} + c_{i-1} \; x \in [x_{i-1}, x_i]$$

also

$$\underbrace{s_3(x_{i-0})}_{lim_{x \to x_i - 0}} = \frac{M_i}{2}(x_i - x_{i-1}) + \frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{M_i - M_{i-1}}{6}(x_i - x_{i-1})$$

$$= (\frac{M_{i-1}}{6} + \frac{M_i}{3})(x_i - x_{i-1}) + \frac{f_i - f_{i-1}}{x_i - x_{i-1}}$$

analog erhält man für den rechtseitigen Grenzwert

$$s_3'(x_{i+0}) = \left(-\frac{M_i}{3} - \frac{M_{i+1}}{6}\right) \cdot (x_{i+1} - xi) + \frac{f_i + 1}{f_i} x_i + 1 - x_i$$

setzt man diese Bedingungen gleich, so erhält man $(n-1)$ Gleichungen für $M_0..M_n$. Für die zwei fehlenden Bedingungen setzt man

$$\begin{aligned} 2M_0 + \lambda_0 M_1 &= d_0 \\ \mu_n M_{n-1} + 2M_n &= d_n \end{aligned}$$

mit vorgegebenen Werten $0 \leq \lambda_0, \mu_n \leq 1$ und d_0, d_n. Zur Erfüllung von $s_3''(a) - s_3''(b) = 0$, muss man alle diese WErte zu Null setzen. Das vollständige lineare System hat die Gestalt:

$$\begin{pmatrix} 2 & \lambda_0 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

mit

$$\begin{aligned} \mu_i &= \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} \\ \lambda_i &= \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} \\ d_i &= \frac{6}{x_{i+1} - x_{i-1}} \left(\frac{f_{i+1} - f_i}{x_{i+1} - x_i} - \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \right) \end{aligned}$$

Nachdem man dieses System gelöst hat, erhält man aus (10), (11) und (12) die Darstellung des kubischen Splines.
Bezüglich des Interpolationsfehlers gilt folgendes Resulat.

Satz Seien $f \in C^4([a,b])$, $h_{\max} = \max\limits_{i=0..n}(x_{i-1} - x_i)$, $h_{\min} = \min\limits_{i=0..n}(x_{i-1} - x_i)$ und $\beta = \frac{h_{\max}}{h_{\min}}$. Dann gilt

$$\max_{x\in[a,b]} \left| f^{(r)}(x) - s_3^{(r)}(x) \right| \leq c_r h_{\max}^{4-r} \max_{x\in[a,b]} \left| f^{(4)}(x) \right|$$

$r \in \{0,1,2,3\}$ mit

$$c_0 = \frac{5}{384} \quad c_1 = \frac{1}{24} \quad c_2 \frac{3}{8} \quad c_3 = \frac{1}{2}\left(\beta + \frac{1}{\beta}\right)$$

■

Das bedeutet, für $h_{\max} \longrightarrow 0$ nähert sich der kubische Spline (samt 1.-3. Ableitung) der vorgegebenen Funktion und ihren Ableitungen an.

7 Numerische Integration

Sei $f : [a,b] \to \mathbb{R}$ eine integrierbare Funktion. Die Berechnung von $I(f) = \int_a^b f(x)dx$ kann aber schwierig oder sogar analytisch nicht durchführbar sein. Jede explizite Formel, die eine Näherung für $I(f)$ darstellt, wird *Quadraturformel* genannt.

7.1 Interpolatorische Quadraturformeln

Eine Näherung $I_n(f)$ von $I(f)$ erhält man, indem man die Funktion $f(x)$ durch eine Approximation $f_n(x)$ ersetzt, die sich einfach integrieren lässt:

$$I_n(f) = \int_a^b f_n(x)\ dx.$$

Funktionen, die sich einfach integrieren lassen sind Polynome. Wählt man n+1 paarweise verschiedene Stützstellen aus $[a,b]$, so hat das Lagrange-Polynom von f die Gestalt

$$p_{n,f}(x) = \sum_{i=0}^{n} \underbrace{\left(\prod_{j=0,\, j\neq 1}^{n} \frac{x - x_j}{x_i - x_j} \right)}_{l_i(x)} f(x_i)$$

Man erhält

$$I_n(f) = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)\ dx.$$

Das ist eine spezielle Form der folgenden Quadraturformel

$$I_n(f) = \sum_{i=0}^{n} \alpha_i f(x_i). \tag{1}$$

Die Punkte x_i in (1) werden *Knoten* genannt und die Werte α_i-Gewichte.

$$I_n(f) = \sum_{i=0}^{n} \underbrace{\alpha_i}_{\text{Gewichte}} \underbrace{f(x_i)}_{\text{Knoten}}$$

Falls f eine konstante Funktion ist, $f(x) = 0$, verlangt man

$$I(f) = I_n(f)$$

Daraus folgt

$$I(f) = c(b-a) = c\left(\sum_{i=0}^{n} \alpha_i\right)$$

$$\sum_{i=0}^{n} = b - a$$

Als *Ordnung* oder *Genauigkeit einer Quadraturformel* definiert man diejenige natürliche Zahl $r \geq 0$, für die gilt:

$$I_n(f) = I(f) \quad \forall f \in P_r$$

d.i. eine Quadraturformel r-ter Ordnung integriert Polynome vom Grad r exakt.
Bei der praktischen Durchführung wird man im Allgemeinen nicht $f(x)$ durch $p_{n,f}(x)$ im gesamten Intervall $[a, b]$ ersetzen. Stattdessen wird man $[a, b]$ zuerst in Teilintervalle $[x_i, x_{i+1}]$, $i = 0..n$, zerlegen und auf jedem Teilintervall $f(x)$ durch $p_{n,f}(x)$ approximieren. Das liefert die sogenannten *zusammengesetzten interpolarischen Quadraturformeln*.

7.1.1 Mittelpunktregel

In dieser Quadraturformel wird f durch eine kosntante Funktion ersetzt, deren Wert der Funktionswert in der Mitte des Intervalls ist:

$$I_0(f) = \int_a^b f\left(\frac{a+b}{2}\right)\, dx = f\left(\frac{a+b}{2}\right)(b-a).$$

PICTURE

Sei $f \in C^2([a, b])$. Taylor-Entwicklung on f im Intervallmittelpunkt liefert

$$f(x) = f\left(\frac{a+b}{2}\right) + f'\left(\frac{a+b}{2}\right) \cdot \left(x - \frac{a+b}{2}\right) + \frac{f''(\xi)}{2} \cdot \left(x - \frac{a+b}{2}\right)^2$$

mit $x, \xi \in [a, b]$. Der Quadraturfehler ist

$$\begin{aligned} I(f) - I_0(f) =& \int_a^b \text{Taylorentwicklung}\, dx - f\left(\frac{a+b}{2}\right) \cdot (b-a) \\ =& \underline{f\left(\frac{a+b}{2}\right) \cdot (b-a)} + f'\left(\frac{a+b}{2}\right) \cdot \mathbf{0} + \frac{f''(\xi)}{3} \cdot \left(\frac{b-a}{2}\right)^3 - \underline{f\left(\frac{a+b}{2}\right) \cdot (b-a)} \\ =& \frac{f''(\xi)}{3}\left(\frac{b-a}{2}\right)^3 \end{aligned}$$

Damit sit die Mittelpunktregel exakt für die konstante und lineare Funktionen, da dort $f''(x) = 0$. Sie hat die Ordnung 1.

$$\begin{aligned} \int_a^b \left(x - \frac{a+b}{2}\right)^2 =& \frac{1}{3}\left(x - \frac{a+b}{2}\right) |_a^b \\ =& \frac{1}{3}\left(\left(\frac{b-a}{2}\right)^3 - \left(\frac{b-a}{2}\right)^3\right) \\ =& \frac{2}{3}\left(\frac{b-a}{2}\right)^3 \end{aligned}$$

Wir zerlegen $[a, b]$ in m gleichlange Teilintervalle der Länge $H = \frac{b-a}{m}$ und bezeichnen

$$x_k = a + \frac{2k+1}{2}H \quad k = 0..m-1$$

Das sind die Mittelpunkte der Teilintervalle. Dann hat die zusammengesetzte Mittelpunktregel die Gestalt

$$I_{0,m}(f) = H \sum_{k=0}^{m-1} f(x_k)$$

$$PICTURE$$

Analog zu oben erhält man für den Quadraturfehler

$$I(f) - I_{q,m}(f) = \frac{b-a}{24} H^2 f''(\xi).$$

7.1.2 Trapezregel

Bei dieser Quadraturformel verwendet man das Lagrange-Interpolationspolynom vom Grad 1 bezüglich a und b

$$PICTURE$$

Man erhält

$$\begin{aligned} I_1(f) =& f(a) \int_a^b \frac{(x-b)}{a-b}\, dx + f(b) \int_a^b \frac{(x-a)}{b-a}\, dx \\ =& \frac{b-a}{2} \left(f(a) + f(b)\right). \end{aligned}$$

Für den Quadraturfehler kann man zeigen, dass

$$I(f) - I_1(f) = \frac{-(b-a)^3}{12} f''(\xi)$$

mit $\xi \in [a, b]$ ist. Damit ist die Trapezregel auch von 1. Ordnung genau. De zusammengesetzte Trapezregel für eine gleichmäßige Zerlegung von $[a, b]$ wie oben hat die Form

$$\begin{aligned} I_{1,m}(f) =& \frac{H}{2} \sum_{k=0}^{m-1} \left(f(x_k) + f_{k+1}\right) \\ =& H \left(\frac{f(a)}{2} + f(x_1) + \ldots + f(x_{m-1}) + \frac{f(b)}{2} \right) \end{aligned}$$

wobei hier $x_k = a + k \cdot H$, $k = 0..m$ die Grenzen der Teilintervalle sind

$$PICTURE$$

7.1.3 Simpson-Regel

Hier würd f durch das Interpolationspolynom 2. Gradesbezüglich der Knoten $x_0 = a$, $x_1 = \frac{b+a}{2}$, $x_2 = b$ approximiert. Man erhält die Quadraturformel

$$I_2(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right).$$

Der Quadraturfehler ist

$$I(f) - I_2(f) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

falls $f \in C^4([a,b])$, mit $\xi \in [a,b]$. Die Simpson-Regel ist von dritter Ordnung genau.
Die zusammengesetzet Simpson-Regel hat die Gestalt

$$I_{2,m}(f) = \frac{H}{6}\left[f(x_0) + 2\sum_{r=1}^{m-1} f(x_{2r}) + 4\sum_{s=0}^{m-1} f(x_{2r+1}) + f(\underbrace{x_{2m}}_{=b})\right].$$

wobei $x_k = a + k \cdot \frac{H}{2}$, $k = 0..2m$, die Knoten sind.

7.2 Die Newton-Cotes-Formeln

Die Newton-Cotes-Formeln (NC) sind die Verallgemeinerung der im Abschnitt 7.1 eingeführten Quadraturformeln. diese formeln basieren auf der Nutzung der Lagrange-Interpolierten und auf einer Zuerlegung von $[a,b]$, bei welcher die Knoten den gleichen Abstand h voneinander besitzen:

$$x_k = x_0 + k \cdot h \quad k = 0..n.$$

Man unterscheidet:

geschlossene Newton-Cotes-Formeln Die Randpunkte des Intervalls sind Knoten der Quadraturformel: $x_0 = a$, $x_n = b$, $h = \frac{b-a}{n}$

offene Newton-Cotes-Formeln Die Randpunkte a, b sind keine Knoten der Quadraturformel, man wählt $x_0 = a + h$, $x_n = b - h$, $h = \frac{b-a}{n+2}$, $n \geq 0$.

Nach der Festlegung der Knoten müssen jetzt noch die Gewichte α_i in den Newton-Cotes-Formeln berechnet werden. Eine wichtige Eigenschaft der Newton-Cotes-Formeln ist, dass diese Gewichte von n und h abhängen, jedoch nicht vom Integrationsintervall $[a,b]$. Damit können die Gewichte unabhängig von $[a,b]$ tabelliert werden.
Diese Eigenschaft betrachten wir für die geschlossenen Newton-Cotes-Formeln. Mit der Variablentransformation

$$x = x_0 + th$$

erhält man

$$\begin{aligned}
\alpha_i &= \int_a^b l_i(x)\,dx \\
&= \int_a^b \left(\prod_{j=0}^{n} \frac{x - x_j}{x_i - x_j} \right) dx \\
&= \int_0^n \left(\prod_{j=0, j\neq i}^{n} \frac{a + th - (a + jh)}{a + ih - (a + jh)} \right) h\,dt \\
&= h \int_0^n \left(\underbrace{\prod_{j=0, j\neq i}^{n} \frac{t - j}{i - j}}_{\phi_i(\mathbf{t})} \right) dt \\
&= h \underbrace{\int_0^n \phi_i(t)\,dt}_{\omega_i}
\end{aligned}$$

Man erhält die geschlossene Newton-Cotes-Formel

$$I_n(f) = h \sum_{i=0}^{n} \omega_i f(x_i) \quad \text{mit } \omega_i = \int_0^n \phi_i(t)\,dt$$

Für die offenen Newton-Cotes-Formeln erhält man die gleiche Darstellung mit

$$\omega_i = \int_{-1}^{n+1} \phi_i(t)\,dt$$

Aus $\sum_{i=0}^{n} \alpha_i = b - a$ folgt

$$\sum_{i=0}^{n} \omega_i = \frac{1}{h} \sum_{i=0}^{n} \alpha_i = \frac{b-a}{h} = \begin{cases} n & \text{geschlossene Newton-Cotes-Formel} \\ n+2 & \text{offene Newton-Cotes-Formel} \end{cases}$$

Man erhält speziell

n	ω_0	ω_1	ω_2		ω_0	ω_1	ω_2	
0	/	/	/		2	/	/	7.1.1
1	$\frac{1}{2}$	$\frac{1}{2}$	/	7.1.2	$\frac{3}{2}$	$\frac{3}{2}$	/	
2	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{1}{3}$	7.1.3	$\frac{8}{3}$	$-\frac{\mathbf{4}}{\mathbf{3}}$	$\frac{8}{3}$	

Man beachte das negative Gewicht für die offeene Newton-Cotes-Formel mit $n = 2$. Newton-Cotes-Formeln höheren Frades ($n \geq 8$ für geschlossene, $n \geq 2$ für offene) besitzen negative

Gewichte. Diese Gewichte können eine Quelle numerischer Instabilität sein, insbesodnere wegen Rundungsfehlern (Auslöschung).
Für den Integrationsfehler gilt:

Satz Sei $f \in C^{n+2}([a,b])$. Für n gerade sind die Newton-Cotes-Formeln von $(n+1)$-ter Ordnung genau und für n ungerade von n-ter Ordnung.

7.3 Gaußsche Quadraturformeln

Bei den Newton-Cotes-Formeln wird die zu integrierende Funktion f durch ein Polynom n-ten Grades approximiert und man erhält Quadraturformeln, von n-ter bzw. $(n+1)$-ter Ordnung Genauigkeit. Dies ist jedoch nicht die maximale Genauigkeit, die man mit einem Polynom n-ten Grades erreichen kann. Um eine höhere Genauigkeit als bei den Newton-Cotes-Formeln zu erreichen, muss man die Einschränkung, dass die Stützstellen äquidistant sein sollen, fallen lassen. Man hat also als Freiheitsgrade

- $n+1$ Stützstellen
- $n+1$ Gewichte

minus der Nebenbedingung, dass die Summe der Gewichte gleich $b-a$ sein soll. D.u., die Gesamtanzahl der Freiheitsgrade ist $2n+1$. Das ist die maximale Ordnung, die erreichba ist. Zur Konstruktion von Quadraturformeln, die diese Ordnung besitzen, den sogenannten *Gaußschen Quadraturformeln*, benötigt man spezielle Polynome.

Definition Sei $\omega : [a,b] \longrightarrow \mathbb{R}$ eine Gewichtsfunktion mit positiven Funktionswerten. Eine Menge $\{p_0(x), \ldots, p_n(x)\}$ von Polynomen heißt *orthogonal* bzgl. $\omega(x)$, falls

$$\int_a^b \omega(x) p_j(x) p_k(x)\, dx = 0 \qquad \forall j \neq k$$

■

Im Fall $[a,b] = [-1,1]$ und $\omega(x) \equiv 1$ bilden die so genannten *Legendre-Polynome* die Menge der orthogonalen Polynome:

$$\begin{aligned} p_0(x) &= 1 \\ p_1(x) &= x \\ p_2(x) &= \frac{1}{2}\left(3x^2 - 1\right) \\ p_3(x) &= \frac{1}{2}\left(5x^3 - 3x\right) \\ &\ldots \end{aligned}$$

Für diese Polynome gilt:

$$\int_{-1}^{1} p_j(x) p_j(x)\, dx = \frac{2}{2j+1}$$

Man kann orthogonale Polynome rekursiv definieren und man kann zeigen, dass alle Nullstellen dieser Polynome reell sind und in $]a, b[$ liegen. Den Zusammenhang zu Quadraturformeln stellt folgender Satz her:

Satz Zu jeden $n \in \mathbb{N}$ gibt es eindeutig bestimmte Knoten x_i und Gewichte α_i, $i = 0..n$, so dass

$$\int_a^b \omega(\alpha) p(x)\, dx = \sum_{i=0}^{n} \alpha_i p(x_i) \quad \forall p \in P_{n+1}$$

Die x_i sind die Nullstellen von $p_{n+1}(x)$, des Orthogonalpolynoms vom Grad $n+1$ bezgl. ω. Die α_i ergeben sich als Lösung des linearen Gleichungssystems

$$\begin{pmatrix} p_0(x_0) & p_0(x_0) & \dots & p_0(x_n) \\ p_1(x_0) & p_1(x_0) & \dots & p_2(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ p_n(x_0) & p_n(x_0) & \dots & p_n(x_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \begin{pmatrix} \int_a^b \omega(x) p_0^2(x)\, dx \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

wobei $\{p_0(x), \dots, p_n(x)\}$ die Menge der ersten $(n+1)$ Orthogonalpolynome bzgl. ω ist.
Es gilt: $\alpha_i > 0$, $\forall i = 0..n$

■

Verwendet man die Gewichtsfunktion $\omega(x) \equiv 1$, so spricht man von *Gauß-Legendre-Quadratur*. Man geht wie folgt vor:

$$\begin{aligned} \int_a^b f(x)\, dx &= \int_{-1}^{1} f(t) \frac{(b-a)}{2}\, dt \\ &\overset{\text{numerische Quadratur}}{\approx} \frac{b-a}{2} \sum_{i=0}^{n} \underbrace{\alpha_i}_{\text{Gewichte}} f(\underbrace{t_i}_{\text{Knoten}}) \\ t &= \frac{2x - (a+b)}{b-a} \end{aligned}$$

wobei $(n+1)$ der Grad des Legendre-Polynoms ist, dessen Nullstellen die Knoten $t_i \in]-1, 1[$ sind und α_i sind die entsprechenden Gewichte. Knoten und Gewichte sind tabelliert:

N		
1	$x_{0,1} = \pm\frac{\sqrt{3}}{3}$	$\alpha_{0,1} = 1$
2	$x_{0,2} = \pm\sqrt{\frac{3}{5}}$	$\alpha_{0,2} = \frac{5}{9}$
	$x_1 = 0(?)$	$\alpha_1 = \frac{5}{9}(?)$

8 Numerische Methoden zur Lösung gewöhnlicher Differentialgleichungen

In diesem Kapitel werden Verfahren vorgestellt, mit denen man die Lösung für Anfangswertprobleme (AWP) eines expliziten Systems gewöhnliche Differentialgleichungen 1. Ordnung

$$\vec{y}' = \vec{f}(x, \vec{y}), \qquad \vec{y}(x_0) = \vec{y_0}$$

numerisch approximieren kann. Die Betrachtung von Systemen 1. Ordnung ist keine große Einschränkung, da man ka explizite Differentialgleichungen höherer Ordnung un solch ein System überführen kann. Der einfacheren Darstellung wegen werden jedoch alle Verfahren für Anfangswertprobleme von skalaren Differentialgleichungen 1. Ordnung beschrieben:

$$y' = f(x, y), \qquad y(x_0) = y_0 \tag{1}$$

Die Erweiterung der Verfahren auf Systeme ist im allgemeinen unkompliziert.
Im Allgemeinen ist eine numerische Approximation der Lösung von (1) notwendig, da man nur in den wenigsten Fällen eine analytische Lösung findet. Es gibt sogar Differentialgleichungen, wie

$$y' = x^2 + y^2$$

für die man eine analytische Lösung nachweisbar nicht durch elementare Funktion und Integration ausgeben kann.

8.1 Theorie

Zunächst werden Resultate zur Existenz und Eindeitigkeit der Lösung von (1) aus der Analysis zusammengefasst. Wir suchen eine Lösung von (1) mit $y \in C^1(I)$.

- Lokale Existenz und Eindeutigkeit (Picard-Lindelöf). Sei $f(x, y)$ in einer Umgebung (x_0, y_0) Lippschitz-stetig bzgl. y, d.i. es gibt Umgebungen

  $$U_x = (x_0 - a, x_0 + a) \subseteq I,\ U_y(y_0 - b, y_0 + b)$$

  so dass

  $$|f(x, y_1) - f(x, y_2)| \leq L\,|y_1 - y_2|$$

  für alle $x \in U_x$, $y_1, y_2 \in U_y$
  Dann besitzt (1) eine eindeutige Lösung in einer Umgebung r_0 von x_0 mit

  $$0 < r_0 < \min\{s, \frac{b}{M}, \frac{1}{L}\}$$

  mit

  $$M = \max_{(x,y) \in U_x \times U_y} |f(x, y)|$$

- Globale Existenz und Eindeitigkeit. Die Gleichung (1) besitzt eine eindeitige globale Lösung, falls die Bedingung für die Ideale Lösung für $U_x = i$ und $U_y = \mathbb{R}$ gilt. D.i. f ist gleichmäßig Lippschitz-stetig bezgl. y.

Zur Untersuchung der Stabilität von (1) betrachten wir das folgende gestörte Problem:

$$z'(x) = \quad f(x,z(x)) + \delta(x) \quad x \in I \tag{2}$$

$$z(x_0) = \quad y_0 + \delta_0$$

wobei $\delta_0 \in \mathbb{R}$ und $\delta(x) \in C(I)$.

Definition Sei I eine beschränkte Menge. Das Problem (1) ist *stabil im Sinne von Ljapunov* auf I, falls für alle Störungen $(\delta_0, \delta(x))$ mit

$$|\delta_0| < \epsilon, \quad |\delta(x)| < \epsilon, \quad \delta x \in I$$

wobei ϵ hinreichend klein ist, so dass das gestörte Problem (3) eine eindeutige Lösung besitzt, gilt, dass ein $C > 0$ existiert, welches unabhängig von ϵ ist, so dass

$$|y(X) - z(x)| \leq C\epsilon \quad \forall x \in I$$

Ist I nicht von oben beschränkt, sagt man, dass (1) *asymptotisch stabil* ist, falls (1) Ljapunoc-stabil in jedem beschränkten Teilintervall ist und außerdem

$$\lim_{x \longrightarrow \infty} |y(x) - z(x)| = 0$$

■

Die Bedingung, dass (1) stabil ist, ist äquivalent zur Bedingung, dass (1) gut gestellt ist im Sinne von Abschnitt 3.

8.2 Einführendes Beispiel: das explizite Euler-Verfahren

Leonard Euler (1707-1783)
Dabei handelt es sich um das einfachste numerische Verfahren zur Lösung von Differentialgleichungen. Es wird auch *Vorwärts-Euler-Verfahren* genannt.
Wir betrachten das Anfangswertproblem (1) im Intervall $[x_0, x_e]$. Dieses Intervall wird on N gleichlange (der Einfachheit halber) Teilintervalle zerlegt $\longrightarrow$ *Gitter*.
Es sei $h = \dfrac{x_e - x_0}{n}$

$$PICTURE$$

interpretiert man(1) von x_0 bis x, erhält man die äquivalente Integralgleichung

$$\int_{x_0}^{x} y'(t)\, dt = \int_{x_0}^{x} f(t, y(t))\, dt \qquad \Leftrightarrow$$

$$y(x) = y(x_0) + \int_{x_0}^{x} f(t, y(t))\, dt.$$

Wir sind nun an einer Approximation der Lösung in x_1 interessiert. Es ist

$$y(x_1) = y_0 + \int_{x_0}^{x_1} f(t, \underbrace{y(z)}_{\textbf{nicht bekannt}}) \, dt$$

Der Wert des Integrals wird nun auf (gröbste) Weise approximiert. Integrand an der untersten Integrationsgrende mal Länge des Intervalls.

$$\begin{aligned} y_1 =& y_0 + f(x_0, y(x_0))(x_1 - x_0) \\ =& y_0 + h \cdot f(x_0, y_0) \end{aligned}$$

Das ist eine Näherung für $y(x_1)$.

Schreibweise

Lösung von (1) $y(x_k)$

numerische Näherung y_k

Nun kann man, startend von y_1 eine Näherung y_2 von $y(x_2)$ auf die selbe Art und Weise berechnen. Damit erhält man das *explizite Euler-Verfahren*

$$y_{k+1} = y_k + h \cdot f(x_k, y_k). \tag{3}$$

Bemerkung Setzt man etwas höhere Differenzierbarkeit für $y(x)$ vorraus, kann man das Verfahren auch über die Taylor-Entwicklung von $y(k)$ in x_{k+1} herleiten.

■

geometrische Interpretation Das Verfahren lässt sich einfach geometrisch interpretieren

PICTURE

Im Punkt (x_k, y_k) nimmt man den (ungefähren) Anstieg der Lösung

$$y'(x_k) \overset{(1)}{=} f(x_k, y(x_k)) \approx \underbrace{f(x_k, y_k)}_{\text{bekannt}}$$

und geht auf einem Geradenstück mit diesem Anstief bis zum Punkt mit der x-Koordinate x_{k+1}.
Anstieg der Geraden von (x_k, y_k) nach (x_{k+1}, y_{k+1}):

$$\frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{y_{k+1} - y_k}{h} = f(x_k, y_k) \Leftrightarrow (3)$$

Beispiel

$$y' = x, \quad y(x_0 = 0) = 1, \quad x \in [0, 1]$$

Wir nehmen die Stützstellen: $x_k = \frac{k}{10}$, $k = 0..10 \Rightarrow h = \frac{1}{10}$. Man erhält mit (3):

$$\begin{array}{lll} y_1 & = y_0 + h \cdot f(x_0, y_0) & = 1 + 0 \qquad\quad = 1 \\ y_2 & = y_1 + h \cdot f(x_1, y_1) & = 1 + \frac{1}{10} \cdot \frac{1}{10} \quad = 1.01 \\ \vdots & & \\ y_{10} & = y_9 + h \cdot f(x_9, y_9) & = \ldots \qquad\quad = 1.45 \end{array}$$

Nun stellen sich folgende Fragen:

- Funktionert das explizite Euler-Verfahren immer (unabhängig von $f(x, y)$, h)?
- Wird die Lösung immer besser, wenn man die Gitterweite h verkleinert? Wie groß ist die Verbesserung?

8.3 Eigenschaften von Einschritt-Verfahren

Das explizite Euler-Verfahren gehört zur Klasse der expliziten Einschritt-Verfahren (ESV).

Definition Ein *explizites Einschritt-Verfahren* zur Bestimmung einer Näherungslösung $y_k + 1$ von (1) auf einem Gitter

$$I_N = \{x_0, x_1, \ldots, x_N = x_e\}$$

mit

$$x_0 < x_1 < \ldots < x_N$$

und den Schrittweiten

$$h_k = x_{k+1} - x_k$$

hat die Gestalt

$$y_0 = y(x_0), \quad y_{k+1} = y_k + h \cdot \Phi(x_k, y_k, h_k). \tag{4}$$

Hierbei wird Φ die *Verfahrensfunktion* oder die *Zuwachsfunktion* des Einschritt-Verfahrens genannt.

■

Beispiel explizit Euler (3):

$$\Phi = f(x_k, y_k)$$

■

Bei der qualitativen Beurteilung von Einschritt-Verfahren spielt der lokale Fehler eine zentrale Rolle.

Definition sei $\widehat{y}_{k+1}$ das Resultat eines Schrittes von (4) mit dem Startwert auf der Lösung $y(x)$, d.i.

$$y_{k+1} = \underbrace{y(x_k)}_{\text{exakter Wert}} + h_k \Phi(x_k, y(x_k), h_k).$$

Dann heißt

$$le(x_{k+1}) = le_{k+1} = y(x_{k+1}) - \widehat{y}_{k+1}$$

lokaler Fehler.

PICTURE

Für ein brauchbares Verfahren wird man fordern, dassder lokale Fehler in einem geeigneten Sinne „klein“ ist. Konkret fordert man, dass der Anstieg des lokalen Fehlers gegen Null konvergiert, wenn die Schrittweite h_k immer kleiner wird für jeden Punkt des zugehörigen Gitters.

Definition Seien $y(x)$ die Lösung des Anfangswertproblems (1), $h_{\max} = \max\limits_k h_k$, $S = \{(x, y) \ : \ x \in [x_0, x_e], \ y \in \mathbb{R}\}$. Dann heißt das Einschritt-Verfahren (4) *konsistent* wenn für alle $f \in C(S)$, die in S einer Lippschitz-Bedingung bzgl. y genügen, gilt

$$\lim_{h_{\max} \longrightarrow 0} \left(\max_{x_k \in I_N} \frac{|le(x_k + h)|}{h} \right) = 0$$

oder

$$\lim_{h_{\max} \longrightarrow 0} \left(\max_{x_k \in I_N} |f(x, y(x)) - \Phi(x, y(x), h_k)| \right) = 0$$

(Beachte: $N \longrightarrow \infty$ für $h_{\max} \longrightarrow 0$)
Beide Forderungen sind äquivalent.

PICTURE

Beispiel Im expliziten Euler-Verfahren ist

$$\Phi(x, y(x), h_k) = f(x, y(x)).$$

Somit ist die zweite Forderung trivialerweise erfüllt und das Verfahren ist konsistent.

■

Für praktische Dinge ist nicht nur die Konsistenz, sondern auch die Güte der Approximation wesentlich. Diese gestattet den Vergleich verschiedener Einschritt-Verfahren. Sei $h_k = h \forall k$.

Definition Ein Einschritt-Verfahren besitzt die *Konsistenzordnung* p wenn p die größte positive ganze Zahl ist, so dass für jede Funktion $f \in C(S)$ die bzgl. y einer Lippschitz-Bedingung genügt, gilt

$$|le(x_k + h)| \leq c \cdot h^{p+1}$$

für alle $x_k \in I_N$, für alle I_N mit $h \in]0, H]$, mit einer von h unabhängigen Konstanten c.

■

Die konstante c kann abhängen von Ableitungen der Funktionen y und f und von paritellen Ableitungen von f.

Beispiel Explizites Euler-Verfahren, sei y zweimal stetig differenzierbar:

$$\begin{aligned} le(\underbrace{x_k + h}_{x_{k+1}}) =& y(x_k + h) - \widehat{y}_{k+1} \\ \overset{\text{T.E.}}{=}& \underline{y(x_k)} + \underline{h \cdot y'(x_k)} + \frac{h^2}{2} y''(x_k + \Theta h) - \underline{y(x_k)} - \underline{h \cdot \underbrace{f(x_k, y(x_k))}_{y'(x_k) \text{ aus (1)}}} \quad \Theta \in]0,1[\\ =& \frac{h^2}{2} y''(x_k + \Theta h) \\ =& \underbrace{\frac{h^2 \|y\| c^2([x_0, x_e])}{2}}_{=C} \end{aligned}$$

Das Verfahren hat die Konsistenzordnung 1.

■

Die Konsistenz ist eine lokale Eigenschaft des Verfahrens. Für praktische Belange ist jedoch die Frage wichtig, ob die numerische Lösung gegen die analytische Lösung der Differentialgleichung konvergiert, wenn man das Fitter immer mehr verfeinert.

$$y' = f(x, y) \qquad y(x_0) = y_0$$

Explizites Eulerverfahren

$$y_{k+1} = y_k + h \cdot f(x_k, y_k)$$

Explizite Einschritt-Verfahren

$$y_{k+1} = y_k + h_k \Phi(x_k, y_k, h_k)$$

Klassifizierung: Konsitenz, Konsistenzordnung Das Einschritt-Verfahren liefert eine Näherung y_k für die Lösung in den Gitterpunkten x_k, $k = 0..N$. Verbindet man diese Punkte mit Geradenstücken von (x_k, y_k) nach (x_{k+1}, y_{k+1}), so erhält man eine stpckweise lineare Näherung für die Lösung, die auf $[x_0, x_e]$ definiert ist. Diese Funktion nennen wir $y^h(x)$.

Definition Ein Einschritt-Verfahren heißt *konvergent* für das Anfangswertproblem (1) auf dem Intervall $I = [x_0, x_e]$, wenn für jede Folge von Gittern $\{I_k\}$ mit $h_{\max} = \max\limits_{h_k \in I_k} h_k \longrightarrow 0$ für den *globalen Fehler*

$$e(x, h) = y(x) - y^h(x)$$

gilt

$$\max_{x \in I_k} |e(x, h)| \longrightarrow 0 \qquad \forall h_{\max} \longrightarrow 0$$

Das Einschritt-Verfahren besitzt die *Konvergenzordnung* p^*, wenn p^* die größte positive ganze Zahl ist, so dass für alle Schrittweiten $h_{\max} \in [0, H]$ gilt

$$|e(x, h)| \leq C \cdot h_{\max}^{p^*}$$

für alle $x \in I_k$, wobei C unabhängig von $h_{\max}$ ist. ■
Es zeigt sich, dass unter bestimmten Bedingungen aus der Konsistenz eines Einschritt-Verfahren die Konvergenz des Einschritt-Verfahren folgt. Ist die Konsistenzordnung gleich p, so ist auch die Konvergenzordnung gleich p.

Satz Sei $y(x)$ die Lösung des Anfangswertproblems (1) mit $f \in C(S)$, wobei f in S einer Lippschitz-Bedingung mit der Lippschitz-Konstanten L genügt. Ferner gelte für den lokalen Fehler die Abschätzung

$$|le(x+h)| \leq C \cdot h^{p+1}$$

für $x \in I_k$, $h \leq h_{\max} \in]0, H]$. Dann gilt für den globalen Fehler

$$|le(x,h)| \leq \frac{C}{L} \underbrace{[\exp(L(x-x_0)-1)]}_{\textbf{groß für } x \gg x_0} h_{\max}^p$$

für alle $x \in I_k$, wobei C unabhängig von $h_{\max}$ ist.

8.4 Runge-Kutta-Verfahren

Runge (1856-1927)
Kutta (1867-1944)
Die Grundidee dieser Verfahren besteht darin, die Verfahrensfunktion $\Phi(x, y, h)$ durch eine Linearkombination von Werten von $f(x, y)$ in diskreten Punkten anzusetzen. Man erhält dadruch Verfahren höherer Ordnung für den Preis, das man mehr Funktionswerte berechnen muss.

Definition Ein *Runge-Kutta-Verfahren (RKV)* hat die Gestalt

$$\begin{aligned} y_0 &= y(x_0) \\ y_{k+1} &= y_k + h \cdot \Phi(x, y, h) \end{aligned}$$

($h = h_k$ der Einfachheit halber),
wobei die Verfahrensfunktion mit Hilfe der Größen

$$K_i(x,y) = f\left(x + \mathbf{c_i}h, y + h\sum_{j=1}^{s} \mathbf{a_{ij}} K_j(x,y)\right)$$

$$\Phi(x,y) = \sum_{i=1}^{s} \mathbf{b_i} K_i(x,y)$$

definiert ist, mit $c_1..c_s$, $b_1..b_s$, $a_{ij} \in \mathbb{R}$.
Die Größen $K_i(x, y)$ werden *Steigungen* genannt, s ist die Anzahl der *Stufen* des Verfahrens. ■

Der besseren Pbersichtlichkeit halber schreibt man ein Runge-Kutta-Verfahren im Allgemeinen in einem Parameterschema, dem sogenannten *Butcher-Schema*

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} = \begin{array}{c|c} c & A \\ \hline & b \end{array}$$

1. c Knotenvektor
2. A Verfahrensmatrix
3. b Gewichtsvektor

Definition Ein numerisches Verfahren zur Lösung von (1) wird *explizit* genannt, falls y_{k+1} direkt durch bereits berechnete Werte y_i, $i \leq k$, berechnet werden kann. Andernfalls heißt das Verfahren *implizit*. ■

Implizite Verfahren benötigen in jedem Schritt die Lösung eines im Allgemeinen nichtlinearen Gleichungssystems zur Berechnung von y_{k+1}.

8.4.1 Explizite Runge-Kutta-Verfahren

Bei expliziten Runge-Kutta-Verfahren können die Steigungen nacheinander berechnet werden.

$$\begin{aligned} K_1(x,y) =& f(x,y) \\ K_2(x,y) =& f\left(x + C_2 h, y + h a_{21} K_1(x,y)\right) \\ =& \vdots \\ K_s(x,y) =& f\left(x + C_s h, y + h \sum_{j=1}^{s-1} a_{ij} \cdot K_j(x,y)\right) \end{aligned}$$

Das Butcher-Schema hat die Gestalt

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

Beispiel Das explizite Euler-Verfahren ist ein explizites Runge-Kutta-Verfahren mit dem Butcher-Schema

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

■

Satz Ein explizites Runge-Kutta-Verfahren ist unter der Bedingung

$$\sum_{i=1}^{s} b_i = 1$$

konsistent.
Gilt

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \quad i \geq 2,$$

so ist $K_i(x, y)$ eine Approximation von mindestens 1. Ordnung an $y'(x + c_i h)$, d.i.

$$y'(x + c_i h) - K_i(x, y) = \mathcal{O}(h^2).$$

Das ziel besteht nun darin, die Koeffizienten a_{ij}, b_i so zu bestimmen, dass das explizite Runge-Kutta-Verfahren eine möglichst hohe Konsistenzordnung besitzt. Die Konsistenzordnung eines s-Stufigen Runge-Kutta-Verfahren kann aus der Taylor-Entwiclung des lokalen Fehlers hergeleitet werden.

Beispiel: $s = 2$ Wir betrachten das sogenannte *autonome Anfangswertproblem*

$$y' = f(y), \qquad y(x_0) = y_0.$$

Steigungen:

$$\begin{aligned} K_1(y) =& f(y) \\ K_2(y) =& f\left(y + ha_{21}K_1(y)\right) = \qquad\qquad f(y + ha_{21} \underbrace{f(y)}_{\textbf{als Konstante auffassen}}) \\ \overset{T.E.}{=}& f(y) + ha_{21}f(y)\underbrace{f_y}_{\frac{df}{dy}}(y) + \mathcal{O}(h^2) \end{aligned}$$

Dann gilt für die Verfahrensfunktion

$$\begin{aligned} \Phi(y, h) =& b_1K_1(y) + b_2K_2(y) \\ =& (b_1 + b_2)f(y) + b_2ha_{21}f(y)f_y(y) + \mathcal{O}(h^2) \end{aligned}$$

und die Lösung

$$y(x + h) \overset{T.E.}{=} y(x) + h\underbrace{y'(x)}_{f(y)} + \frac{h^2}{2}y''(y) + \mathcal{O}(h^3)$$

wobei

$$\begin{aligned} y''(x) =& \frac{d}{dx}\left(y'(x)\right) \\ \overset{DGL}{=}& \frac{d}{dx}\left(f(y(x))\right) \\ \overset{Kettenregel}{=}& f_y(y)y'(x) \\ \overset{DGL}{=}& f_y(y)f(y)\ . \end{aligned}$$

Es folgt für den lokalen Fehler:

$$\begin{aligned} le(x+h) =& y(x+h) - y(x) - h\Phi(y,h) \\ \overset{einsetzen}{=}& \underline{y(x)} + hf(y) + \frac{h^2}{2}f_y(y)f(y) + \mathcal{O}(h^3) \\ & - \underline{y(x)} - h(b_1+b_2)f(y) \\ & - h^2 b_2 a_{21} f_y(y)f(y) \\ =& h(1-(b_1+b_2))f(y) + h^2\left(\frac{1}{2} - b_2a_{21}\right)f_y(y)f(y) + \mathcal{O}(h^3) \end{aligned}$$

Für eine möglichst hohe Konsistenzordnung müssen die ersten beiden Terme verschwinden, also

$$\mathbf{b_1 + b_2 = 1}$$

(Das ist ohnehin für Konsitenz nötig!)

und

$$\mathbf{b_2 a_{21} = \frac{1}{2}} \quad \overset{\mathbf{c_2 = a_{21}}}{\Longleftrightarrow} \quad \mathbf{b_2 c_2 = \frac{1}{2}}$$

Mit diesen beiden Bedingungen sind alle 2-stufigen expliziten Runge-Kutta-Verfahren gekennzeichnet, die die Konsistenz- und Konvergenzordnung 2 besitzen:

$$\begin{array}{c|cc} 0 & & \\ c_2 & c_2 & \\ \hline & 1-\frac{1}{2c_2} & \frac{1}{2c_2} \end{array} \text{ mit } c_2 \neq 0$$

Methode von Heun (1900): $c_2 = \frac{1}{2}$

$$\begin{array}{c|cc} \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

Methode von Kutta (1895): $c_2 = 1$

$$\begin{array}{c|cc} 1 & 1 & \\ \hline & \frac{3}{2} & \frac{1}{2} \end{array}$$

Analog kann man Bedingungen an die Koeffizienten des Runge-Kutta-Verfahrens finden, um höhere Orndungen zu erreichen. Es bleibt noch die Frage, was die Mindestanzahl von Stufen ist (in einem expliziten Runge-Kutta-Verfahren), um eine gewisse Ordnung erreichen zu können. Antworten gab Butcher (1963, 1965, 1985).

p	1	2	3	4	5	6	7	8
s	1	2	4	5	6	7	9	11

Beispiel

$$y' = x^3 y^2 \qquad y(0) = 1$$

$$\Leftrightarrow \qquad y = -\frac{4}{x^4 - 4}$$

8.4.2 Implizite Runge-Kutta-Verfahren

Implizite Runge-Kutta-Verfahren können mit Hilfe der Integraldarstellung des Anfangswertproblems (1) hergeleitet werden. Der einfachheit halber, hänge die rechte Seite von (1) nur von x ab. Dann lautet die Integraldarstellung von (1):

$$y(x) = y_0 + \int_{x_0}^{x} f(t)\ dt\ .$$

Sei $y(x)$ für $x = x_k$ gegeben, dann ist

$$y(x_{k+1}) = y(x_k) \int_{x_k}^{x_{k+1}} f(t)\ dt\ .$$

Die Idee von impliziten Runge-Kutta-Verfahren besteht nun einfach darin, das Integral auf der rechten Seite durch eine geignete Quadraturformel zu approximieren:

$$\int_{x_k}^{x_{k+1}} f(t)\ dt \approx h \cdot \sum_{j=1}^{s} b_j f(x_k + c_j h)$$

wobei b_j die Gewichte, und $x_k + c_j h$ die Knoten sind.

Man kann zeigen, dass für jedes implizite Runge-Kutta-Verfahren mit Gewichten b_j und Knoten $x_k + c_j h$ eine entsprechende Gaußsche Quadraturformel mit den gleichen Gewichten und Knoten existiert.
Seien zum Beispiel die Koeffizienten $c_1..c_s$ die Nullstellen des Legendre-Polynoms $p_3(t)$ in den Variablen

$$t = \frac{2}{h}(x - x_k) - 1 \longrightarrow t \in [-1, 1]\ .$$

Hat man $c_1..c_s$ berechnet (oder nachgeschlagen) kann man die Koeffizienten a_{ij} und b_j so bestimmen, dass man ein Verfahren der Ordnung $\underline{2s}$ erhält. Dazu muss ma ndie linearen Systeme

$$\sum_{j=1}^{s} c_j^{k-1} a_{ij} = \frac{c_i^k}{k} \quad k = 1..s$$

$$\sum_{j=1}^{s} c_j^{k-1} a_{ij} = \frac{1}{k} \quad k = 1..s$$

lösen, siehe Literatur.

Einige Klassen von impliziten Runge-Kutta-Verfahren

Gauß-Legendre Runge-Kutta-Verfahren Es werden die Gauß-Legendre Quadraturknoten benutzt. ein s-stufiges Verfahren besitzt die (maximal mögliche) Ordnung $2s$.
Beispiel:

- *Implizite Mittelpunktregel*

$$y_{k+1} = y_k + h_k f\left(x_k + \frac{h_k}{2}, \frac{1}{2}\left(y_k + y_{k+1}\right)\right)$$

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \qquad s = 1,\ p = 2$$

Gauß-Radan-Verfahren Diese Verfahren sind dadurch charakterisiert, dass einer der beiden Endpunkte des Intervalls $[x_k, x_{k+1}]$ zu den Quadraturknoten gehört. Ein s-stufiges Verfahren dieser Klasse kann maximal von Ordnung $2s - 1$ sein.
Beispiele:

-

$$\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array} \qquad s = 1,\ p = 1$$

- *Implizites Euler-Verfahren*

$$y_{k+1} = y_k + h_k f(\mathbf{x_{k+1}}, \mathbf{yk + 1})$$

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad s = 1,\ p = 1$$

Gauß-Lobatto-Verfahren In diesem Verfahren sind beide Endpunkte des Intervalls $[x_k, x_{k+1}]$ Quadraturpunkte. Ein s-stufiges Verfahren dieser Art kann maximal von Ordnung $2s - 2$ sein.
Beispiele:

- *Trapezregel, Crank-Nicolson-Verfahren*

$$y_{k+1} = y_k + \frac{h_k}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right)$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad s = 2,\ p = 2$$

-

$$\begin{array}{c|cc} 0 & \frac{1}{2} & 0 \\ 1 & \frac{1}{2} & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad s = 2,\ p = 2$$

Bei einem s-stufigen impliziten Runge-Kutta-Verfahren hat man ein gekoppelts nichtlineares System für $K_1(x, y)..K_s(x, y)$ in jedem Schritt zu lösen. Für höhere s ist das unter Umständen teuer. Ein Kompromiss besteht darin sogenannte *diagonal-implizite Runge-Kutta-Verfahren (DIRK)* zu nutzen.

$$\begin{array}{c|ccccc} c_1 & a_{11} & 0 & 0 & \dots & 0 \\ c_2 & a_{21} & a_{22} & 0 & \dots & \vdots \\ c_3 & a_{31} & a_{32} & a_{33} & \ddots & \\ \vdots & \vdots & \ddots & & \ddots & 0 \\ c_s & a_{s1} & \dots & & & a_{ss} \\ \hline & b_1 & \dots & & & b_s \end{array}$$

Für diagonal-implizite Runge-Kutta-Verfahren hat man s unabhängige nichtlineare Gleichungen für die Steigungen zu lösen.

8.5 Lineare Stabilitätstheorie von Einschritt-Verfahren

Für die Untersuchung der Stabilität von numerischen Verfahren zur Lösung von (1) untersucht man ihr Verhalten bei der Lösung des Modellproblems

$$y' = \lambda y, \quad y(0) = y_0, \quad \lambda \in \mathbb{C} \tag{5}$$

Die analytische Lösung des Anfangswertproblems lautet

$$y(x) = e^{\lambda x} y_0 \ text.$$

Wird nun die Anfangsbedingung leicht gestört auf den Wert $y_0 + \delta y$, dann ist die Lösung des gestörten Anfangswertproblems

$$\begin{aligned} \tilde{y}(x) =& e^{\lambda x}(y_0 + \delta y) \\ =& e^{\lambda x} y_0 + \rceil^{\lambda x} \delta y \\ =& y(x) + e^{\lambda x} \delta y \ . \end{aligned}$$

Falls $\Re(\lambda) > 0$ ist, wird die Differenz

$$|y(x) - \tilde{y}(x)| = \left|\delta y e^{\lambda x}\right| \tag{6}$$

Für jedes $\delta y \neq 0$ beliebig groß, falls nur x hinreichend groß ist. Das Anfangswertproblem ist schlecht gestellt. Ist andererseits $\Re(\lambda) < 0$, ann wird (6) beliebig klein für $x \longrightarrow \infty$. In diesem Fall ist das Anfangswertproblem stabil.

Wir betrachten nun ein Einschritt-Verfahren mit Schrittweite h zur Lösung von (5). Die Lösung von (5) im Gitterpunkt x_k ist

$$y(x_k) = e^{\lambda x} y_0$$

und im Gitterpunkt y_{k+1}

$$\begin{aligned} y(x_{k+1}) =& e^{\lambda x_{k+1}} y_0 \\ =& e^{\lambda(x_k+h)} y_0 \\ =& e^{\lambda h} e^{\lambda x_k} y_0 \\ =& e^{\lambda \mathbf{h}} y(x_k) \quad \mathbf{z} := \lambda \mathbf{h} \\ =& e^{\mathbf{z}} y(x_k) \end{aligned}$$

Als nächstes werden wir uns ansehen, wie die Formeln verschiedener Einschritt-Verfahren beim Schritt von x_k zu x_{k+1} aussehen.

8.5.1 Beispiel: Explizites Euler-Verfahren

allgemeine Gestalt:

$$y_{k+1} = y_k + h f(x_k, y_k)$$

speziell für (5):

$$\begin{aligned} y_{k+1} =& y_k \underbrace{h\lambda}_{=z} y_k \\ =& (\mathbf{1} + \mathbf{z}) y_k \\ =& \mathbf{R}(\mathbf{z}) x_k \ . \end{aligned}$$

Es gilt, unabhängig von $\Re(z)$:

$$\lim_{|z| \longrightarrow \infty} |R(z)| = \infty \ .$$

8.5.2 Beispiel: Implizites Eulerverfahren

allgemeine Gestalt:

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

speziell für (5):

$$\begin{aligned} y_{k+1} =& y_k + h\lambda y_{k+1} \quad \Leftrightarrow \\ (1 - z) y_{k+1} =& y_k \\ y_{k+1} =& \frac{1}{\mathbf{1} - \mathbf{z}} y_k \\ =:& \mathbf{R}(\mathbf{z}) y_k \end{aligned}$$

In diesem Fall gilt, unabhängig von $\Re(z)$:

$$\lim_{|z|\longrightarrow\infty} |R(z)| = 0 \ .$$

8.5.3 Beispiel: Trapezregel

allgemeine Gestalt:

$$y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right)$$

speziell für (5):

$$\begin{aligned} x_{k+1} =& y_k + \frac{h}{2}(\lambda y_k + \lambda y_{k+1}) \quad \Leftrightarrow \\ \left(1 - \frac{z}{2}\right) y_{k+1} =& \left(1 + \frac{z}{2}\right) y_k \qquad\qquad \Leftrightarrow \\ y_{k+1} =& \frac{\left(\mathbf{1} + \frac{\mathbf{z}}{\mathbf{2}}\right)}{\left(\mathbf{1} - \frac{\mathbf{z}}{\mathbf{2}}\right)} y_k \\ =:& \mathbf{R}(\mathbf{z}) y_k \end{aligned}$$

8.5.4 Beispiel: s-stufiges explizites Runge-Kutta-Verfahren

Man kann zeigen, dass

$$y_{k+1} = P_s(z) y_k$$

ist, wobei $P_s(z)$ ein Polynom s-ten Grades mit reellen Koeffizienten ist, d.i.

$$\lim_{|z|\longrightarrow\infty} |P_s(z)| = \infty \ .$$

Definition Die Funktion $R(z)$ heißt *Stabilitätsfunktion* des Verfahrens. ■

9 Klausurstreß

Hi zusammen!
Die jetzt kommende Vorlesung kann ich nicht mitschreiben - wäre aber sehr verbunden wenn die mir jemand zu kommen lassen kann - egal wie ;)