panikzettel.htwr-aachen.de

Numerisches Rechnen Panikzettel

"der Dude", Luca Oeljeklaus,
Tobias Polock, Philipp Schröer, Caspar Zecha

Version 1 — 04.08.2024

Inhaltsverzeichnis

1 Einleitung

Dieser Panikzettel ist über die Vorlesung Numerisches Rechnen. Er basiert auf dem Foliensatz von Prof. Dr. Martin Grepl aus dem Wintersemester 17/18.

Dieser Panikzettel ist Open Source. Wir freuen uns über Anmerkungen und Verbesserungsvorschläge auf `https://git.rwth-aachen.de/philipp.schroer/panikzettel`.

2 Fehleranalyse: Kondition, Rundungsfehler, Stabilität

2.1 Kondition

Der *absolute Fehler* in Daten x und gestörten Daten $\tilde{x}$ ist $\|\Delta_x\| := \|\tilde{x} - x\|$. Der *relative Fehler* ist $\delta_x := \frac{\|\Delta_x\|}{\|x\|}$.

Für eine Funktion f ist der Fehler in den Ausgabedaten $\tilde{y} = f(\tilde{x})$.

Mit *Kondition* ist meist die *relative Kondition* gemeint: $\frac{\delta_y}{\delta_x}$. Diese beschreibt die *Sensitivität* des Problems bei Störungen in den Eingabedaten. Die *absolute Kondition* ist $\frac{\|\Delta_y\|}{\|\Delta_x\|}$. Wir sagen, dass ein Problem *gut konditioniert* ist, wenn seine Kondition klein (≤ 1) ist.

Als Approximation (über Taylorentwicklung 1. Ordnung) an die Kondition benutzen wir die *Konditionszahl*.

$$\kappa_{\text{rel}}(x) := \left| f'(x) \frac{x}{f(x)} \right|$$

2.2 Gleitpunktdarstellung

Da die Menge der Reellen Zahlen nicht im Computer darstellbar ist[Citation needed], approximieren wir diese durch Maschinenzahlen.

$$\mathbb{M}(b,m,r,R) = \left\{ \pm \sum_{i=1}^{m} d_i b^{e-i} \;\middle|\; r \leq e \leq R,\ 0 \leq d_i < b \right\}$$

- Relative Maschinengenauigkeit: $\text{eps} := \frac{b^{1-m}}{2} = \inf\{\, \delta > 0 \mid \text{fl}(1+\delta) > 1 \,\}$
- Die betragsmäßig kleinste Zahl ($\neq 0$): $x_{\text{MIN}} = b^{r-1}$
- Die betragsmäßig größte Zahl: $x_{\text{MAX}} = (1 - b^{-m})b^R$

Für IEEE Double-precision floating point-Zahlen ist $\text{eps} = 2.2204 \times 10^{-16}$.

Die Abbildung $\text{fl} : \mathbb{R} \to \mathbb{M}(b,m,r,R)$ rundet. Es gilt $\text{fl}(x) = x(1+\delta)$ für ein $|\delta| \leq \text{eps}$.

Für Multiplikation und Division von Maschinenzahlen ist $\kappa_{\text{rel}} \leq 1$. Diese Operationen sind also immer gut konditioniert. Bei Addition und Subtraktion gilt dagegen $\kappa_{\text{rel}} \gg 1$, wenn das Ergebnis gegen Null geht.

2.3 Stabilität

Stabile Algorithmen erzeugen Fehler nur in der Größenordnung des Fehlers, der durch die Kondition des Problems unvermeidbar ist.

Ein *rückwärts stabiles* Verfahren $\tilde{f}$, das eine Funktion f approximiert, gibt die exakte Antwort auf eine nahezu richtige Frage: Für alle $x \in X$ gilt

$$\tilde{f}(x) = f(\tilde{x}) \text{ für ein } \tilde{x} \text{ mit } \frac{\|x - \tilde{x}\|}{\|x\|} = \mathcal{O}(\text{eps}).$$

Dann wissen wir: $\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \mathcal{O}(\kappa(x)\text{eps})$.

2.4 Taylorentwicklung

Funktionen lassen sich als die Taylorentwicklung darstellen. Für ein ξ zwischen x und x_0 ist

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i + \frac{f^{(n)}(\xi)}{n!}(x - x_0)^n.$$

Somit lässt sich eine Funktion f durch das sogenannte Taylorpolynom n-ten Grades approximieren:

$$p_n(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i$$

3 Lineare Gleichungssysteme, direkte Lösungsverfahren

3.1 Kondition und Störungssätze

Die *Konditionszahl einer Matrix* ist $\kappa(A)_{\|\cdot\|} = \|A\|\|A^{-1}\|$ bzgl. der $\|\cdot\|$-Norm.

Für ein lineares Gleichungssystem $Ax = b$ mit Störung in b bzw. in A und b gilt:

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

3.2 LR-Zerlegung (Gauß-Elimination)

Bei der Gauß-Elimination wird eine Matrix und die zugehörige rechte Seite durch elementare Zeilentransformationen in rechte obere Dreiecksform gebracht, die anschließend durch Rückwärtseinsetzen einfach gelöst werden kann.

Bei der *LR-Zerlegung* werden diese Zeilentransformationen zusätzlich in einer Matrix L gespeichert. So wird $A \in \mathbb{R}^{n \times n}$ in $A = LR$ zerlegt. L ist eine normierte linke untere und R eine rechte obere Dreiecksmatrix.

Etwa ist $L_{2,1} = \frac{A_{2,1}}{A_{1,1}}$, also der Faktor mit dem man Zeile 1 von Zeile 2 subtrahiert, um $A_{2,1}$ zu eliminieren.

So können wir $Ax = b$ über $Ly = b$ und $Rx = y$ lösen.

Um die Kondition des Problems zu verbessern, können wir mit Diagonalmatrix D_Z *äquilibieren*. Wir teilen jede Zeile durch ihre Betragssumme. Das gibt uns die minimale Konditionszahl bezüglich der ∞-Norm.

Die *LR-Zerlegung mit Spaltenpivotisierung* ist eine Zerlegung $PA = LR$ mit einer Permutationsmatrix P. P entsteht, indem in jeder Spalte das betragsgrößte Element als *Pivotelement* wählt und dafür die Matrixzeilen vertauscht. Dies ist nötig, da sonst das Pivotelement verschwinden könnte (Division durch null!) und es ist sowieso gut für Stabilität.

3.3 Cholesky-Zerlegung (LDL^T-Zerlegung)

Für symmetrisch positiv definite Matrix $A \in \mathbb{R}^{n \times n}$ existiert eine Zerlegung

$$A = LDL^T$$

wobei L eine normierte linke untere Dreiecksmatrix und D eine Diagonalmatrix ist.

$$d_{k,k} = a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}^2 d_{j,j}$$
$$l_{i,k} = \left(a_{i,k} - \sum_{j=1}^{k-1} l_{i,j} d_{j,j} l_{k,j} \right) / d_{k,k}$$

Zur Berechnung läuft man einfach durch die Spalten und berechnet $d_{1,1}, l_{2,1}, \ldots$. Auch cool: Ist $d_{i,i} > 0$ für alle i und L normierte Dreiecksmatrix, dann ist A symmetrisch positiv definit! Cholesky kann also erst angewandt werden und dann das Ergebnis auf Gültigkeit geprüft werden.

Für $n = 3$ ergibt sich beispielsweise

$$\begin{aligned} LDL^T &= \begin{pmatrix} 1 & & \\ l_{2,1} & 1 & \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix} \begin{pmatrix} d_{1,1} & & \\ & d_{2,2} & \\ & & d_{3,3} \end{pmatrix} \begin{pmatrix} 1 & l_{2,1} & l_{3,1} \\ & 1 & l_{3,2} \\ & & 1 \end{pmatrix} \\ &= \begin{pmatrix} d_{1,1} & l_{2,1}d_{1,1} & l_{3,1}d_{1,1} \\ l_{2,1}d_{1,1} & l_{2,1}^2 d_{1,1} + d_{2,2} & l_{2,1}l_{3,1}d_{1,1} + l_{3,2}d_{2,2} \\ l_{3,1}d_{1,1} & l_{2,1}l_{3,1}d_{1,1} + l_{3,2}d_{2,2} & l_{3,1}^2 d_{1,1} + l_{3,2}^2 d_{2,2} + d_{3,3} \end{pmatrix} \end{aligned}$$

3.4 QR-Zerlegung

Bei der QR-Zerlegung wird eine Matrix A durch orthogonale Transformationen Q^T in rechte obere Dreiecksform gebracht. Rechte obere Dreiecksform bedeutet, dass die ersten k Spalten von R den selben Raum wie die ersten k Einheitsvektoren aufspannen. Es geht also bei der QR-Zerlegung darum, durch orthogonale Transformationen den Raum, der von den ersten k Spalten von A aufgespannt wird, in die von den ersten k Einheitvektoren aufgespannte Hyperebene abzubilden.

Da orthogonale Transformationen genau Kombinationen von Rotationen und Spiegelungen sind, ergeben sich folgende drei Optionen für die Transformationen.

3.4.1 Gram-Schmidt-Orthogonalisierung

Ausführlich erklärt in unserem Panikzettel zu Lineare Algebra. Dieses Verfahren ist nicht stabil.

3.4.2 Givens-Rotation

Die Givens-Rotationsmatrix $G_{i,k}$ rotiert in der x_i-x_k-Ebene die Matrix A so, dass in $G_{i,k}A$ der Eintrag an Position k, i Null wird.

$G_{i,i} = c = G_{k,k}$, $G_{i,k} = s$ und $G_{k,i} = -s$. Dazu Einsen auf die Diagonale und sonst Nullen.

$$G_{i,k} = \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & 0 & \\ & & 1 & & & & & & \\ & & & c & 0 & \cdots & 0 & s & \\ & & & 0 & 1 & & & 0 & \\ & & & \vdots & & \ddots & & \vdots & \\ & & & 0 & & & 1 & 0 & \\ & & & -s & 0 & \cdots & 0 & c & \\ & & & & & & & & 1 \\ & 0 & & & & & & & \ddots \\ & & & & & & & & & 1 \end{pmatrix}$$

$$\begin{aligned} r &= \pm \left\| \begin{pmatrix} A_{i,i} \\ A_{k,i} \end{pmatrix} \right\|_2 \\ c &= \frac{A_{i,i}}{r} \\ s &= \frac{A_{k,i}}{r} \end{aligned}$$

Wir wenden also diese Rotationsmatrizen auf alle Elemente unterhalb der Diagonalen von A an, sodass wir eine QR-Zerlegung für A erhalten:

$$R = \underbrace{\dots G_{2,3} \dots G_{1,2}}_{Q^T} A \quad \text{bzw.} \quad A = \underbrace{G_{1,2}^T \dots G_{2,3}^T \dots}_{Q} R$$

Q wird nie explizit berechnet. Zur Lösung von $QRx = b$: Berechne $y = \dots G_{2,3} \dots G_{1,2} b$ und löse $Rx = y$ durch Rückwärtseinsetzen.

3.4.3 Householder-Transformation

Die Householder-Transformation entspricht Spiegelungen. Die Householder-Matrix Q_v spiegelt an der zu v orthogonalen Hyperebene (v ist Normalenvektor).

$$Q_v = I - 2\frac{vv^T}{v^Tv}$$

vv^T ist eine Dyade. v^Tv ist ein Skalar zur richtigen Skalierung. Die 2 taucht auf, weil man ja nicht genau einmal den Normalenvektor in die Hyperebene, sondern genau zwei mal den Normalenvektor durch (auf die andere Seite) der Hyperebene will.

Die folgenden Eigenschaften sind schön zu beweisen:

- Orthogonalität: $Q_v^{-1} = Q_v^T$.
- Selbstinverse: $Q_v^2 = I$.
  Zweimalige Spiegelung ergibt den ursprünglichen Punkt.
- $Q_{\alpha v} = Q_v, \alpha \in \mathbb{R} \setminus \{0\}$.
  Skalierung des Normalenvektors ändert nicht die Spiegelebene.
- $Q_v y = y \Leftrightarrow y^Tv = 0$.
  Ursprünglicher und gespiegelter Punkt sind nur identisch, wenn der Punkt in der Spiegelebene liegt.
- $Q_v v = -v$.
  Spiegelung des Normalenvektors vertauscht das Vorzeichen.

Mit Householder schauen wir uns immer die erste Spalte a_1 an und spiegeln die auf die Ebene zum Einheitsvektor e_1. Danach wiederholen wir das Gleiche auf der Teilmatrix A ohne erste Zeile und Spalte (in den Q_v fügen wir passen Nullen ein).

$$v = a_1 + \text{sign}(a_{1,1})\|a_1\|e_1$$

Mit Q_i als Q_v mit v für Teilmatrix i ist dann die QR-Zerlegung:

$$Q_{n-1} \dots Q_2 Q_1 A = R \text{ bzw. } A = Q_1^T Q_2^T \dots Q_{n-1}^T R = QR$$

Übrigens ist allgemeiner $v = a_1 \pm \|a_1\|e_1$ möglich, aber wir möchten Auslöschung vermeiden.

4 Lineare Ausgleichsrechnung

Bei einem *Linearen Ausgleichsproblem* wollen wir ein überbestimmtes LGS so lösen, dass der Fehler möglichst klein ist. Für $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$ mit $m < n$ suchen wir x^*:

$$x^* = \underset{x \in \mathbb{R}^m}{\text{argmin}} \|Ax - b\|$$

Im Folgenden werden wir annehmen, dass es sich bei der gewählten Norm um die 2-Norm handelt, da das Problem sonst nicht eindeutig lösbar ist.

Geometrische Vorstellung: Wir wollen x^* so, dass das Residuum $r = b - Ax^*$ orthogonal auf Ax^* steht, denn dann wird r am kleinsten.

Für die Kondition des Problems gilt mit $\cos\Theta = \frac{\|Ax^*\|_2}{\|b\|_2}$ und $\kappa_2(A) = \frac{\max_{\|x\|=1}\|Ax\|_2}{\min_{\|x\|=1}\|Ax\|_2}$:

$$\frac{\|\tilde{x} - x^*\|}{\|x^*\|} \leq \frac{\kappa_2(A)}{\cos\Theta} \frac{\|\tilde{b} - b\|_2}{\|b\|_2}$$
$$\frac{\|\tilde{x} - x^*\|}{\|x^*\|} \leq (\kappa_2(A) + \kappa_2(A)^2 \tan\Theta) \frac{\|\tilde{A} - A\|_2}{\|A\|_2}$$

4.1 Normalengleichungen

Mit der Lösung von *Normalengleichungen* lässt sich das Problem auf einfache LGS zurückführen:

$$\left(A^T A\right) x^* = A^T b$$

Es gibt immer eine Lösung. Sie ist genau dann eindeutig, wenn A vollen Rang hat.

Ausgleichsrechnung über Normalengleichungen ist allerdings numerisch instabil.

4.2 QR-Zerlegung

Ein numerisch stabiles Verfahren zur Lösung des Linearen Ausgleichsproblems ergibt sich mithilfe der QR-Zerlegung.

Sei eine QR-Zerlegung gegeben mit $Q \in \mathbb{R}^{n\times n}$ und $R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \in \mathbb{R}^{n\times m}$, wobei $\hat{R} \in \mathbb{R}^{m\times m}$. Weil orthogonale Projektionen die 2-Norm nicht ändern, gilt:

$$x^* = \underset{x\in\mathbb{R}^m}{\operatorname{argmin}} \|Ax - b\|_2 = \underset{x\in\mathbb{R}^m}{\operatorname{argmin}} \left\| Rx - Q^T b \right\|_2$$

Berechnen wir $\hat{b} \in \mathbb{R}^n$ mit $Q^T b = \begin{pmatrix} \hat{b} \\ r \end{pmatrix}$, können wir $\hat{R}x^* = \hat{b}$ durch Rückwärtseinsetzen lösen. Dann ist auch die Norm des Residuums durch $\|r\|_2$ gegeben.

4.3 Singulärwertzerlegung (SVD)

Die *Singulärwertzerlegung* einer Matrix $A \in \mathbb{R}^{n\times m}$ ist $A = U\Sigma V^T$.

Dabei sind $U \in \mathbb{R}^{n\times n}$ und $V \in \mathbb{R}^{m\times m}$ orthogonalen Matrizen und $\Sigma \in \mathbb{R}^{n\times m}$ Diagonalmatrix. Die nicht-null Einträge von Σ heißen Singulärwerte von A, die Spalten von U heißen Linkssingulärvektoren, die Spalten von V heißen Rechtssingulärvektoren.

SVD gibt die Pseudoinverse $A^+ := V\Sigma^+ U^T$ mit Σ^+ der Diagonalmatrix, die die invertierten Singulärwerte von A enthält.

Auch damit können wir das Lineare Ausgleichsproblem stabil lösen:

$$x^* = A^+ b \implies x^* = \underset{x\in\mathbb{R}^m}{\operatorname{argmin}} \|Ax - b\|$$

Sind die Singulärvektoren nach betragsmäßiger Größe geordnet, ergibt sich die Lösung mit kleinster 2-Norm.

5 Nichtlineare Gleichungssysteme

5.1 Fixpunktiteration

Für eine Funktion $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ heißt $x \in \mathbb{R}^n$ Fixpunkt von Φ gdw. $\Phi(x) = x$.

Damit die Fixpunktiteration auf Φ funktioniert, muss gelten:

- Selbstabbildung auf einem kompaktem $E \subseteq \mathbb{R}^n$: $\Phi : E \to E$.
- Kontraktion: $\|\Phi(x) - \Phi(y)\| \leq L\|x - y\|$ für alle $x, y \in E$ mit $L < 1$.

Nach dem *Banachschen Fixpunktsatz* gilt dann: Φ hat einen eindeutigen Fixpunkt $x^* \in E$ und die Fixpunktiteration $x^{i+1} = \Phi(x^i)$ konvergiert für alle $x^0 \in E$.

Fehlerabschätzungen:

$$\|x^k - x^*\| \leq \frac{L^k}{1-L}\|x^1 - x^0\| \qquad \text{bzw. } \log\left((1-L)\frac{\|x_k - x^*\|}{\|x^1 - x^0\|}\right) / \log(L) \leq k \qquad \textit{(a priori)}$$

$$\|x^k - x^*\| \leq \frac{L}{1-L}\|x^k - x^{k-1}\| \qquad \textit{(a posteriori)}$$

Das Nullstellenproblem $f(x^*) = 0$ ist äquivalent zum Fixpunktproblem

$$x^* = \Phi(x^*) \quad \text{mit} \quad \Phi(x) := x - M_{x^*} f(x)$$

und mit einer Matrix M_{x^*}, die in einer Umgebung von x^* invertierbar ist.

5.2 Newton-Verfahren

Es sei $F : \mathbb{R}^n \to \mathbb{R}^n$ stetig differenzierbar. Beim *(klassischen) Newton-Verfahren* lösen wir in jedem Schritt:

$$F'(x^k) \underbrace{(x^{k+1} - x^k)}_{:=s^k} = -F(x^k) \quad \text{bzw.} \quad x^{k+1} = x_k - (F'(x^k))^{-1} \cdot F(x^k)$$

s^k ist die Schrittweite ($x^{k+1} = x^k + s^k$). Das Newton-Verfahren konvergiert lokal quadratisch.

Das *vereinfachte Newton-Verfahren* verwendet statt $F'(x^k)$ einfach $F'(x^0)$ in jedem Schritt, d.h. die Iteration ist gegeben durch:

$$x^{k+1} = x^k - (F'(x^0))^{-1} \cdot F(x^k)$$

Dabei geht quadratische Konvergenz verloren, aber über LR-Zerlegung gewinnen wir Effizienz (in $F'(x_0)(x^{k+1} - x^k) = -F(x^k)$ wiederholen sich linke Seiten).

6 Nichtlineare Ausgleichsrechnung

Beim *nichtlinearen Ausgleichsproblem* sucht man für eine (nicht unbedingt lineare) Funktion $F : \mathbb{R}^n \to \mathbb{R}^m$ ein x^* sodass

$$x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \|F(x)\|_2.$$

6.1 Gauß-Newton-Verfahren

Die lineare Approximation von $F(x)$ ist $F(x^k) + F'(x^k)(x - x^k)$ (Taylor-Entwicklung).

Das *Gauß-Newton-Verfahren* ist eine Fixpunktiteration, bei der $F(x)$ linear approximiert wird:

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^n} \| \underbrace{F(x^k)}_{\widehat{=} -b} + \underbrace{F'(x^k)}_{\widehat{=} A} \underbrace{(x - x^k)}_{\widehat{=} s^k} \|_2$$

In jedem Schritt wird also ein lineares Ausgleichsproblem $\arg\min_{s^k \in \mathbb{R}^n} \|As^k - b\|_2$ gelöst.

Wenn $F'(x)$ vollen Rang hat, können wir auch schreiben (über Normalengleichung):

$$x^{k+1} = x^k - (F'(x^k)^T F'(x^k))^{-1} \cdot F'(x^k)^T F(x^k)$$

6.2 Bonus: Newton-Verfahren

Eigentlich nur hier zum Vergleich der Gauß-Newton-Fixpunktiteration mit der Newton-Iteration.

Wir wissen, dass gilt:

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^n} \|F(x)\|_2 = \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} \|F(x)\|_2^2 = \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} F(x)^T \cdot F(x)$$

$\phi(x) := \frac{1}{2} F(x)^T \cdot F(x)$ hat an einer Stelle x^* ein lokales Minimum genau wenn $\nabla\phi(x^*) = 0$ und wenn $\phi''(x^*)$ symmetrisch positiv definit ist. Also könnte man mit dem Newton-Verfahren eine Nullstelle von $\nabla\phi(x)$ bestimmen:

$$\begin{aligned} x^{k+1} &= x^k - (\phi''(x^k))^{-1} \nabla\phi(x^k) \\ &= x^k - (F'(x^k)^T F'(x^k) + \underbrace{\sum_{i=1}^{m} F_i(x^k) F_i''(x^k))^{-1} \nabla\phi(x^k)}_{\text{hässliche Summe}} \end{aligned}$$

Die hässliche Summe wird beim Gauß-Newton-Verfahren ignoriert.

6.3 Levenberg-Marquardt-Verfahren

Das *Levenberg-Marquardt-Verfahren* ist eine Modifikation des Gauß-Newton-Verfahrens, bei dem bei der Berechnung der Schrittweite $s^k = x^{k+1} - x^k$ ein Dämpfungsfaktor $\mu > 0$ eingefügt wird:

$$s^k = \operatorname*{argmin}_{s \in \mathbb{R}^n} (\|F'(x^k)s + F(x^k)\|_2^2 + \mu^2 \|s\|_2^2) = \operatorname*{argmin}_{s \in \mathbb{R}^n} \| \underbrace{\begin{pmatrix} F'(x^k) \\ \mu I \end{pmatrix}}_{\substack{\text{immer voller} \\ \text{Rang}}} s + \begin{pmatrix} F(x^k) \\ 0 \end{pmatrix} \|_2$$

Als Fixpunktiteration:

$$x^{k+1} = x^k - (F'(x^k)^T F'(x^k) + \mu^2 I)^{-1} \cdot F'(x^k) F(x^k)$$

7 Interpolation

Den Raum der Polynome vom Grad n bezeichnen wir als Π_n, und deren Elemente als $P_n(x)$. Wir haben Daten (z.B. Messwerte) $(x_i, f(x_i) = y_i)$, $i \in \{0, \ldots, n\}$ gegeben und wollen diese durch ein Polynom $P_n(x)$ darstellen: Es soll also $P_n(x_i) = y_i$ für alle i gelten. Ein solches Polynom existiert immer eindeutig.

7.1 Lagrange-Polynominterpolation

Die *Lagrange-Fundamentalpolynome* $\ell_i(x)$ bilden eine Orthogonalbasis von Π_n:

$$\ell_i(x) := \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}, \quad \text{dafür gilt} \quad \ell_i(x_j) = \delta_{ij} = \begin{cases} 1, & \text{falls } i = j \\ 0, & \text{falls } i \neq j \end{cases}$$

Dann ist das *Lagrange-Interpolationspolynom*:

$$P_n(f|x_0, \ldots, x_n) := \sum_{j=0}^{n} f(x_j)\ell_j = P_n(x)$$

7.2 Auswertung des Polynoms durch das Neville-Aitken-Schema

Wir wollen nun den Wert unseres Interpolationspolynoms an *einer festen Stelle* x auswerten, ohne das Polynom explizit aufzustellen und dann erst auszuwerten.

Wir definieren:

$$P_{i,k} := P(f|x_{i-k}, \ldots, x_i)(x),$$

also insbesondere

$$P_{i,0} = P(f|x_i)(x) = f(x_i) \quad \text{und} \quad P_{n,n} = P(f|x_0, \ldots, x_n)(x)$$

Das *Neville-Aitken-Schema* berechnet $P_n(f|x_0, \ldots, x_n)(x) = P_{n,n}$ rekursiv:

$$P_{i,0} = f(x_i)$$
$$P_{i,k} = P_{i,k-1} + \frac{x - x_i}{x_i - x_{i-k}}(P_{i,k-1} - P_{i-1,k-1})$$

	$P_{i,0}$	$P_{i,1}$	$P_{i,2}$	$\ldots$
x_0	$f(x_0)$			
x_1	$f(x_1)$	$P_{1,1}$		
x_2	$f(x_2)$	$P_{2,1}$	$P_{2,2}$	
$\vdots$	$\vdots$			
x_n	$f(x_n)$	$\ldots$	$\ldots$	$P_{n,n}$

7.3 Berechnung der Potenzform über ein LGS

Wir können das Interpolationspolynom $P_n(x)$ auch über das folgende LGS bestimmen, welches wegen der Vandermode-Matrix im Allgemeinen lächerlich schlecht konditioniert ist:

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}}_{\text{Vandermonde-Matrix } V_n} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

7.4 Auswertung der Potenzform mit dem Horner-Schema

Falls die Potenzform des Polynoms $P_n(x)$ vorliegt, kann diese durch geschickte Klammerung schnell (etwa halber Rechenaufwand) für ein x ausgewertet werden.

$$a_0 + a_1 x + \ldots + a_n x^n = a_0 + x \cdot (a_1 + x \cdot (a_2 + x \cdot (\ldots x \cdot (a_{n-1} + x a_n) \ldots)))$$

7.5 Newtonsche Interpolationsformel

Sei $P(f|x_0, \ldots, x_{n-1})$ gegeben und wir wollen nun mit möglichst wenig Aufwand $P(f|x_0, \ldots, x_n)$ berechnen.

Es gilt:

$$P(f|x_0, \ldots, x_n)(x) = P(f|x_0, \ldots, x_{n-1})(x) + \underbrace{\delta_n (x - x_0) \cdots (x - x_{n-1})}_{\textit{Korrekturterm}}$$

mit δ_n der Koeffizient der höchsten Potenz x^n, bzw. $[x_0, \ldots, x_n]f$ weil der Koeffizient eben von $x_0, \ldots, x_n$ abhängt:

$$\delta_n =: [x_0, \ldots, x_n]f =: \frac{f(x_n) - P(f|x_0, \ldots, x_{n-1})(x_n)}{(x_n - x_0) \ldots (x_n - x_{n-1})}$$

So lässt sich auch von einer einzigen Stützstelle (z.B. $P(f|x_0)$) aus das gesamte Interpolationspolynom $P(f|x_0, \ldots, x_n)$ rekursiv bestimmen. Wir erhalten die *Newtonsche Interpolationsformel*:

$$\begin{aligned} P(f|x_0, \ldots, x_n)(x) = {} & [x_0]f + (x - x_0) \cdot [x_0, x_1]f \\ & + (x - x_0)(x - x_1) \cdot [x_0, x_1, x_2]f \\ & + \ldots \\ & + (x - x_0) \cdots (x - x_{n-1}) \cdot [x_0, \ldots, x_n]f \end{aligned}$$

Die sich wiederholenden Polynome $\omega_i(x)$ bilden die *Newtonsche Basis* von Π_n:

$$\omega_0 := 1, \qquad \omega_1 := (x - x_0), \qquad \ldots, \qquad \omega_n := (x - x_0) \ldots (x - x_{n-1})$$

Für die Newtonsche Interpolationsformel haben wir das rekursive *Schema der dividierten Differenzen* für verschiedene Stützstellen $x_i, \ldots, x_k$:

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

	$[x_i]f$		$[x_i, x_{i+1}]f$		$[x_i, x_{i+1}, x_{i+2}]f$
x_0	$[x_0]f$				
		$>$	$[x_0, x_1]f$		
x_1	$[x_1]f$			$>$	$[x_0, x_1, x_2]f$
		$>$	$[x_1, x_2]f$		
x_2	$[x_2]f$				

7.6 Fehlerabschätzung

Der Fehler der Polynominterpolation $P_n(f|x_0, \ldots, x_n)(x)$ zur interpolierten Funktion $f(x)$, auf dem Intervall $[a, b]$, kann abgeschätzt werden durch:

$$|f(x) - P_n(x)| \leq |\omega_{n+1}(x)| \max_{\xi \in [a,b]} \frac{\left| f^{(n+1)}(\xi) \right|}{(n+1)!}$$

8 Numerische Differentiation

Wir reduzieren die numerische Differentiation auf Polynominterpolation: Die abzuleitende Funktion wird erst durch ein Polynom interpoliert. Das Interpolationspolynom kann dann leicht abgeleitet werden. Die n-te Ableitung eines Polynoms ist $n! \cdot$ (Koeffizient n). Bei der Newton-Interpolationsformel ist dieser n-te Koeffizient genau $[x_0, \ldots, x_n]f$. Wir erhalten:

$$f^{(n)}(x) \quad \approx \quad P(f|x_0, \ldots, x_n)^{(n)}(x) = n! \cdot [x_0, \ldots, x_n]f$$

Spezialfälle für äquidistante Stützstellen $x_j = x_0 + j \cdot h$ und $x \in [x_0, x_n]$:

$$\begin{aligned} f'(x) &\approx [x_0, x_1]f = \frac{f(x_1) - f(x_0)}{h} \\ f''(x) &\approx 2! \cdot [x_0, x_1, x_2]f \\ &= \frac{f(x_2) - 2 \cdot f(x_1) + f(x_0)}{h^2} \end{aligned}$$

Fehler ε in den Daten, $|f(y) - \tilde{f}(y)| \leq \varepsilon$, können zu Auslöschung führen. Für die zweite Ableitung mit äquidistanten Stützstellen mit Differenzenquotienten:

$$|\tilde{f}'' - f''(x)| \leq \underbrace{\frac{4 \cdot \varepsilon}{h^2}}_{\text{Rundungsfehler}} + \underbrace{c \cdot h^2}_{\text{Diskretisierungsfehler}}$$

Die Fehlerschranke wird für $h = \sqrt[4]{4 \cdot \varepsilon / c}$ minimal.

9 Numerische Integration

Der Ansatz bei der numerischen Integration ist einfach: Das Integral $\int_a^b f(x)\,dx$ wird in Teilstücke aufgeteilt und dann werden diese Teilintegrale mithilfe einer einfach integrierbaren Funktion approximiert.

Formal teilen wir also das Intervall $[a, b]$ in Teilintervalle $[t_{k-1}, t_k]$ auf und approximieren f durch ein g_k für jedes Intervall:

$$\int_a^b f(x)\,dx = \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} f(x)\,dx \approx \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} g_k(x)\,dx$$

9.1 Kondition

Zunächst die Kondition der Integration mit Approximation. Wir approximieren f durch $\tilde{f}$.

$$I = \int_a^b f(x)\,dx \qquad\qquad \tilde{I} = \int_a^b \tilde{f}(x)\,dx$$

Dann gilt: $|I - \tilde{I}| \leq (b - a)\|f - \tilde{f}\|_\infty$. Also ist die Kondition:

$$\frac{|I - \tilde{I}|}{|I|} \leq \underbrace{\frac{\int_a^b \|f\|_\infty dx}{\|\int_a^b f(x)dx\|}}_{\kappa_{\text{rel}}} \cdot \frac{\|f - \tilde{f}\|_\infty}{\|f\|_\infty}$$

9.2 Quadraturformeln

Eine Approximation einer Funktion f mit Stellen $x_0, \ldots, x_m \in [c,d]$ zur Integration wird *Quadraturformel* genannt:

$$I_m(f) := \int_c^d P(f|x_0, \ldots, x_m)(x)\, dx$$

Ist f ein Polynom mit Grad $\leq m$, dann gibt die Quadraturformel ein exaktes Ergebnis.

In die Quadraturformeln wird nun die Definition des Lagrange-Integrationspolynoms eingesetzt. Ein wenig Umformen ergibt die folgenden Formeln. Sei dazu $h = d - c$.

$$I_m(f) = h \sum_{j=0}^{m} c_j f(x_j) \qquad\qquad c_j := \frac{1}{h} \int_c^d \underbrace{\prod_{\substack{k=0 \\ k \neq j}}^{m} \frac{x - x_k}{x_j - x_k}}_{\ell_{jm}(x)} dx$$

Die Lagrange-Fundamentalpolynome ℓ_{jm} sind nur von den Stützstellen, aber nicht von den Funktionswerten abhängig.

9.3 Newton-Cotes-Formeln

Wir setzen die Stützstellen äquidistant mit der Form:

$$\begin{array}{cc} m = 0 & m > 0 \\ x_0 = c + \frac{1}{2}h & x_j = c + \frac{j}{m}h \end{array}$$

Man erhält dann die *Newton-Cotes-Formeln* mit normierten Stützstellen ξ_j und Gewichten c_j. Die Gewichte sind unabhängig vom Intervall $[c,d]$.

$$I_m(f) = h \sum_{j=0}^{m} c_j f(c + \xi_j h)$$

m		ξ_j	c_j	$I_m(f) - \int_c^d f(x)dx$
0	Mittelpunktsregel	$\frac{1}{2}$	1	$-\frac{1}{24}h^3 f^{(2)}(\xi)$
1	Trapezregel	$0, 1$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{12}h^3 f^{(2)}(\xi)$
2	Simpson-Regel	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$\frac{1}{90}\left(\frac{1}{2}h\right)^5 f^{(4)}(\xi)$
3	$\frac{3}{8}$-Regel	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{3}{80}\left(\frac{1}{3}h\right)^5 f^{(4)}(\xi)$
4	Milne-Regel	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{8}{945}\left(\frac{1}{4}h\right)^7 f^{(6)}(\xi)$

Man beachte, dass der Exaktheitsgrad der Newton-Cotes-Formeln immer ungerade ist, d.h. entweder m oder $m+1$.

Mit der Integralaufteilung, die in der Kapiteleinleitung beschrieben wurde, erhalten wir die *summierten Newton-Cotes-Formeln*. Jetzt ist $h = \frac{b-a}{n}$.

$$\int_a^b f(x)dx \quad \approx \quad \sum_{k=1}^{n} h \sum_{j=0}^{m} c_j f(a + (k-1)h + \xi_j h)$$

9.4 Gauß-Quadratur

Die *Gauß-Quadratur* erreicht einen Exaktheitsgrad von $2m + 1$, also hat doppelt so hohen Exaktheitsgrad wie benötigte Funktionsauswertungen.

Es existieren Stützstellen $x_0, \ldots, x_m \in (c, d)$ (offenes Intervall!) und positive Gewichte $\omega_0, \ldots, \omega_m$, so dass mit $h = d - c$ gilt:

$$h \sum_{i=0}^{m} \omega_i f(x_i) = \int_c^d f(x)dx + E_f(x)$$

Dabei ist $E_f(h)$ der Fehler mit passendem $\xi \in [c, d]$. Für $f \in \Pi_{2m+1}$ ist $E_f(h) = 0$.

$$\left|E_f(h)\right| = \frac{((m+1)!)^4}{((2m+2)!)^3(2m+3)}\, h^{2m+3} \left|f^{(2m+3)}(\xi)\right|$$

Wir schreiben auch $I_{k,n}$ für die Approximation des Integrals mit n Teilintervallen der Länge $\frac{b-a}{n} = h$, wobei in jedem Teilintervall die Gauß-Quadratur auf $k = m + 1$ Stützstellen angewandt wird.

Für den kleinsten Fehler: Mehr oder weniger Teilintervalle für größere Präzision? Für $I_{2k,n}$ und $I_{k,2n}$ wird die Anzahl der Funktionsauswertungen verdoppelt im Vergleich zu $I_{k,n}$. Außerdem kann man herausfinden, dass $|I - I_{2k,n}| \ll |I - I_{k,2n}|$. Also wird in der Praxis oft n klein gewählt, oft auch $n = 1$.

10 Lineare Gleichungssysteme: Iterative Lösungsverfahren

Für dünnbesetzte lineare Gleichungssysteme eignen sich iterative Lösungsverfahren. Dazu wird die folgende Fixpunktiteration durchgeführt, wobei A^{-1} durch C gut appproximiert wird.

$$x^{k+1} = x^k + C(b - Ax^k)$$

Die Iteration konvergiert, wenn für den Spektralradius $\rho(I - CA) < 1$ gilt. Der Spektralradius $\rho(A)$ ist der Betrag des betragsmäßig größten Eigenwerts von A.

Da $\rho(A) \leq \|A\|$ für jede Vektornorm-induzierte Matrixnorm $\|\cdot\|$ ist, konvergiert die Iteration also insbesondere, wenn $\|A\| < 1$ für eine beliebige Matrixnorm $\|\cdot\|$ ist.

Im Folgenden sei die Zerlegung $A = D - L - U$, wobei D nur Einträge auf, L unterhalb und U oberhalb der Diagonalen enthalten.

10.1 Jacobi-Verfahren

Beim *Jacobi-Verfahren* ist $C = D^{-1}$.

Für Konvergenz reicht das *starke Zeilensummenkriterium* (bzw. Spaltensummenkriterium):

$$\sum_{\substack{j=1 \\ j \neq i}} |A_{ij}| < |A_{ii}| \qquad \forall\, i \in \{\, 1, \ldots, n \,\}$$

10.2 Gauß-Seidel-Verfahren

Beim *Gauß-Seidel-Verfahren* ist $C = (D - L)^{-1}$. Konvergenz ist wie beim Jacobi-Verfahren.

$$x_i^{k+1} = \underbrace{a_{ii}^{-1}}_{D^{-1}} \left(b_i - \underbrace{\sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1}}_{L} - \underbrace{\sum_{j=i+1}^{n} a_{ij}\, x_j^{k}}_{U} \right)$$

10.3 SOR-Verfahren

Beim *Successive Over-Relaxation (SOR)*-Verfahren ist $C = (\frac{1}{\omega} D - L)^{-1}$ für ein $\omega \in (0, 2)$.

$$x_i^{k+1} = x_i^k + \omega \cdot \underbrace{a_{ii}^{-1}}_{D^{-1}} \left(b_i - \underbrace{\sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1}}_{L} - \underbrace{\sum_{j=i+1}^{n} a_{ij}\, x_j^{k}}_{U} \right)$$