Numerische Mathematik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

WS 2021/2022

Vorlesung 1

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Probleme und Motivation I

- Konstruktion numerischer Verfahren zur näherungsweisen Lösung von "Gleichungen" (Theorie $\rightarrow$ Numerische Analysis)
- Einfluß von Rundungsfehlern auf das Resultat mathematischer Operationen und Methoden

Beispiel 1: (Berechnung der inversen Matrix)

$$a_{ij} = \sin(i^2 + j), i, j = 1..N$$

- Exakt mit der Cramerschen Regel, es müssen nur Determinanten berechnet werden.
- Numerisches Resultat macht nicht immer Sinn. ????

Motivation II

Der Satz von Fermat sagt aus, dass die Gleichung

$$a^n + b^n = c^n, a, b, c, n \in \mathbb{N}$$

für $n > 2$ keine Lösungen besitzt. Manche Personen versuchen sich trotzdem noch an Gegenbeispielen...

Inhalt

- Rundungsfehler, Computerzahlen, Konditionszahlen
- Große lineare Gleichungssysteme
- Interpolation und Approximation
- Numerische Differentiation
- Numerische Lösung gewöhnlicher Differentialgleichungen (neu)+ Individuelle Projekte 1.5 ECTS
- Numerische Integration
- Mündliche Prüfung

Einfluß von Rundungsfehlern

- Reelle Zahlen - unendliche Dezimalbrüche
- Normalisierte Darstellung (exponentielle oder Gleitkommadarstellung)

$$x = \pm \underbrace{0.z_1 z_2 \ldots}_{\textit{Mantisse}} * 10^p; \quad z_1 \neq 0; \quad z_i \in \{0, 1, \ldots, 9\}, \quad p \text{ Exponent} \in \mathbb{Z}$$

$$25.86 = 0.2586 \cdot 10^2, \quad 0.00314 = 0.314 \cdot 10^{-2}$$

Problem:

Wir haben im Computer nur eine endliche Anzahl von Stellen für die Mantisse und den Exponenten verfügbar. Relle Zahlen müssen gerundet werden.

$$x = 0 \quad \Rightarrow \quad rd(0) = 0.00 \ldots 0 * b^{-e_1}, \; -e_1 \text{ kleinster möglicher Exponent}$$

$$\Downarrow$$

Exakte Darstellung

Einfluß von Rundungsfehlern

$$x \neq 0 \quad \Rightarrow \quad x = \pm 0.z_1 z_2 \cdots * b^e, \quad z_1 \neq 0$$

$\Downarrow$

1. Fall $e < -e_1$ $\Rightarrow$ $rd(x) = 0$ Unterlauf

2. Fall $-e_1 \leq e \leq e_2$ $\Rightarrow$ **1. unsymmetrisches Runden**

$rd(x) = \pm 0.z_1 z_2 \ldots z_t * b^e$

Abschneiden nach der t-ten Stelle

(leicht zu implementieren)

3. Fall $e > e_2$ $\Rightarrow$ Überlauf

2. symmetrisches Runden

$$rd(x) = \begin{cases} \pm 0.z_1 z_2 \ldots z_{t-1} z_t * b^e & z_{t+1} \leq \frac{b}{2} \\ \pm (0.z_1 z_2 \ldots z_{t-1} z_t + b^{-t}) * b^e & z_{t+1} > \frac{b}{2} \end{cases}$$

symmetrisches Runden zeigt gutes Verhalten. Trotzdem sind einige Probleme zu berücksichtigen. Im Folgenden steht b für die Basis des Zahlensystems.

Beispiele

Beispiel: $t = 3$, Dezimalzahlen (b=10),
$x = 0.5698 \cdots * 10^0 \quad \Longrightarrow \quad rd(x) = 0.570 * 10^0$
$8 > 5$, d.h., das Runden beeinflußt nicht nur die letzte Stelle.

$x = 0.9998 \cdots * 10^0 \quad \Longrightarrow \quad rd(x) = 1.0 = 0.1 * 10^1$
$8 > 5$, d.h., neue Normalisierung, alle Positionen in Mantisse und Exponent werden verändert.

Fehlerdiskussion

1. Fall: Unterlauf

$$\varepsilon(x) = \frac{\mathrm{rd}(x) - x}{x} = -1 \quad \text{für} \quad x \neq 0$$

d.h., relativer Fehler beträgt 100%, absoluter Fehler ist sehr klein, ganze Information ist verloren.

Beispiele

2. Fall: unsymmetrisches Runden

$$0 \geq \varepsilon(x) = \frac{\mathrm{rd}(x) - x}{x} \geq -b^{1-t}$$

d.h., der Fehler ist immer negativ oder Null $\rightarrow$ unsymmetrisch

3. Fall: symmetrisches Runden

$$-\frac{1}{2}b^{1-t} \leq \varepsilon(x) = \frac{\mathrm{rd}(x) - x}{x} \leq \frac{1}{2}b^{1-t}$$

$$\Longrightarrow |\varepsilon(x)| = \left|\frac{\mathrm{rd}(x) - x}{x}\right| \leq \nu = \begin{cases} b^{1-t} & \text{unsym.} \\ 0.5b^{1-t} & \text{sym.} \end{cases}$$

ν heißt *relative Computergenauigkeit*.

Eine andere Möglichkeit der Fehlerbetrachtung basiert auf

$$\mathrm{rd}(x) = x(1 + \varepsilon(x)).$$

Beispiele und neue Probleme

$\rightarrow$ Arithmetische Grundoperationen auf dem Computer.

$b = 10; \quad t = 3; \quad e_1 = e_2 = 20$

$x_1 = 0.123 \cdot 10^0; \quad y_1 = 0.456 \cdot 10^{-3}$

$x_1 + y_1 = 0.123456 \cdot 10^0 \quad \longrightarrow \quad 0.123 \cdot 10^0$ unsym. runden

$x_1 - y_1 = 0.122544 \cdot 10^0 \quad \longrightarrow \quad 0.122 \cdot 10^0$ unsym. runden

$x_1 \cdot y_1 = 0.56088 \cdot 10^{-4} \quad \longrightarrow \quad 0.561 \cdot 10^{-4}$ sym. runden

$x_1 / y_1 = 0.26973\ldots \cdot 10^2 \quad \longrightarrow \quad 0.270 \cdot 10^2$ sym. runden

oder zum Beispiel,

$$x_1 = 0.1 \cdot 10^1; \quad y_1 = 0.1 \cdot 10^{-3}$$
$$x_1 + y_1 = 1.0001 \quad \longrightarrow \quad 1$$
$$x_1 - y_1 = 0.9999 \quad \longrightarrow \quad 1$$

!!! $x_1 + y_1 = x_1 - y_1$ for $y_1 \neq 0$!!!

Problem: Widersprüche zu Axiomen der reelen Zahlen

Beispiele und neue Probleme

wichtige Aufgabe:

Besseres Verständnis der Fehlerfortpflanzung in arithmetischen Operationen. Bewertung numerischer Verfahren.

Ziel:

Kleine Fehler in Eingangsdaten $\longrightarrow$ kleine Fehler in Ausgangsdaten, Resultat.

Beispiel: $s = x + y + z$

theoretisch haben wir: $(x + y) + z = x + (y + z)$

$t = 3; \quad b = 10; \quad e_1 = e_2 = 20$ unsymm. runden

$$s_1 = \mathrm{rd}(\mathrm{rd}(x + y) + z)$$

$$s_2 = \mathrm{rd}(x + \mathrm{rd}(y + z))$$

$x = 0.123 \cdot 10^0; \quad y = 0.456 \cdot 10^{-3} \quad z = 0.789 \cdot 10^{-2}$

$s_1 = 0.130 \cdot 10^0, \qquad s_2 = 0.131 \cdot 10^0$ ok.

Beispiele und neue Probleme

$$x = -0.123 \cdot 10^0; \quad y = 0.124 \cdot 10^0 \quad z = 0.987 \cdot 10^{-2}$$
$$\Longrightarrow \qquad s_1 = 0.198 \cdot 10^{-2}, \qquad s_2 = 0.100 \cdot 10^{-2}$$

$$\text{exaktes Resultat:} \quad s = 0.1987 \cdot 10^{-2}$$
$$\left|\frac{s_1 - s}{s}\right| = 0.0035 \quad , \quad \left|\frac{s_2 - s}{s}\right| = 0.5$$

Fehler unterscheiden sich um Faktor 140.

Relative Konditionszahlen, arithmetische Grundoperationen

$$x, \quad y; \qquad \text{Störungen} \quad \delta x, \quad \delta y$$

$$z = P(x, y) \quad , \quad \text{gesucht ist} \quad \delta z$$

$P(x, y) = x \pm y$:

$$\delta z = (x + \delta x) \pm (y + \delta y) - (x \pm y) = \delta x \pm \delta y$$

$$\frac{\delta z}{z} = \frac{x}{x \pm y}\frac{\delta x}{x} \pm \frac{y}{x \pm y}\frac{\delta y}{y} \qquad \text{relativer Fehler}$$

Die Koeffizienten der relativen Fehler der Eingangsdaten heißen **relative Konditionszahlen**.

$$\frac{x}{x \pm y}, \quad \frac{y}{x \pm y}$$

kann beliebig groß werden (wenn $|x \pm y| \ll |x|$ bzw. $|x \pm y| \ll |y|$)

Relative Konditionszahlen, arithmetische Grundoperationen

$P(x, y) = x \cdot y$:

$$\delta z = (x + \delta x)(y + \delta y) - (xy) = y\delta x + x\delta y + \delta x\delta y$$

$$\approx y\delta x + x\delta y$$

$$\Longrightarrow \frac{\delta z}{z} = \frac{\delta x}{x} + \frac{\delta y}{y} \qquad \text{rel. Konditionszahlen} = 1$$

Das Produkt hat kleine relative Fehler, wenn die Faktoren kleine relative Fehler haben.

$P(x, y) = x/y, \quad y \neq 0$:

$$\delta z = \frac{x + \delta x}{y + \delta y} - \frac{x}{y} = \frac{xy + y\delta x - xy - x\delta y}{y^2 + y\delta y} \approx \frac{\delta x}{x} - \frac{\delta y}{y^2}$$

$$\Longrightarrow \frac{\delta z}{z} = \frac{\delta x}{y} - \frac{\delta y}{y} \qquad \text{rel. Konditionszahlen} = 1, \quad -1$$

Multiplikation und Division sind nicht "gefährlich", die relativen Fehler addieren sich.

Relative Konditionszahlen für Funktionen

Beispiel: Man bewerte die Qualität der numerischen Berechnung der Funktion

$$y = 1 - \sqrt{1-x} \qquad 0 \leq x \leq 1.$$

$$\begin{aligned} \delta y &= P(x+\delta x) - P(x) \approx \frac{dP(x)}{dx}\delta x \qquad \text{(Taylorsche Formel)} \\ &= \frac{1}{2\sqrt{1-x}}\delta x \\ \frac{\delta y}{y} &= \left[\frac{x}{y}\frac{dP(x)}{dx}\right]\frac{\delta x}{x} = \left[\frac{x}{1-\sqrt{1-x}} \cdot \frac{1}{2\sqrt{1-x}}\right]\frac{\delta x}{x} \end{aligned}$$

Die Berechnung ist schlecht konditioniert (*ill-posed*) (beliebig große Konditionszahlen) für $x \to 1$; überraschend !!!

Relative Konditionszahlen für Funktionen

Allgemeiner Fall einer reellwertigen Funktion von n reellen Variablen:

$$
\begin{aligned}
z &= P(x_1, \ldots, x_n) \\
\delta z &= P(x_1 + \delta x_1, \ldots, x_n + \delta x_n) - P(x_1, \ldots, x_n) \\
&\approx \sum_{i=1}^{n} \frac{\partial P}{\partial x_i}(x_1, \ldots, x_n)\, \delta x_i \\
\frac{\delta z}{z} &= \sum_{i=1}^{n} \frac{1}{P(x_1, \ldots, x_n)} \frac{\partial P}{\partial x_i}(x_1, \ldots, x_n)\, \delta x_i
\end{aligned}
$$

$$
\boxed{\frac{\delta z}{z} = \sum_{i=1}^{n} \frac{x_i}{P(x_1, \ldots, x_n)} \frac{\partial P}{\partial x_i}(x_1, \ldots, x_n)\, \frac{\delta x_i}{x_i}}
$$

$\frac{x_i}{P(x_1, \ldots, x_n)} \frac{\partial P}{\partial x_i}(x_1, \ldots, x_n)$ heißen relative Konditionszahlen

Beispiel - Inneres Produkt

$$z = P(x_1, \cdots, x_n, y_1, \cdots, y_n) = (\vec{x} \cdot \vec{y}) = \sum_{k=1}^{n} x_k y_k$$

$$\frac{\partial P(x_1, \cdots, x_n, y_1, \cdots, y_n)}{\partial x_i} = \frac{\partial \sum_{k=1}^{n} x_k y_k}{\partial x_i} = \sum_{k=1}^{n} \delta_{ik} y_k = y_i$$

$$\frac{\partial P(x_1, \cdots, x_n, y_1, \cdots, y_n)}{\partial y_i} = \frac{\partial \sum_{k=1}^{n} x_k y_k}{\partial y_i} = \sum_{k=1}^{n} \delta_{ik} x_k = x_i$$

$$\frac{\delta z}{z} = \sum_{i=1}^{n} \frac{x_i}{P(x_1, \ldots, x_n)} \frac{\partial P}{\partial x_i}(x_1, \ldots, x_n) \frac{\delta x_i}{x_i} =$$

$$= \sum_{i=1}^{n} \frac{x_i y_i}{(\vec{x} \cdot \vec{y})} \frac{\delta x_i}{x_i} + \sum_{i=1}^{n} \frac{y_i x_i}{(\vec{x} \cdot \vec{y})} \frac{\delta y_i}{y_i}$$

Numerische Mathematik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

WS 2021/2022

Vorlesung 2

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Matrixnormen und Rundungsfehler

Motivation: Wir betrachten eine Standardsituation

$$A = (a_{ij})_{i,j=1}^{n}, \quad n = 1000, \quad |\delta a_{ij}| \approx 10^{-11}, \quad |a_{ij}| \approx 1.$$

Man löse das lineare Gleichungssystem mit Hilfe des Gauß-Algorithmus. $\Longrightarrow$ Algorithmus benötigt $\frac{n^3}{3}$ Operationen.

Ein Aufsummieren der Rundungsfehler unter Benutzung der Resultate zu relativen Konditionszahlen bei arithmetischen Grundoperationen führt auf:

$$\frac{1000^3}{3} \cdot 10^{-11} = \frac{10^9 \cdot 10^{-11}}{3} = \frac{10^{-2}}{3}$$

Die mathematische Aufgabe ist es, die Genauigkeit der Lösung von $Ax = b$ oder der Berechnung von $x = A^{-1}b$ besser abzuschätzen.

- Wie kann die Störung/der Fehler von A gemessen werden.
- Wie kann der Fehler der Lösung x gemessen werden?
- Alle Ideen müssen verträglich mit der Theorie linearer Vektorräume sein.
- Können wir für Matrizen eine Norm einführen?

Matrixnormen - erste Ideen

$$\|A\|_2 = \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}$$

Wir müssen Matrixnormen in Verbindung mit Vektornormen betrachten und verlangen

$$\|Ax\|_2 \leq \|A\|_2 \, \|x\|_2.$$

Mit Hilfe der Cauchy-Schwarz-Ungleichung schätzen wir wie folgt ab:

$$\begin{aligned} \|Ax\|_2^2 &= \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij} x_j \right)^2 \leq \sum_{i=1}^{n} \left(\sum_j a_{ij}^2 \right) \left(\sum_j x_j^2 \right) \\ &= \left(\sum_{i,j} a_{ij}^2 \right) \|x\|^2 = ||A||_2 ||x||_2 \end{aligned}$$

$||A||$ heißt *verträgliche Matrixnorm*.

Matrixnormen

Definition

Wenn $||Ax||_X \leq ||A|| \cdot ||x||_Y$, dann heißt die Matrixnorm *konsistent*.

Wir erinnern an einige wichtige Vektornormen:

Definition

$$x \in \mathbb{R}^n, ||x||_1 := \sum_{i=1}^{n} |x_i| \text{ heißt } l_1\text{- Norm or Summennorm}$$

$$x \in \mathbb{R}^n, ||x||_p := (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} \text{ heißt } l_p\text{-Norm}$$

$$x \in \mathbb{R}^n, ||x||_\infty := \max_{i=1,,,n} |x_i| \text{ heißt } l_\infty\text{-Norm oder Supremumnorm}$$

Wir studieren den Fall $X = l_1$ und versuchen, einen Raum Y zu finden, so daß wir mit einer geeigneten Definition einer Matrixnorm eine konsistente Matrixnorm erhalten.

Matrixnormen

Beispiel

$$\begin{aligned}\|Ax\|_1 &= \sum_i \left| \sum_j a_{ij} x_j \right| \leq \sum_i \sum_j |a_{ij}||x_j| \leq \sum_i \sum_j |a_{ij}| \max_j |x_j| \\ &\leq \sum_i \sum_j |a_{ij}| \|x\|_\infty = \|A\|_G \|x\|_\infty\end{aligned}$$

Bemerkung

Üblicherweise gibt es verschiedene konsistente Matrixnormen.

Hausaufgabe: Man finde eine geeignete Matrixnorm $||A||$, so daß

$$\|Ax\|_\infty \leq \|A\|_? \|x\|_1$$

Matrixnormen und Konditionszahlen

Wir betrachten das lineare Gleichungssystem $Ax = b$, $A_{(n,n)}$, $b \in \mathbb{R}^n$, nehmen an, daß A^{-1} existiert undbetrachten das gestörte System

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

Weiter wird vorausgesetzt, daß $\|\delta A\|$ "klein" ist, so daß auch $(A + \delta A)^{-1}$ existiert. Wir suchen eine Darstellung des absoluten Fehlers der unbekannten Lösung x.

$$\begin{aligned}
(A + \delta A)(x + \delta x) &= Ax + (\delta A)x + A(\delta x) + \delta A \delta x \\
(A + \delta A)(x + \delta x) &= b + \delta b \\
\cancel{Ax} + (\delta A)x + A(I + A^{-1}\delta A)\delta x &= \cancel{b} + \delta b
\end{aligned}$$

$$\Longrightarrow \qquad \boxed{\delta x = (I + A^{-1}\delta A)^{-1} A^{-1}(\delta b - (\delta A)x)}$$

$$\Longrightarrow \quad \|\delta x\| \leq \|(I + A^{-1}\delta A)^{-1}\| \, \|A^{-1}\|(\|\delta b\| + \|\delta A\| \, \|x\|)$$

$$\Longrightarrow \text{relativer Fehler } \frac{\|\delta x\|}{\|x\|} \leq \|(I + A^{-1}\delta A)^{-1}\| \, \|A^{-1}\| \left(\frac{\|\delta b\|}{\|x\|} + \|\delta A\| \right)$$

Matrixnormen und Konditionszahlen

Diese Ungleichung muß vereinfacht werden und wir müssen dabei die unbekannte Norm der Lösung x auf der rechten Seite entfernen.

Theorem (Banach)

Sei $B_{(n,n)}$, $\|B\| < 1$, dann ist die Matrix $I - B$ regulär und es gilt

$$\|(I - B)^{-1}\| \leq \frac{1}{1 - \|B\|}$$

Der Beweis ist konstruktiv und benutzt Ideen der Theorie geometrischer Reihen. Mit der Einheitsmatrix I können wir schreiben

$$(I + B + B^2 + B^3 + ... + B^n)(I - B) = (I - B^{n+1})$$

Mit $\|B^n\| \leq \|B\|^n \to 0$ für $n \to \infty$ erhalten wir

$$(\sum_{k=0}^{\infty} B^k)(I - B) = I$$

und das bedeutet, daß der zweite Faktor die Inverse des ersten ist.

Bemerkung

Das Resultat hat auch praktische Bedeutung. Für Matrizen spezieller Struktur können Partialsummen eine gute und effiziente Approximation von of $(I - B)^{-1}$ sein. Kandidaten sind z. B. schwach besetzte Matrizen (wo die Berechnung von B^n "billig" ist).

Benutzen wir wieder unser Resultat

$$(\sum_{k=0}^{\infty} B^k)(I - B) = I \text{ or } (\sum_{k=0}^{\infty} B^k) = (I - B)^{-1}$$

schließen wir wie folgt:

$$\|(I - B)^{-1}\| = \|(\sum_{k=0}^{\infty} B^k)\| \leq \sum_{k=0}^{\infty} \|B\|^k = \frac{1}{1 - \|B\|}$$

Das führt uns direkt zur Fehlerabschätzung in der folgenden Form

$$\|\sum_{k=0}^{N} B^k - (I - B)^{-1}\| \leq \frac{\|B\|^{N+1}}{1 - \|B\|}.$$

Matrixnormen und Konditionszahlen

Wir setzen die Abschätzung des relativen Fehlers der Lösung x unseres linearen Gleichungssystems fort.

$$\frac{\|\delta x\|}{\|x\|} \leq \|(I + A^{-1}\delta A)^{-1}\| \, \|A^{-1}\| \left(\frac{\|\delta b\|}{\|x\|} + \|\delta A\| \right)$$

Mit $\|(I + A^{-1}\delta A)^{-1}\| = \|(I - \underbrace{(-A^{-1})\delta A}_{B})^{-1}\|$ und

$\|A^{-1}\delta A\| \leq \|A^{-1}\|\|\delta A\|$ erhalten wir

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

Wir benutzen

$$x = A^{-1}b \Longleftrightarrow Ax = b \Longrightarrow \|Ax\| = \|b\|$$

$$\Rightarrow \quad \|A\| \, \|x\| \geq \|b\| \Rightarrow \|x\| \geq \frac{\|b\|}{\|A\|}$$

$$\Rightarrow \quad \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

Matrixnormen und Konditionszahlen

$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|(A^{-1}\|}{1-\|A^{-1}\|\,\|\delta A\|}\left(\frac{\|\delta b\|}{\|x\|}+\|\delta A\|\right)$ und $\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$ führen zu

$$\begin{aligned}\frac{\|\delta x\|}{\|x\|} &\leq \frac{\|A^{-1}\|}{1-\|A^{-1}\|\,\|\delta A\|}\left(\frac{\|A\|\,\|\delta b\|}{\|b\|}+\|\delta A\|\right)\\ &\leq \frac{\|A^{-1}\|\,\|A\|}{1-\|A^{-1}\|\,\|A\|\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)\end{aligned}$$

Wir sehen, daß die wesentlichen Parameter die bekannten Größen

$$\frac{\|\delta A\|}{\|A\|}, \frac{\|\delta b\|}{\|b\|} \quad \text{und das Produkt} \|A^{-1}\|\,\|A\| \textit{sind}.$$

Definition

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

heißt *Konditionszahl* der Matrix A.

Matrixnormen und Konditionszahlen

Bemerkung

MATLAB und MAPLE beinhalten vordefinierte Kommandos für die Berechnung verschiedener Matrixnormen.

Beispiel

$A = (a_{ij})_{i,j=1}^{n}, n = 1000, |\delta a_{ij}| \approx 10^{-11}, |a_{ij}| \approx 1, cond(A) = 10^5$

Das System $Ax = b$ wird mit Hilfe des Gauß-Algorithmus gelöst. Wir benötigen $\frac{n^3}{3}$ Operationen. Durch Aufsummieren der Rundungsfehler erhalten wir

$$\frac{1000^3}{3} \cdot 10^{-11} = \frac{10^9 \cdot 10^{-11}}{3} = \frac{10^{-2}}{3}.$$

Unter Benutzung der Konditionszahl von A ergibt sich

$$\|\delta A\| \leq \sqrt{\sum |\delta a_{ij}|^2} \leq 10^3 \cdot 10^{-11} = 10^{-8}$$

Matrixnormen und Konditionszahlen

$$\|A\| \approx 10^3 \Rightarrow \frac{\|\delta A\|}{\|A\|} \approx 10^{-11}, \text{ nehmen an, daß } \frac{\|\delta b\|}{\|b\|} \approx 10^{-11}$$

$$\Rightarrow \frac{\|\delta x\|}{\|x\|} \leq \frac{10^5}{1 - 10^5 \cdot 10^{-11}}(2 \cdot 10^{-11}) = 2 \cdot 10^{-6}$$

- Gleiche Matrix, gleiches Sytem, beide Abschätzungen der Fehlerfortpflanzung liefern sehr verschiedene Resultate
- Wenn z. B. eine Genauigkeit von 10^{-6} gefordert ist, dann ist das zweite Resultat ok. Im ersten Fall brauchen wir eine höhere Genauigkeit der Eingangsdaten:

$$\|\delta a_{ij}\| \leq 10^{-15}$$

(mehr Speicher, größere CPU-Zeit notwendig, ...)

Numerische Mathematik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

WS 2021/2022

Vorlesung 3

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Interpolation und Approximation

Gegeben $f : D \longrightarrow \mathbb{R}, \quad f : D \longrightarrow \mathbb{R}^n$.

- an endlich vielen Stellen $x_i \in D$ sind Funktionswerte bzw. Ableitungen von f bekannt.
- gesucht ist eine Funktion aus einer gewissen Klasse, die die Vorgaben erfüllt.

Bsp. Interpolation: $x_i \in D$, $i = 1, \ldots, n$

- Gegeben: $f(x_i) = g(x_i), \quad i = 1, \ldots, n$
  gesucht: Polynom vom Grad $n-1$ (max.) mit $P(x_i) = g_i$
- Gegeben: $f(x_i) = g_{i0}, \quad f'(x_i) = g_{i1}, \quad i = 1, \ldots, n$
  gesucht: Polynom vom Grad $2n-1$ mit
  $P(x_i) = g_{i0}, \quad P'(x_i) = g_{i1}, \quad i = 1, \ldots, n$

Bsp. Approximation:

- Funktionswerte an endlich vielen Stellen gegeben
- Daten an endlich vielen Stellen (Messwerte, fehlerbehaftet)
- gesucht: einfache Funktion aus gewisser Klasse, die im Sinne eines geeigneten Abstandes nahe den gegebenen Daten, liegt.

Interpolation und Aprroximation

Bsp.: An einigen Stellen liegen für ein Argument verschiedene Daten vor $\Longrightarrow$ keine Interpolation möglich.

$$\varphi_1, \ldots \varphi_n \; : \; D \longrightarrow \mathbb{R} \qquad \text{geg. Funktion}$$

Forderung:

$$\sum_{i=1}^{n} \alpha_1 \varphi_i(x) = \Phi(x)$$

erfüllt z.B.

$$\max_{x \in [a,b]} |f(x) - \Phi(x)| \longrightarrow \min$$

oder

$$\int_a^b |f(x) - \Phi(x)|^2 \; dx \longrightarrow \min$$

oder

$$\sum_{i=1}^{n} |f(x_i) - \Phi(x_i)|^2 \longrightarrow \min$$

Lagrange-Interpolation

$$x_i \in [a,b],\ i = 0,\ldots,n;\ x_i \neq x_i \quad \text{für} \quad i \neq i$$

$$f(x_i) = g_i,\ i = 0,\ldots,n$$

Definieren Basispolynome für Spezialfall

$$\omega_j(x_i) = \delta_{ij} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

$$\omega_i = \frac{(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_{i-1})(x-x_{i+1})\cdot\ldots\cdot(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdot\ldots\cdot(x_i-x_{i-1})(x_i-x_{i+1})\cdot\ldots\cdot(x_i-x_n)}.$$

Damit Lösung der Ausgangsaufgabe durch

$$(L_n f)(x) = \sum_{i=0}^{n} f(x_i)\,\omega_i(x).$$

Probe:

$$(L_n f)(x_k) = \sum_{i=0}^{n} f(x_i)\,\omega_i(x_k) = \sum_{i=0}^{n} f(x_i)\,\delta_{ik} = f(x_k)$$

Lagrange-Interpolation

Bsp: $y = \sin(x)$ L_4, L_{10} in $(-3, 3)$
$y = |x|$ L_4, L_{10} in $(-3, 3)$

Wir erkennen deutliche Unterschiede in der Genauigkeit $\Longrightarrow$ Fehlerabschätzung nötig.

Vorteile:

- Koeffizienten explizit bekannt
- Abhängigkeit von Daten gut erkennbar

Nachteile:

- $\{x_i\}_{i=0}^{n} \longrightarrow \{x_i\}_{i=0}^{n+1}$ erfordert völlig neue Berechnung

Newton-Interpolation

Wir arbeiten mit dem Ansatz

$$(L_n f)(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1})$$

Aus $(L_n f)(x_i) = p_i$ sind Koeffizienten $c_0, \ldots, c_n$ zu bestimmen.

$$\begin{aligned}
(L_n f)(x_0) &= c_0 = p_0 \\
(L_n f)(x_1) &= c_0 + c_1(x_1 - x_0) = p_1 \\
(L_n f)(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = p_2 \\
&\vdots \\
(L_n f)(x_n) &= c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \ldots \\
&\quad + c_n(x_n - x_0)(x_n - x_1) \cdot \ldots \cdot (x_n - x_{n-1}) = p_n
\end{aligned}$$

$$\Updownarrow$$

Newton-Interpolation

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & 0 & \dots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \dots & (x_n - x_0) \cdot \dots \cdot (x_n - x_{n-1}) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}$$

Gleichungssystem bei verschiedenen Stützstellen für beliebige Vorgaben stets eindeutig lösbar!

Vorteil: Hinzunahme einer neuen Stützstelle bedeuet Addition einer Gleichung, praktisch ist diese neue Gleichung nach den neuen unbekannten Koeffizienten aufzulösen (1 lineare Gleichung).

Bemerkung: Lagrange-Polynom und Newton-Polynom stimmen bei gleichen Vorgaben $(x_i, y_i)_{i=0}^n$ überein. Unterschiedlich ist nur die Berechnung der Koeffizienten und die "Handhabbarkeit".

Interpolation - Beispiele

$$y = |x|, n = 4, \quad x_0 = -2, x_1 = -1, x_2 = 0, x_3 = 1, x_4 = 2$$
$$y_0 = 2, \quad y_1 = 1, \quad y_2 = 0, y_3 = 1, y_4 = 2$$

Lagrange-Interpolation

$$\begin{aligned}(L_4 y)(x) &= 2\frac{(x+1)x(x-1)(x-2)}{(-1)(-2)(-3)(-4)} + 1\frac{(x+2)x(x-1)(x-2)}{1(-1)(-2)(-3)} \\ &\quad +1\frac{(x+2)(x+1)x(x-2)}{4\cdot 3\cdot 1(-1)} + 2\frac{(x+2)(x+1)x(x-1)}{4\cdot 3\cdot 2\cdot 1}\end{aligned}$$

Newton-Interpolation

$$\begin{aligned}(L_4 y)(x) &= c_0 + c_1(x+2) + c_2(x+2)(x+1) + c_3(x+2)(x+1)x + \\ &+ c_4(x+2)(x+1)x(x-1)\end{aligned}$$

Wir bestimmen die Koeffizienten sukzessive.

Fortsetzung Interpolation - Beispiele

$$
\begin{aligned}
c_0 &= 2 \\
c_0 + c_1 = 1 \quad &\Longrightarrow c_1 = 1 \\
c_0 + 2c_1 + 2c_2 = 0 \quad &\Longrightarrow c_2 = 0 \\
c_0 + 3c_1 + 6c_2 + 6c_3 = 1 \quad &\Longrightarrow c_3 = \frac{1}{3} \\
c_0 + 4c_1 + 12c_2 + 24c_3 + 24c_4 = 2, \quad &\Longrightarrow c_4 = -\frac{1}{6}
\end{aligned}
$$

$$(L_4 y)(x) = 2 - (x+2) + \frac{1}{3}(x+2)(x+1)x - \frac{1}{6}(x+2)(x+1)x(x-1)$$

Probe auf Übereinstimmung:

$$(L_4 y)(x) = \frac{7}{6}x^2 - \frac{1}{6}x^4$$

Aufwandsdiskussion:

- Lagrange $\longrightarrow n^2 + n$
- Newton $\longrightarrow 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{n^2}{2}$

Steigungen bzw. dividierte Differenzen

Hilfreich für Berechnung der Koeffizienten im Nowton-Polynom bzw. zur Fehleranschätzung.

Die rekursive Definition ist motiviert durch Wachstumsbeobachtungen bei Polynomen.

Bsp: $p(x) = x^3 + 2x^2 - 3x + 4$

x	$p(x)$				
0	4	0	10	6	0
1	4	10	16	6	0
1	14	26	22	6	0
3	40	48	28	6	
4	88	76	34		
5	164	110			
6	274	↓			

Differenzen von Funktionswerten

↓

Differenzen der Differenzen der Funktionswerte

Steigungen bzw. dividierte Differenzen

Definition

$[x_i, x_{i+1}, \ldots, x_{i+r}]$ heißt Steigung r-ter Ordnung bzgl. (x_i, y_i), wobei

$$[x_i] = y_i, r = 0, i = 0, \ldots, n \qquad \text{(Steigung 0-ter Ordnung)}$$

$$[x_i, x_{i+1}] = \frac{[x_{i+1}] - [x_i]}{x_{i+1} - x_i} \qquad \text{(Steigung 1-ter Ordnung)}$$

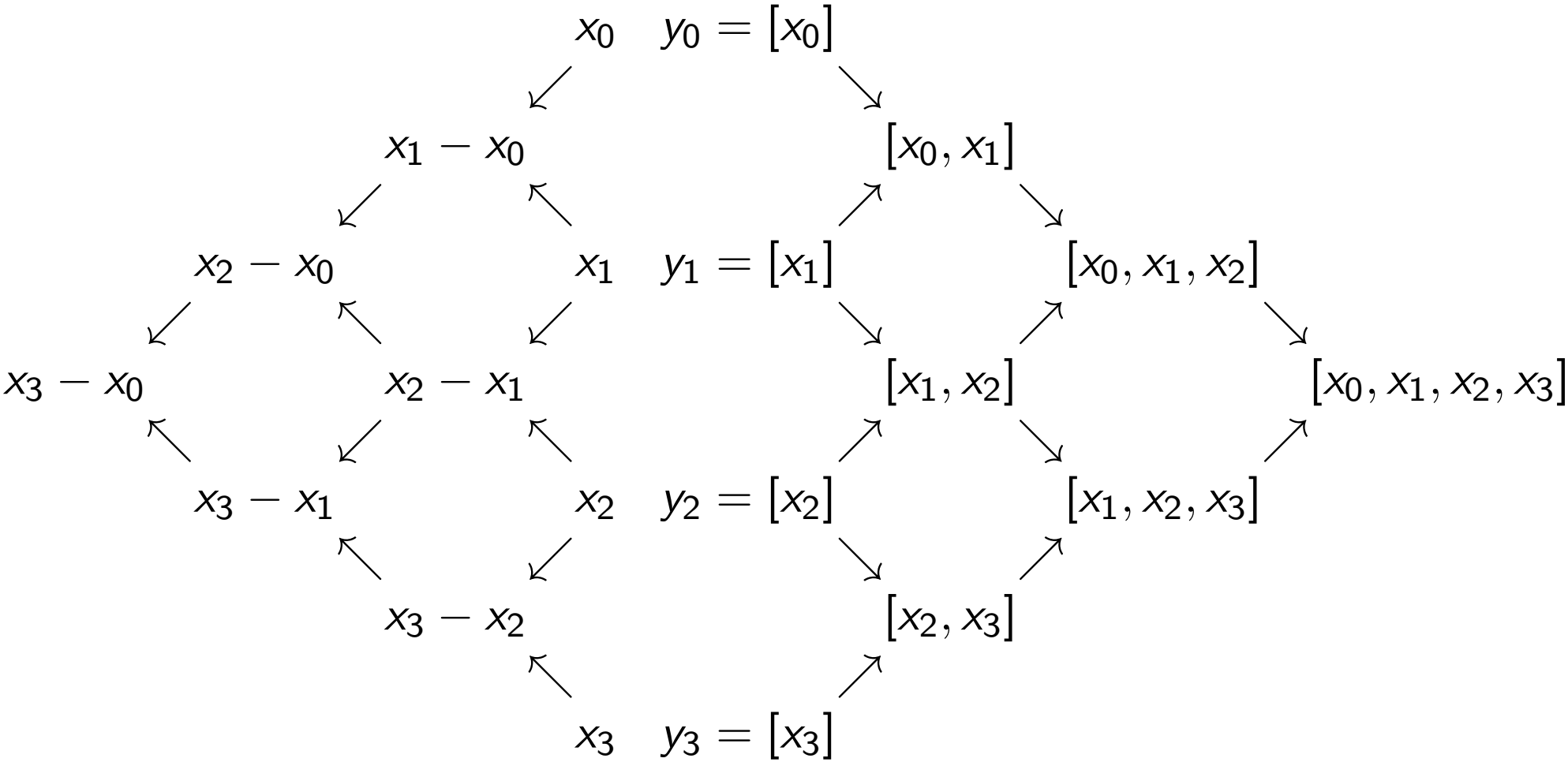
$$\vdots$$

$$[x_i, x_{i+1}, \ldots, x_{i+r}] = \frac{[x_{i+1}, \ldots, x_{i+r}] - [x_i, \ldots, x_{i+r-1}]}{x_{i+r} - x_i}$$

$$\text{für } r = 1, \ldots, n; i = 0, \ldots, n - r$$

Steigungsspiegel

Formalisierung durch Steigungsspiegel (wichtig auch für Programmierung)



Beispiel Steigungsspiegel

$$p(x) = \frac{7}{6}x^2 - \frac{1}{6}x^4, x_0 = -2, x_1 = -1, x_2 = 0, x_3 = 1, x_4 = 2$$

Man berechne die Steigungen:

$$\begin{array}{lll} [x_0] = 2 & [x_0, x_1] = \frac{-1}{1} = -1 & [x_0, x_1, x_2] = \frac{[x_1, x_2] - [x_0, x_1]}{x_2 - x_0} = 0 \\ [x_1] = 1 & [x_1, x_2] = \frac{-1}{1} = -1 & [x_1, x_2, x_3] = \frac{[x_2, x_3] - [x_1, x_2]}{x_3 - x_1} = 1 \\ [x_2] = 0 & [x_2, x_3] = \frac{1}{1} = 1 & [x_2, x_3, x_4] = \frac{[x_3, x_4] - [x_2, x_3]}{x_4 - x_2} = 0 \\ [x_3] = 1 & [x_3, x_4] = \frac{1}{1} = 1 & \\ [x_4] = 2 & & \end{array}$$

$$[x_0, x_1, x_2, x_3] = \frac{[x_1, x_2, x_3] - [x_0, x_1, x_2]}{x_3 - x_0} = \frac{1}{3}$$

$$[x_1, x_2, x_3, x_4] = \frac{[x_2, x_3, x_4] - [x_1, x_2, x_3]}{x_4 - x_1} = -\frac{1}{3}$$

$$[x_0, x_1, x_2, x_3, x_4] = \frac{[x_1, x_2, x_3, x_4] - [x_0, x_1, x_2, x_3]}{x_4 - x_0} = -\frac{1}{6}$$

Fortsetzung Beispiel

Wir beobachten, dass

$$[x_0] = 2, [x_0, x_1] = -1, [x_0, x_1, x_2] = 0, [x_0, x_1, x_2, x_3] = \frac{1}{3},$$

$$[x_0, x_1, x_2, x_3, x_4] = -\frac{1}{6}$$

gerade die Koeffizienten des Newton-Polynoms sind.

Hausaufgabe: Nachweis, dass das immer richtig ist.

$$[x_0, \ldots, x_r] = c_r \qquad r = 0, \ldots, n$$

Numerische Mathematik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

WS 2021/2022

Vorlesung 4-5

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Fehlerabschätzung Newton-Interpolation

Betrachten nun Steigungen einer differenzierbaren Funktion f mit

$$f(x_i) = y_i$$

$$[x_i; f] = f(x_i)$$

$$[x_i, x_{i+1}; f] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = f'(\xi)$$

$$[x_i, x_{i+1}, x_{i+2}; f] = \frac{[x_{i+1}, x_{i+2}] - [x_i, x_{i+1}]}{x_{i+2} - x_i} =$$

$$= \left[\frac{f(x_{i+2}) - f(x_{i+1})}{x_{i+2} - x_{i+1}} - \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}\right] \frac{1}{x_{i+2} - x_i} = (*)$$

$$= \left[\frac{f(x_{i+2}) - f(x_{i+1})}{h} - \frac{f(x_{i+1}) - f(x_i)}{h}\right] \frac{1}{2h}$$

unter der Voraussetzung

$$(x_{i+1} - x_i = h).$$

Fehlerabschätzung Newton-Interpolation

$$
\begin{aligned}
(*) &= \frac{1}{2h^2}\left[f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)\right] = \\
&= \frac{1}{2h^2}\left[f(x_{i+1}) + hf'(x_{i+1}) + \frac{h^2}{2}f''(x_{i+1}) + \frac{h^3}{6}f'''(x_{i+1}) + \frac{h^4}{24}f^{(4)}(\xi_+)\right. \\
&\qquad - 2f(x_{i+1}) \\
&\qquad \left. + f(x_{i+1}) - hf'(x_{i+1}) + \frac{h^2}{2}f''(x_{i+1}) - \frac{h^3}{6}f'''(x_{i+1}) + \frac{h^4}{24}f^{(4)}(\xi_-)\right] \\
&= \frac{1}{2}f''(x_{i+1}) + o(h^2) = \cdots = \frac{1}{2}f''(\xi)
\end{aligned}
$$

allgemein:

$$[x_i, x_{i+1}, \ldots, x_{i+r}; f] = \frac{1}{r!}f^{(r)}(\xi) \quad \text{für} \quad f \in C^{(r)}[a, b]$$

Versuchen, diese Resultate für eine Fehlerabschätzung auszunutzen.

Fehlerabschätzung Newton-Interpolation

Bemerkung

Diese Aussage begründet, warum Steigungen $(r+1)$-ter Ordnung eines Polynoms r-ter Ordnung verschwinden müssen.

Problem:

$$(x_i, y_i)_{i=0}^n, y_i = f(x_i) \Rightarrow (L_n f)(x)$$
$$\Longrightarrow r_n(x) = (L_n f)(x) - f(x) \qquad \text{Integrationsfehler}$$
$$r_n(x_i) = 0 \qquad i = 0, \ldots, n \qquad \text{nach Konstruktion}$$

Frage: Wie verhält sich $r_n(x)$ außerhalb der Stützstellen?

konkreter:

$$x = \bar{x} \in [x_0, x_n], \bar{x} \neq x_0, \ldots, x_n, \bar{x} := x_{n+1} \text{ formal bezeichnet}$$

Betrachten L_{n+1} bzgl. aus Stützstellen $\{x_0, x_1, \ldots, x_n, \bar{x}\}$ und stellen Zusammenhang zu L_n her.

Fehlerabschätzung Newton-Interpolation

Newton:

$$(L_{n+1}f)(x) = (L_n f)(x) + c_{n+1}\underbrace{(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_n)}_{b_{n+1}(x)}$$

$$\begin{aligned} c_{n+1} &= c_{n+1}(\bar{x}) = [x_0, x_1, \ldots, x_n, \bar{x}; f] \\ b_{n+1}(x) &= (x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_n) \\ x = \bar{x} \Longrightarrow (L_{n+1}f)(\bar{x}) &= (L_n f)(\bar{x}) + c_{n+1}(\bar{x})b_{n+1}(\bar{x}) = f(\bar{x}) \\ \Longrightarrow f(\bar{x}) &= (L_n f)(\bar{x}) + R_n(\bar{x}) \qquad \text{mit} \end{aligned}$$

$$x = \bar{x} \Longrightarrow L_{n+1} = L_n(\bar{x}) + c_{n+1}(\bar{x})b_{n+1}(\bar{x}) = f(\bar{x})$$
$$\Longrightarrow f(\bar{x}) = L_n(\bar{x}) + R_n(\bar{x}) \qquad \text{mit}$$

$$R_n(\bar{x}) = f(\bar{x}) - L_n(\bar{x}) = c_{n+1}(\bar{x})b_{n+1}(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}b_{n+1}(\bar{x})$$

Fehlerabschätzung Newton-Interpolation

also gilt

$$|f(\bar{x}) - L_n(\bar{x})| \leq \frac{1}{(n+1)!} \max_{x \in [x_0, x_n]} \left| f^{(n+1)}(\xi) \right| \; |b_{n+1}(\bar{x})|$$

möglich wäre auch globale Abschätzung von $|b_{n+1}(\bar{x})|$

Beispiel

$f(x) = \sin x, \; x \in [-\pi, \pi]$, *Fehlertoleranz* $\leq 10^{-6}$, *Frage* $n \geq ?$

$$\begin{aligned} |f(x) - L_n(x)| &\leq \frac{1}{(n+1)!} \max_{x \in [-\pi, \pi]} \left| \sin^{(n+1)}(\xi) \right| \; |b_{n+1}(x)| \\ \max_{x \in [-\pi, \pi]} |f(x) - L_n(x)| &\leq \frac{1}{(n+1)!} \max_{x \in [-\pi, \pi]} |b_{n+1}(x)| \\ b_{n+1}(x) &= (x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_n) \end{aligned}$$

Annahme: gleichmäßige Unterteilung von $[-\pi, \pi]$

$|b_{n+1}(x)| = |x - x_0| \cdot |x - x_1| \cdot \ldots \cdot |x - x_n|$, $|x - x_i| \leq 2\pi$

Fehlerabschätzung Newton-Interpolation

$|\sin x - (L_n f)(x)| \leq \frac{1}{(n+1)!}(2\pi)^{n+1} \leq 10^{-6}$ (?)

$n \geq 26 \Rightarrow |\sin x - (L_n f)(x)| \leq 10^{-6}$

Verbesserung

$|x - x_0| \leq h$, $|x - x_1| \leq h$, $|x - x_2| \leq 2h$,...,$|x - x_n| \leq n \cdot h$,

mit $h = \frac{2\pi}{n} \Rightarrow |b_{n+1}(x)| \leq n! h^{n+1} = n! \frac{(2\pi)^{n+1}}{n^{n+1}}$ und es gilt

$|\sin x - (L_n f)(x)| \leq \frac{1}{(n+1)!} n! \frac{(2\pi)^{n+1}}{n^{n+1}} = \frac{1}{(n+1)} \frac{(2\pi)^{n+1}}{n^{n+1}} \leq 10^{-6}$, falls $n \geq 15$

Interpolationsfehler

Weitere Verbesserung: variable Stützstellen

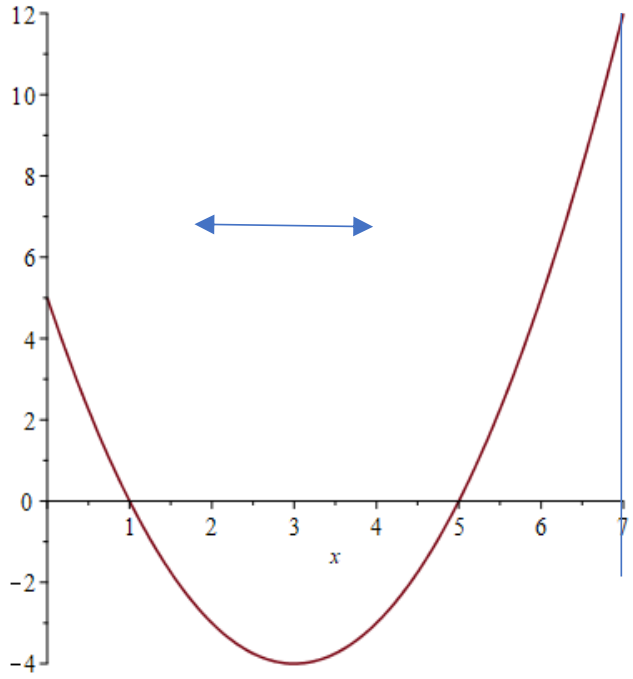
$F(x_0, ..., x_n) := \max_{x \in [x_0, x_n]} |b_{n+1}(x)|$

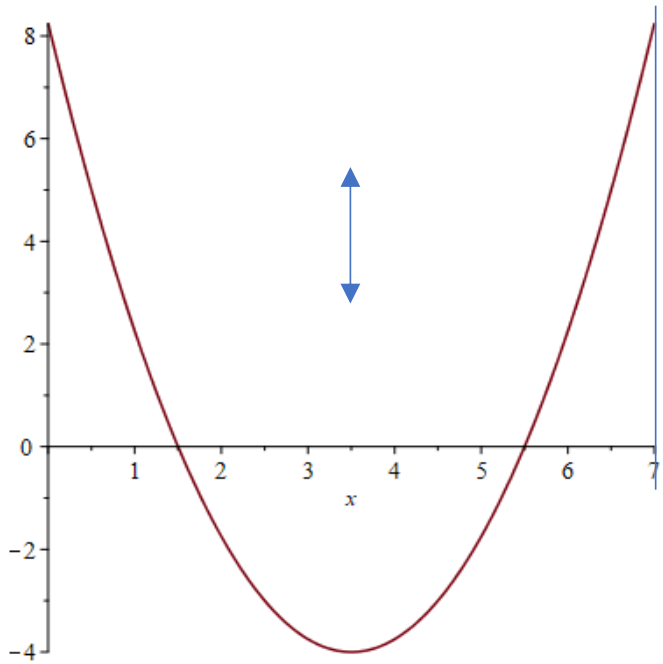
Gesucht: $\min_{(x_0, ..., x_n)} F(x_0, ..., x_n)$

Beispiel

$n = 1$

$b_2(x) = (x - x_0)(x - x_1) \qquad |b_2(x)| = |(x - x_0)(x - x_1)|$





Interpolationsfehler

$b_2(\frac{x_0+x_1}{2}) = -\frac{1}{4}(x_0 - x_1)^2 \qquad \left|b_2(\frac{x_1+x_0}{2})\right| = \frac{1}{4}(x_1 - x_0)^2$

$b_2(a) = (a - x_0)(a - x_1), \qquad b_2(b) = (b - x_0)(b - x_1)$
$|b_2(a)| = (x_0 - a)(x_1 - a), \qquad |b_2(b)| = (b - x_0)(b - x_1)$
für welche x_0, x_1 ist $\left|b_2(\frac{x_1+x_0}{2})\right| = |b_2(a)| = |b_2(b)|$?

allgemeines Resultat:

$$x_i = \frac{b + a + t_i(b - a)}{2}$$

$$t_i = \cos\left(\frac{2i + 1}{n + 1} \cdot \frac{\pi}{2}\right), \quad i = 0, ..., n$$

$\Rightarrow \max_{x\in[a,b]} |b_{n+1}(x)| = \frac{(b-a)^{n+1}}{2^{2n+1}}$

für Bsp: $a = -\pi, \quad b = \pi, \quad f = \sin x, \quad \delta \leq 10^{-6} \quad \Rightarrow n \geq 11$

Numerik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

Wintersemester 2021/2022

Vorlesung 6

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Hermite-Interpolation

$x_i, \quad i = 0, ..., n$, gesucht Polynom $P(x)$ mit den Eigenschaften $P(x_i) = f(x_i),\ P'(x_i) = f'(x_i)$ für $i = 0, ..., n$.

Annahmen: $f \in C^1[a, b],\ x_0, ..., x_n$ Stützstellen, $x_i \neq x_j$, für $i \neq j$

gesucht: Polynom H (minimalen Grades) mit

$$H(x_i) = f(x_i), i = 0, ..., n$$

$$H'(x_i) = f'(x_i), i = 0, ..., n$$

$\Rightarrow H := H_{2n+1}$

Idee: suchen Basis-Polynome:

$$H_{n,j}(x_k) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases} \quad \text{und } H'_{n,j}(x_k) = 0\ \forall k$$

$$\tilde{H}_{n,j}(x_k) = 0\ \forall k \qquad \text{und } \tilde{H}'_{n,j}(x_k) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases}$$

falls solche Polynome existieren, ist die Aufgabe gelöst:

$$H_{2n+1}(x) = \sum_{i=0}^{n} f(x_i) H_{n,j}(x) + \sum_{i=0}^{n} \tilde{H}_{n,j}(x) f'(x_i)$$

Hermite-Interpolation

Konstruktion:

$$\tilde{H}_{n,j}(x_k) = 0 \quad \forall k, \text{ und } \tilde{H}'_{n,j}(x_k) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases} \text{ bedeutet,}$$

daß $\tilde{H}_{n,j}$ an den Stellen $x_k \neq x_j$ (mindestens) doppelte Nullstellen haben muß und an der Stelle $x = x_j$ eine einfache Nullstelle besitzen muß.

$$\begin{aligned} \tilde{H}_{n,j}(x) &= \frac{(x-x_0)^2(x-x_1)^2 \ldots (x-x_{j-1})^2 (x-x_j)(x-x_{j+1})^2 \ldots (x-x_n)^2}{(x_j-x_0)^2(x_j-x_1)^2 \ldots (x_j-x_{j-1})^2 \cdot (x_j-x_{j+1})^2 \ldots (x_j-x_n)^2} \\ &= (x-x_j)(L_{n,j}(x))^2 \end{aligned}$$

Überprüfen Ableitung:

$$\tilde{H}'_{n,j} = (L_{n,j}(x))^2 + 2(x-x_j)L_{n,j}(x)L'_{n,j}(x) \Rightarrow$$

$$\tilde{H}'_{n,j}(x_k) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases}$$

Hermite-Interpolation

$H_{n,j}(x) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases}$ und $H'_{n,j}(x_k) = 0 \ \forall k$ bedeutet, daß

$H_{n,j}(x)$ für $x = x_k, \ k \neq j$ doppelte Nullstellen besitzt.

$$\begin{aligned} H_{n,j}(x) &= \frac{(x-x_0)^2 \ldots (x-x_{j-1})^2 (x-x_{j+1})^2 \ldots (x-x_n)^2}{(x_j-x_0)^2 \ldots (x_j-x_{j-1})^2 \cdot (x_j-x_{j+1})^2 \ldots (x_j-x_n)^2} (\alpha x + \beta) \\ &= (L_{n,j}(x))^2 (\alpha x + \beta) \end{aligned}$$

$H_{n,j}(x_j) = 1 \Rightarrow \alpha x_j + \beta = 1.$

$$H'_{n,j}(x_j) = 0 \Rightarrow \underbrace{(L_{n,j}(x_j))^2}_{1} \alpha + 2 \underbrace{L_{n,j}(x_j)}_{1} L'_{n,j}(x_j) \underbrace{(\alpha x_j + \beta)}_{1} = 0 \Rightarrow$$

$$\alpha = -2L'_{n,j}(x_j), \ \beta = 1 - \alpha x_j, \ \beta = 1 + 2L'_{n,j}(x_j) x_j$$

$$\alpha x + \beta = -L'_{n,j}(x_j) x + 1 + 2L'_{n,j}(x_j) x_j = 1 - 2(x - x_j) L'_{n,j}(x_j)$$

$$H_{n,j}(x) = (1 - 2(x - x_j) L'_{n,j}(x_j))(L_{n,j}(x))^2$$

Abstrakte Interpolation

Definition (Lineares Funktional)

$L : X \to \mathbb{R}$, X linearer Vektorraum, heißt lineares Funktional, wenn $L(\alpha f + \beta g) = \alpha L(f) + \beta L(g) \quad \forall \alpha, \beta \in \mathbb{R}, \forall f, g \in X$

Beispiel

$$X = C[a, b], \quad Lf := \int_a^b f(x)dx$$

Beispiel

$$X = C[a, b], \quad L_j f := f(x_j)$$

Beispiel

$X = C[-\pi, \pi]$ *oder* $X = L_2[-\pi, \pi]$

$$L_{2n} f := \int_a^b \cos(nx) f(x)dx$$

$$L_{2n+1} f := \int_a^b \sin(nx) f(x)dx$$

Fourier-Koeffizienten sind auch lineare Funktionale

Abstrakte Interpolation

Sei X ein linearer Vektorraum über $\mathbb{R}$ der Dimension n und $L_1, ..., L_n$ seien lineare Funktionale. Für gegebene Werte $g_1, ..., g_n \in \mathbb{R}$ suchen wir ein $x \in X$ mit den Eigenschaften

$$L_k(x) = g_k, k = 1, ..., n.$$

Sei $x_1, ..., x_n$ eine Basis in X (nicht eindeutig)$\Rightarrow$

$$x = \sum_{j=1}^{n} \alpha_j x_j \Rightarrow$$

$$L_i(x) = \sum_{j=1}^{n} L_i(x_j)\alpha_j = g_i$$

$$\begin{pmatrix} L_1(x_1) & L_1(x_2) & ... & L_1(x_n) \\ L_2(x_1) & L_2(x_2) & ... & L_2(x_n) \\ ... & ... & ... & ... \\ L_n(x_1) & L_n(x_2) & ... & L_n(x_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ ... \\ g_n \end{pmatrix}$$

Dieses System ist eindeutig lösbar, wenn $\det((L_i(x_j)) \neq 0$

Abstrakte Interpolation

Beispiel

$f_1(x) = \cos x,\ f_2(x) = \sin x,\ x \in [-\pi, \pi],\ f \in C[-\pi, \pi],$

$x_1, x_2 \in [-\pi, \pi],\ x_1 \neq x_2$

$$T_2(x) = \alpha \cos x + \beta \sin x$$

$$T_2(x_i) = g_i,\ i = 1, 2 \Rightarrow \begin{pmatrix} \cos x_1 & \sin x_1 \\ \cos x_2 & \sin x_2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$$

$$\det \begin{pmatrix} \cos x_1 & \sin x_1 \\ \cos x_2 & \sin x_2 \end{pmatrix} = \cos x_1 \sin x_2 - \sin x_1 \cos x_2 = \sin(x_2 - x_1) = 0$$

falls $x_2 - x_1 = \pi$

oder $x_1 = -\pi,\ x_2 = \pi$

Dieses Interpolationsproblem ist nicht immer eindeutig lösbar. (Die Funktionen $\sin x$ und $\cos x$ sind orthogonal auf $[-\pi, \pi]$ und damit auch linear unabhängig!)

Abstrakte Interpolation

Beispiel

$X = span\{1, x^2\}, x \in [-1, 1], x_1, x_2 \in [-1, 1], x_1 \neq x_2,$
$L_1(f) = f(x_1)$, $L_2(f) = f(x_2)$, f *gerade Funktion*
$P_2(x) = a_0 + a_2 x^2$

$$\det \begin{pmatrix} 1 & x_1^2 \\ 1 & x_2^2 \end{pmatrix} = x_2^2 - x_1^2 = (x_2 + x_1)(x_2 - x_1) = 0,$$

wenn $x_2 = -x_1$, Interpolationsproblem nicht lösbar.

Beispiel

$X = span\{1, x, x^2\}, x \in [-1, 1], x_1 < 0, x_3 > 0, x_2 = 0$

$$\det \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & 0 & 0 \\ 1 & x_3 & x_3^2 \end{pmatrix} = x_1^2 x_3 - x_1 x_3^2 = x_1 x_3 (x_3 - x_1) \neq 0,$$

Interpolationsproblem lösbar.
Spezialfall: $x_3 = -x_1$, für gerade Funktion
$L_2(f)(x) = f_2 + 0x + \frac{f_1 - f_2}{x_1^2} x^2 \Rightarrow$
gerade Funktion liefert gerades IP-Polynom (vgl. voriges Beispiel)

Trigonometrische Interpolation

Idee für trigonometrische Interpolation:

$1, \cos x, \cos nx, \sin x, \ldots, \sin nx$

Wir suchen:

$(T_n f)(x) = \alpha_0 + \sum_{k=1}^{n} \alpha_k \cos(kx) + \sum_{k=1}^{n} \beta_k \sin(kx)$

mit $(T_n f)(x_j) = f(x_j)$ in $[0, 2\pi)$, $x_l \neq x_j$ für $l \neq j$, $\quad j = 0, \ldots, 2n$

Untersuchen Determinante $\det(L_k(x_j))$

$$\det \begin{pmatrix} 1 & \cos x_0 & \sin x_0 & \cos(2x_0) & \ldots & \cos(nx_0) & \sin(nx_0) \\ 1 & \cos x_1 & \sin x_1 & \cos(2x_1) & \ldots & \cos(nx_1) & \sin(nx_1) \\ 1 & \cos x_2 & \sin x_2 & \cos(2x_2) & \ldots & \cos(nx_2) & \sin(nx_2) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \cos x_{2n} & \sin x_{2n} & \cos(2x_{2n}) & \ldots & \cos(nx_{2n}) & \sin(nx_{2n}) \end{pmatrix}$$

- multiplizieren 3.,5.,7.,... Spalte mit i und addieren zur 2.,4.,6.,...
  $\Rightarrow D = \det(1, e^{ix_j}, \sin x_j, e^{2ix_j}, \sin(2x_j), \ldots, e^{inx_j}, \sin(nx_j))$
- multiplizieren 3.,5.,7.,... Spalte mit $-2i$ und add. 2. zur 3., 4. zur 5.,... und erhalten

  $(-2i)^n D = \det(1, e^{ix_j}, e^{-ix_j}, e^{2ix_j}, e^{-2ix_j}, \ldots, e^{inxj}, e^{-inx_j})$

Trigonometrische Interpolation

- multiplizieren mit e^{inx_j} und vertauschen die Spalten

  $e^{in(x_0+\ldots+x_{2n})}(-1)^{n(n+1)}(-2i)^n D = \det(1, e^{ix_j}, e^{2ix_j}, \ldots, e^{i2nx_j})$

  $= \det(1, e^{ix_j}, (e^{ix_j})^2, \ldots, (e^{ix_j})^{2n}) \neq 0$, falls $e^{ixj} \neq e^{ixl} \Leftrightarrow x_j \neq x_l$

  Die erhaltene Determinante ist eine Vandermonde-Determinate.

Numerik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

Wintersemester 2021/2022

Vorlesung 7

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Spline-Interpolation

Ziel: Interpolation bei sehr großen Datenmengen unter Vermeidung hoher Polynomgrade.

Problem: Bei der klassischen Interpolation passen dann Anzahl der Freiheitsgrade und Anzahl der Datenpunkte nicht mehr zusammen.

Idee: Stückweise polynomiale Interpolation

Annahme: $x_0 < x_1 < x_2 < ... < x_{n-1} < x_n$ Stützstellen, Knoten

Definition

Eine Splinefunktion s (oder Spline) vom Grad k ($k \geq 1$) und dem Defekt r ($0 \leq r \leq k$) ist erklärt durch:

1. In $[x_i, x_{i+1}]$ ist $s(x)$ im Polynom s_i vom Grad $\leq k$
2. Im gesamten Intervall $[x_0, x_n]$ besitzt $s(x)$ stetige Ableitungen bis zur Ordnung $k - r$ (einschließlich)

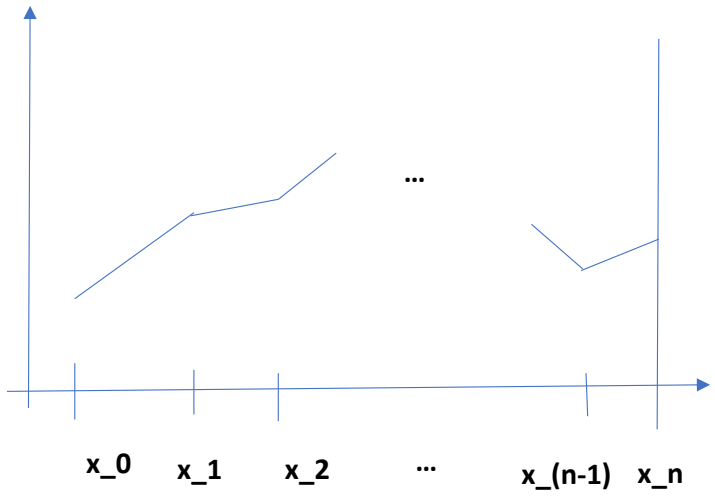
Alle diese Spline-Funktionen werden mit $S^{k,r}$ bezeichnet.

Splines sind stückweise Polynome, möglichst glatt "zusammen geheftet".

Lineare Splines

Beispiel

$k = 1, r = 1$, *Daten:* $x_k, f(x_k) = y_k$



$$\Rightarrow s(x) = \begin{cases} y_i + m_i(x - x_i), & x \in [x_i, x_{i+1}] \\ 0, & x \neq [x, x_{i+1}] \end{cases}$$

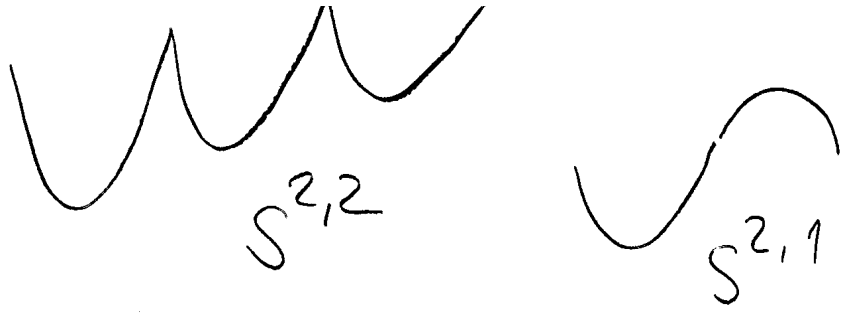
$s_i(x_i) = y_i, s_i(x_{i+1}) = s_{i+1}(x_{i+1}) = y_{i+1}$
$y_i + m_i(x_{i+1} - x_i) = y_{i+1} \Rightarrow m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$

$$\Rightarrow s_i = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \text{ oder } s_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} y_i + \frac{x - x_i}{x_{i+1} x_i} y_{i+1}$$

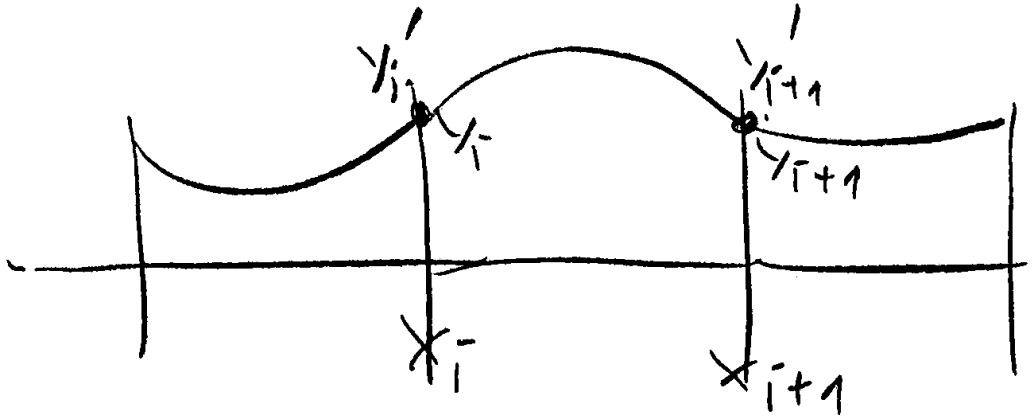
Quadratische Splines

Häufig werden Splines (für Programmierzwecke) auf einem Referenzintervall , z. B. $[0, 1]$, angegeben und bei Bedarf transformiert.

$$t = \frac{x - x_i}{x_{i+1} - x_i} \Rightarrow s_i(x_i + th_i) = (1 - t)y_i + ty_{i+1}, h_i = x_{i+1} - x_i$$

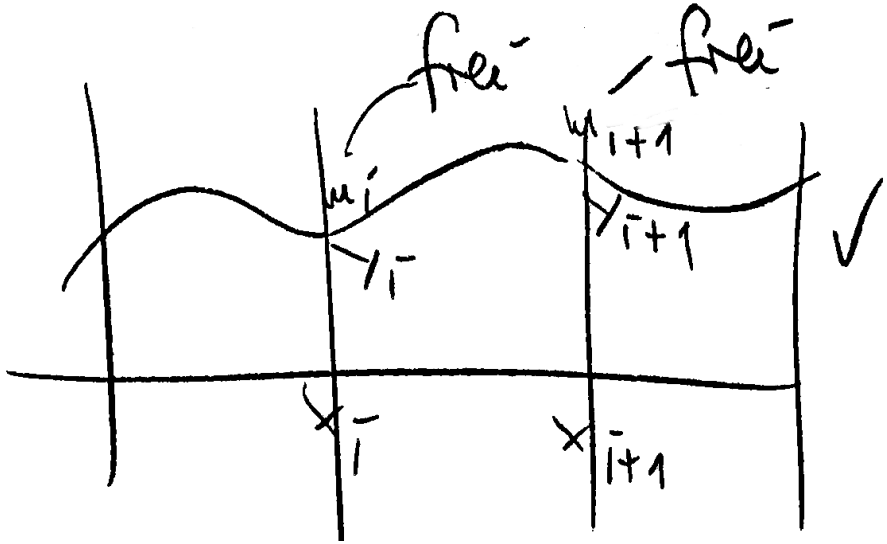


Wir suchen eine Splinefunktion, die stückweise ein Polynom 2. Grades ist und stetig und stetig differenzierbar in $[x_0, x_n]$ ist.



Quadratische Splines

Widerspruch: 4 Bedingungen $\leftrightarrow$ 3 Freiheitsgrade



4 Bedingungen, 4 Freiheitsgrade

Ansatz : $s_i(x_i + th_i) = y_i + th_i m_i + t^2(y_{i+1} - y_i - h_i m_i), \quad 0 \leq t \leq 1$

Bedingungen:

$$s_i(x_i) = y_i, s_i(x_{i+1}) = s_{i+1}(x_{i+1}) = y_{i+1}$$

$$s_i'(x_i) = m_i, s_i'(x_{i+1} - 0) = s_{i+1}'(x_{i+1} + 0) = m_{i+1}$$

$$s_i(x) = s_i(x_i + th_i), t = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_i}{h_i}, x = x_i + th_i$$

Quadratische Splines

$$\frac{ds_i}{dt}(x_i + th_i) = \frac{ds_i}{dx}\frac{dx}{dt} = \frac{ds_i}{dx}h_i \Leftrightarrow, s_i'(x) = \frac{1}{h_i}\frac{ds_i}{dt}$$

$$\begin{aligned} s_i'(x_{i+1}) &= \frac{1}{h_i}[h_i m_i + 2(y_{i+1} - y_i - h_i m_i)] = \frac{2(y_{i+1} - y_i)}{h_i} - m_i = \\ &= 2d_i - m_i, \quad i = 0, 1, ... \\ s_{i+1}'(x_{i+1}) &= m_{i+1} \end{aligned}$$

Stetigkeitsbedingung:

$$s_i'(x_i) = s_{i+1}'(x_i) \Rightarrow m_i = 2d_{i-1} + m_{i-1}, \quad i = 1, ..., n-1$$

Das heißt, dass ein Freiheitsgrad übrig ist, der für zusätzliche Bedingungen an die Splinefunktion verwendet werden kann. Die anderen Parameter ergeben sich aus einer einfachen rekursiven Vorschrift. Eine Möglichkeit ist z. B. $m_0 = y_0'$. Denkbar sind aber auch Forderungen, dass m_0 so gewählt wird, das die Kurvenlänge des Splines minimal wird (HA) oder m_0 so gesetzt wird, dass die mittlere quadratische Krümmung minimal wird.

Quadratische Splines

Wir erinnern an die Formel für die Krümmung

$$k(x) = \frac{s''(x)}{(1 + (s'(x))^2)^{\frac{3}{2}}} \approx s''(x).$$

Bedingung: $\int_{x_0}^{x_n} (s''(x))^2 dx \longrightarrow \min$

$$k(m_0) = \int_{x_0}^{x_n} s''(x)dx = \sum_{i=0}^{n-1} h_i (s_i'')^2 = 4 \sum_{i=0}^{n-1} \frac{(d_i - m_i)^2}{h_i}$$

Wir suchen: Eine explizite Formel für m_i in Abhängigkeit von m_0

$a_0 = 0, \quad a_i = d_{i-1} - a_{i-1}, \quad i = 1, ..., n-1$

$\Rightarrow m_i = 2a_i + (-1)^i m_0, \quad i = 0, 1, ..., n-1$

$$\Rightarrow k(m_0) = 4 \sum_{i=0}^{n-1} \frac{(d_i - 2a_i - (-1)^i m_0)^2}{h_i} \rightarrow \min$$

$$\text{Wir lösen } k'(m_0) = 0 \Rightarrow m_0 = \frac{\sum_{i=0}^{n-1} \frac{(-1)^i (d_i - 2a_i)}{h_i}}{\sum_{i=0}^{n-1} \frac{1}{h_i}}$$

Kubische Splines

$k = 3, \quad r = 2, \quad S^{3,2}$
$y_i = s(x_i), \quad y_{i+1} = s(x_{i+1}) = s_{i+1}(x_{i+1})$
$m_i = y_i' = s'(x_i) = s_{i-1}'(x_i), \; m_{i+1} = y_{i+1}' = s'(x_{i+1}) = s_{i+1}'(x_{i+1})$

Ansatz:
$s_i(x_i + th) =$
$(1-t)^2(1+2t)y_i + t^2(3-2t)y_{i+1} + t(1-t)h_i[(1-t)m_i - tm_{i+1}]$

Probe $\surd$, alle Bedingungen sind erfüllt. Die Werte und die ersten Ableitungen können vorgegeben werden und passen zusammen.

$k = 3, \quad r = 1, \quad S^{3,1}$
"Werte, 1. Ableitung und 2. Ableitung sollen zusammenpassen".
$\Rightarrow$ 4 Freiheitsgrade, 6 Bedingungen
$\Rightarrow$ Stetigkeit Werte, 1. und 2. Ableitung, m_i, m_{i+1} nicht vorgeben
$\Rightarrow$ 6 Freiheitsgrade, 6 Bedingungen

Kubische Splines

Bedingungen:

$s(x_i) = y_i, s(x_{i+1}) = y_{i+1}$

$s'(x_i) = s'_{i+1}(x_i) = s'_i(x_i) = m_i,$

$s'(x_{i+1}) = s'_i(x_{i+1}) = s'_{i+1}(x_{i+1}) = m_{i+1}$

$\Longrightarrow$ diese Bedingungen sind bereits vom vorigen Ansatz erfüllt, sie gelten für beliebige m_i, m_{i+1}. Die Freiheitsgrade wurden also noch nicht benutzt und können durch die geforderte Stetigkeit der zweiten Ableitungen bestimmt werden.

$$s''(x_i - 0) = s''_{i+1}(x_i) = s''_i(x_i + 0) = s''_i(x_i), i = 1, 2, ..., n-1$$

$$s''_i(x_i + th_i) = \frac{1}{h_i^2}(6(1-2t)(y_{i+1}-y_i)+2h_i((-2+3t)m_i+(-1+3t)m_{i+1})$$

Wir vergleichen $\lim_{t\to 1} s''_{i-1}(x_{i-1} + th_{i-1})$ mit $\lim_{t\to 0} s''_i(x_i + th_i)$

$$\frac{6(y_i - y_{i-1})+2h_{i-1}(m_{i-1} + 2m_i)}{h_{i-1}^2} = \frac{6(y_{i+1} - y_i)-2h_i(m_i + 2m_{i+1})}{h_i^2}, i=1, ..., n-1$$

Kubische Splines

Umstellen:

$$v_i m_{i-1} + 2m_i + u_i m_{i+1} = 3(v_i d_{i-1} + u_i d_i), i = 1, ..., n-1$$

$$u_i = \frac{h_{i-1}}{h_{i-1} + h_i}, v_i = \frac{h_i}{h_{i-1} + h_i}, u_i, v_i > 0, u_i + v_i = 1$$

Lineares Gleichungssystem für $n+1$ Unbekannte mit $n-1$ Gleichungen. Zwei zusätzliche Bedingungen können formuliert werden, z.B.

$$s_0''(x_0) = s''(x_0) = 0 \quad \Rightarrow \quad 2m_0 + m_1 = 3d_0$$

$$s_{n-1}''(x_n) = s''(x_n) = 0 \quad \Rightarrow \quad m_{n-1} + 2m_n = 3d_{n-1}$$

Matrix-Vektorform des Systems

$$\begin{pmatrix} 2 & 1 & 0 & 0 & ... & 0 \\ v_1 & 2 & u_1 & 0 & ... & 0 \\ 0 & v_2 & 2 & u_2 & ... & 0 \\ \cdots & & & & & \\ 0 & \cdots & 0 & v_{n-1} & 2 & u_{n-1} \\ 0 & \cdots & 0 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ \cdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} 3d_0 \\ 3(v_1 d_0 + u_1 d_1) \\ 3(v_2 d_1 + u_2 d_2) \\ \cdots \\ \cdots \\ 3d_{n-1} \end{pmatrix}$$

Kubische Splines

Beobachten: Matrix ist schwach besetzt, Bandmatrix, Tridiagonalmatrix, Matrix ist diagonal-dominant, d.h.

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \ \forall i$$

Ein Gleichungssystem mit diagonal-dominanten Matrix ist stets eindeutig lösbar.

Bemerkung

Notwendig sind schnelle und stabile Löser (Gauß zu langsam)

Fehlerabschätzungen

Beispiel (Splines aus $S^{1,1}$ (stetig linear))

Wir verwenden die Fehlerabschätzung für das Lagrange-Polynom, eingeschränkt auf ein Teilintervall.

$$|f(x) - (L_n f)(x)| \leq \frac{1}{(n-1)!} \max_{x \in [x_0, x_n]} |f^{(n+1)}(x)||b_{n+1}(x)|$$

In unserem Fall gilt:
$n = 1, x_0 = x_i, x_n = x_{i+1}, b_2(x) = (x - x_i)(x - x_{i+1})$

$$|f(x) - S(x)| \leq \frac{1}{2} \max_{x \in [x_i, x_{i+1}]} |f''(x)| \cdot |b_2(x)|$$

$$|b_2(x)| = |(x - x_i)(x - x_{i+1})| = |x - x_i||x_i - x_{i+1}| \leq h_i^2$$

$$\Rightarrow |f(x) - S(x)| \leq \tfrac{1}{2} M_2 h_i^2, \quad \forall x \in [x_i, x_{i+1}]$$

$$\Rightarrow |f(x) - S(x)| \leq \tfrac{1}{2} M_2 h^2, \quad h = \max_i h_i$$

HA: $f(x) = \sin x, x \in [-\pi, \pi]$, $\varepsilon = 10^{-6}$ gewünschte Genauigkeit, $n =$?, Aufwandsvergleich mit $(L_{26} f)(k)$

Basis-Splines

Beispiel

Stetige lineare Splines aus $S^{1,1}$.

$$B_{1,i}(x) = \begin{cases} 1 & x = x_{i+1} \\ 0 & x = x_j,\ j \neq i+1 \end{cases}$$

$$B_{1,i}(x) = \begin{cases} \frac{x - x_i}{x_{i+1} - x_i} & x \in [x_i, x_{i+1}) \\ \frac{x_{i+2} - x}{x_{i+2} - x_{i+1}} & x \in [x_{i+1}, x_{i+2}] \\ 0 & \textit{sonst} \end{cases}$$

Setzen formal: $x_{-1} = x_0$, $x_{n+1} = x_n$

$$\Rightarrow s(x) = \sum_{i=-1}^{n-1} y_{i+1} B_{1,i}(x)$$

Numerik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

Wintersemester 2021/2022

Vorlesung 8

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Beste Approximation

Sei X ein Hilbert-Raum mit dem Skalarprodukt $(\cdot,\cdot)$, $X_n \subset X$ sei ein n-dimensionaler Unterraum, $\{x_1, ..., x_n\}$ sei eine Basis von X_n und $f \in X$ ein beliebiges Element.

Wir suchen $f_n \in X_n$ mit

$$||f_n - f||_x = \min_{\varphi \in X_n} ||\varphi - f||_X.$$

f_n heißt dann beste Approximation von f in X_n.

$$\varphi = \sum_{k=1}^{n} a_k x_k$$

$||\varphi - f|| \longrightarrow \min$ bedeutet, dass $(\varphi - f, \varphi - f) = ||\varphi - f||^2 \to \min$

$(\sum_{i=1}^{n} a_i x_i - f, \sum_{j=1}^{n} a_j x_j - f) =$

$$\underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j (x_i, x_j) - 2\sum_{i=1}^{n} a_i (x_i, f) + ||f||^2}_{F(a_1,...,a_n) \longrightarrow \min} \longrightarrow \min$$

Beste Approximation

Notwendige Bedingung für Extrema:

$$\frac{\partial F(a_1, ..., a_n)}{\partial a_k} = 0, \quad k = 1, ..., n$$

$$\frac{\partial}{\partial a_k}(\sum_i \sum_j a_i a_j(x_i, x_j) - 2\sum_i a_i(x_i, f)) = 0, \quad k = 1, ..., n$$

$$\sum_i \sum_j (\delta_{ik} a_j(x_i, x_j) + a_i \delta_{jk}(x_i, x_j)) - 2\sum_i \delta_{ik}(x_i, f) = 0, \ k = 1, ..., n$$

$$\sum_j a_j(x_k, x_j) + \sum_i a_i(x_i, x_k) - 2(x_k, f) = 0, \ k = 1, 2, ...$$

$$2\sum_j a_j(x_j, x_k) - 2(f, x_k) = 0, \ k = 1, 2, ...$$

$$\sum_{j=1}^{n} (x_j, x_k) a_j = (f, x_k), \ k = 1, 2, ..., n$$

Beste Approximation

$$\begin{pmatrix} (x_1, x_1) & (x_2, x_1) & \dots & (x_n, x_1) \\ \dots & & & \\ (x_1, x_n) & (x_2, x_n) & \dots & (x_n, x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} (f, x_1) \\ \dots \\ (f, x_n) \end{pmatrix} \Leftrightarrow$$

$$G \cdot a = g$$

Wenn $\{x_1, \dots, x_n\}$ ONS

$\Rightarrow G = I \Rightarrow a = g$

$\Rightarrow a_i = (f, x_i)$ Fourier-Koeffizienten.
G heißt Gramsche Matrix.

HA: Man zeige, dass die Matrix G regulär ist, wenn $x_1, \dots, x_n$ linear unabhängig sind.

Methode der kleinsten Fehlerquadrate

Gegeben: $(x_i, y_i), i = 1, ..., n, x_i \neq x_j$ für $i \neq j$ nicht zwingend vorausgesetzt

Ansatzfunktionen: $\varphi_j(x), \quad j = 1, ..., m \quad m \neq n$ möglich (i.a. $n >> m$)

Bilden Linearkombination $f_{a_1 ... a_m}(x) = \sum_{j=1}^{m} a_j \varphi_j(x)$

Gesucht:

$$a_1, ..., a_m : \sum_{i=1}^{n} |f_{a_1 a_2 ... a_m}(x_i) - y_i|^2 \xrightarrow[a_1,...,a_m]{} \min$$

Wir definieren $F(a_1, ..., a_m) = \sum_{i=1}^{n} (\sum_{j=1}^{m} a_j \varphi_j(x) - y_i)^2$

Notwendige Bedingung für Extremum $\frac{\partial F}{\partial a_k} = 0, k = 1, ..., m$

$$\frac{\partial F}{\partial a_k} = \frac{\partial}{\partial a_k} \sum_{i=1}^{n} (\sum_{j=1}^{m} a_j \varphi_j(x_i) - y_i)^2 = 0$$

Methode der kleinsten Fehlerquadrate

$$\sum_{i=1}^{n} 2(\sum_{j=1}^{m} a_j \varphi_j(x_i) - y_i)\varphi_k(x_i)) = 0$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \varphi_j(x_i)\varphi_k(x_i)a_j = \sum_{i=1}^{n} y_i\varphi_k(x_i), k = 1, ..., m$$

Definieren $A = (a_{ij})_{i=1,j=1}^{n,m}$, $a_{ij} = \varphi_j(x_i)$, $a = (a_j)_{j=1}^{m}$, $y = (y_i)_{i=1}^{n}$

$$\sum_{i=1}^{n} y_i\varphi_k(x_i) = \sum_{i=1}^{n} \varphi_k(x_i)y_i = (A^T y)_k$$

$$\sum_{j=1}^{m}\sum_{i=1}^{n} \varphi_j(x_i)\varphi_k(x_i)a_j = \sum_{j=1}^{m}\sum_{i=1}^{n} a_{ij}a_{ik}a_j = \sum_{i=1}^{n} a_{ik}(\sum_{j=1}^{m} a_{ij}a_j)$$

$$= \sum_{i=1}^{n} a_{ik}(Aa)_i = \sum_{i=1}^{n}(A^T)_{k,i}(Aa)_i = (A^T Aa)_k$$

Methode der kleinsten Fehlerquadrate

Insgesamt:

$$(A^T A)a = A^T y \quad \text{lineares Gleichungssystem}$$

$A_{(n,m)}, A^T_{(m,n)}, A^T A_{(m,m)}, (A^T y)_{(m,1)}$ quadratisches System.
$(A^T A)^T = A^T A$ ist eine symmetrische Matrix.
Frage: $\det(A^T A) \neq 0$

Bemerkung

Das Minimierungsproblem läßt sich umschreiben als:
$F(a_1, ..., a_m) = ||Aa - y||^2_{\mathbb{R}^n} \to \min.$

Beispiel (lineare Regression)

$\varphi_1(x) \equiv 1, \varphi_2(x) = x$

$$A = (\varphi_j(x_i))^{n,2}_{i=1,j=1} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ ... & \\ 1 & x_n \end{pmatrix}, \; A^T = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ x_1 & x_2 & x_3 & ... & x_n \end{pmatrix},$$

$$A^T A = \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}$$

Methode der kleinsten Fehlerquadrate

Zu lösen ist:

$$\begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix}$$

$$\det(A^T A) = n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2 =$$

$$n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)(\sum_{j=1}^n x_j) = n \sum_{i=1}^n x_i^2 - \sum_{i=1}^n \sum_{j=1}^n x_i x_j \geq \ldots$$

benutzen: $(x_i - x_j)^2 \geq 0 \Rightarrow \begin{array}{l} x_i x_j \leq \frac{1}{2}(x_i^2 + x_j^2) \\ -x_i x_j \geq -\frac{1}{2}(x_i^2 + x_j^2) \end{array}$

$$\ldots \geq n \sum_{i=1}^n x_i^2 - \underbrace{\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (x_i^2 + x_j^2)}_{n \sum x_i^2} = 0$$

$\det(A^T A) > 0$, falls nicht alle $x_i = x_1$ sind.

Numerische Differentiation

Situation: $y = f(x), a \leq x \leq b$, ist nur in den Punkten x_i bekannt:
$x_i, y_i, i = 1, ..., N$
Gesucht sind Näherungen für $f'(x_i)$,..., $f^{(k)}(x_i)$
Grundlegende Strategien:

- Wir betrachten Ableitungen als Grenzwerte von Differenzenquotienten, approximieren die Ableitung durch Differenzenquotienten.
- Approximieren f durch ein Interpolationspolynom, Splines, ... und differenzieren diese Approximation
- Bemerkung: Numerische Differentiation ist "gefährlich" (kann instabil sein)

Beispiel

$f(x) = x$
Approximation $f_n(x) = x + \frac{1}{n}\sin(nx) \Rightarrow f_n(x) \longrightarrow f(x), n \to \infty, \forall x$
$f'(x) \equiv 1$
$f_n'(x) = 1 + \cos(nx)$, *nicht konvergent für* $n \to \infty$

Numerische Differentiation

Beginnen mit Variante 1 für 1. Ableitungen:

$$f'(x_i) = \lim_{h\to 0} \frac{f(x_i+h)-f(x_i)}{h} \Rightarrow$$

$$f'(x_i) \approx y'_{i,r} = \frac{y_{i+1}-y_i}{h},\ h = x_{i+1}-x_i, y_i = f(x_i), y_{i+1} = f(x_{i+1})$$

$$f'(x_i) \approx y'_{i,l} = \frac{y_i - y_{i-1}}{h},\ h = x_i - x_{i-1}, y_{i-1} = f(x_{i-1})$$

$$f'(x_i) \approx y'_{i,z} = \frac{y_{i+1}-y_{i-1}}{2h},\ 2h = x_{i+1}-x_{i-1}$$

$$f'(x_i) - \tfrac{f(x_{i+1})-f(x_i)}{h} =$$

$$f'(x_i) - \tfrac{1}{h}\left(\underbrace{f(x_i)+hf'(x_i)+\frac{h^2}{2}f''(\xi)}_{\text{Taylor}} - f(x_i)\right) =$$

$$\underbrace{f'(x_i) - \frac{1}{h}hf'(x_i)}_{=0} - \tfrac{h}{2}f''(\xi), \quad \xi \in (x_i, x_{i-1})$$

Numerische Differentiation-Fehlerabschätzung

$$\left| f'(x) - \frac{f(x_{i+1}) - f(x_i)}{h} \right| = \frac{h}{2} \left| f''(\xi) \right| \leq \frac{h}{2} M_2$$

$M_2 = \max_{x \in [x_0, x_n]} |f''(x)|$, falls f'' existiert und stetig ist.

$$f'(x_i) - \frac{1}{2h} \left[f(x_{i+1}) - f(x_{i-1}) \right] =$$

$$= f'(x_i) - \frac{1}{2h} \left[f(x_i) + h f'(x_i) + \frac{h^2}{2} f''(x_i) + \frac{h^3}{6} f'''(\xi_1) \right.$$

$$\left. - f(x_i) + h f'(x_i) - \frac{h^2}{2} f''(x_i) + \frac{h^3}{6} f'''(\xi_2) \right]$$

mit $\xi_1 \in (x_i, x_{i+1})$, $\xi_2 \in (x_{i-1}, x_i)$

$$\left| f'(x_i) - \frac{1}{2h} (f(x_{i+1}) - f(x_{i-1}) \right| \leq \frac{h^3}{6} \cdot \frac{1}{2h} (\underbrace{\left| f'''(\xi_1) \right|}_{\leq M_3} + \underbrace{\left| f'''(\xi_2) \right|}_{\leq M_3}) \leq \frac{h^2}{6} M_3,$$

falls f''' existiert und stetig ist.

Numerische Differentiation

Approximation der 2. Ableitung

$$y''(x_i) = \lim_{h \to 0} \frac{y'(x_i + h) - y'(x_i)}{h} \approx \frac{y'(x_i + h) - y'(x_i)}{h} \approx$$

- "rechts - rechts"

$$\frac{1}{h}\left(\frac{y_{i+2} - y_{i+1}}{h} - \frac{y_{i+1} - y_i}{h}\right) = \frac{1}{h^2}\left(y_{i+2} - 2y_{i+1} + y_i\right)$$

- "links - rechts"

$$\frac{1}{h}\left(\frac{y_{i+1} - y_i}{h} - \frac{y_{i+1} - y_i}{h}\right) = 0 \text{ immer}$$

- "links - links"

$$\frac{1}{h}\left(\frac{y_{i+1} - y_i}{h} - \frac{y_i - y_{i-1}}{h}\right) = \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right)$$

Numerische Differentiation

Fehlerabschätzung für die Approximation der 2. Ableitung

$$y''(x_i) - \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right) =$$

$$y'' - \frac{1}{h^2}\left(y_i + hy_i' + \frac{h^2}{2}y_i'' + \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_1)\right.$$

$$-2y_i$$

$$\left.+y_i - hy_i' + \frac{h^2}{2}y_i'' - \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_2)\right)$$

$$= y_i'' - \frac{1}{h^2}\left(h^2 y_i'' + \frac{h^4}{24}y^{(4)}(\xi_1) + \frac{h^4}{24}y^{(4)}(\xi_2)\right)$$

mit $\xi_1 \in (x_i, x_{i+1})$, $\xi_2 \in (x_{i-1}, x_i)$, falls f 4x stetig differenzierbar.

$\left|y''(x_i) - \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right)\right| = \frac{h^2}{24}\left|y^{(4)}(\xi_1) + y^{(4)}(\xi_2)\right|$

$\leq \frac{h^2}{24}\left(\left|y^{(4)}(\xi_1)\right| + \left|y^{(4)}(\xi_2)\right|\right) \leq \frac{h^2}{12}M_4,\ M_4 = \max_{x\in[x_0,x_n]}\left|y^{(4)}(x)\right|$

Numerische Differentiation-Taylorabgleich

Beispiel

$f''(x_i) \approx A_{i-1}y_{i-1} + A_i y_i + A_{i+1}y_{i+1}$, $y_k = f(x_k)$

$$f''(x_i) - (A_{i-1}y_{i-1} + A_i y_i + A_{i+1}y_{i+1}) =$$

$$= y_i'' - A_{i-1}\left(y_i - hy_i' + \frac{h^2}{2}y_i'' - \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_1)\right)$$

$$-A_i y_i$$

$$-A_{i+1}\left(y_i - hy_i' + \frac{h^2}{2}y_i'' - \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_2)\right)$$

Wir versuchen, die Differenz möglichst klein zu machen

1. $A_{i-1} + A_i + A_{i+1} = 0$
2. $A_{i-1} - A_{i+1} = 0$
3. $1 - \frac{h^2}{2}A_{i-1} - A_{i+1}\frac{h^2}{2} = 0$

und erhalten ein lineares Gleichungssystem für die A_k.

Numerische Differentiation-Taylorabgleich

$$\Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ \frac{h^2}{2} & 0 & \frac{h^2}{2} \end{pmatrix} \begin{pmatrix} A_{i-1} \\ A_i \\ A_{i+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \Rightarrow$$

$$A_{i-1} = \frac{1}{h^2},\ A_{i+1} = \frac{1}{h^2},\ A_i = -\frac{2}{h^2}$$

$A_{i-1},\ A_i,\ A_{i+1} = O(h^{-2}) \Rightarrow$ Fehler $O(\frac{h^3}{h^2}) = O(h)$ aber:

$\frac{h^3}{6}A_{i-1} - \frac{h^3}{6}A_{i+1} = 0$ linear abhängig von Gleichung (2)$\Rightarrow$

$$\text{Fehler } = O(\frac{h^4}{h^2}) = O(h^2)$$

HA: Man approximiere die 2. Ableitung $f''(x_i)$ durch $A_{i-2}y_{i-2} + A_{i-1}y_{i-1} + A_iy_i + A_{i+1}y_{i+1} + A_{i+2}y_{i+2}$

HA: Man bestimme die beste Approximation von $f'''(x_i)$.

HA: Man entwickle einen Algorithmus für die Approximation von $f^{(k)}(x_i)$.

Numerische Differentiation

Beispiel (Gewichtete Approximation)

Man untersuche die Approximation

$$u_i' \approx \lambda(\frac{1}{h}(u_{i+1} - u_i)) + (1-\lambda)\frac{1}{h}(u_i - u_{i-1}).$$

$$u_i' - \frac{1}{h}\left(\lambda(u_i + hu_i' + \frac{h^2}{2}u_i'' + \frac{h^3}{6}u_i''' + \ldots - u_i) + \right.$$

$$\left. +(1-\lambda)(u_i - u_i + hu_i' - \frac{h^2}{2}u_i'' + \frac{h^3}{6}u_i''' - \ldots)\right) =$$

$$= \underbrace{u_i' - \lambda u_i' - (1-\lambda)u_i'}_{0} - \underbrace{\lambda\frac{h}{2}u_i'' + (1-\lambda)\frac{h}{2}u_i''}_{-(1-2\lambda)\frac{h}{2}u_i'' - \frac{h^2}{6}u_i''' + \ldots} - \lambda\frac{h^2}{6}u_i''' - (1-\lambda)\frac{h^2}{6}u_i''' + \ldots$$

Für $\lambda \neq \frac{1}{2} \Rightarrow$ Approximationsordnung $= 1$, $O(h)$.
Für $\lambda = \frac{1}{2} \Rightarrow$ Approximationsordnung $= 2$, $O(h^2)$
$\lambda = \frac{1}{2}$ bedeutet: $\frac{1}{2h}(u_{i+1} - u_i + u_i - u_{i-1}) = \frac{u_{i+1} - u_{i-1}}{2h}$

Numerische Differentiation

HA: Ungleichmäßiges Gitter

$$u_i'' \approx \frac{1}{\tilde{h}_i}\left[\frac{u_{i+1}-u_i}{h_{i+1}} - \frac{u_i - u_{i+1}}{h_i}\right], x_{i+1}-x_i = h_{i+1}, x_i - x_{i-1} = h_i$$

$$\tilde{h}_i = \frac{h_i + h_{i+1}}{2}$$

Man schätze die Approximationsordnung ab und bewerte das Resultat.

Numerik

Prof. Klaus Gürlebeck
Bauhaus-Universität Weimar

Wintersemester 2021/2022

Vorlesung 9-10

Diese Folien dienen der Unterstützung der Vorlesung. Sie sind kein vollständiges Skript. Die Folien sind nur zur persönlichen Verwendung gedacht.

Numerische Differentiation-Taylorabgleich

Beispiel

$f''(x_i) \approx A_{i-1}y_{i-1} + A_i y_i + A_{i+1}y_{i+1}$, $y_k = f(x_k)$

$$f''(x_i) - (A_{i-1}y_{i-1} + A_i y_i + A_{i+1}y_{i+1}) =$$

$$= y_i'' - A_{i-1}\left(y_i - hy_i' + \frac{h^2}{2}y_i'' - \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_1)\right)$$

$$-A_i y_i$$

$$-A_{i+1}\left(y_i - hy_i' + \frac{h^2}{2}y_i'' - \frac{h^3}{6}y_i''' + \frac{h^4}{24}y^{(4)}(\xi_2)\right)$$

Wir versuchen, die Differenz möglichst klein zu machen

1. $A_{i-1} + A_i + A_{i+1} = 0$
2. $A_{i-1} - A_{i+1} = 0$
3. $1 - \frac{h^2}{2}A_{i-1} - A_{i+1}\frac{h^2}{2} = 0$

und erhalten ein lineares Gleichungssystem für die A_k.

Numerische Differentiation-Taylorabgleich

$$\Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ \frac{h^2}{2} & 0 & \frac{h^2}{2} \end{pmatrix} \begin{pmatrix} A_{i-1} \\ A_i \\ A_{i+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \Rightarrow$$

$$A_{i-1} = \frac{1}{h^2},\ A_{i+1} = \frac{1}{h^2},\ A_i = -\frac{2}{h^2}$$

$A_{i-1},\ A_i,\ A_{i+1} = O(h^{-2}) \Rightarrow$ Fehler $O(\frac{h^3}{h^2}) = O(h)$ aber:

$\frac{h^3}{6}A_{i-1} - \frac{h^3}{6}A_{i+1} = 0$ linear abhängig von Gleichung (2)$\Rightarrow$

$$\text{Fehler } = O(\frac{h^4}{h^2}) = O(h^2)$$

HA: Man approximiere die 2. Ableitung $f''(x_i)$ durch $A_{i-2}y_{i-2} + A_{i-1}y_{i-1} + A_i y_i + A_{i+1}y_{i+1} + A_{i+2}y_{i+2}$

HA: Man bestimme die beste Approximation von $f'''(x_i)$.

HA: Man entwickle einen Algorithmus für die Approximation von $f^{(k)}(x_i)$.

Numerische Differentiation

Beispiel (Gewichtete Approximation)

Man untersuche die Approximation

$$u_i' \approx \lambda\left(\frac{1}{h}(u_{i+1} - u_i)\right) + (1-\lambda)\frac{1}{h}(u_i - u_{i-1}).$$

$$u_i' - \frac{1}{h}\left(\lambda(u_i + hu_i' + \frac{h^2}{2}u_i'' + \frac{h^3}{6}u_i''' + \ldots - u_i) + \right.$$

$$\left. +(1-\lambda)(u_i - u_i + hu_i' - \frac{h^2}{2}u_i'' + \frac{h^3}{6}u_i''' - \ldots)\right) =$$

$$= \underbrace{u_i' - \lambda u_i' - (1-\lambda)u_i'}_{0} - \underbrace{\lambda\frac{h}{2}u_i'' + (1-\lambda)\frac{h}{2}u_i''}_{-(1-2\lambda)\frac{h}{2}u_i'' - \frac{h^2}{6}u_i''' + \ldots} - \lambda\tfrac{h^2}{6}u_i''' - (1-\lambda)\tfrac{h^2}{6}u_i''' + \ldots$$

Für $\lambda \neq \frac{1}{2} \Rightarrow$ Approximationsordnung $= 1$, $O(h)$.
Für $\lambda = \frac{1}{2} \Rightarrow$ Approximationsordnung $= 2$, $O(h^2)$
$\lambda = \frac{1}{2}$ bedeutet: $\frac{1}{2h}(u_{i+1} - u_i + u_i - u_{i-1}) = \frac{u_{i+1} - u_{i-1}}{2h}$

Numerische Differentiation

HA: Ungleichmäßiges Gitter

$$u_i'' \approx \frac{1}{\tilde{h}_i}\left[\frac{u_{i+1}-u_i}{h_{i+1}} - \frac{u_i - u_{i+1}}{h_i}\right], x_{i+1} - x_i = h_{i+1}, x_i - x_{i-1} = h_i$$

$$\tilde{h}_i = \frac{h_i + h_{i+1}}{2}$$

Man schätze die Approximationsordnung ab und bewerte das Resultat.

Numerische Lösung von gewöhnlichen Differentialgleichungen 1. Ordnung

Wir betrachten das Anfangswertproblem

$$\begin{aligned} y' &= f(x,y) \\ y(x_0) &= y_0. \end{aligned}$$

Approximieren

$$y'(x) = \frac{y(x+h) - y(x)}{h}$$

$$x = x_0 + h \quad \Rightarrow y(x_0 + h) = hf(x_0, y_0) + y(x_0)$$

Nächster Schritt

$$y(x_0 + 2h) = y(x_0 + h) + hf(x_0 + h, y(x_0 + h))$$

...

$h > 0$	muss gewählt werden, $y_0 = y(x_0)$ gegeben
x_n	$= x_0 + nh \qquad n = 0, 1, 2, \ldots$
y_{n+1}	$= y_n + hf(x_n, y_n)$

Explizites Eulerverfahren

Wir benötigen eine Fehlerabschätzung. Wir nehmen an, dass f zweimal stetig differenzierbar auf $[x_0 - a, x_0 + a] \times [y_0 - b, y_0 + b]$ ist. Die Stetigkeit der zweiten Ableitung muss später noch einmal betrachtet werden, weil die Lösung einer Differentialgleichung erster Ordnung in der Regel nur einmal differenzierbar ist.

$$\begin{aligned} y(x_1) = y(x_0 + h) &= y_0(x_0) + hy'(x_0) + \frac{1}{2}h^2 y''(x_0 + \theta h) \quad 0 < \theta < 1 \\ &= \underbrace{y_0 + hf(x_0, y_0)}_{y_1, \text{ siehe Algorithmus}} + \frac{1}{2}h^2 y''(x_0 + \theta h) \end{aligned}$$

$$\Rightarrow \quad \underline{y(x_1) - y_1} = \frac{1}{2}h^2 y''(x_0 + \theta h) = \underline{O(h^2)}$$

Das ist eine lokale Abschätzung für den ersten Schritt unter der Annahme, dass der Wert y_0 exakt ist.

Explizites Eulerverfahren

n-ter Schritt: (warum nicht auch $O(h^2)$?)

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{1}{2}h^2 y''(x_n + \theta_n h) \\ &= y(x_n) + hf(x_n, y(x_n)) + \frac{1}{2}h^2 y''(x_n + \theta_n h) \quad \theta_n \in (0,1) \end{aligned}$$

wegen $y_{n+1} = y_n + hf(x_n, y_n)$ erhalten wir

$$y(x_{n+1}) - y_{n+1} = y(x_n) - y_n + h[\underbrace{f(x_n, y(x_n)) - f(x_n, y_n)}] + \frac{1}{2}h^2 \underbrace{y''(x_0 + \theta_n h)}_{|\,| \leq B}$$

$$= \underbrace{\frac{\partial f}{\partial y}(x_n, y_n^*)}_{|\,| \leq A}(y(x_n) - y_n)$$

und schließlich,

$$y(x_{n+1}) - y_{n+1} = \left[1 + h\frac{\partial f}{\partial y}(x_n, y_n^*)\right](y(x_n) - y_n) + \frac{1}{2}h^2 y''(x_0 + \theta_n h)$$

$$|y(x_{n+1}) - y_{n+1}| \leq (1 + hA)|y(x_n) - y_n| + \frac{1}{2}h^2 B.$$

Explizites Eulerverfahren

Definieren wir $y(x_k) - y_k =: \delta_k$, so erhalten wir rekursiv

$$
\begin{aligned}
\delta_{n+1} &\leq (1+hA)\delta_n + \frac{1}{2}h^2B \\
&\leq (1+hA)[(1+hA)\delta_{n-1} + \frac{1}{2}h^2B] + \frac{1}{2}h^2B \\
&= (1+hA)^2\delta_{n-1} + (1+hA)\frac{1}{2}h^2B + \frac{1}{2}h^2B \\
&\leq (1+hA)^3\delta_{n-2} + (1+hA)^2\frac{1}{2}h^2B + (1+hA)\frac{1}{2}h^2B + \frac{1}{2}h^2B \\
&\vdots \\
&\leq (1+hA)^n\frac{1}{2}h^2B + (1+hA)^{n-1}\frac{1}{2}h^2B + \ldots + (1+hA)\frac{1}{2}h^2B + \\
&+ \frac{1}{2}h^2B \\
&= [(1+hA)^n + \ldots + (1+hA) + 1]\frac{1}{2}h^2B
\end{aligned}
$$

Explizites Eulerverfahren

$$= \left[\frac{1-(1+hA)^{n+1}}{1-(1+hA)}\right]\frac{1}{2}h^2B = \left[\frac{-1+(1+hA)^{n+1}}{hA}\right]\frac{1}{2}h^2B$$

$$= \frac{-1+(1+hA)^{n+1}}{A}\frac{B}{2}h \leq \frac{B}{2}\frac{e^{(x_{n+1}-x_0)A}-1}{A}h$$

Für den letzten Schritt haben wir die folgenden Abschätzungen benutzt:

$$\begin{aligned} 1+x &\leq e^x \\ 1+hA &\leq e^{hA} \\ (1+hA)^{n+1} &\leq e^{(n+1)hA} \\ (1+hA)^{n+1}-1 &\leq e^{(x_{n+1}-x_0)A}-1 \end{aligned}$$

Bemerkung

- *Globale Genauigkeit ist nicht sehr gut, aber die Methode ist einfach und effizient.*
- *h muss sehr klein gewählt werden, um eine akzeptable Genauigkeit zu erreichen.*
- *Gute Idee, um ersten Eindruck über die Lösung zu erhalten.*

Stabilität des expliziten Eulerverfahrens

Wir haben im numerischen Beispiel gesehen, dass das Verfahren instabil wird, wenn h nicht klein genug ist. Das ist eine zweite Anforderung an h, zusätzlich zur Genauigkeit der Approximation. Wir suchen nach möglichen Gründen und betrachten das Beispiel

$$y'(x) = -ky(x), k > 0, y(0) = 1$$

Die Anwendung des expliziten Eulerverfahrens bedeutet

$$y_{n+1} = y_n - hky_n = (1-hk)y_n = (1-hk)^2 y_{n-1} = \ldots = (1-hk)^{n+1} y_0.$$

Notwendig für die Konvergenz ist

$$\begin{aligned} |1 - hk| &< 1 \\ -1 < 1 - hk &< 1 \\ -2 < -hk \iff hk < 2 \iff h &< \frac{2}{k} \end{aligned}$$

und das ist unsere Stabilitätsbedingung.

Implizites Eulerverfahren

Eine formal kleine Anpassung führt zu ganz anderen Ergebnissen. Wir betrachten das *implizite Eulerverfahren*.

$$\begin{aligned} y_{n+1} &= y_n + hf(x_{n+1}, y_{n+1}), n = 0, 1, ... \\ y_0 &= y(0) \end{aligned}$$

Hausaufgabe: Man untersuche die Approximationsordnung!

Untersuchen wir wieder die Stabilität für unser Beispiel, so erhalten wir das folgende Resultat:

$$\begin{aligned} y_{n+1} &= y_n - hky_{n+1} \\ y_{n+1}(1 + hk) &= y_n \\ y_{n+1} &= \frac{1}{1 + hk} y_n = ... = \frac{1}{(1 + hk)^{n+1}} y_0. \end{aligned}$$

Wegen $\frac{1}{1+hk} < 1 \ \forall h > 0$ ergibt sich die sogenannte *unbedingte Stabilität*.

Approximation und Stabilität

Basierend auf unseren Beobachtungen für unser einfaches Beispiel wollen wir nun die Zusammenhänge zwischen Approximation, Stabilität und Konvergenz allgemeiner studieren. Mit diesen Resultaten werden wir dann gewöhnliche Differentialgleichungen höherer Ordnung untersuchen.
Zu lösen sei eine Differentialgleichung $Au = f$ im Intervall Ω. Wir approximieren diese Gleichung durch eine Gleichung $A_h u_h = f_h$ auf einer Diskretisierung Ω_h von Ω.

$$\begin{aligned} y''(x) &= f(x), x \in (0,1) \\ Ay &= y'' \\ \Omega &= (0,1) \end{aligned}$$

$$\begin{aligned} \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} &= f_i, i = 1, ..., N-1 \\ (A_h y)_i &= \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} \\ \Omega_h &= \{x_0, x_1, ..., x_{n-1}, x_n\} \end{aligned}$$

Approximation und Stabilität

Wenn

$$||A_h(u)_h - (f)_h|| \leq C \cdot h^{\alpha}$$

dann sagen wir, dass A durch A_h mit der Ordnung α approximiert wird.

Wenn

$$||u_h|| \leq \tilde{C}||f_h||$$

dann sagen wir, dass das diskrete Problem $A_h u_h = f_h$ gut konditioniert ist (oder stabil). Wir fragen nun nach Konsequenzen der Stabilität.

- $A_h u_h = f_h \wedge ||u_h|| \leq \tilde{C}||f_h|| \Longrightarrow A_h^{-1}$ existiert.
- $u_h = A_h^{-1} f_h \Longrightarrow ||u_h|| \leq ||A_h^{-1}||\,||f_h|| \Longrightarrow ||A_h^{-1}||$ ist gleichmäßig beschränkt.
- $\Longrightarrow \kappa(A_h) = ||A_h||\,||A_h^{-1}||$ ist gleichmäßig beschränkt bzgl. h.

Nehmen an, dass

$$||A_h(u)_h - (f)_h|| \leq C \cdot h^{\alpha} \text{ und } ||u_h|| \leq \tilde{C}||f_h||.$$

Approximation und Stabilität

$$
\begin{aligned}
A_h(u_h - (u)_h) &= f_h - A_h(u)_h = \underbrace{f_h - (Au)_h}_{} + \underbrace{\overbrace{(Au)_h}^{(f)_h} - A_h(u)_h}_{z_h} \\
&= \underbrace{f_h - (f)_h}_{0} + z_h = z_h \\
||u_h - (u)_h|| &\leq \tilde{C}||z_h|| \leq \tilde{C} \cdot C \cdot h^{\alpha}
\end{aligned}
$$

Das bedeutet: "Approximation + Stabilität = Konvergenz".
Das ist ein sehr wichtiges Resultat, auch für praktische Zwecke.
Die Überprüfung der Approximation erfordert einige Rechnungen und Abschätzungen, die auf der Taylorschen Formel basieren, aber nicht wirklich schwierig sind. Zur Charakterisierung der Stabilität werden wir einige allgemeine Sätze formulieren, die sofort in der Praxis anwendbar sind. Damit kann auch die Frage nach der Konvergenz in der Praxis beantwortet werden, ohne nur auf Computersimulationen zu schauen.