Numerik 2

Hermann Schichl

Skriptum zur Vorlesung WS 2009/10

Inhaltsverzeichnis

1 Einleitung II

Dieser zweite Teil der Vorlesungsserie „Numerische Mathematik“ unterscheidet sich grundlegend vom ersten Teil in mehreren Gesichtspunkten.

Zum ersten enthält er in den Kapiteln über Optimierung (**??** und **??**), und Differentialgleichungen (**??** und 9) die Anfangsgründe für eigenständige große Forschungsgebiete. In den nächsten Jahren wird es einige Spezialvorlesungen geben, die sich mit diesen Gebieten näher beschäftigen werden.

Weiters treten in den Kapiteln über Statistik (7), mehrdimensionale Integration (8) und globale Optimierung (**??**) zum ersten Mal Algorithmen auf, die nicht mehr garantieren können, daß sie ein Resultat liefern. Es wird lediglich garantiert, daß mit zunehmender Laufzeit die Wahrscheinlichkeit, daß das Ergebnis gefunden wird, gegen Eins geht. Das ist einer der typischen Tricks der numerischen Mathematik: Wird ein Problem zu komplex, dann versuche zuerst es approximativ zu lösen. Ist auch dafür der Aufwand zu hoch, dann erfinde einen Algorithmus, der das Problem mit wachsender Wahrscheinlichkeit löst. Oft sind diese Methoden jedoch nur Zwischenschritte zu deterministischen Algorithmen, die mit ähnlichem Aufwand bessere Ergebnisse erzielen.

Darüber hinaus verlassen die analytischen Aufgaben, das Lösen von nichtlinearen Gleichungssystemen (Kapitel 5) und das Interpolieren (Kapitel 6), den „sicheren Hafen“ der Eindimensionalität. Wir werden sehen, daß das einen erstaunlich großen Unterschied macht, ja daß die Erhöhung der Dimension auch eine zusätzliche Dimension der Schwierigkeit bedeutet.

Besonders im Kapitel über die numerische Lösung von partiellen Differentialgleichungen werden wir auf ein anderes Phänomen treffen, das im ersten Teil nicht aufgetreten ist. Wir werden Methoden zur numerischen Lösung von mathematischen Problemen entwickeln, von denen wir (noch) nicht beweisen können, daß die Lösung eindeutig ist, ja oft nicht einmal, daß eine Lösung existiert. Mit Hilfe der numerischen Algorithmen werden wir jedoch Funktionen bestimmen können, die sich in den Anwendungen als verwendbar erweisen, auch wenn wir nicht beweisen können, daß sie Approximationen für die wirklichen Lösungen sind — was das auch immer bedeuten mag.

Schließlich, der wahrscheinlich auffälligste Unterschied liegt in den Anwendungen. Es gibt nur wenige Probleme, für deren Lösung man die Algorithmen des ersten Teils direkt verwenden kann, und diese sind meist starke Vereinfachungen oder kleine Teile von tatsächlich interessanten Modellen. Die Methoden und mathematischen Probleme des zweiten Teils sind dagegen durchaus geeignet, echte Anwendungsprobleme zu lösen — auch wenn viele der Verfahren noch verbessert und verfeinert werden können, auch wenn die moderne mathematische Forschung neue schnellere Algorithmen erfindet, deren genaue Funktionsweise nur in Spezialvorlesungen erklärt werden kann. Trotz allem werden die Verfahren aus dem ersten Teil nicht überflüssig — im Gegenteil. Lediglich der Gesichtspunkt wird verschoben. Die Methoden der linearen Algebra und der eindimensionalen Analysis werden die grundlegenden Bausteine sein, aus denen viele der noch zu entwickelnden Algorithmen zusammengesetzt werden. Die sorgfältige Untersuchung dieser Verfahren wird sich im Nachhinein noch als außerordentlich nützlich erweisen.

Bevor wir allerdings numerische Verfahren für einige der oben erwähnten mathematischen Probleme entwickeln, wollen wir einige der interessanten Anwendungen und die daraus resultierenden Modelle studieren.

2 Modellierung II

2.1 Beispiele

2.1.1 Architektur: Baustatik, Brückenbau, Festigkeitslehre

2.1.2 Biologie und Wirtschaft: Fischpopulationen

2.1.3 Pharmazie: Proteinfaltung

2.1.4 Wirtschaft: Buchung von Flugzeugsitzen

2.1.5 Informatik: Kryptographie

2.1.6 Graphik und Werbung: Photorealistische Darstellung

2.1.7 Biologie, Medizin: Epidemologie

2.1.8 Meteorologie: Wettervorhersage

2.1.9 Meteorologie: Klimamodelle, Ozonloch, Treibhauseffekt

2.1.10 Astrophysik: Sterne

2.1.11 Verkehr: Crash-Tests

2.1.12 Verkehr: Aerodynamik, Flugzeugbau

2.1.13 Physik: Gasdynamik

2.1.14 Physik: Halbleiter, Transportprobleme

2.2 Andere Anwendungen

.

3 Numerische Lineare Algebra III: Eigenwertprobleme

3.1 Grundlagen

Die Berechnung der Eigenwerte einer Matrix spielt in vielen Anwendungsgebieten eine große Rolle. Die Modelle **??** und **??** aus Kapitel **??** sind gute Beispiele dafür.

3.1.1 Normalformen

Eine Eigenwertzerlegung einer quadratischen Matrix $A \in \mathbb{K}^{n\times n}$ ist eine Faktorisierung

$$A = X\Lambda X^{-1},$$

wobei X eine reguläre Matrix, gebildet aus einer Basis von Eigenvektoren zu A, und $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ die aus den Eigenwerten zusammengesetzte Diagonalmatrix sind. Aus der linearen Algebra ist bekannt, daß nicht jede Matrix eine Eigenwertzerlegung besitzt.

Die Frage welche Matrizen diagonalisierbar sind (eine Eigenwertzerlegung besitzen) führt über den folgenden algebraischen Ausdruck:

Definition 3.1.1.1. *Das* charakteristische Polynom *der Matrix A ist definiert durch den Ausdruck*

$$p_A(x) := (-1)^n \det(A - x\mathbb{I}).$$

Das Vorzeichen wurde dabei so gewählt, dass p monisch ist. Die Nullstellen des charakteristischen Polynoms sind genau die Eigenwerte von A. Die Vielfachheit der Nullstelle heißt die algebraische Vielfachheit *des Eigenwertes.*

Die Menge aller Eigenvektoren zum fix gewählten Eigenwert λ bildet zusammen mit dem Nullvektor einen Vektorraum E_λ, den Eigenraum *zum Eigenwert λ. Die Dimension von E_λ, also die maximale Anzahl linear unabhängiger Eigenvektoren zu λ, heißt die* geometrische Vielfachheit *von λ.*

Stimmen für jeden Eigenwert geometrische und algebraische Vielfachheit überein, so existiert eine Basis aus Eigenvektoren, und daher ist A diagonalisierbar.

Definition 3.1.1.2. *Für jede invertierbare Matrix T ist die Abbildung*

$$A \mapsto TAT^{-1}$$

eine Ähnlichkeitstransformation*. Zwei Matrizen A und B heißen* ähnlich*, wenn es eine Ähnlichkeitstransformation gibt, die A und B verbindet; d.h. es gibt eine reguläre Matrix T mit $B = TAT^{-1}$.*

Das folgende Resultat ist auch bestens bekannt aus der linearen Algebra:

Proposition 3.1.1.3. *Zwei ähnliche Matrizen A und B haben dasselbe charakteristische Polynom, dieselben Eigenwerte mit denselben algebraischen und geometrischen Vielfachheiten.*

Beweis. Nachdem A und B ähnlich sind, existiert eine Matrix T mit $B = TAT^{-1}$. Dann ist aber

$$\begin{aligned} p_{TAT^{-1}}(x) &= \det(x\mathbb{I} - TAT^{-1}) = \det(T(x\mathbb{I} - A)T^{-1}) = \\ &= \det(T)\det(x\mathbb{I} - A)\det(T)^{-1} = \det(x\mathbb{I} - A) = p_A(x). \end{aligned}$$

Daher stimmen die Eigenwerte und deren algebraische Vielfachheiten überein. Ist E_λ der Eigenraum der Matrix A. Dann ist $V_\lambda := TE_\lambda$ der Eigenraum zum Eigenwert λ der Matrix $B = TAT^{-1}$. Weil T invertierbar ist, stimmen die Dimensionen von E_λ und V_λ überein. □

Es existieren jedoch Matrizen, die nicht diagonalisierbar sind. Für diese Matrizen existiert eine Verallgemeinerung der Eigenwertzerlegung, die *Jordan–Zerlegung*:

$$A = TJT^{-1},$$

wobei J eine Blockdiagonalmatrix ist aufgebaut aus sogenannten Jordanblöcken. Ein solcher Jordanblock hat die Gestalt

$$\begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda & 1 \\ 0 & \cdots & \cdots & 0 & \lambda \end{pmatrix}.$$

Es gibt jedoch noch andere Normalformen, die die Eigenwerte einer Matrix verraten. Die wichtigste von diesen ist die Schur–Zerlegung. Sie ist eine Verallgemeinerung des Satzes über symmetrische (hermitesche) Matrizen, der besagt, dass jede hermitesche Matrix A unitär diagonalisierbar ist, d.h. es gibt eine unitäre Matrix U mit

$$A = U\Lambda U^*.$$

Die Verallgemeinerung führt zur Schur–Zerlegung:

Theorem 3.1.1.4. *Jede quadratische Matrix $A \in \mathbb{K}^{n\times n}$ besitzt eine Faktorisierung der Form*

$$A = URU^*,$$

die Schur–Zerlegung*, mit einer unitären Matrix U und einer oberen Dreiecksmatrix R.*

Beweis. Der Beweis erfolgt mittels Induktion. Für $n = 1$ ist die Behauptung trivial. Nehmen wir also an, dass $n \geq 2$ ist. Sei x irgendein normierter Eigenvektor von A und λ der dazugehörige Eigenwert. Sei U_1 eine unitäre Matrix, deren erste Spalte mit x übereinstimmt. Dann hat $U_1^*AU_1$ die Gestalt

$$U_1^*AU_1 = \begin{pmatrix} \lambda & y \\ 0 & C \end{pmatrix}$$

mit einem Vektor y und einer $(n-1)\times(n-1)$–Matrix C. Nach der Induktionsannahme existiert eine Matrix U_2, die C Schur–faktorisiert $C = U_2R_2U_2^*$. Setzen wir

$$U = U_1 \begin{pmatrix} 1 & 0 \\ 0 & U_2 \end{pmatrix},$$

so gilt

$$U^*AU = \begin{pmatrix} \lambda & yU_2 \\ 0 & R_2 \end{pmatrix} =: R.$$

□

Ist A hermitesch, so stimmt die Schur–Zerlegung mit der Eigenwertzerlegung überein. Nachdem die Determinante einer oberen Dreiecksmatrix R genau das Produkt der Hauptdiagonalelemente ist, sind ebendiese Hauptdiagonalelemente (wegen der Gestalt des charakteristischen Polynoms) auch die Eigenwerte von R. Aus Proposition 3.1.1.3 folgt, dass man mit Hilfe einer Schur-Zerlegung die Eigenwerte einer beliebigen Matrix A bestimmen kann.

3.1.2 Allgemeines über Algorithmen zur Eigenwertberechnung

Obwohl die obigen Resultate einen großen theoretischen Wert haben, sind sie zur numerischen Bestimmung der Eigenwerte nur bedingt verwendbar. Besonders das Standardverfahren, das man in der linearen Algebra lernt, ist unbrauchbar. Dort berechnet man die Eigenwerte von A, indem man die Nullstellen des charakteristischen Polynomes $p_A(x)$ sucht. Die Gründe für die Unverwendbarkeit des Standardzuganges sind die folgenden: Erstens ist die Berechnung des charakteristischen Polynoms sehr aufwändig ($O(n!)$), und zweitens ist die Nullstellenberechnung von Polynomen numerisch sehr instabil (Nullstellen von Polynomen sind schlecht konditioniert).

Das charakteristische Polynom ermöglicht es jedoch einen wesentlichen mathematischen Aspekt der Eigenwertberechnung zu beleuchten. Sei

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

ein beliebiges monisches Polynom. Es gilt

$$p(x) = (-1)^n \det \begin{pmatrix} -x & & & & & -a_0 \\ 1 & -x & & & & -a_1 \\ & 1 & -x & & & -a_2 \\ & & \ddots & \ddots & & \vdots \\ & & & 1 & -x & -a_{n-2} \\ & & & & 1 & (-x-a_{n-1}) \end{pmatrix}$$

$$= \det(x\mathbb{I} - \begin{pmatrix} 0 & & & & & -a_0 \\ 1 & 0 & & & & -a_1 \\ & 1 & 0 & & & -a_2 \\ & & \ddots & \ddots & & \vdots \\ & & & 1 & 0 & -a_{n-2} \\ & & & & 1 & -a_{n-1} \end{pmatrix});$$

p ist also das charakteristische Polynom obiger Matrix. Ist λ eine Nullstelle von p, so ist λ auch ein Eigenwert von A und ein Eigenvektor zu λ ist $(1, \lambda, \lambda^2, \ldots, \lambda^{n-1})^*$. A heißt *Begleitmatrix* zu λ. Die Existenz dieser Begleitmatrix zeigt, daß das Bestimmen von Polynomnullstellen und die Berechnung von Eigenwerten äquivalente mathematische Probleme sind, und an dieser Stelle kommt ein algebraisches Resultat ins Spiel, das auf Arbeiten von Abel und Galois im neunzehnten Jahrhundert aufbaut und von Abel im Jahr 1824 bewiesen worden ist.

Theorem 3.1.2.1 (Abel 1824)**.** *Für jedes $n \geq 5$ existiert ein Polynom p mit rationalen Koeffizienten vom Grad n, das eine reelle Nullstelle r besitzt mit der Eigenschaft, daß r nicht geschrieben werden kann als algebraischer Ausdruck, der rationale Zahlen, Additionen, Subtraktionen, Multiplikationen, Divisionen und k–te Wurzeln enthält. Anders ausgedrückt existiert keine Formel und damit kein endlicher algebraischer Algorithmus, der aus den Koeffizienten eines Polynoms vom Grad $n \geq 5$ die Nullstellen berechnet.*

Der oben zitierte Satz impliziert über die Äquivalenz zwischen Eigenwerten und Nullstellen von Polynomen auch die Unmöglichkeit der Bestimmung der Eigenwerte allgemeiner Matrizen A mit Hilfe endlicher Algorithmen. Das hat die folgende Konsequenz: In den Kapiteln **??** und **??** haben wir Verfahren zur Lösung elementarer linear algebraischer Probleme kennengelernt, die könnten wir sie in exakter Arithmetik implementieren die behandelten Probleme exakt lösen würdeen. *Für Eigenwertberechnung ist eine solche Vorgangsweise unmöglich!* Daher muß jeder Algorithmus zur Lösung des Eigenwertproblemes approximativ sein. Fast alle Eigenwert–Algorithmen funktionieren *iterativ*.

Die meisten Verfahren zur Eigenwertberechnung versuchen eine Schur–Zerlegung von A zu berechnen, indem sie eine Folge von unitären Ähnlichkeitstranformationen mit Matrizen Q_i ausführen, sodaß das Produkt

$$\underbrace{Q_j^* \ldots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \ldots Q_j}_{Q}$$

gegen eine obere Dreiecksmatrix R konvergiert für $j \to \infty$.

Um den Aufwand zu minimieren geschieht das in zwei Phasen. Zuerst wird die Matrix A mittels unitären Ähnlichkeitstransformationen in Hessenbergform (siehe Abschnitt 3.2) transformiert und dann wird die Hessenbergmatrix H iterativ Schur–faktorisiert (siehe Abschnitt 3.3).

3.2 Transformationsverfahren

Abhängig von der Gestalt der Matrix A transformiert man sie mittels einem direkten Verfahren entweder in *Hessenberggestalt*, eine *Hessenbergmatrix* ist eine Matrix der Form

$$H = \begin{pmatrix} * & * & \cdots & \cdots & * \\ * & * & \ddots & & \vdots \\ 0 & * & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * & * \\ 0 & \cdots & 0 & * & * \end{pmatrix},$$

wenn A allgemeine Gestalt hat, oder in Tridiagonalform, wenn A hermitesch ist.

3.2.1 Transformation auf Hessenberggestalt

Die Transformation einer allgemeinen Matrix in Hessenbergform funktioniert analog zur QR–Zerlegung, die in Kapitel **??** vorgestellt worden ist. Man verwendet Householder–Transformationen Q_j um sukzessive die Nullen spaltenweise zu erzeugen. Anhand einer (5×5)–Matrix sei die Methode beschrieben (Hervorgehoben sind diejenigen Elemente, die sich im jeweiligen Schritt ändern.):

$$\underset{A}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}} \xrightarrow{Q_1^* \cdot} \underset{Q_1^* A}{\begin{pmatrix} * & * & * & * & * \\ \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ \boldsymbol{0} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ \boldsymbol{0} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ \boldsymbol{0} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \end{pmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^* A Q_1}{\begin{pmatrix} * & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ * & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \end{pmatrix}}$$

Man führt also zuerst einen Schritt wie bei einer QR–Zerlegung aus mit der einzigen Änderung, daß man bei der Wahl der Householdermatrix die erste Zeile von A ignoriert. Danach multipliziert man von rechts mit der hermitesch konjugierten der Householdermatrix. Dies läßt die erste Spalte unverändert, und daher bleiben die bereits erzeugten Nullen unberührt. Die weiteren Schritte sehen dann etwa folgendermaßen aus:

$$\underset{A}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{pmatrix}} \xrightarrow{Q_2^* \cdot} \underset{Q_2^* Q_1^* A Q_1}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & \boldsymbol{0} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & \boldsymbol{0} & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \end{pmatrix}} \xrightarrow{\cdot Q_2} \underset{Q_2^* Q_1^* A Q_1 Q_2}{\begin{pmatrix} * & * & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ * & * & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \\ & & \boldsymbol{*} & \boldsymbol{*} & \boldsymbol{*} \end{pmatrix}}$$

Wiederholt man $n-2$ solche Householderschritte, so erhält man eine unitäre Ähnlichkeitstransformation auf Hessenberggestalt

$$\underbrace{Q_{n-2}^* \ldots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \ldots Q_{n-2}}_{Q} =: H.$$

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix}$$

Der untenstehende Algorithmus ist eine Abwandlung von Algorithmus **??** aus Kapitel **??**. Der Algorith-

---

Hessenberg Transformation

for $k = 1$ **to** $n-2$ **do**
$x = A_{k+1:n,k}$
$v_k = e^{i\arg(x_1)}\|x\|_2 e_1 + x$
$v_k = v_k/\|v_k\|_2$
$A_{k+1:n,k:n} = A_{k+1:n,k:n} - 2v_k(v_k^* A_{k+1:n,k:n})$
$A_{1:n,k+1:n} = A_{1:n,k+1:n} - 2(A_{1:n,k+1:n}v_k)v_k^*$
done

---

mus unterscheidet sich nur durch die letzte Zeile innerhalb der Schleife vom Householder–Schmidt Verfahren zur Berechnung der *QR*–Zerlegung. Wieder wird das Produkt $Q = \prod_{i=1}^{n-2} Q_i$ nicht explizit berechnet. Sollten Matrix–Vektor–Produkte benötigt werden, können sie mittels der Algorithmen **??** und **??** aus Kapitel **??** iterativ bestimmt werden.

Der Aufwand zur Bestimmung der Hessenbergtransformierten ist in etwa doppelt so groß wie für eine *QR*–Zerlegung. Asymptotisch beträgt er

$$\sim \frac{10}{3}n^3.$$

3.2.2 Transformation auf Tridiagonalgestalt

Im hermiteschen Fall führen die unitären Ähnlichkeitstransformationen A wieder in eine hermitesche Matrix über. Wendet man also den Algorithmus zur Hessenbergtransformation auf eine hermitesche Matrix A an, so erhält man als Resultat eine hermitesche obere Hessenbergmatrix H, also eine hermitesche Tridiagonalmatrix T. In diesem Fall kann man die meisten Rechenschritte zur Berechnung der Produkte mit den Householdermatrizen von rechts einsparen, da man das Ergebnis bereits kennt, und man kann die Symmetrie von Ausgangs– und Resultatmatrizen verwenden, um den Aufwand auf

$$\sim \frac{4}{3}n^3$$

elementare Rechenoperationen zu reduzieren.

3.2.3 Gutartigkeit der Transformationen

Wie aus der Konstruktion zu erwarten ist, ist die Hessenbergtransformation ein gutartiger Algorithmus. Erstens wird mit unitären Matrizen gearbeitet und zweitens ist er sehr ähnlich zum Householder–Schmidt–Verfahren zur Berechnung einer *QR*–Zerlegung. In der Tat gilt der folgende Satz:

Theorem 3.2.3.1. *Sei die Hessenbergtransformation $A = QHQ^*$ der Matrix A aus Algorithmus 3.2.1 berechnet und seien die rundungsfehlerbehafteten Ergebnisse mit $\tilde{Q}$ und $\tilde{H}$ bezeichnet. In dieser Situation gilt*

$$\tilde{Q}\tilde{H}\tilde{Q}^* = A + \Delta A, \quad \frac{\|\Delta A\|}{\|A\|} = O(\mathsf{eps})$$

für eine Fehlermatrix ΔA.

3.3 QR–Verfahren

In diesem Abschnitt wollen wir das verbreitetste Verfahren zur Berechnung der Eigenwerte einer oberen Hessenbergmatrix vorstellen. Man kann den Algorithmus auch für allgemeine Matrizen A anwenden, doch der Aufwand ist in diesem Fall im allgemeinen $O(n^4)$ oder mehr.

Im folgenden wird der Algorithmus aus Motivationsgründen nur für reelle symmetrische Matrizen, also für Tridiagonalmatrizen, entwickelt; er läßt sich jedoch beinahe ohne Änderung auf Hessenbergmatrizen verallgemeinern. Das einzige, das wirklich geändert werden muß, ist die Wahl der richtigen Shiftstrategie, doch darauf wird in Abschnitt 3.3.6 genauer eingegangen.

3.3.1 Der Rayleigh–Quotient

Um Schätzwerte für Eigenwerte zu berechnen, ist der folgende Ausdruck wesentlich:

Definition 3.3.1.1. *Der* Rayleigh–Quotient *eines Vektors $x \in \mathbb{K}^n$ bezüglich der Matrix A ist der Skalar*

$$r(x) = \frac{x^* A x}{\langle x, x \rangle}.$$

Ist y Eigenvektor zum Eigenwert λ, so gilt $r(y) = \lambda$. Die Formel läßt sich aus der Lösung eines Kleinste–Quadrate–Problems motivieren. Man sucht die Zahl $r(x)$, die $\|Ax - r(x)x\|_2$ minimiert. Stellt man für dieses Problem die Normalengleichungen auf (x ist die Matrix, Ax ist der Vektor b, die rechte Seite), so erhält man $r(x)x^*x = x^*Ax$, und eine Umformung dieser Beziehung führt zum Rayleigh–Quotienten.

Betrachtet man $r(x)$ als Abbildung $\mathbb{K}^n \to \mathbb{R}$, so kann man den Gradienten berechnen. Eine einfache Anwendung der Quotientenregel liefert

$$\nabla r(x) = \frac{\nabla(x^*Ax)\langle x,x\rangle - x^*Ax\nabla(\langle x,x\rangle)}{\langle x,x\rangle^2} = \frac{2Ax}{\langle x,x\rangle} - \frac{2x(x^*Ax)}{\langle x,x\rangle^2} = \frac{2}{\langle x,x\rangle}\big(Ax - r(x)x\big).$$

Werten wir $\nabla r(x)$ wieder an einem Eigenvektor y zum Eigenwert λ aus, so ist das Resultat

$$\nabla r(y) = \frac{2}{\langle y,y\rangle}\big(\lambda y - \lambda y\big) = 0;$$

die Eigenvektoren sind also gerade die stationären Punkte der Funktion r. Aus dieser Tatsache und der Taylorentwicklung der Funktion r erhalten wir schließlich noch das Ergebnis

Proposition 3.3.1.2. *Sei q einer der Eigenvektoren von A. Dann gilt*

$$r(x) - r(q) = O(\|x-q\|_2^2), \quad \textit{für } x \to q.$$

Der Rayleigh–Quotient ist also ein Schätzwert für den Eigenwert zum Eigenvektor q, der in einer Umgebung von q quadratisch genau *ist.*

3.3.2 Potenziteration und inverse Iteration

Wie man der Abbildung 3.1 entnehmen kann, kann man erwarten, daß die Folge $\frac{v}{\|v\|}, \frac{Av}{\|Av\|}, \frac{A^2v}{\|A^2v\|}, \frac{A^3v}{\|A^3v\|}, \ldots$ mit einem (beinahe) beliebigen Startvektor v gegen einen Eigenvektor zum größten Eigenwert λ_1 konvergiert. Nehmen wir an, daß der Startvektor v der Folge nicht orthogonal auf q_1, den Eigenvektor zu λ_1 steht. Dann gilt eine Gleichung der Gestalt

$$\frac{v}{\|v\|} = a_1 q_1 + a_2 q_2 + \cdots + a_n q_n,$$

wobei die q_i die Eigenvektoren von A bezeichne. Die einzelnen Folgenglieder erfüllen dann die Beziehung

$$\frac{A^k v}{\|A^k v\|} = c_k(\lambda_1^k a_1 q_1 + \lambda_2^k a_2 q_2 + \cdots + \lambda_n^k a_n q_n).$$

für irgendeine Konstante c_k. Die Gleichung kann noch etwas umgeformt werden zu

$$\frac{A^k v}{\|A^k v\|} = c_k \lambda_1^k \left(a_1 q_1 + a_2 \left(\tfrac{\lambda_2}{\lambda_1}\right)^k q_2 + \cdots + a_n \left(\tfrac{\lambda_n}{\lambda_1}\right)^k q_n \right).$$

Weil die Konstanten c_k so gewählt sind, daß die Norm der Folgenglieder konstant gleich eins bleibt, erhalten wir den folgenden Satz.

Abbildung 3.1: Potenziteration

Theorem 3.3.2.1. *Unter den Voraussetzungen* $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$ *und* $\langle q_1, v\rangle \neq 0$ *konvergiert die Folge* $(v^{(k)} := A^k v/\|A^k v\|)$ *und die Folgenglieder erfüllen*

$$\|v^{(k)} - (\pm q_1)\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right), \quad |\lambda^{(k)} - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$$

für $k \to \infty$*, wobei* $\lambda^{(k)} := r(v^{(k)})$ *den zu* $v^{(k)}$ *gehörigen Rayleigh–Quotienten bezeichne. Das* $\pm$ *bedeutet, daß in jedem Schritt entweder die eine oder die andere Wahl des Vorzeichens die Abschätzung richtig macht.*

Den Algorithmus der Potenziteration kann man folgendermaßen formulieren: Das Abbruchkriterium ist

---

Potenziteration

$v^{(0)}$ ist beliebig mit $\|v^{(0)}\| = 1$.
for $k = 1$ **to** ? **do**
  $w = Av^{(k-1)}$
  $v^{(k)} = w/\|w\|$
  $\lambda^{(k)} = (v^{(k)})^\top Av^{(k)}$ `; Rayleigh--Quotient`
done

---

in diesem Fall nicht angegeben, da gerade an diesem Punkt sich Top–Qualitäts–Software auszeichnet. Der mathematische Hintergrund dazu ist meist nicht so ausgeprägt; es zählen eher Erfahrungstatsachen und statistische Untersuchungen.

Leider ist die Potenziteration nur von beschränktem Nutzen, da sie erstens nur den größten Eigenwert und einen dazupassenden Eigenvektor finden kann und zweitens die Konvergenz nur linear mit dem Faktor $|\lambda_2/\lambda_1|$ ist, der für viele Matrizen nahe bei 1 liegt; die Konvergenz ist also meist sehr langsam. Glücklicherweise lassen sich diese beiden Fakten durch einen einfachen Trick ändern.

Für jedes $\mu \in \mathbb{R}$, das nicht Eigenwert von A ist, stimmen die Eigenvektoren von A und $(A - \mu\mathbb{I})^{-1}$ überein. Ist nämlich $Ax = \lambda x$, so folgt

$$(A - \mu\mathbb{I})x = (\lambda - \mu)x \implies (\lambda - \mu)^{-1}x = (A - \mu\mathbb{I})^{-1}x.$$

Nehmen wir nun an, daß μ sehr nahe an einem bestimmten Eigenwert λ_j ist, so ist $(\lambda_j - \mu)^{-1}$ sehr viel größer als $(\lambda_i - \mu)^{-1}$ für $i \neq j$. Wenden wir also die Potenziteration statt auf die Matrix A auf die Matrix

$(A-\mu\mathbb{I})^{-1}$ an, so wird die erzeugte Folge relativ rasch gegen den zu λ_j gehörenden Eigenvektor q_j konvergieren. Das entstehende Verfahren heißt *inverse Iteration* und ist ein Standardverfahren zur Bestimmung eines Eigenvektors bei gegebenem Schätzwert für den Eigenwert. Aus Theorem 3.3.2.1 kann man sofort das folgende Theorem ablesen:

Theorem 3.3.2.2. *Sei λ_j der Eigenwert, der μ am nächsten liegt, und sei λ_k der zweitnächste. Dann gilt $|\mu-\lambda_j| < |\mu-\lambda_k| \le |\mu-\lambda_i|$ für $i \neq j$. Sei weiters $\langle q_j, v\rangle \neq 0$; dann konvergiert die Folge $((A-\mu\mathbb{I})^{-k}/\|(A-\mu\mathbb{I})^{-k}\|)$ und ihre Glieder $v^{(k)}$ erfüllen*

$$\|v^{(k)} - (\pm q_j)\| = O\left(\left|\frac{\mu-\lambda_j}{\mu-\lambda_k}\right|^k\right), \quad |\lambda^{(k)} - \lambda_j| = O\left(\left|\frac{\mu-\lambda_j}{\mu-\lambda_k}\right|^{2k}\right)$$

für $k \to \infty$, wobei $\lambda^{(k)}$ wieder die Rayleigh–Quotienten zu $v^{(k)}$ bezeichnet.

Diese Konvergenzbeschleunigung ist ein außerordentlich wichtiges Werkzeug bei der numerischen Eigenwert/Eigenvektor–Suche. Der folgende Algorithmus faßt sie zusammen:

Inverse Iteration

$v^{(0)}$ beliebig mit $\|v^{(0)}\| = 1$.
for $k = 1$ **to** ? **do**
  Löse $(A-\mu\mathbb{I})w = v^{(k-1)}$ nach w.
  $v^{(k)} = w/\|w\|$
  $\lambda^{(k)} = (v^{(k)})^\top A v^{(k)}$ `; Rayleigh--Quotient`
done

3.3.3 Rayleigh–Quotient Iteration

Wir kennen also jetzt eine Möglichkeit, einen Eigenwert zu schätzen, wenn wir einen Schätzwert für einen Eigenvektor kennen (den Rayleigh–Quotienten); und wir kennen eine Möglichkeit, einen Vektor zu berechnen, der näher an einem Eigenvektor liegt, wenn wir einen guten Schätzwert für einen Eigenwert kennen (die inverse Iteration). Eine naheliegende Idee ist also, die beiden Methoden miteinander zu kombinieren. Daraus resultiert die *Rayleigh–Quotient Iteration*.

Rayleigh–Quotient Iteration

$v^{(0)}$ beliebig mit $\|v^{(0)}\| = 1$.
$\lambda^{(0)} = (v^{(0)})^\top A v^{(0)}$
for $k = 1$ **to** ? **do**
  Löse $(A-\lambda^{(k-1)}\mathbb{I})w = v^{(k-1)}$ nach w.
  $v^{(k)} = w/\|w\|$
  $\lambda^{(k)} = (v^{(k)})^\top A v^{(k)}$ `; Rayleigh--Quotient`
done

Die kleine Änderung im Algorithmus, das Anpassen des Shiftwertes μ in jedem Schritt, hat dramatische Auswirkungen auf die Geschwindigkeit des Algorithmus. Jetzt kombinieren sich nämlich die lineare Konvergenz der inversen Iteration und die quadratische Genauigkeit des Rayleigh–Quotienten. Es ist also nicht weiter verwunderlich, daß das folgende Resultat gilt.

Theorem 3.3.3.1. *Die Rayleigh–Quotient Iteration konvergiert für fast alle (alle bis auf eine Menge vom Maß Null) Startwerte $v^{(0)} \in \mathbb{R}^n$ gegen ein Eigenwert/Eigenvektor–Paar. Wenn sie konvergiert, so konvergiert*

sie schließlich mit kubischer *Geschwindigkeit. Ist also $v^{(0)}$ genügend nahe am Grenzvektor q_j, so gelten*

$$\|v^{(k+1)} - (\pm q_j)\| = O(\|v^{(k)} - (\pm q_j)\|^3)$$

und

$$|\lambda^{(k+1)} - \lambda_j| = O(|\lambda^{(k)} - \lambda_j|^3)$$

für $k \to \infty$. Die Vorzeichen in der ersten Gleichung müssen auf beiden Seiten geeignet gewählt werden.

Um zu unterstreichen wie schnell kubische Konvergenz ist (sie verdreifacht in jedem Iterationsschritt die Anzahl der korrekten Stellen), sei das folgende Beispiel zitiert:

Beispiel 3.3.3.2. *Sei die Matrix*

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{pmatrix}$$

gegeben. Mit dem Startwert $v^{(0)} = 1/\sqrt{3}(1,1,1)^\top$ sind die ersten Glieder der Folge der $\lambda^{(k)}$:

$$\lambda^{(0)} = \underline{5}, \quad \lambda^{(1)} = \underline{5.21}31\ldots, \quad \lambda^{(2)} = \underline{5.214319743}184\ldots$$

Der wahre Wert des Eigenwertes zu demjenigen Eigenvektor, der $v^{(0)}$ am nächsten liegt, ist $\lambda = 5.214319743377$. Nach drei Iterationen ist also bereits zehnstellige Genauigkeit erreicht. Drei weitere Iterationsschritte würden bei genügend genauer Rechnung die Zahl der korrekten Stellen auf etwa 270 erhöhen.

Jeder Schritt der Rayleigh–Quotienten Iteration benötigt die Lösung eines linearen Gleichungssystems. Weil die Matrix in jedem Schritt verändert wird, kann man auch nicht im Vorhinein *LR*–zerlegen, und damit ist der Aufwand für einen Iterationsschritt $O(n^3)$.

Ist allerdings A eine obere Hessenbergmatrix, so senkt sich der Aufwand durch die große Anzahl von Nullen auf $O(n^2)$, und für Tridiagonalmatrizen gar auf $O(n)$, was die Wichtigkeit der ersten Phase, der Hessenbergtransformation, noch einmal unterstreicht.

3.3.4 QR–Verfahren ohne Shifts

Obwohl die Rayleigh–Quotienten Iteration eine hervorragende Möglichkeit bietet, einzelne Eigenwert/Eigenvektor–Paare effizient zu berechnen, ist das Verfahren doch zu aufwändig, wenn es darum geht, alle Eigenwerte einer Matrix zu berechnen. Der folgende absurd einfach wirkende Algorithmus ist der Start zur Lösung dieses Problems.

QR–Verfahren ohne Shifts

$A^{(0)} = A$
for $k = 1$ **to** ? **do**
  $Q^{(k)}R^{(k)} = A^{(k-1)}$; *QR*--`Zerlegung von` $A^{(k-1)}$
  $A^{(k)} = R^{(k)}Q^{(k)}$; `Ausmultiplizieren in umgekehrter Reihenfolge`
done

Auf den ersten Blick erscheint es unklar, warum dieser Algorithmus überhaupt etwas vernünftiges berechnen soll. Trotzdem steht die Behauptung, daß die Folge $(A^{(k)})$ gegen eine obere Dreiecksmatrix konvergiert (bzw. gegen eine Diagonalmartix im hermiteschen Fall). Das einzige, das man leicht überprüfen kann ist, daß wirklich unitäre Ähnlichkeitstransformationen ausgeführt werden:

$$A^{(k)} = {Q^{(k)}}^* A^{(k-1)} Q^{(k)}.$$

Der Rest hängt stark mit einem weiteren Verfahren zusammen, der gleichzeitigen Iteration, welche wiederum auf der Blockpotenziteration aufbaut.

Blockpotenziteration

Was mag passieren, wenn man bei der Potenziteration anstelle eines einzelnen Vektors eine Menge $\{v_0^{(0)}, \ldots, v_m^{(0)}\}$ von linear unabhängigen Vektoren gleichzeitig verwendet? Nachdem A regulär ist, sind die Mengen $\{A^k v_0^{(0)}, \ldots, A^k v_m^{(0)}\}$ jeweils wieder linear unabhängig. In Matrixnotation können wir das etwa folgendermaßen schreiben:

$$V^{(0)} := (v_0^{(0)} | \cdots | v_m^{(0)})$$

sei die Startmatrix. Definiere $V^{(k)}$ als

$$V^{(k)} := A^k V^{(0)} = (v_0^{(k)} | \cdots | v_m^{(k)}).$$

Nachdem unser Hauptinteresse dem Raum aufgespannt von den Spalten von $V^{(k)}$ gilt, bestimmen wir eine Orthonormalbasis dafür mit Hilfe einer reduzierten QR–Zerlegung

$$\hat{Q}^{(k)} \hat{R}^{(k)} = V^{(k)}.$$

Nachdem der von $V^{(k)}$ aufgespannte Raum aus Gründen, die analog sind zu den Gründen für die Konvergenz der Potenziteration, gegen den Raum konvergiert, der von den Eigenvektoren zu den m betragsgrößten Eigenwerten konvergiert, konvergiert die Matrix $\hat{Q}^{(k)}$ gegen $Q = (q_1 | \cdots | q_m)$, die Matrix gebildet aus ebendiesen Eigenvektoren.

Der Beweis dafür ist ebenfalls analog zum Beweis für die Konvergenz der Potenziteration. Man entwickelt alle $v_j^{(0)}$ in die Eigenvektoren von A und verwendet danach die Formeln für die Gram–Schmidt Orthonormalisierung. Es resultiert der folgende Satz:

Theorem 3.3.4.1. *Seien die m betragsgrößten Eigenwerte vom Betrag her verschieden*

$$|\lambda_1| > \cdots > |\lambda_m| \geq |\lambda_{m+1}| \geq \cdots \geq 0,$$

und sind die m führenden Hauptminoren der Matrix $\hat{Q}^\top V^{(0)}$ nichtverschwindend, wobei $\hat{Q} = (q_1 | \cdots | q_m)$ ist. Dann konvergieren die Spalten der Matrizen $Q^{(k)}$ mit linear gegen die Eigenvektoren zu den m betragsgrößten Eigenwerten von A:

$$\|q_j^{(k)} - (\pm q_j)\| = O(C^k),$$

für $1 \leq j \leq m$, wobei

$$C = \max_{1 \leq k \leq m} \frac{|\lambda_{k+1}|}{|\lambda_k|} < 1$$

ist. Wieder muß man das Vorzeichen $\pm$ in jedem Schritt und für jedes j geeignet wählen.

Beweis. Sei $\hat{Q}$ ergänzt zu einer orthogonalen Matrix Q aus Eigenvektoren von A. Dann gilt $A = Q\Lambda Q^\top$, und sei $\hat{\Lambda}$ der führende $(m \times m)$–Teil der Matrix Λ. Dann gilt

$$V^{(k)} = A^k V^{(0)} = Q\Lambda^k Q^\top V^{(0)} = \hat{Q}\hat{\Lambda}^k \hat{Q}^{(k)} V^{(0)} + O(|\lambda_{m+1}|^k)$$

für $k \to \infty$. Aus der Hauptminorenbedingung folgt, daß $\hat{Q}^\top V^{(0)}$ regulär ist. Daher kann man die obige Gleichung umformen zu

$$V^{(k)} = \big(\hat{Q}\hat{\Lambda}^k + O(|\lambda_{m+1}|^k)\big)\hat{Q}^\top V^{(0)}.$$

Nachdem $\hat{Q}^\top V^{(0)}$ regulär ist, ist der Spaltenraum von $V^{(k)}$ derselbe wie der Spaltenraum von

$$\hat{Q}\hat{\Lambda}^k + O(|\lambda_{m+1}|^k).$$

Klarerweise konvergiert dieser Raum linear gegen den Spaltenraum von $\hat{Q}$. Nachdem das Argument auch für alle führenden Teile der Matrix $V^{(k)}$ gilt, weil jeder der ersten m Hauptminoren ungleich Null ist, konvergieren auch alle führenden Teilmengen der Spalten von $V^{(k)}$ gegen die entsprechende führende Spaltenmenge von $\hat{Q}$. Aus der Definition der QR–Zerlegung folgt der Rest der Behauptungen. □

Gleichzeitige Iteration

Für numerische Zwecke ist das Verfahren der Blockpotenziteration denkbar ungünstig, da jeder einzelne der Vektoren $v_j^{(0)}$ gegen ein Vielfaches des Eigenvektors zum betragsgrößten Eigenwert konvergiert. Die Kondition der Basis $\{v_0^{(k)}, \ldots, v_m^{(k)}\}$ wird also mit zunehmendem k immer schlechter. Die Rundungsfehler würden also mit der Zeit die Informationen zerstören. Die Lösung des Problemes ist jedoch simpel. Man orthonormalisiert die Basis in jedem Schritt der Iteration. Das verändert den aufgespannten Raum nicht, verhindert aber die Verschlechterung der Kondition. Dieses Vorgehen führt zum Algorithmus der *gleichzeitigen Iteration*.

Gleichzeitige Iteration

Wähle $Q^{(0)}$ mit orthogonalen Spalten
for $k = 1$ **to** ? **do**
  $Z = A\hat{Q}^{(k-1)}$
  $\hat{Q}^{(k)}\hat{R}^{(k)} = Z$; `reduzierte QR--Zerlegung` **done**

Theorem 3.3.4.2. *Aus der Form dieses Algorithmus folgt, daß er unter den Voraussetzungen von Theorem 3.3.4.1 auch analoge Ergebnisse wie in Theorem 3.3.4.1 gelten.*

Zusammenhang zwischen gleichzeitiger Iteration und *QR*–Verfahren

Das Ziel dieses Abschnittes ist es zu zeigen, daß das *QR*–Verfahren äquivalent ist zur gleichzeitigen Iteration angewendet auf die Startmatrix $Q^{(0)} = \mathbb{I}$. Da die Startmatrix und damit auch alle iterierten Matrizen $Q^{(k)}$ quadratisch sind, muß man keine reduzierten *QR*–Zerlegungen berechnen sondern kann volle *QR*–Faktorisierungen verwenden. Im folgenden seien die erzeugten Matrizen statt mit $\hat{Q}^{(k)}$ mit $\underline{Q}^k$ und statt $\hat{R}^{(k)}$ mit $R^{(k)}$ bezeichnet. Verfolgt man mit diesen Bezeichnungen den Algorithmus 3.3.4, so kann man die folgenden Beziehungen zwischen den einzelnen Matrizen ablesen. Die Gleichungen (3.1), (3.2) und (3.3) folgen direkt aus dem Algorithmus. (3.4) ist eine Definition für die Matrizen $A^{(k)}$:

$$\underline{Q}^{(0)} = \mathbb{I} \tag{3.1}$$
$$Z^{(k)} = A\underline{Q}^{(k-1)} \tag{3.2}$$
$$Z^{(k)} = \underline{Q}^{(k)}R^{(k)} \tag{3.3}$$
$$A^{(k)} = \left(\underline{Q}^{(k)}\right)^\top A\underline{Q}^{(k)} \tag{3.4}$$

Auf ähnliche Weise kann man den *QR*–Algorithmus in Gleichungen zusammenfassen. Beziehung 3.8 dient wieder als Definition der Matrix $\underline{Q}^{(k)}$:

$$A^{(0)} = A \tag{3.5}$$
$$A^{(k-1)} = Q^{(k)}R^{(k)} \tag{3.6}$$
$$A^{(k)} = R^{(k)}Q^{(k)} \tag{3.7}$$
$$\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\ldots Q^{(k)} \tag{3.8}$$

Die Übereinstimmung in den Bezeichnungen $A^{(j)}$ und $\underline{Q}^{(j)}$ in den Gleichungen für beide Algorithmen wurden nicht von ungefähr gewählt. Der nächste Satz wird zeigen, daß es sich dabei um jeweils dieselben Matrizen handelt. Für beide Algorithmen sei zusätzlich noch die Bezeichnung

$$\underline{R}^{(k)} = R^{(k)}R^{(k-1)}\ldots R^{(1)} \tag{3.9}$$

eingeführt. Mit Hilfe dieser Notation kann man den folgenden Satz beweisen

Theorem 3.3.4.3. *Die Algorithmen 3.3.4 und 3.3.4 erzeugen identische Folgen von Matrizen $\underline{R}^{(k)}$, $\underline{Q}^{(k)}$ und $A^{(k)}$, die definiert sind durch die QR–Faktorisierung der k–ten Potenzen von A*

$$A^k = \underline{Q}^{(k)}\underline{R}^{(k)}.$$

$A^{(k)}$ ist die Projektion

$$A^{(k)} = \left(\underline{Q}^{(k)}\right)^\top A\underline{Q}^{(k)}.$$

Beweis. Der Beweis erfolgt mittels Induktion nach k. Für $k = 0$ ist die Übereinstimmung offensichtlich. Für beide Algorithmen gilt nämlich $A^0 = \underline{Q}^{(0)} = \underline{R}^{(0)} = \mathbb{I}$ und $A^{(0)} = A$, wie man leicht aus den Gleichungen (3.1), (3.4), (3.5) und (3.9) ablesen kann. Für $k \geq 1$ folgt für die gleichzeitige Iteration

$$A^k = A\underline{Q}^{(k-1)}\underline{R}^{(k-1)} = Z^{(k)}\underline{R}^{(k-1)} = \underline{Q}^{(k)}R^{(k)}\underline{R}^{(k-1)} = \underline{Q}^{(k)}\underline{R}^{(k)},$$

wobei man in dieser Reihenfolge die Induktionsvoraussetzung und die Gleichungen (3.2), (3.3), (3.9) verwendet.

Andererseits beweist man für das *QR*–Verfahren

$$\begin{aligned} A^k &= A\underline{Q}^{(k-1)}\underline{R}^{(k-1)} = A^{(0)}Q^{(1)}Q^{(2)}\dots Q^{(k-1)}\underline{R}^{(k-1)} = \\ &= Q^{(1)}A^{(1)}Q^{(2)}\dots Q^{(k-1)}\underline{R}^{(k-1)} = \dots = \\ &= Q^{(1)}Q^{(2)}\dots Q^{(k-1)}A^{(k-1)}\underline{R}^{(k-1)} = \underline{Q}^{(k-1)}Q^{(k)}R^{(k)}\underline{R}^{(k-1)} = \underline{Q}^{(k)}\underline{R}^{(k)}, \end{aligned}$$

wobei man in dieser Reihenfolge die Induktionsvoraussetzung, dann die Gleichungen (3.8), $k-1$ Mal die Gleichungen (3.6) und (3.7), wieder (3.7) und zuletzt (3.9) verwendet.

Damit ist die Äquivalenz der Algorithmen gezeigt und die erste Behauptung. Die zweite Behauptung folgt zum Beispiel aus (3.4). □

Aus der Äquivalenz von *QR*–Verfahren und gleichzeitiger Iteration können wir auch Resultate über die Konvergenz des *QR*–Verfahrens herleiten. Aus (3.4) folgt, daß die Diagonalelemente der Matrizen $A^{(j)}$ Rayleigh–Quotienten von A zu den Spalten von $\underline{Q}^{(k)}$ sind. Wenn diese Spalten gegen Eigenvektoren von A konvergieren, konvergieren die Hauptdiagonalelemente gegen die zugehörigen Eigenwerte. Die Elemente abseits der Hauptdiagonale werden aus inneren Produkten zweier verschiedener Spalten von $\underline{Q}^{(k)}$ gebildet, und da diese Spalten orthogonal werden, verschwinden sie im Grenzwert. Also konvergiert $A^{(k)}$ gegen eine Diagonalmatrix. Ferner folgt aus Theorem 3.3.4.2 das Resultat über die Konvergenz des *QR*–Verfahrens

Theorem 3.3.4.4. *Sei der Algorithmus 3.3.4 angewendet auf die reelle symmetrische Matrix A, deren Eigenwerte*

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$$

erfüllen und deren entsprechende Eigenvektormatrix Q nichtverschwindende Hauptminoren hat. Dann gilt für $k \to \infty$, daß $A^{(k)}$ linear mit Konstante $\max_k |\lambda_{k+1}|/|\lambda_k|$ gegen $\operatorname{diag}(\lambda_1, \dots, \lambda_n)$ konvergiert. Ferner konvergiert $\underline{Q}^{(k)}$ mit derselben Geschwindigkeit gegen Q.

3.3.5 QR–Verfahren mit Shifts

Niemand würde sich für das *QR*–Verfahren interessieren, wenn die erreichbare Konvergenzordnung nur linear ist. Doch ein Trick, der dem Übergang von der Potenziteration zur Rayleigh–Quotienten Iteration entspricht, verleiht dem Algorithmus Flügel. Es ist dies die Einführung von Shifts $A \to A - \mu\mathbb{I}$. Eine weitere Verbesserung, die *Deflation* zerteilt die Matrix $A^{(k)}$ in Untermatrizen. Der praktisch verwendete *QR*–Algorithmus ist im folgenden zusammengefaßt. Dieser Algorithmus ist das Standardverfahren zur Bestimmung der Eigenwerte und Eigenvektoren einer Matrix A seit Anfang der Sechzigerjahre. Erst zu Beginn der Neunzigerjahre ist ein neues Verfahren entwickelt worden, das beginnt das *QR*–Verfahren zu verdrängen, die Divide-and-conquer Algorithmen, die in Abschnitt 3.4.3 kurz beschrieben werden.

QR–Verfahren mit Shifts

Wähle eine Toleranz ε.
$\left(Q^{(0)}\right)^\top A^{(0)} Q^{(0)} = A$; `tridiagonalisieren von` A
for $k = 1$ **to** ? **do**
  Wähle einen Shift $\mu^{(k)}$
  $Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} \mathbb{I}$; *QR*`--Zerlegung`
  $A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} \mathbb{I}$
  if $A_{j,j+1} < \varepsilon$ **then**
    $A_{j,j+1} = 0$
    $A_{j+1,j} = 0$
    Zerlege $A^{(k)}$ in zwei Teilmatrizen $\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}$
    Wende das *QR*–Verfahren ab nun auf A_1 und A_2 getrennt an.
  endif
done

Definiert man $\underline{Q}^{(k)}$ und $\underline{R}^{(k)}$ analog zu Abschnitt 3.3.4, so kann man leicht herleiten, daß $\underline{Q}^{(k)}$ und $\underline{R}^{(k)}$ eine *QR*–Zerlegung einer geshifteten Potenz von A bilden:

$$(A - \mu^{(k)}\mathbb{I})(A - \mu^{(k-1)}\mathbb{I}) \dots (A - \mu^{(1)}\mathbb{I}) = \underline{Q}^{(k)} \underline{R}^{(k)}.$$

Die erste Spalte von $\underline{Q}^{(k)}$ ist das Resultat einer wiederholt geshifteten Potenziteration mit dem Startvektor e_1. Die letzte Spalte ist das Resultat einer wiederholt geshifteten inversen Iteration angewendet auf den Startvektor e_n. Sind die Shifts gute Schätzwerte für einen Eigenwert, so konvergiert diese letzte Spalte sehr schnell gegen einen Eigenvektor.

Wie man $\mu^{(k)}$ am besten wählt, ist Untersuchungsgegenstand des nächsten Abschnittes.

3.3.6 Shift–Strategien

Aus den Resultaten über die Rayleigh–Quotienten Iteration kann man herauslesen was sehr gute Eigenwertschätzwerte $\mu^{(k)}$ sind. Man wählt am besten den Rayleigh–Quotienten

$$\mu^{(k)} = \frac{\left(q_n^{(k)}\right)^\top A q_n^{(k)}}{q_n^{(k)}} = \left(q_n^{(k)}\right)^\top A q_n^{(k)}.$$

Wie wir schon zuvor festgestellt haben, benötigt man keinen weiteren Rechenaufwand zur Bestimmung von $\mu^{(k)}$. Es gilt nämlich

$$\left(q_n^{(k)}\right)^\top A q_n^{(k)} = e_n^\top \underline{Q}^{(k)\top} A \underline{Q}^{(k)} e_n = e_n^\top A^{(k)} e_n = A_{nn}^{(k)},$$

man kann also den benötigten Rayleigh–Quotienten von der Hauptdiagonale der Matrix $A^{(k)}$ ablesen. Die Wahl $\mu^{(k)} = A_{nn}^{(k)}$ heißt wegen dieser Zusammenhänge auch *Rayleigh–Quotienten Shift*. Aus den Verbindungen zwischen *QR*–Verfahren mit Shifts und der Rayleigh–Quotienten Iteration folgt, daß das *QR*–Verfahren mit Rayleigh–Quotienten Shift schließlich kubisch konvergiert. Trotz der phantastischen Konvergenzgeschwindigkeit ist dieser Shift nicht erste Wahl.

Der Grund dafür ist, daß die Konvergenz des Verfahrens nicht garantiert werden kann. Betrachtet man nämlich die Matrix

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

so sieht man daß das *QR*–Verfahren ohne Shift nicht konvergiert:

$$A = Q^{(1)}R^{(1)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$A^{(1)} = R^{(1)}Q^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = A.$$

Der Rayleigh–Quotienten Shift $\mu = A_{nn}$ hat keinen Effekt, da $A_{nn} = 0$, also versagt auch das *QR*–Verfahren mit Rayleigh–Quotienten Shift. Das Problem ist, daß die Eigenwerte der Matrix gleich 1 und -1 sind. Soll also die Schätzung 0 für die Eigenwerte verbessert werden, so ist wegen der Symmetrie unklar welcher Eigenwert favorisiert werden soll, und der Schätzwert wird gar nicht verbessert. In so einem Fall versagt der Rayleigh–Quotienten Shift, und das *QR*–Verfahren konvergiert mit dieser Shift–Strategie gar nicht.

Um Symmetrien dieser Art zu brechen untersuchen wir die rechte untere 2×2 Teilmatrix von $A^{(k)}$:

$$B = \begin{pmatrix} A_{n-1,n-1} & A_{n-1,n} \\ A_{n,n-1} & A_{nn} \end{pmatrix}.$$

Der *Wilkinson–Shift* ist definiert als derjenige Eigenwert von B, der näher an A_{nn} liegt. Im Fall eines Unentschiedens wählt man irgendeinen. Eine numerisch stabile Formel für den Wilkinson–Shift ist

$$\mu = A_{nn} - \frac{\operatorname{sgn}(\delta) A_{n,n-1}^2}{|\delta| + \sqrt{\delta^2 + A_{n-1,n-1}^2}},$$

wobei $\delta = \frac{1}{2}(A_{n-1,n-1} - A_{nn})$ ist. Sollte $\delta = 0$ gelten, so setzt man $\operatorname{sgn}(\delta) = 1$. Im generischen Fall erzielt der Wilkinson–Shift ebenso wie der Rayleigh–Quotienten Shift kubische Konvergenz. Außerdem gilt, daß im schlimmsten Fall wenigstens noch quadratische Konvergenz erreicht wird, das *QR*–Verfahren zusammen mit dem Wilkinson–Shift konvergiert also im Gegensatz zum Rayleigh–Quotienten Shift immer.

Im obigen Beispiel ist der Wilkinson–Shift gleich 1, und die Konvergenz erfolgt in einem Schritt.

Es gibt noch andere Shift–Strategien, besonders solche, die die explizite Subtraktion $A^{(k-1)} - \mu^{(k)}\mathbb{I}$ vermeiden, um Auslöschungseffekte zu minimieren. Solche *implizite Shift–Strategien* verbreiten sich in der letzten Zeit immer mehr. Sie sind aber vor allem dann von Vorteil, wenn unsymmetrische Hessenbergmatrizen Schur–faktorisiert werden sollen. Im unsymmetrischen Fall muß auch das Auftreten komplexer Eigenwerte in Betracht gezogen werden. Ein *QR*–Doppelschritt ist dann die Lösung. Genauere Beschreibungen dieser Verfahren können etwa in [Golub, Van Loan 1996, chapters 7 and 8] nachgeschlagen werden.

Zum Abschluß der Abhandlungen über das *QR*–Verfahren muß noch etwas über die Gutartigkeit des Algorithmus ausgesagt werden.

Theorem 3.3.6.1. *Sei eine reelle symmetrische Matrix $A \in \mathbb{K}^{n\times n}$ diagonalisiert mittels Algorithmus 3.3.5, wobei $\mu^{(k)}$ der Wilkinson–Shift sei, und es mögen $\tilde{\Lambda}$ und $\tilde{Q}$ die berechneten Faktoren bezeichen. Dann gilt*

$$\tilde{Q}\tilde{\Lambda}\tilde{Q}^* = A + \Delta A, \quad \frac{\|\Delta A\|}{\|A\|} = O(\mathsf{eps})$$

für eine geeignete Fehlermatrix ΔA. Daher erfüllen die bestimmten Eigenwerte $\tilde{\lambda}_j$ die Abschätzung

$$\frac{|\tilde{\lambda}_j - \lambda_j}{\|A\|} = O(\mathsf{eps}).$$

Der Algorithmus ist also gutartig. Der Aufwand von $\sim \frac{4}{3}n^3$ ist umso bemerkenswerter als er etwa ein Drittel des Aufwandes zur standardmäßigen Berechnung eines Matrixproduktes zweier $n \times n$ Matrizen ist.

3.4 Andere Verfahren

Im Laufe der Zeit sind neben dem *QR*–Verfahren noch andere Algorithmen zur Eigenwertberechnung interessant gewesen oder geworden.

3.4.1 Jacobi Verfahren

Der älteste Algorithmus wurde 1845 von Jacobi konstruiert. Er basiert auf der Tatsache, daß zwar Matrizen der Größe 5×5 oder mehr nicht explizit diagonalisiert werden können, kleinere Matrizen abar schon.

Im Jacobi Verfahren diagonalisiert man immer wieder 2×2 Teilmatrizen in der Form

$$J^\top \begin{pmatrix} a & b \\ b & c \end{pmatrix} J = \begin{pmatrix} \neq 0 & 0 \\ 0 & \neq 0 \end{pmatrix}$$

und hofft, daß dabei die gesamte Matrix gegen eine Diagonalmatrix konvergiert. Das Standardverfahren wählt für J Rotationsmatrizen der Form

$$J = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

mit

$$\tan(2\theta) = \frac{2b}{c-a};$$

solch eine Matrix wird in der Literatur mit *Jacobirotation* bezeichnet.

Als zu diagonalisierende Untermatrix wählt man üblicherweise diejenige 2×2 Teilmatrix, die das betragsgrößte Nebendiagonalelement enthält. Man kann zeigen, daß bei dieser Wahl die Matrix mit quadratischer Geschwindigkeit gegen eine Diagonalmatrix konvergiert. Die Konvergenzordnung ist also 2 im Gegensatz zu 3 beim *QR*–Verfahren.

Der Hauptvorteil des Jacobi–Verfahrens ist die gute Parallelisierbarkeit. Nachdem in jedem Schritt immer nur zwei Zeilen und Spalten betroffen sind, kann man gleichzeitig mehrere Jacobirotationen beinahe unabhängig voneinander anwenden. Ist die Zahl der zur Verfügung stehenden CPUs sehr groß, so übertrifft das Jacobi–Verfahren das nur schlecht parallelisierbare *QR*–Verfahren. Seit dem Aufkommen von Divide–and–conquer Algorithmen, die ebenfalls gut parallelisierbar sind aber viel schneller konvergieren, ist die Bedeutung des Jacobi–Verfahrens stark zurückgegangen.

3.4.2 Bisektion

Das Bisektionsverfahren dient vor allem der Bestimmung eines kleinen Teils der Eigenwerte. Mit Hilfe der *Bisektion* kann man etwa die größten 5 Eigenwerte, die kleinsten 10 oder alle im Intervall $[a,b]$ finden. Sind die gesuchten Eigenwerte bestimmt, so kann man die zugehörigen Eigenvektoren durch einen inversen Iterationsschritt berechnen.

Die Idee zur Bestimmung der Eigenwerte liegt darin, die Nullstellen des charakteristischen Polynoms $p_A(x) = \det(x\mathbb{I} - A)$ zu suchen. Ja, obwohl a priori die Nullstellenbestimmung bei Polynomen instabil ist! Man versucht jedoch nicht, die Nullstellen des Polynoms aus den Koeffizienten zu berechnen (das ist instabil), sondern verwendet eine etwas subtilere Variante.

Beginnen wir mit einer irreduziblen symmetrischen Tridiagonalmatrix A (irreduzibel bedeutet, daß kein Nebendiagonalelement verschwindet). Andernfalls könnte man A auf Tridiagonalgestalt transformieren und in Teilmatrizen zerlegen. Mit $A^{(1)}, A^{(2)}, \ldots$ seien die Hauptuntermatrizen von A bezeichnet. Das wesentliche mathematische Resultat ist jetzt, daß die Eigenwerte von $A^{(k)}$ und die Eigenwerte von $A^{(k+1)}$ folgende Gleichung erfüllen:

$$\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k+1)};$$

die Eigenwerte von $A^{(k)}$ liegen also zwischen denen von $A^{(k+1)}$. Diese Eigenschaft ist es, die einem ermöglicht, die Anzahl der Eigenwerte in einem gegebenen Intervall zu zählen.

Es gilt nämlich, daß die Anzahl der negativen Eigenwerte der Tridiagonalmatrix $A \in \mathbb{K}^{n\times n}$ gleich der Anzahl der Vorzeichenwechsel in der Folge

$$1, \det(A^{(1)}), \det(A^{(2)}), \ldots, \det(A^{(n)})$$

ist; diese Folge ist auch unter dem Namen *Sturmsche Kette* bekannt. Tritt 0 in der Folge auf, so definiert man als Vorzeichenwechsel auch einen Übergang von 0 auf $-$ oder $+$ aber nicht einen Übergang von $-$ oder $+$ auf 0. Indem man A um ein Vielfaches der Identität verschiebt, kann man mit dieser Methode auch die Anzahl der Eigenwerte im Intervall $(-\infty, b)$ oder im Intervall (a, ∞) zählen. Schneidet man diese Intervalle, so ist es auch möglich, die Anzahl der Eigenwerte in Intervallen der Form $[a, b)$ zu bestimmen.

Eine weitere wesentliche Beobachtung ist, daß die Determinanten von $A^{(k)}$, $A^{(k-1)}$ und $A^{(k-2)}$ für eine Tridiagonalmatrix durch eine Drei–Terme–Rekurrenz zusammenhängen:

$$\det(A^{(k)}) = A_{kk} \det(A^{(k-1)}) - A_{k-1,k}^2 \det(A^{(k-2)}),$$

und damit kann man auch das charakteristische Polynom mit relativ geringem Aufwand berechnen:

$$p^{(k)}(x) = (A_{kk} - x) p^{(k-1)}(x) - A_{k-1,k}^2 p^{(k-2)}(x)$$

ist eine Rekurrenz, die die charakteristischen Polynome für die einzelnen Teilmatrizen $A^{(k)}$ zueinander in Beziehung setzt. Als Anfangswerte setzt man $p^{(-1)}(x) = 0$ und $p^{(0)}(x) = 1$. Wertet man sukzessive die Polynome $p^{(k)}(x)$ an verschiedenen Stellen x aus und zählt man Vorzeichenwechsel, so kann man beliebig kleine Intervalleinschließungen für alle Eigenwerte von A berechnen.

3.4.3 Divide–and–conquer

Die modernsten Verfahren zur Eigenwertberechnung sind die sogenannten Divide–and–conquer Algorithmen. Sie stellen die bedeutendste Innovation auf dem Gebiet der Eigenwertberechnung dar seit Erfindung des *QR*–Verfahrens in den frühen Sechzigerjahren. Von Cuppen 1981 eingeführt dauerte es etwa zehn Jahre bis sich der Algorithmus durchsetzte. Obwohl er etwa zweimal so schnell wie das *QR*–Verfahren ist und er sich viel besser parallelisieren läßt, haben Stabilitätsprobleme die Verbreitung des Verfahrens behindert.

Im folgenden wird auf die Probleme, die mit der Stabilisierung des Algorithmus zusammenhängen, nicht näher eingegangen. Es wird im Gegenteil nur die Grundidee präsentiert.

Sei T eine symmetrische irreduzible Tridiagonalmatrix. Dann kann T in zwei Untermatrizen zerlegt werden:

$$T = \left[\begin{array}{cc} T_1 & \beta \\ \beta & T_2 \end{array}\right] = \left[\begin{array}{cc} \hat{T}_1 & \\ & \hat{T}_2 \end{array}\right] + \left[\begin{array}{cc} \beta & \beta \\ \beta & \beta \end{array}\right].$$

Hier entsteht die Matrix $\hat{T}_1$ durch Subtrahieren von β vom rechten unteren Element von T_1. Analog entsteht $\hat{T}_2$ durch Subtraktion von β vom linken oberen Element von T_2. Zusammenfassend kann man also feststellen, daß *sich jede Tridiagonalmatrix schreiben läßt als die Summe einer* 2×2 *Blockdiagonalmatrix mit tridiagonalen Blöcken und einer Rang–1 Korrektur*.

Der *Divide–and–conquer Algorithmus* funktioniert durch fortgesetzte Unterteilung der Matrix T gemäß dem obigen Verfahren. Oft stoppt man bevor man 2×2 Matrizen erreicht hat, deren Eigenwerte man explizit berechnen kann, und bestimmt schon für 10×10 Untermatrizen die Eigenwertzerlegung mit dem *QR*–Verfahren. Danach berechnet man jeweils aus den Eigenwerten von den Untermatrizen $\hat{T}_1$ und $\hat{T}_2$ und der Rang–1 Korrektur die Eigenwerte von T. Dies geschieht folgendermaßen: Angenommen wir haben Diagonalisierungen

$$\hat{T}_1 = Q_1 D_1 Q_1^*, \quad \hat{T}_2 = Q_2 D_2 Q_2^*$$

bereits berechnet. Dann folgt aus der Zerlegung von T die Beziehung

$$T = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \left(\begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \beta z z^\top \right) \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix}^\top$$

mit dem Vektor $z^\top = (q_1^\top, q_2^\top)$, wobei q_1 die letzte Spalte von Q_1 und q_2 die erste Spalte von Q_2 bezeichnet.

Wir haben das Problem also darauf reduziert, die Eigenwerte einer Matrix der Gestalt

$$D + ww^\top$$

mit einer Diagonalmatrix D zu berechnen, also einer Diagonalmatrix plus einer Rang–1 Korrektur. Wir können annehmen, daß $w_j \neq 0$ für alle j, da sonst das Problem reduzibel wäre. Ist nun q ein Eigenvektor von $D + ww^\top$ zum Eigenwert λ, so gilt

$$(D + ww^\top)q = \lambda q \implies (D - \lambda \mathbb{I})q + w(w^\top q) = 0.$$

Weil λ kein Eigenwert von D sein kann, ist $(D - \lambda\mathbb{I})$ invertierbar und es folgt

$$q + (D - \lambda\mathbb{I})^{-1} w(w^\top q) = 0 \implies w^\top q + w^\top (D - \lambda\mathbb{I})^{-1} w(w^\top q) = 0,$$

was man umschreiben kann als

$$f(\lambda)(w^\top q) = 0$$

mit der Funktion

$$f(x) = 1 + \sum_{j=1}^{n} \frac{w_j^2}{d_j - x}.$$

Nachdem $w^\top q \neq 0$ ist — andererseits wäre q Eigenvektor von D, hätte also genau eine nichtverschwindende Komponente q_j; dann wäre aber $w_j = 0$, ein Widerspruch — gilt

$$f(\lambda) = 0$$

genau dann, wenn λ ein Eigenwert von $D + ww^\top$ ist. Diese Gleichung heißt auch *Sekulargleichung*. Die Form der rationalen Funktion $f(x)$ impliziert, daß sie genau n Nullstellen besitzt, die mit einem Newton–Verfahren (siehe Kapitel **??**) schnell bestimmt werden können. Zur Bestimmung jeder einzelnen Nullstelle benötigt man $O(1)$ Aufwand, gesamt also $O(n)$ für alle n Nullstellen. Der Gesamtaufwand für Unterteilung und Rekombination bei Lösung der Sekulargleichung beträgt $O(n^2)$.

Speziell bei der Berechnung von Eigenwerten und Eigenvektoren ist der Vorteil des Divide–and–conquer gegenüber dem *QR*–Verfahren offenbar. Die Reduktion der symmetrischen Matrix auf Tridiagonalgestalt (Phase 1) ist bei beiden Algorithmen gleich $\sim \frac{8}{3}n^3$, doch der Iterationsprozeß verbraucht beim *QR*–Verfahren $\sim 6n^3$ Flops, während er beim Divide–and–conquer Algorithmus $\sim \frac{4}{3}n^3$ elementare Rechenschritte verbraucht, was in Summe einer Verbesserung von $\sim 9n^3$ auf $\sim 4n^3$ Flops entspricht. Schwerer wiegt noch die Möglichkeit zur Parallelisierung, die den Siegeszug dieses Verfahrens in den nächsten Jahren unvermeidlich machen wird.

3.5 Berechnung der Singulärwertzerlegung

Endlich, nach Analyse der Eigenwerte berechnenden Algorithmus kann man auch ein Verfahren zur Bestimmung der Singulärwertzerlegung einer Matrix A konstruieren. Wieder wird das Verfahren in zwei Phasen aufgeteilt. In Phase 1 wird die $n \times m$ Matrix A in eine quadratische Bidiagonalmatrix B der Größe $\min(m,n) \times \min(m,n)$ transformiert. Danach werden die singulären Werte von B in Phase 2 iterativ approximiert.

3.5.1 Transformation auf Bidiagonalgestalt

Es gibt zweieinhalb verschiedene Verfahren, eine rechteckige Matrix A auf *Bidiagonalform* zu bringen, also $A = UBV^*$ zu zerlegen mit

$$B = \begin{pmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & * & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & * & * \\ 0 & \cdots & \cdots & 0 & * \end{pmatrix}.$$

Die einfachere ist die Methode der *Golub–Kahan Bidiagonalisierung*, im folgenden mit GK–Bidiagonalisierung bezeichnet. Mit Hilfe von Householder–Spiegelungen erzeugt man abwechselnd Nullspalten und Nullzeilen wie anhand der untenstehenden Matrix angedeutet:

$$\underset{A}{\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}} \xrightarrow{U_1^*\cdot} \underset{U_1^*A}{\begin{pmatrix} * & * & * & * \\ \mathbf{0} & * & * & * \\ \mathbf{0} & * & * & * \\ \mathbf{0} & * & * & * \\ \mathbf{0} & * & * & * \\ \mathbf{0} & * & * & * \end{pmatrix}} \xrightarrow{\cdot V_1} \underset{U_1^*AV_1}{\begin{pmatrix} * & * & \mathbf{0} & \mathbf{0} \\ & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{pmatrix}}$$

$$\xrightarrow{U_2^*\cdot} \underset{U_2^*U_1^*AV_1}{\begin{pmatrix} * & * & & \\ & * & * & * \\ & \mathbf{0} & * & * \\ & \mathbf{0} & * & * \\ & \mathbf{0} & * & * \\ & \mathbf{0} & * & * \end{pmatrix}} \xrightarrow{\cdot V_2} \underset{U_2^*U_1^*AV_1V_2}{\begin{pmatrix} * & * & & \\ & * & * & \mathbf{0} \\ & & * & * \\ & & * & * \\ & & * & * \\ & & * & * \end{pmatrix}}$$

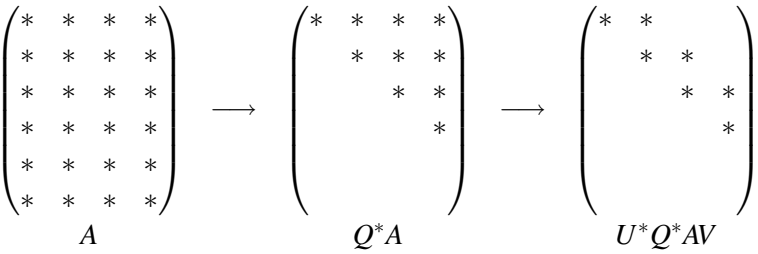
$$\xrightarrow{U_3^*\cdot} \underset{U_3^*\ldots U_1^*AV_1V_2}{\begin{pmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & \mathbf{0} & * \\ & & \mathbf{0} & * \\ & & \mathbf{0} & * \end{pmatrix}} \xrightarrow{U_4^*\cdot} \underset{U_4^*\ldots U_1^*AV_1V_2}{\begin{pmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & & * \\ & & & \mathbf{0} \\ & & & \mathbf{0} \end{pmatrix}}$$

Dieses Verfahren entspricht also zwei gleichzeitig durchgeführten Householder QR–Faktorisierungen, eine von rechts die andere von links. Der Aufwand für die GK–Bidiagonalisierung beträgt

$$\sim 4nm^2 - \tfrac{4}{3}m^3$$

Flops.

Falls $n \gg m$ ist, so erweist sich dieses Verfahren als zu aufwändig, da dann ein Großteil der zu erzeugenden Nullen unterhalb der Hauptdiagonale sitzt. Das von Lawson, Hanson und Chan vorgeschlagene Verfahren, die LHC–Bidiagonalisierung verfolgt daher die folgende Idee:



Man beginnt mit der Berechnung einer *QR*–Zerlegung von $A = QR$. Das erzeugt alle unterhalb der Hauptdiagonale liegenden Nullen in einem Schritt. Für die übrigbleibende $m \times m$ obere Dreiecksmatrix R bestimmt man die Bidiagonalisierung mittels GK–Bidiagonalisierung. Dieser Trick reduziert den Aufwand auf

$$\sim 2nm^2 + 2m^3$$

Flops. Das Verfahren ist billiger als die GK–Bidiagonalisierung, wenn $n > \frac{5}{3}m$.

Das zweieinhalbte Verfahren ist eine leichte Verallgemeinerung des LHC–Verfahrens und wird Drei–Schritt–Bidiagonalisierung genannt. Der Trick ist, die *QR*–Zerlegung nicht zu Beginn durchzuführen sondern irgendwann in der Mitte des Algorithmus. Im GK–Prozeß wird eine Matrix nämlich in jedem Schritt ein klein wenig schmäler. Nach k Schritten hat die noch zu bearbeitende Matrix die Dimensionen $(n-k) \times (m-k)$, und wenn das Verhältnis genügend groß ist, führt man die *QR*–Zerlegung durch und reduziert das Problem wie im LHC–Verfahren auf eine quadratische Dreiecksmatrix. Die maximale Einsparung erzielt man, wenn $(n-k)/(m-k) = 2$ erreicht ist. Diese Wahl liefert die Aufwandsabschätzung

$$\sim 4nm^2 - \tfrac{4}{3}m^3 - \tfrac{2}{3}(n-m)^3,$$

eine moderate Verbesserung gegenüber den anderen beiden Verfahren.

3.5.2 Berechnung der Singulärwerte

Nachdem man das Problem darauf reduziert hat, die Singulärwerte einer quadratischen Bidiagonalmatrix zu errechnen, muß man nur das Problem nur noch mit einem geeigneten Eigenwertproblem in Verbindung bringen:

Sei die Singulärwertzerlegung $A = U\Sigma V^*$ bekannt. Dann gelten die beiden Gleichungen

$$A^*A = V\Sigma^*\Sigma V^* = V\Sigma^2 V^*$$

und

$$\begin{pmatrix} 0 & A^* \\ A & 0 \end{pmatrix} = \begin{pmatrix} V & V \\ U & -U \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \begin{pmatrix} V & V \\ U & -U \end{pmatrix}^*,$$

die jeweils die Singulärwerte und singulären Vektoren in eine Eigenwertzerlegung umschreiben. Es hat jedoch jede der beiden Formulierungen seine Nachteile. Die erste Form, die die Singulärwerte von A mit der *Kovarianzmatrix* A^*A in Verbindung bringt, leidet an der starken Erhöhung der Konditionszahl (es gilt ja $\kappa_2(A^*A) = \kappa_2(A)^2$). Ist die Matrix A schlecht konditioniert, so birgt diese Tatsache eine mögliche Instabilität. Auf der anderen Seite verdoppelt die zweite Formulierung die Dimension der zu behandelnden Matrix und erhöht damit die im *QR*–Verfahren zu leistende Arbeit um einen Faktor 4.

Der gebräuchlichste Algorithmus ist eine leichte Abwandlung des *QR*–Verfahrens, aufbauend auf der Matrix A^*A und der ersten Formulierung. Das Instabilitätsproblem wird dadurch umgangen, dass die Matrix A^*A nie explizit gebildet sondern nur implizit *QR*–zerlegt wird. Die genaue Vorgehensweise, sei im folgenden illustriert:

Sei

$$B = \begin{pmatrix} d_1 & f_1 & 0 & \cdots & 0 \\ 0 & d_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & f_{n-1} \\ 0 & \cdots & \cdots & 0 & d_n \end{pmatrix}$$

die zu zerlegende Bidiagonalmatrix. Dann wählen wir eine Art Wilkinson–Shift, indem wir die Eigenwerte der rechten unteren 2×2–Teilmatrix T der Tridiagonalmatrix B^*B bestimmen (diese Teilmatrix kann fast ohne zusätzlichen Rechenaufwand bestimmt werden)

$$T = \begin{pmatrix} d_{n-1}^2 + f_{n-2}^2 & d_{n-1} f_{n-1} \\ d_{n-1} f_{n-1} & d_n^2 + f_{n-1}^2 \end{pmatrix}$$

und denjenigen auswählen (λ), der näher bei $d_n^2 + f_{n-1}^2$ liegt. Bestimmen wir dann eine Givens–Rotation

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix}^\top \begin{pmatrix} d_1^2 - \lambda \\ d_1 f_1 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix},$$

$c_1 = \cos\theta_1$ und $s_1 = \sin\theta_1$, und setzen wir $G_1 = G(1,2,\theta_1)$.

Danach wenden wir G_1 auf B an. Am folgenden Beispiel mit $n = 6$ seien die weiteren Schritte illustriert:

$$BG_1 = \begin{pmatrix} * & * & & & & \\ \boldsymbol{*} & * & * & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

Der eine Eintrag unterhalb der Hauptdiagonale stört, und daher konstruieren wir sukzessive weitere Givensrotationen $U_1, V_2, U_2, \ldots, V_{n-1}$ und U_{n-1} mit deren Hilfe wir das Element wieder Richtung rechts unten „aus der Matrix schieben“. Die ersten drei Schritte seien im folgenden angegeben:

$$U_1^* BG_1 = \begin{pmatrix} * & * & \boldsymbol{*} & & & \\ & * & * & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

$$U_1^* BG_1 V_2 = \begin{pmatrix} * & * & & & & \\ & * & * & & & \\ & \boldsymbol{*} & * & * & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

$$U_2^* U_1^* BG_1 V_2 = \begin{pmatrix} * & * & & & & \\ & * & * & \boldsymbol{*} & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

und so weiter. Der Vorgang endet mit einer neuen Bidiagonalmatrix $\bar{B}$:

$$\bar{B} = (U_1 \dots U_{n-1})^* B (G_1 V_2 \dots V_{n-1}) = \bar{U}^* B \bar{V}.$$

Implizit hat man damit einen geshifteten QR–Schritt für die symmetrische Tridiagonalmatrix B^*B ausgeführt. Das so konstruierte Verfahren konvergiert mit kubischer Geschwindigkeit gegen eine Diagonalmatrix B und damit gegen eine Singulärwertzerlegung. Jedoch ist es nötig, zwei verschiedene Deflationsschritte zur Verkleinerung des Problems zusätzlich einzubauen. Erstens, wenn sich B zerlegen läßt

$$B = \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix}$$

kann man das Problem direkt in zwei kleinere aufsplitten, doch auch wenn $d_k = 0$ gilt für ein k kann man das Problem zerteilen. Wieder verwendet man geeignete Givensrotationen, um das entsprechende Nebendiagonalelement f_k auch auf Null zu transformieren. Das Verfahren sei ebenfalls anhand eines 6×6 Beispieles illustriert.

$$B = \begin{pmatrix} * & * & & & & \\ & * & * & & & \\ & & 0 & * & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

$$H_1^* B = \begin{pmatrix} * & * & & & & \\ & * & * & & & \\ & & 0 & & * & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

$$H_2^* H_1^* B = \begin{pmatrix} * & * & & & & \\ & * & * & & & \\ & & 0 & & & * \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

$$H_3^* H_2^* H_1^* B = \begin{pmatrix} * & * & & & & \\ & * & * & & & \\ & & 0 & & & \\ & & & * & * & \\ & & & & * & * \\ & & & & & * \end{pmatrix}$$

und das Problem zerfällt. Setzt man alle diese Algorithmenteile zusammen erhält man das implizite QR–Verfahren zur stabilen Berechnung der Singulärwertzerlegung einer Bidiagonalmatrix B.

Der Gesamtaufwand zur Bestimmung der Singulärwertzerlegung wird völlig vom Bidiagonalisierungsverfahren dominiert. Jeder Schritt der Iteration benötigt nämlich lediglich einen Aufwand von $O(n)$, was einen Gesamtaufwand von $O(n^2)$ für die gesamte Phase 2 ergibt. Verwendet man Drei–Schritt–Bidiagonalisierung, ist der asymptotische Aufwand zur Bestimmung einer Singulärwertzerlegung

$$\sim 4nm^2 - \tfrac{4}{3}m^3 - \tfrac{2}{3}(n-m)^3.$$

3.6 Software

An Software seien vor allem wieder die Funktionen der LAPACK Sammlung hervorgehoben. Sie enthält Unterprogramme zur Berechnung von Eigenwerten und Eigenvektoren für symmetrische Matrizen (`syev`, `syevd`, `syevx`), tridiagonale Matrizen (`steqr` mit *QR*–Verfahren, `stedc` mit Divide–and–conquer), für unsymmetrische Matrizen (`geev`, `geevx`) und Hessenbergmatrizen (`hseqr`, `hsein`) und viele mehr. Auch Programme zur Bestimmung von Tridiagonalisierungen, Bidiagonalisierungen und Hessenbergtransformationen sind enthalten. Weiters existieren natürlich auch Programme zur Berechnung einer Singulärwertzerlegung (`gesvd`,...).

Die Programmpakete wie MAPLE, Mathematica und MATLAB enthalten einfache Funktionen zur Berechnung der Eigenwerte und Eigenvektoren. Dort sind meist *QR*–Verfahren implementiert.

4 Numerische Lineare Algebra IV: Iterationsverfahren

4.1 Grundlagen

In den vergangenen Kapiteln haben wir eine Handvoll Methoden kennengelernt, die verschiedenen Aufgaben der linearen Algebra auf numerischem Wege zu lösen. In Kapitel 3, wo es darum ging, Eigenwerte und Eigenvektoren zu bestimmen, sind zum ersten Mal Verfahren aufgetaucht, die selbst bei rundungsfehlerfreier Rechnung keine exakten Resultate mehr liefern. In allen anderen Fällen sind die Algorithmen so konstruiert, daß sie in endlich vielen Rechenschritten im Prinzip das mathematische Problem exakt lösen können; solche Verfahren nennt man *direkte Verfahren*. Auch Eigenwerte berechnende Algorithmen bestehen im herkömmlichen Fall aus einer direkten Phase (z.B. Hessenbergtransformation) und einer iterativen Phase, die aber nur auf stark vereinfachte Probleme angewendet wird. Eine Gemeinsamkeit aller dieser Algorithmen ist die Größenordnung $O(n^3)$ des Rechenaufwandes.

Für wirklich große Probleme (etwa Computertomographie — siehe Modell **??** aus Kapitel **??**) ist dieser Aufwand jedoch viel zu hoch. Er ist auch in dem Sinne unverständlich hoch als in jedes Matrixproblem nur $O(n^2)$ (bei dünnbesetzten Matrizen nur $O(n)$) Zahlen eingehen. Warum sollte dann der Aufwand zur Berechnung immer $O(n^3)$ sein?

In den vergangenen vier Jahrzehnten ist die maximale Dimension der gerade noch lösbaren Matrixberechnungen etwa alle fünfzehn Jahre um eine Zehnerpotenz gestiegen. Von $n = 20$ in den Fünfzigerjahren auf etwa $n = 20000$ Mitte der Neunzigerjahre. Im gleichen Zeitraum ist die maximale Leistungsfähigkeit der Computersysteme hingegen um einen Faktor 10^9 gestiegen. Man kann also den $O(n^3)$ Flaschenhals deutlich erkennen. Trotz einer gewaltigen Verbesserung der Hardware ist die maximale Dimension der Matrixberechnungen „nur“ um einen Faktor 10^3 gestiegen.

Nun, ganz korrekt sind die oben stehenden Aussagen nicht. Es existieren Algorithmen für die meisten Probleme der linearen Algebra, die einen geringeren Aufwand als $O(n^3)$ haben. Für die Lösung eines linearen Gleichungssystems haben Coppersmith und Winograd 1986 ein Verfahren gefunden, das Ordnung $O(n^{2.376})$ hat. Allerdings ist der Koeffizient des höchsten Terms $n^{2.376}$ so groß, daß der Gesamtaufwand für heutzutage berechenbare Probleme trotz des niedrigen Exponenten höher ist als bei der *LR*–Zerlegung mit Spaltenpivotsuche.

Die einzige Idee, die noch hilft, große Systeme mit Dimensionen jenseits von 10^6 zu lösen, ist Lösungen nicht mehr direkt zu bestimmen sondern approximativ durch Verwendung eines Iterationsverfahrens. Im Gegensatz zu den Algorithmen aus Kapitel 3 versucht man jedoch, den Aufwand eines einzelnen Schrittes so stark wie möglich zu reduzieren. Mehr als $O(n)$ Rechenoperationen kann man für einen einzelnen Iterationsschritt in einem Algorithmus für Probleme riesiger Dimension nicht verwenden. Gelingt es dann auch noch, das Verfahren so zu gestalten, daß lediglich $O(1)$ Iterationsschritte nötig sind, so hätte man das ideale Verfahren gefunden, doch meist ist die höchste erreichbare Verbesserung von $O(n^3)$ auf $O(n^2)$. Heutzutage Ende der Neunzigerjahre übertreffen Iterationsverfahren wegen der hohen Konstanten vor den führenden Termen n^2 die direkten Algorithmen in praktischen höchst dimensionalen Anwendungsproblemen um einen Faktor zehn. Doch durch die steigende Leistungsfähigkeit der Computersysteme, die in der Zukunft die Lösung noch größerer Probleme ermöglichen werden, wird der Siegeszug der Iterationsverfahren nicht mehr aufzuhalten sein, da spätestens dann der kleinere Exponent zu wachsenden Geschwindigkeitsunterschieden führen wird.

Iterationsverfahren können $O(n^3)$ jedoch nicht immer übertreffen. Für Zufallsmatrizen ist das sehr unwahrscheinlich. Der Weg zur Geschwindigkeitssteigerung führt über die Ausnützung der Matrixstruktur,

die in den meisten Anwendungsproblemen alles andere als zufällig ist. Hochdimensionale Probleme führen meist auf dünnbesetzte Matrizen (oft gar auf Matrizen mit Bandstruktur), die in jeder Zeile nur ν Einträge haben, die von Null verschieden sind. Für solche Matrizen kann man Matrix–Vektor Produkte anstatt in $O(n^2)$ Rechenschritten mit $O(\nu n)$ elementaren Operationen berechnen, und Matrix–Vektor Produkte sind die einzigen komplexen Rechenoperationen, die in einen Iterationsschritt eingehen.

4.1.1 Krylov–Räume

Nachdem wir schon festgestellt haben, daß wir uns beim Entwurf der Iterationsalgorithmen auf Matrix–Vektor Produkte einschränken wollen, ist zu erwarten, daß Räume, die von Vektoren b und deren iterierte Produkte mit einer Matrix A aufgespannt werden, zentrale Bedeutung erlangen werden:

Definition 4.1.1.1. *Gegeben eine Matrix $A \in \mathbb{K}^{n\times n}$ und einen Vektor $b \in \mathbb{K}^n$, dann ist die dazu assoziierte* Krylovfolge *gegeben durch die iterierten Produkte von A mit b:*

$$b, Ab, A^2b, A^3b, \ldots$$

Der von den ersten n Vektoren der Krylov–Folge aufgespannte Vektorraum $\mathcal{K}_n$ heißt der n–te Krylov–Raum *zu A und b:*

$$\mathcal{K}_n := \langle b, Ab, A^2b, \ldots, A^{n-1}b\rangle.$$

Die Matrix K_n, die als Spalten die ersten n Glieder der Krylov–Folge enthält, heißt die n–te Krylov–Matrix zu A und b:

$$K_n := \left(b \middle| Ab \middle| A^2b \middle| \cdots \middle| A^{n-1}b\right).$$

Die meisten Iterationsalgorithmen funktionieren, indem sie das zu bearbeitende Problem in Krylovräume wachsender Ordnung projizieren und die Lösungen der niedriger dimensionalen Problems zur Approximation der Lösung der hochdimensionalen Aufgabe verwendet. Dabei werden implizit QR–Zerlegungen und Hessenbergreduktionen von Krylov–Matrizen bestimmt.

Die im folgenden vorgestellten Verfahren können entsprechend der folgenden Tabelle angeordnet werden. Dabei steht die Abkürzung CG für das *konjugierte Gradientenverfahren* (englisch: conjugate gradients) und die Abkürzung GMRES für Verallgemeinerte–Minimale–Residuen (englisch: generalized minimal residuals)

	$Ax = \lambda x$	$Ax = b$
$A \neq A^*$	Arnoldi Iteration	GMRES Verfahren
$A = A^*$	Lanczos Iteration	CG Verfahren

4.2 Arnoldi Iteration

Das zentrale Verfahren, auf dem die meisten anderen Iterationsverfahren aufbauen ist die Arnoldi Iteration. In Kapitel **??** haben wir zwei grundverschiedene Methoden kennengelernt, eine QR–Faktorisierung einer Matrix zu berechnen, das Gram–Schmidt Verfahren und das Householder Verfahren. Dort haben wir aus Stabilitätsgründen die Householder–Reflexionen dem Gram–Schmidt Verfahren vorgezogen. Doch das Gram–Schmidt Verfahren hat einen wichtigen Vorteil. Man kann es nach n Schritten unterbrechen und erhält eine unvollständige QR–Zerlegung, eine reduzierte QR–Faktorisierung der ersten n Spalten von A.

Zur Berechnung einer Hessenbergreduktion existieren ebenfalls zwei Standardverfahren — Householder Reflexionen (Kapitel 3) und die Arnoldi Iteration, die also ein Analogon des Gram–Schmidt Verfahrens für Hessenbergreduktion ist.

4.2.1 Das Iterationsverfahren

Nehmen wir an, wir hätten bereits eine Hessenbergreduktion von A bestimmt:

$$A = QHQ^*$$

mit untärer Matrix Q und oberer Hessenbergmatrix H, oder umgeformt $AQ = QH$. Sei Q_m die $m \times n$ Teilmatrix von Q, die aus den ersten m Spalten von Q besteht:

$$Q_m = \left(q_1 \middle| q_2 \middle| \cdots \middle| q_m \right).$$

Mit $\hat{H}_m$ sei die $(m+1) \times m$ linke obere Teilmatrix von H bezeichnet, selbst wieder eine obere Hessenbergmatrix:

$$\hat{H}_m = \begin{pmatrix} h_{11} & & \cdots & h_{1m} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{pmatrix}.$$

Mit diesen Bezeichnungen gilt die Gleichung

$$AQ_m = Q_{m+1}\hat{H}_m.$$

Schreibt man diese Gleichung spaltenweise auf und extrahiert daraus die Beziehung für die letzte, die m–te, Spalte, so erhält man

$$Aq_m = h_{1m}q_1 + \cdots + h_{mm}q_m + h_{m+1,m}q_{m+1}$$

oder umgeformt

$$h_{m+1,m}q_{m+1} = Aq_m - h_{mm}q_m - \cdots - h_{1m}q_1. \tag{4.1}$$

Betrachtet man diese Gleichung in den Spezialfällen $m = 1$ und $m = 2$:

$$\begin{aligned} h_{21}q_2 &= Aq_1 - h_{11}q_1 \\ h_{32}q_3 &= Aq_2 - h_{22}q_2 - h_{12}q_1 = \\ &= \tfrac{1}{h_{21}}A^2q_1 - \tfrac{h_{11}+h_{22}}{h_{21}}Aq_1 + \left(\tfrac{h_{11}h_{22}}{h_{21}} - h_{12}\right)q_1, \end{aligned}$$

so erkennt man, daß für alle i

$$q_i \in \mathscr{K}_i,$$

dem Krylov–Raum zu A und q_1, gilt. Der i–te Vektor erfüllt nach Gleichung (4.1) eine $i+1$–Terme Rekurrenzrelation, die nur sich und die anderen Krylov–Vektoren enthält. Mit Hilfe dieser Gleichung formuliert man den Algorithmus der Arnoldi Iteration

Das Arnoldi Verfahren erzeugt Vektoren q_j, die orthonormal sind und die Krylov–Räume $\mathscr{K}_k$ aufspannen. Das ist äquivalent dazu, daß es eine QR–Zerlegung der zugehörigen Krylov–Matrix K_n

$$K_n = Q_nR_n$$

berechnet. Der Vorteil ist, daß weder K_n noch R_n explizit berechnet werden, denn beide Matrizen sind ausgesprochen schlecht konditioniert, weil alle Spalten von K_n dazu tendieren, denselben dominanten Eigenvektor von A zu approximieren (siehe Abschnitt 3.3.2 in Kapitel 3, die Potenziteration). In Analogie zur Blockpotenziteration kann man annehmen, daß die Matrix Q_m die ersten m dominanten Eigenwerte von A approximiert.

Arnoldi Iteration

```
b beliebig
q_1 = b/||b||
for m = 1 to ? do
    v = A q_m
    for j = 1 to m do
        h_jm = q_j^* v
        v = v - h_jm q_j
    done
    h_{m+1,m} = ||v||
    if h_{m+1,m} = 0 then
      stop
    endif
    q_{m+1} = v/h_{m+1,m}
done
```

Führen wir eine weitere Notation ein: H_m sei die Matrix $\hat{H}_m$ ohne die letzte Zeile. Dann gilt die Beziehung $H_m = Q_m^* Q_{m+1} \hat{H}_m$, weil die Matrizen Q_m und Q_{m+1} die ersten m bzw. $m+1$ Spalten der unitären Matrix Q enthalten. Aus der Zerlegung $A = QHQ^*$ enthalten wir folglich die Beziehung

$$H_m = Q_m^* A Q_m.$$

Diese Matrix repräsentiert die orthogonale Projektion von A auf den Krylov–Raum $\mathscr{K}_m$ dargestellt in der Basis $\{q_1, \ldots, q_m\}$. (Bezüglich der Standardbasis des $\mathbb{K}^m$ ist der orthogonale Projektor gegeben durch die Formel $Q_m Q_m^* A$.) Diese Abbildung nimmt einen Vektor $v \in \mathscr{K}_m$, wendet A darauf an und projiziert das Ergebnis Av wieder auf $\mathscr{K}_m$ zurück. Diese Projektion trägt den Namen *Rayleigh–Ritz Verfahren*, nicht zuletzt, weil die Einträge von H_m gerade die Rayleigh Quotienten bezüglich A der Vektoren q_j sind.

Weil H_m eine Projektion der Matrix A ist, kann man annehmen, daß die Eigenwerte von H_m einen starken Zusammenhang mit den Eigenwerten von A besitzen. Die m Eigenwerte von H_m, im folgenden mit $\theta_j^{(m)}$ bezeichnet, werden auch *Arnoldi Eigenwertschätzer* oder *Ritz–Zahlen* genannt.

Theorem 4.2.1.1. *Die Matrizen Q_m, die die Arnoldi Iteration erzeugt, sind die unitären Faktoren von reduzierten QR–Zerlegungen der Krylov–Matrizen*

$$K_m = Q_m R_m.$$

Die Hessenbergmatrizen H_m sind die zugehörigen Projektionen

$$H_m = Q_m^* A Q_m$$

auf die Krylov–Räume $\mathscr{K}_m$, und die Iterationsschritte hängen über die Gleichung

$$A Q_m = Q_{m+1} \hat{H}_m$$

zusammen.

4.2.2 Polynomiale Approximation

Haben wir einen Vektor $x \in \mathscr{K}_m$ gegeben, so kann man ihn in die ursprüngliche Basis $\{b, Ab, \ldots, A^{m-1}b\}$ entwickeln:

$$x = a_0 b + a_1 Ab + \cdots + a_{m-1} A^{m-1} b = q(A)b,$$

wobei q das Polynom

$$q(z) = a_0 + a_1 z + \cdots + a_{m-1} z^{m-1}$$

bezeichnet. Aus diesem Grund kann man Krylov–Räume immer mit Hilfe von Matrixpolynomen analysieren. Sei für die folgende Analyse $\mathbb{K}^k_{\text{mon}}[z]$ die Bezeichnung der Menge der monischen Polynome höchstens k–ten Grades in einer Variablen. Dann löst das Arnoldi Verfahren das folgende Approximationsproblem:

> Finde das Polynom $p^m \in \mathbb{K}^m_{\text{mon}}[z]$, das die Norm
>
> $$\|p^n(A)b\|_2$$
>
> minimiert.

Dieses Approximationsproblem hat genau eine Lösung, sofern $\dim \mathscr{K}_m = m$ ist, nämlich das charakteristische Polynom von H_m. Die Nullstellen dieses charakteristischen Polynoms sind genau die Ritz–Zahlen $\theta_j^{(m)}$, welche über die Arnoldi Iteration bestimmt werden können.

Die Polynominterpretation ermöglicht es auch, die Invarianzeigenschaften der Arnoldi Iteration zu bestimmen:

Theorem 4.2.2.1. *Wird die Arnoldi Iteration angewendet auf die Matrix $A \in \mathbb{K}^{n\times n}$ dann gilt*

Translationsinvarianz *Wird A um $\sigma\mathbb{I}$ verschoben ($A \to A + \sigma\mathbb{I}$), so ändern sich die Ritz–Zahlen zu $\theta_j^{(m)} + \sigma$.*

Skalierungsinvarianz *Wird A durch σA ersetzt, so werden auch die Ritz–Zahlen skaliert: $\theta_j^{(m)} \to \sigma\theta_j^{(m)}$.*

Unitäre Invarianz *Ersetzt man A durch eine unitär ähnliche Matrix UAU^*, und verändert man b zu Ub, so bleiben die Ritz–Zahlen gleich.*

In allen drei Fällen verändern sich die Ritz–Vektoren $Q_m y_j$, die den Eigenvektoren y_j von H_m entsprechen, nicht.

4.2.3 Arnoldi Iteration und Eigenwerte

Die letzten beiden Resultate zeigen auch schon auf, wie die Arnoldi Iteration Eigenwerte von A auffindet. Betrachten wir dazu eine Matrix, die nur $m \ll n$ verschiedene Eigenwerte besitzt. Dann besitzt A ein Minimalpolynom p vom Grad m, dessen Nullstellen genau die Eigenwerte von A sind, und es gilt $p(A) = 0$. Dieses Polynom ist also dann jenes Polynom p^m, das von der Arnoldi Iteration berechnet wird (jedenfalls, wenn der Startvektor b Komponenten in Richtung jedes Eigenraumes von A besitzt).

Im generischen Fall versucht man dieses Minimalpolynom zu approximieren, weil es zu hohen Grad hat. Deswegen wählt man das Polynom m–ten Grades, das $\|p^m(A)\|_2$ minimiert und erhält anstelle eines Minimalpolynoms ein „Pseudo–Minimalpolynom“. Die Ritz–Werte, die Nullstellen dieses Pseuso–Minimalpolynoms, sind demnach gute Approximationen für die Nullstellen des Minimalpolynoms, die Eigenwerte von A.

Mit zunehmendem Polynomgrad approximieren die Nullstellen von p^m die Eigenwerte von A immer besser, und dieser Fortschritt kann graphisch mit Hilfe der *Arnoldi–Lemniskaten* dargestellt werden. Eine Lemniskate ist eine Familie von Kurven

$$\{z \in \mathbb{C} : |q(z)| = C\},$$

wo q ein Polynom ist und $C \in \mathbb{R}$ eine reelle Zahl. Wählt man speziell $q = p^m$ und

$$C = \frac{\|p^m(A)b\|}{\|b\|},$$

so erhält man die m–te Arnoldi–Lemniskate. Diese Arnoldi–Lemniskaten approximieren das Spektrum von A dahingehend, daß sie mit steigender Iterationszahl die geometrische Form des Spektrums immer besser

Abbildung 4.1: Arnoldi–Lemniskaten zu den Iterationsschritten $m = 1, 2, 4, 8, 12, 16$ für eine 100×100 Matrix A. Die kleinen Punkte sind die Eigenwerte von A, währen die großen Punkte die Ritz–Zahlen repräsentieren.

approximieren. Abbildung 4.1 stellt diese Lemniskaten für verschiedene Iterationsschritte dar. In der Abbildung erkennt man auch, daß die Arnoldi–Lemniskaten zuerst kleine Teile abspalten, die die extremen Eigenwerte einhüllen und sich mit geometrischer Geschwindigkeit zusammenziehen. Die „großen Wolken" werden immer besser eingeschlossen je höher der Polynomgrad wird, und die Ritz–Zahlen verteilen sich immer mehr innerhalb dieser Wolken.

Die Konvergenz der Arnoldi Iteration genauer zu analysieren ist eine äußerst schwierige Aufgabe, und sie wird bis heute noch nicht vollständig verstanden.

4.3 Das GMRES Verfahren

Das GMRES Verfahren ist eine Abwandlung der Arnoldi Iteration, die dazu verwendet wird, lineare Gleichungssysteme $Ax = b$ zu lösen. Die Idee ist, den Lösungsvektor $x_* := A^{-1}b$ durch denjenigen Vektor $x_m \in \mathscr{K}_m$ zu approximieren, der die Norm des Residuums $r_m = b - Ax_m$ minimiert. Man löst also in anderen Worten ein Kleinste–Quadrate Problem, um x_* zu bestimmen.

Die Formulierung des Kleinste–Quadrate Problems ist einfach. Sei K_m die m–te Krylov–Matrix zu A und b. Dann ist unser Problem, denjenigen Vektor v zu finden, der

$$\|AK_m v - b\|_2$$

minimiert. Ist dieser Vektor erst gefunden, erhält man $x_n = K_n v$. Das Kleinste–Quadrate Problem könnte man z.B. dadurch lösen, daß man eine QR–Zerlegung der Matrix AK_m berechnet.

Dieses Verfahren wäre jedoch numerisch instabil (aus den schon bekannten Gründen: die katastrophale Kondition von K_m). Stattdessen berechnet man mit Hilfe der Arnoldi Iteration eine Folge von Matrizen Q_m, die Orthonormalbasen für die aufeinander folgenden Krylov–Räume $\mathscr{K}_m$ bilden. Wir schreiben also $x_m = Q_m y$ statt $x_m = K_m c$. Daher lautet unser neues Kleinste–Quadrate Problem: Finde einen Vektor $w \in \mathbb{K}^m$ mit

$$\|AQ_m w - b\|_2 \text{ ist minimal.}$$

Mit dieser Formulierung und Gleichung (4.1) kann man die Dimension des Problems reduzieren und das Kleinste–Quadrate Problem umformulieren zu

$$\|Q_{m+1}\hat{H}_m w - b\|_2$$

ist ein Minimum. Nun sind beide Vektoren innerhalb der Norm im Spaltenraum von Q_{m+1}, und man kann ohne die Norm zu verändern mit Q_{m+1}^* von links multiplizieren. Verwendet man überdies noch die Gleichung $Q_{m+1}^* b = \|b\|_2 e_1$, die aus der Konstruktion der Q_i folgt, so erhält man schließlich die GMRES Formulierung des Kleinste–Quadrate Problems

$$\|\hat{H}_m w - \|b\|_2 e_1\|_2 = \min .$$

Im n–ten Schritt des GMRES Verfahrens lösen wir dieses Problem nach w und setzen $x_m = Q_m w$. Dies läßt sich als Algorithmus formulieren:

GMRES Verfahren

$q_1 = b/\|b\|_2$
for $m = 1$ **to** ? **do**
  Führe Schritt n der Arnoldi Iteration (Alg. 4.2.1) aus
  Suche w, das $\|\hat{H}_m w - \|b\|_2 e_1\|_2$ minimiert
  $x_m = Q_m w$
done

Der Schritt „Suche w,…“ ist ein $(m+1) \times m$ Kleinste–Quadrate Problem, das mit Hilfe eines Update Prozesses gelöst werden kann. Hat man die QR–Zerlegung der Matrix $\hat{H}_m$ berechnet, so kann man mit Hilfe einer einzelnen Givens–Rotation eine QR–Faktorisierung von $\hat{H}_{m+1}$ berechnen, und somit reduziert sich der Aufwand zur Lösung dieses Kleinste–Quadrate–Problems auf $O(m)$.

GMRES hängt ebenso wie das Arnoldi Verfahren mit Polynomapproximation zusammen. Man sucht wieder ein Polynom p_m, das

$$\|p_m(A)b\|_2$$

minimiert. Diesmal wählt man das Polynom jedoch nicht im Raum $\mathbb{K}^m_{\text{mon}}[z]$ sondern in $\mathbb{K}^m_1[z]$, der Menge der Polynome q höchstens m–ten Grades, die $q(0) = 1$ erfüllen. GMRES löst genau dieses Optimierungsproblem. Wie die Arnoldi Iteration hat auch das GMRES Verfahren diverse Invarianzeigenschaften:

Theorem 4.3.0.1. *Sei das GMRES Verfahren auf die Matrix A angewendet. Dann gilt*

Skalierungsinvarianz *Wenn man A durch σA ersetzt, für eine komplexe Zahl σ, und wenn gleichzeitig b gegen σb ausgetauscht wird, so verändern sich die Residuen r_m zu σr_m.*

Unitäre Invarianz *Der Übergang $A \to UAU^*$ und $b \to Ub$ für eine unitäre Matrix U führt zu einer Veränderung der Residuen von r_m auf $U^* r_m$.*

Das GMRES Verfahren ist nicht translationsinvariant wie die Arnoldi Iteration, da die Menge der Polynome mit $q(0) = 1$ im Gegensatz zu den monischen Polynomen nicht translationsinvariant ist.

Die Konvergenzgeschwindigkeit des GMRES Algorithmus ist (wie bei fast allen modernen Iterationsverfahren) eine schwierig zu untersuchende Angelegenheit. Lediglich zwei Tatsachen sind offensichtlich.

- GMRES konvergiert monoton, d.h.
$$\|r_{k+1}\|_2 \leq \|r_k\|_2.$$
Das folgt, weil offensichtlich $\mathscr{K}_k \subset \mathscr{K}_{k+1}$.
- Nach höchstens n Schritten gilt $\|r_n = 0\|$, das Problem ist also gelöst; wenigstens bei rundungsfehlerfreier Rechnung.

Diese beiden Aussagen sind im numerischen Zusammenhang leider völlig unbrauchbar. Damit der GMRES Algorithmus brauchbar ist, muß Konvergenz (oder beinahe Konvergenz) nach $m \ll n$ Schritten eintreten.

Was wir noch wissen, ist, daß

$$\|r_m\|_2 = \|p_m(A)b\|_2 \leq \|p_m(A)\|_2\|b\|_2$$

minimal ist. Für generische A und b bestimmt demnach die Zahl

$$\frac{\|r_m\|_2}{\|b\|_2} \leq \inf_{p_m \in \mathbb{K}_1^m[z]} \|p_m(A)\|_2$$

die Konvergenzrate von GMRES. Das führt auf die mathematisch sehr anspruchsvolle Frage, wie klein $\|p_m(A)\|_2$ werden kann wenn man eine Matrix A und eine natürliche Zahl m gegeben hat. Für diagonalisierbare Matrizen kann man diese Zahl abschätzen:

Theorem 4.3.0.2. *Sei $A = V\Lambda V^{-1}$ eine Eigenwertzerlegung von A. Dann gilt nach dem m–ten Schritt der GMRES Iteration für das Residuum r_m die Abschätzung*

$$\frac{\|r_m\|_2}{\|b\|_2} \leq \inf_{p_m \in \mathbb{K}_1^m[z]} \|p_m(A)\|_2 \leq \kappa_2(V) \inf_{p_m \in \mathbb{K}_1^m[z]} \|p_m\|_{\Lambda(A)},$$

wobei wir für eine beliebige Menge M

$$\|p\|_M := \sup_{z \in M} |p(z)|$$

setzen und $\Lambda(A)$ das Spektrum von A (die Menge der Eigenwerte von A) bezeichnet.

Ist A also nicht zu weit davon entfernt normal zu sein, ist also $\kappa_2(V) \not\gg 1$, und können Polnome gefunden werden, deren Größe auf dem Spektrum von A rasch mit wachsendem Polynomrad fällt, so konvergiert das GMRES Verfahren schnell. Ansonsten kann die Konvergenz kriechend sein. Durch geeignete Vorkonditionierung (siehe Abschnitt 4.7) kann die Matrix meist so verändert werden, daß eine hohe Konvergenzgeschwindigkeit erzielt werden kann.

4.4 Lanczos Iteration

Die Lanczos Iteration ist die Spezialisierung der Arnoldi Iteration auf hermitesche Matrizen A. In diesem Fall sind die Matrizen H_m symmetrisch und tridiagonal. Daher sind ihre Eigenwerte, die Ritz–Zahlen oder Lanczos Schätzungen, reell. Die $(n+1)$–Terme Rekurrenz (4.1) wird zu einer Drei–Terme Rekurrenz, und mit der Notation

$$H_m = \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{m-1} \\ & & & b_{m-1} & a_m \end{pmatrix}$$

kann man Algorithmus 4.2.1 umformulieren:

Dieser Algorithmus benötigt in jedem Iterationsschritt eine Matrix–Vektor Multiplikation, ein inneres Produkt und einige wenige Vektoroperationen. Für dünnbesetzte Matrizen bedeutet das Aufwand $O(n)$ mit einer sehr kleinen Konstante für jeden Schritt. Der nächste Satz faßt einige Eigenschaften der Lanczos Iteration zusammen:

Theorem 4.4.0.3. *Die Matrizen Q_m gebildet aus den berechneten Vektoren q_j sind reduzierte QR–Faktoren der Krylov–Matrizen K_m:*

$$K_m = Q_m R_m.$$

Die tridiagonalen Matrizen H_m

$$H_m = Q_m^* A Q_m$$

Lanczos Iteration

b beliebig
$q_1 = b/\|b\|_2$
$b_0 = 0$
$q_0 = 0$
for $m = 1$ **to** ? **do**
  $v = Aq_m$; $Aq_m - b_{m-1}q_{m-1}$ `für größere Stabilität`
  $a_m = q_m^\top v$
  $v = v - b_{m-1}q_{m-1} - a_m q_m$
  $b_m = \|v\|_2$
  if $b_m = 0$ **then**
    stop
  endif
  $q_{m+1} = v/b_m$
done

sind die entsprechenden Projektionen auf die Krylov–Räume, und aufeinanderfolgende Ergebnisse der Iteration hängen folgendermaßen zusammen:

$$AQ_m = Q_{m+1}\hat{H}_m.$$

Diese Gleichung kann man auch in Form einer Drei–Terme Rekurrenz umschreiben:

$$Aq_m = b_{m-1}q_{m-1} + a_m q_m + b_m q_{m+1}.$$

Solange die Lanczos Iteration nicht abbricht, also der Krylov–Raum $\mathscr{K}_m$ Dimension m besitzt, ist das charakteristische Polynom von H_m das eindeutige Polynom $p^m \in \mathbb{K}^m_{\text{mon}}[z]$, das das Approximationsproblem

$$\|p^m(A)b\| = \min$$

erfüllt.

Üblicherweise findet auch die Lanczos Iteration die extremen Eigenwerte der Matrix A zuerst. Üblicherweise konvergieren einige Lanczos–Schätzwerte linear gegen diese extremen Eigenwerte. Genauer tendiert die Lanczos Iteration dazu, gegen Eigenwerte in den Regionen zu konvergieren, die zu „dünn mit Eigenwerten besetzt sind“. Gibt es Regionen $[a,b]$, in denen die Eigenwerte „beinahe gleichverteilt“ sind, so konvergiert die Lanczos Iteration gegen Tschebyscheff Punkte über diesem Intervall — Die m Tschebyscheff Punkte im Intervall $[-1,1]$ sind definiert als

$$x_j = \cos\left(\frac{(j-\frac{1}{2})\pi}{m}\right), \quad 1 \le j \le m.$$

Rundungsfehler haben einen sehr großen Einfluß auf die Lanczos Iteration, und es ist wichtig, diese im Auge zu behalten. Der Grund dafür ist, daß im Gegensatz zur Arnoldi Iteration die Vektoren $q_1, q_2, \ldots$ durch die $(m+1)$–Terme Rekurrenz explizit dazu gezwungen werden, aufeinander orthogonal zu stehen. Das Ausnützen der Symmetrie in der Lanczos Iteration bedingt, daß eine mathematische Identität Grund für die Orthogonalität wird, diese quasi aus dem Nichts durch die Daten entsteht. Dieser Prozeß wird durch die Rundungsfehler mitunter empfindlich gestört.

Daß die Lanczos Iteration in der Praxis trotzdem außerordentlich wertvoll ist, ist ebenfalls ein Phänomen, das noch nicht vollständig analysiert ist. Was üblicherweise passiert, ist das Auftreten von „Geistereigenwerten“. Besonders exponierte Eigenwerte werden im Verlauf des Verfahrens bei späteren Iterationsschritten

plötzlich mehrfach getroffen, wobei die Vielfachheiten, die der Algorithmus produziert, mit den aktuellen Vielfachheiten der Eigenwerte nichts zu tun haben. Eine vernünftig klingende Erklärung für das Auftreten dieser Geistereigenwerte ist die folgende: Konvergenz eines Ritz–Wertes gegen einen Eigenwert annihiliert die entsprechende Eigenvektorkomponente im Vektor b. Durch Rundungsfehler wird diese Komponente jedoch nicht vollständig entfernt; es bleibt ein winzig kleiner Rest übrig. Dieser wird jedoch in den folgenden Iterationsschritten wieder so weit verstärkt, daß im weiteren Verlauf wieder eine neue Ritz–Zahl konstruiert werden muß, die diese Pseudokomponente erneut annihiliert wird, usw.

Ein Detail am Rande: Die Lanczos Iteration hängt stark mit orthogonalen Polynomen zusammen, und die Drei–Terme Beziehung der Lanczos Iteration entspricht der Drei–Terme Beziehung bei der Konstruktion von orthogonalen Polynomen. Genauere Zusammenhänge kann man etwa in [Trefethen, Bau 1997, 37] nachlesen.

4.5 Konjugierte Gradienten

Obwohl in der heutigen Zeit das Verfahren der konjugierten Gradienten als Spezialfall einer Anwendung der Arnoldi Iteration auftritt, war dieses Verfahren 1952 der eigentliche Beginn der Krylov–Raum Iterationsverfahren. Hestenes und Stiefel führten es ein, um symmetrische positiv definite lineare Gleichungssysteme schnell zu lösen, wenn die Eigenwerte gut verteilt sind.

Die Aufgabe, ein Gleichungssystem $Ax = b$ zu lösen für eine hermitesche Matrix A könnte man mit dem üblichen GMRES Algorithmus lösen, doch dieser benötigt mehr Aufwand als nötig. Durch die Symmetrie kann man wieder die $(m+1)$–Terme Rekurrenz durch eine Drei–Terme Rekurrenz ersetzen. Diese Verfahren sind auch unter den Namen *Konjugierte Residuen* oder *MINRES* (englisch: minimal residuals) bekannt.

Hier sei im folgenden der einfachste Fall beschrieben — die Lösung eines linearen Gleichungssystemes mit symmetrischer positiv definiter Koeffizientenmatrix A.

Das Verfahren der *konjugierten Gradienten*, im folgenden CG genannt, ist ein System von Rekurrenzgleichungen, das die eindeutige Folge von Punkten $x_m \in \mathscr{K}_m$ generiert, das die Eigenschaft hat, in jedem Schritt die A–Norm des Fehlers

$$\|e_n\|_A := \|x_* - x_n\|_A$$

zu minimieren ($x_* = A^{-1}b$ sei wieder die exakte Lösung). Dabei ist die A–Norm eines Vektors y gegeben durch

$$\|y\|_A = \sqrt{y^\top A y}.$$

Der Originalalgorithmus von Hestenes und Stiefel läuft folgendermaßen ab:

Konjugierte Gradienten Iteration

$x_0 = 0$
$r_0 = b$
$p_0 = r_0$
for $m = 1$ **to** ? **do**
  $\alpha_m = (r_{m-1}^\top r_{m-1})/(p_{m-1}^\top A p_{m-1})$ `; Schrittweite`
  $x_m = x_{m-1} + \alpha_m p_{m-1}$ `; Näherungslösung`
  $r_m = r_{m-1} - \alpha_m A p_{m-1}$ `; Residuum`
  $\beta_m = (r_m^\top r_m)/(r_{m-1}^\top r_{m-1})$ `; Verbesserung in diesem Schritt`
  $p_m = r_m + \beta_m p_{m-1}$ `; neue Suchrichtung`
done

Der Algorithmus ist außerordentlich einfach und sehr unaufwändig. Die einzige Schwierigkeit in der Implementation ist die Wahl eines Konvergenzkriteriums; dies ist jedoch eine Sache für Spezialisten und nicht Gegenstand dieser Vorlesung. In jedem Schritt tritt neben einigen Vektoroperationen nur ein Matrix–Vektor-Produkt auf. D.h. für dünnbesetzte Matrizen ist der Aufwand für einen Iterationsschritt $O(n)$.

Warum der Algorithmus funktioniert, sei im folgenden kurz erläutert. Alles beginnt mit dem folgenden Satz

Theorem 4.5.0.4. *Sei A symmetrisch und positiv definit und b beliebig. Solange der Iterationsprozeß nicht konvergiert hat ($r_{m-1} \neq 0$) treten im Algorithmus keine Divisionen durch Null auf, und es gelten die folgenden Identitäten:*

$$\begin{aligned}\mathscr{K}_m = \langle x_1, x_2, \dots x_m \rangle = \langle p_0, p_1, \dots p_{m-1} \rangle = \\ = \langle r_0, r_1, \dots r_{m-1} \rangle = \langle b, Ab, \dots A^{m-1} b \rangle.\end{aligned}$$

Außerdem sind die Residuen orthogonal

$$r_m^\top r_j = 0, \quad \textit{für } j < m$$

und die Suchrichtungen A–konjugiert

$$p_m^\top A p_j = 0, \quad \textit{für } j < m.$$

Beweis. Der Beweis ist ein Induktionsbeweis ohne wesentliche technische Schwierigkeiten. □

Diese Orthogonalitätseigenschaften enthalten bereits die wesentlichen Aussagen über das Verfahren. Der folgende Satz ist eigentlich nur eine Folgerung daraus.

Theorem 4.5.0.5. *Hat das Verfahren noch nicht konvergiert, dann ist x_m der eindeutige Punkt in $\mathscr{K}_m$, der $\|e_m\|_A$ minimiert. Die Konvergenz ist monoton*

$$\|e_m\|_A \leq \|e_{m-1}\|_A,$$

und $e_m = 0$ wird für ein $m \leq n$ erreicht.

In Gleitkommaarithmetik ist die Garantie für Konvergenz nach höchstens n Schritten leider nicht mehr haltbar. In der Praxis ist der erforderliche Genauigkeit bereits nach $m \ll n$ Schritten erreicht, was die theoretische exakte Konvergenz für $m = n$ unwichtig erscheinen läßt. Worauf man jedoch besonders zu achten hat ist eine gute Vorkonditionierung (siehe Abschnitt 4.7), um eine kriechende Konvergenz zu vermeiden.

Der bekannteste Satz über die Konvergenz des CG Verfahrens ist der folgende:

Theorem 4.5.0.6. *Sei das Konjugierte–Gradienten Verfahren angewendet auf die symmetrische positiv definite Matrix A mit $\kappa_2(A) = \kappa$ und den Vektor b. Sei x_0 der Startvektor des Verfahrens, und sei*

$$e_j = x_j - x_*,$$

wobei $x_* = A^{-1}b$ für die exakte Lösung steht. Dann erfüllen die A–Normen der Fehler

$$\frac{\|e_m\|_A}{\|e_0\|_A} \leq 2 \Big/ \left(\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^m + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^{-m} \right) \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)^m.$$

Für $k \to \infty$ verhält sich der Fehler asymptotisch wie

$$\sim \left(1 - \frac{2}{\sqrt{\kappa}} \right)^k.$$

Man kann also für moderate Konditionszahlen κ erwarten, daß der Algorithmus in $O(\sqrt{\kappa})$ Schritten konvergiert.

Eine der wesentlichen Bedeutungen des CG Verfahrens liegt übrigens darin, dass man es als Algorithmus zur lokalen Optimierung einer reellwertigen Funktion verwenden kann. Diese Anwendung ist Gegenstand von Teil **??**, Kapitel **??**.

4.6 Überblick über andere Iterationsverfahren

Der Nachteil der Krylov–Iterationen für nichtsymmetrische Probleme ist der in jedem Schritt steigende Aufwand durch Verwendung einer $(m+1)$–Terme Rekurrenz. Durch Einführung eines zweiten Krylov–Raumes (für A^*) kann man auch für unsymmetrische Probleme Drei–Terme Rekurrenzen herleiten. Die Verfahren, die so aufgebaut sind heißen *Biorthogonalisierungsverfahren*. Die beiden bekanntesten Vertreter sind das BCG (biconjugate gradients — bikonjugierte Gradienten) und das CGN (auch CGNR) (conjugate gradients for normal equations — konjugierte Gradienten für die Normalengleichungen) Verfahren.

4.6.1 CGN Verfahren

Das CGN Verfahren ist einfach das CG Verfahren angewendet auf die Normalengleichungen

$$A^*Ax = A^*b.$$

A^*A ist hermitesch und positiv definit, und daher ist das CG Verfahren anwendbar. Die entsprechenden Krylov–Räume werden erzeugt durch die Vektoren

$$\{A^*b, (A^*A)A^*b, \ldots, (A^*A)^{m-1}A^*b\}$$

Die Matrix (A^*A) wird dabei nicht explizit gebildet sondern durch zwei Multiplikationen ersetzt. Das CG Verfahren minimiert die (A^*A)–Norm von $x_m - x_*$, und es gilt

$$\|e_m\|_{A^*A} = \sqrt{e_m^* A^* A e_m} = \|Ae_m\|_2 = \|r_m\|_2.$$

Das CGN Verfahren minimiert also die 2–Norm des Residuums in jedem Schritt; es ist also eine Minimale–Residuen Methode wie GMRES. Da sich jedoch die Krylov–Räume unterscheiden, sind die beiden Methoden ebenfalls grundverschieden. Die Konvergenzrate für das CGN Verfahren kann aus derjenigen für das CG Verfahren hergeleitet werden: es gilt

$$\frac{\|r_m\|_2}{\|r_0\|_2} \leq 2\left(\frac{\kappa-1}{\kappa+1}\right),$$

weil $\kappa_2(A^*A) = \kappa_2(A)^2$. Für große κ impliziert das, dass man mit $O(\kappa)$ Iterationen rechnen muß bis zur Konvergenz auf vorgegebene Genauigkeit, ein Wert, der für praktische Anwendungen viel zu hoch ist; speziell wenn die Matrizen nicht sehr gut konditioniert sind.

Das CGN Verfahren ist jedoch dann im Vorteil gegenüber dem CMRES Verfahren, wenn die Matrix brave Singulärwerte besitzt, sich die Eigenwerte aber in einem Ring rund um den Ursprung anordnen. Dann benötigt das CMRES Verfahren $m \approx n$ Iterationsschritte während CGN relativ rasch konvergiert.

4.6.2 BCG Verfahren

Das BCG Verfahren basiert auf der Idee, eine Drei–Terme Rekurrenz dadurch zu erzwingen, dass es eine Faktorisierung der Form

$$A = VTV^{-1}$$

bestimmt mit tridiagonaler Matrix T. Verwendet man auch noch die adjungierte Gleichung

$$A^* = V^{-*}T^*(V^{-*})^{-1},$$

so kann man beide Beziehungen verwenden, um die Faktorisierung zu berechnen. Dabei hilft, dass zwar V nicht mehr unitär ist, aber immer noch die beiden Matrizen V und V^{-*} „orthogonal zueinander“ sind:

$$(V^{-*})^*V = \mathbb{I};$$

daher auch der Ausdruck Biorthogonalisierungsverfahren.

Setzen wir $W = V^{-*}$, so stehen die Spalten v_i bzw. w_j biorthogonal aufeinander im folgenden Sinn:

$$w_i^* v_j = \delta_{ij}.$$

Fassen wir wieder die ersten m Vektoren in die Matrix V_m bzw. W_m zusammen, so kann man die Biorthogonalitätsbeziehungen schreiben als

$$W_m^* V_m = V_m^* W_m = \mathbb{I}.$$

Daraus folgen schließlich die zur Arnoldi Iteration analogen Gleichungen

$$\begin{aligned} AV_m &= V_{m+1}\hat{T}_m \\ A^*W_m &= W_{m+1}\hat{S}_m \\ T_m = S_m^* &= W_m^* A V_m, \end{aligned}$$

mit Tridiagonalmatrizen T_m und S_m. T_m (S_m) erhält man wieder aus $\hat{T}_m$ ($\hat{S}_m$) durch Löschung der letzten Zeile (Spalte). Diese Gleichungen entsprechen den beiden Drei–Terme Rekurrenzen

$$\begin{aligned} Av_m &= \gamma_{m-1}v_{m-1} + \alpha_m v_m + \beta_m v_{m+1}, \\ A^*w_m &= \bar{\beta}_{m-1}w_{m-1} + \bar{\alpha}_m w_m + \bar{\gamma}_m w_{m+1}, \end{aligned}$$

wobei die α_j die Hauptdiagonale, die β_j die untere Nebendiagonale und die γ_j die obere Nebendiagonale von T besetzen.

Die Vektoren v_j liegen im Krylov–Raum zu A und v_1, während die w_j im Krylov–Raum zu A^* und w_1 liegen. Der klassische Algorithmus, der diese Beziehungen ausnützt, ist der folgende:

Bikonjugierte Gradienten

$x_0 = 0$
$p_0 = r_0 = b$
$q_0 = s_0 =$beliebig
for $m = 1$ **to** ? **do**
  $\alpha_m = (s_{m-1}^* r_{m-1})/(q_{m-1}^* A p_{m-1})$
  $x_m = x_{m-1} + \alpha_m p_{m-1}$
  $r_m = r_{m-1} - \alpha_m A p_{m-1}$
  $s_m = s_{m-1} - \alpha_m A q_{m-1}$
  $\beta_m = (s_m^* r_m)/(s_{m-1}^* r_{m-1})$
  $p_m = r_m + \beta_m p_{m-1}$
  $q_m = s_m + \beta_m q_{m-1}$
done

Es gilt $s_m^* r_j = 0$ und $q_m^* A p_j = 0$, woraus wieder die Konvergenz folgt.

Es existieren noch einige weitere Biorthogonalisierungsverfahren, wie CGS (cojugate gradient squared), QMR (quasi minimal residuals), TFQMR (transpose–free QMR), ... alle haben ihre Vor– und Nachteile. Bislang konnte sich jedoch noch keiner dieser Algorithmen durchsetzen. Die Weiterentwicklung der Verfahren stellt im Moment einen großen Forschungsschwerpunkt im Bereich der numerischen linearen Algebra dar, und die Zukunft wird zeigen, welches der Verfahren schließlich die Oberhand gewinnen wird.

4.7 Vorkonditionierung

Wie schon mehrfach in diesem Kapitel erwähnt, ist *Vorkonditionierung* ein wesentlicher Schritt vor der Lösung eines linearen Gleichungssystemes. Der mathematische Hintergrund ist der folgende:

Für jede invertierbare Matrix M haben

$$Ax = b \quad \text{und} \quad M^{-1}Ax = M^{-1}b$$

dieselben Lösungen. Wenn man allerdings das Gleichungssystem mit einem Iterationsverfahren löst, so hängt die Konvergenzgeschwindigkeit im einen System von den Eigenschaften von A, im anderen System jedoch von den Eigenschaften von $M^{-1}A$ ab. Wählt man den *Vorkonditionierer* M gut, so kann das zweite Gleichungssystem mitunter um ein Vielfaches schneller gelöst werden als das erste.

Nun kann man aber $M^{-1}A$ nicht explizit berechnen. Daher muß man den Vorkonditionierer so wählen, dass man Multiplikationen $M^{-1}Ax$ effizient berechnen kann. Außerdem soll das Produkt $M^{-1}A$ schöne Eigenschaften haben, etwa dass $\|M^{-1}A - \mathbb{I}\|$ klein ist, oder dass $M^{-1}A$ wenigstens fast normal ist und die Eigenwerte in einem Cluster liegen.

Es gibt eine Menge an Verfahren, die solche Vorkonditionierer erzeugen, wie etwa *unvollständige Cholesky–Zerlegungen*, *Diagonalskalierungen* oder *Block–Vorkonditionierer*. Alle oder auch nur einige dieser Vorkonditionierer zu dikutieren würde aber den Rahmen dieser Vorlesung sprengen, und daher sei hier nur noch einmal deutlich auf die Notwendigkeit verwiesen und auf die Literatur zu diesem Thema ([Golub, Van Loan 1996] wäre eventuell ein guter Punkt, um eine Beschäftigung mit dieser Problematik zu beginnen).

5 Nichtlineare Gleichungssysteme II: Mehrdimensionaler Fall

5.1 Grundlagen

Wie wir schon in Teil 1 Kapitel **??** gesehen haben ist die Lösung von Gleichungssystemen eine der wichtigsten Teilaufgaben der numerischen Mathematik.

Die Behandlung linearer Gleichungssysteme kann mit Hilfe der Methoden der numerischen linearen Algebra (Teil 1, Kapitel **??**, **??**, 4) auf einfache Weise algorithmisiert werden. Daß dies relativ einfach mit vorherbestimmbarem Aufwand möglich ist, folgt aus der Linearität der Gleichung. Im linearen Fall (alle Variablen treten in allen Termen aller Gleichungen linear auf — nicht innerhalb transzendenter Funktionen) kann man aus den Parametern der Gleichungen algorithmisch ablesen ob es eine eindeutige Lösung gibt, bzw. wieviel dimensional der Vektorraum der Lösungen ist. In diesem Sinn sind alle linearen Probleme abgesehen von numerischen Schwierigkeiten äquivalent.

Im nichtlinearen Fall ist das nicht so einfach. Weder kann man durch die genaue Analyse *eines* Problems Rückschlüsse auf alle anderen ziehen noch gibt es für den größten Teil der nichtlinearen Probleme effiziente Algorithmen, um alle Lösungen zu finden. Ist die Gleichung eindimensional (eine Gleichung in einer Variablen) dann kann man durch Monotonieuntersuchungen und Bisektion mit relativ geringem Aufwand *alle* Lösungen einer nichtlinearen Gleichung finden - jedenfalls bis auf numerisch bedingte Ungenauigkeiten.

In höheren Dimensionen ist dies nicht mehr so einfach möglich. Die Auffindung aller Lösungen einer beliebigen nichtlinearen Gleichung ist NP–hart und kann als Spezialfall eines globalen Optimierungsproblems betrachtet werden. In Kapitel **??** werden wir Algorithmen zur Behandlung des allgemeinen Problems besprechen. Möchte man nicht extremen Aufwand treiben, so muß man sich damit begnügen nur eine Lösung des Gleichungssystems approximativ zu finden.

5.1.1 Problemstellung

Wir behandeln in diesem Kapitel also die folgende Fragestellung: Sei $f : E \to F$ eine Funktion zwischen zwei Mengen E und F. Wir suchen zu einem $\eta \in F$ *ein* $\xi \in E$ mit $f(\xi) = \eta$. Ohne Einschränkung ist diese Aufgabenstellung natürlich viel zu allgemein. Wir können nicht erwarten, sie für beliebige E, F und f numerisch behandeln zu können. Um analytische Methoden anwenden zu können, müssen wir zuerst E und F auf Teilmengen des $\mathbb{R}^n$ einschränken. Doch selbst dann ist das Problem noch hoffnungslos, was dessen Analyse betrifft. Beliebige Funktionen zwischen beliebigen Teilmengen des $\mathbb{R}^n$ erlauben viel zu allgemeine Konstellationen, um auf numerischem Wege untersucht werden zu können.

Als Funktionen f wird man daher im allgemeinen nur (stückweise) stetige - oder noch stärker eingeschränkte - zulassen; die Mengen E und F sollten am besten abgeschlossen und zusammenhängend sein. Es zeigt sich, daß auch dieses Problem, wenn man E und F durch Gleichungen beschreiben kann, äußerst schwierig handzuhaben ist (siehe Kapitel **??**), und daher wollen wir uns fürs erste auf den einfachsten Fall beschränken:

> Sei $f : \mathbb{R}^n \to \mathbb{R}^n$ stetig. Wir suchen $\xi \in \mathbb{R}^n$ mit $f(\xi) = 0$.

Man beachte, daß „$= 0$“ keine Einschränkung bedeutet, da die Gleichungen im Fall $E = F = \mathbb{R}^n$ immer so umgeformt werden können, daß die Aufgabe auf ein Nullstellenproblem reduziert wird.

Ferner lohnt es noch, sich klarzumachen, daß Minimierungsprobleme und Nullstellenprobleme für Funktionen im $\mathbb{R}^n$ stark zusammenhängen. Sei nämlich $h : \mathbb{R}^n \to \mathbb{R}$ stetig differenzierbar, dann ist $\xi \in \mathbb{R}^n$ nur

dann ein lokales Minimum von h, wenn ξ Nullstelle des Gradienten $\nabla h = (\frac{\partial h}{\partial x_1}, \ldots, \frac{\partial h}{\partial x_n})^\top : \mathbb{R}^n \to \mathbb{R}^n$ ist. Man kann also die Aufgabe

$$\xi = \min_{x \in \mathbb{R}^n} h(x), \tag{5.1}$$

ξ sei dabei ein *lokales* Minimum, lösen, indem man eine Nullstelle des Gradienten $f := \nabla h$ sucht und dann diese Nullstelle mit Hilfe der zweiten Ableitungen weiter untersucht. Umgekehrt ist jede Lösung des Gleichungssystemes $f(\xi) = 0$ lokales Minimum der Funktion

$$h(x) = \|f(x)\|_2^2.$$

Aufgabe (5.1) ist ein typisches Beispiel für ein lokales Optimierungsproblem ohne Nebenbedingungen. In diesem Sinn sind in diesem Abschnitt bereits die Anfangsgründe für die Verfahren zur lokalen Optimierung aus Kapitel **??** enthalten.

5.1.2 Iterationsverfahren

Die meisten Methoden zur Lösung des Nullstellenproblems

$$f(\xi) = 0$$

gehen iterativ vor. Die grundlegenden Begriffe, die Iterationsverfahren betreffen, sind schon im Teil 1, in den Kapiteln **??** und **??** behandelt worden. Wir wollen diese hier nur noch einmal kurz zusammenfassen.

Sei also im weiteren das folgende Iterationsverfahren gegeben:

$$x_{i+1} = \Phi(x_i), \quad \text{mit Startwert } x_0.$$

Φ heißt in diesem Fall die *Iterationsfunktion*.

Definition 5.1.2.1. *Sei ξ ein Fixpunkt der Iterationsfunktion Φ, d.h.*

$$\Phi(\xi) = \xi,$$

und es gelte für alle Startvektoren x_0 aus einer Umgebung U von ξ und jede zugehörige Folge $(x_n)_n$ die folgende Abschätzung:

$$\|x_{i+1} - \xi\| \leq C\|x_i - \xi\|^p,$$

wobei für $p = 1$ gelte $C < 1$. In diesem Fall nennt man das durch Φ erzeugte Iterationsverfahren ein Verfahren mindestens p–ter Ordnung.

Theorem 5.1.2.2. *Jedes Verfahren mindestens erster Ordnung ist* lokal konvergent *in dem Sinne, daß es zu jedem Fixpunkt ξ eine Umgebung U gibt, sodaß für alle Startpunkte $x_0 \in U$ die durch die Iterationsfunktion Φ erzeugte Folge gegen ξ konvergiert. Kann man U auf den gesamten Raum $\mathbb{R}^n$ ausdehnen, so nennt man das Verfahren* global konvergent.

Zusätzlich werden wir noch einen wichtigen Begriff aus der Analysis benötigen, den wir hier ebenfalls noch einmal definieren werden.

Definition 5.1.2.3. *Eine Teilmenge $C \subseteq \mathbb{R}^n$ heißt* konvex*, falls für jedes Paar $(x,y) \in C \times C$ von Punkten aus C auch deren Verbindungsstrecke*

$$\overline{xy} := \{x + t(y-x) \,|\, t \in [0,1]\}$$

in C liegt, also $\overline{xy} \subseteq C$.

5.2 Fixpunktverfahren

Wir wollen nun einfache Iterationsverfahren konstruieren, die mit möglichst hoher Konvergenzordnung das Nullstellenproblem

$$f(\xi) = 0$$

lösen. Wie im eindimensionalen Fall beginnen wir mit der einfachen Idee der linearen Approximation von f. Die Konstruktion verschiedener Iterationsfunktionen Φ ist die Grundidee in diesem Abschnitt.

5.2.1 Newton–Verfahren

Das allgemeine n–dimensionale Newton–Verfahren wird ganz analog zum eindimensionalen konstruiert. Die Funktion $f : \mathbb{R}^n \to \mathbb{R}^n$ wird durch ihr lineares Taylorpolynom approximiert

$$f(x) \approx f(y) + Df(y)(x-y).$$

Dann wird die Nullstelle der Approximationsfunktion gesucht und als neuer Wert für die Approximation verwendet. Genauer kann man das Iterationsverfahren schreiben als

$$x_{i+1} = x_i - (Df(x_i))^{-1} f(x_i),$$

sofern nur $Df(x_i)$ regulär ist. Die Iterationsfunktion Φ ist also im Fall des *allgemeinen Newton–Verfahren* folgendermaßen definiert:

$$\Phi(x) = x - (Df(x))^{-1} f(x).$$

Die Iteration wird so lange wiederholt bis der Wert von f im Absolutbetrag unter eine vorgegebene Schranke ε gefallen ist. Algorithmisch können wir das folgendermaßen fassen.

Allgemeines Newton–Verfahren

Wähle x_0 und Toleranz ε
$x = x_0$
$v = f(x)$
while $\|v\| \geq \varepsilon$ **do**
    Berechne $Df(x)$
    Löse $Df(x)w = v$
    $x = x - w$
    $v = f(x)$
done

Die Abschätzung des Aufwandes ist nicht ganz einfach, da wir noch nichts über die Konvergenz(ordnung) der Iteration wissen. Wir können jedoch einiges über den Aufwand eines einzelnen Iterationsschrittes aussagen. Der Hauptaufwand liegt in der Berechnung von $Df(x)$ und in der Lösung des darauf folgenden linearen Gleichungssystemes. Die Bestimmung von $Df(x)$ erfordert die Berechnung von n^2 Zahlen. Die Lösung des linearen Gleichungssystemes erfordert zusätzlich noch $O(n^3)$ elementare Rechenoperationen, wie wir aus Teil 1 Kapitel **??** wissen. Bei solch hohem Aufwand sollte die Anzahl der Iterationsschritte höchstens $O(1)$ sein, da sonst der Algorithmus für hochdimensionale Probleme hoffnungslos langsam wäre.

Glücklicherweise stimmt im Mehrdimensionalen, was wir aus den Erfahrungen mit dem eindimensionalen Newton–Verfahren erwarten. Das Resultat ist im folgenden Satz zusammengefaßt:

Theorem 5.2.1.1. *Es seien eine offene Menge $U \subseteq \mathbb{R}^n$ und eine konvexe Menge C mit $\overline{C} \subseteq U$ gegeben. Ferner wollen wir eine Funktion $f : U \to \mathbb{R}^n$, die an allen $x \in C$ differenzierbar ist und an allen $x \in U$ stetig ist, betrachten. Gibt es für ein $x_0 \in C$ positive Konstanten $r, \alpha, \beta, \gamma, h$ mit*

(a) $B_r(x_0) \subseteq C$,

(b) $h := \frac{\alpha\beta\gamma}{2} < 1$,

(c) $r := \frac{\alpha}{1-h}$,

wobei $B_r(x) := \{y \in \mathbb{R}^n \mid \|y-x\| < r\}$, *und*

(d) $\|Df(x) - Df(y)\| \leq \gamma\|x-y\|$ *für alle* $x, y \in C$,

(e) für alle $x \in C$ *ist* $Df(x)$ *regulär und es gilt* $\|(Df(x))^{-1}\| \leq \beta$,

(f) $\|(Df(x_0))^{-1} f(x_0)\| \leq \alpha$,

so haben wir die folgenden Aussagen:

1. *Wählt man* x_0 *als Startwert, so ist jedes Element der durch Iteration erzeugten Folge*
$$x_{i+1} := x_i - Df(x_i)^{-1} f(x_i)$$
wohldefiniert, und es gilt $x_i \in B_r(x_0)$ *für alle* $i \geq 0$.
2. $\lim_{k\to\infty} x_k = \xi$ *existiert,* $\xi \in \overline{B_r(x_0)}$ *und* $f(\xi) = 0$.
3. *Für alle* $k \geq 0$ *gilt die Abschätzung*
$$\|x_k - \xi\| \leq \alpha \frac{h^{2^k-1}}{1-h^{2^k}}.$$
Wegen der Voraussetzung $0 < h < 1$ *ist das Newton–Verfahren also mindestens lokal* ***quadratisch konvergent***.

Beweis. Aus Annahme (e) folgt, daß x_{k+1} so lange wohldefiniert bleibt wie $x_k \in B_r(x_0)$ gilt. Für $k = 0$ und $k = 1$ folgt das aus den Voraussetzungen (d) und (f). Für die übrigen k beweisen wir die Aussage mittels vollständiger Induktion.

$$\begin{aligned}\|x_{k+1} - x_k\| &= \|-(Df(x_k))^{-1} f(x_k)\| \leq \beta\|f(x_k)\| = \\ &= \beta\|f(x_k) - f(x_{k-1}) - Df(x_{k-1})(x_k - x_{k-1})\|.\end{aligned}$$

Um den letzten Term abschätzen zu können, benötigen wir ein Resultat, das im wesentlichen aus dem Satz von Taylor und der Kettenregel folgt:

Lemma 5.2.1.2. *Existiert* $Df(x)$ *für alle* $x \in K \subseteq \mathbb{R}^n$, K *konvex und gibt es eine Konstante* λ *mit*
$$\|Df(y) - Df(z)\| \leq \lambda\|y-z\|$$
für alle $y, z \in K$, *dann gilt für alle* $y, z \in K$
$$\|f(y) - f(z) - Df(z)(y-z)\| \leq \tfrac{\lambda}{2}\|y-z\|^2.$$

Beweis. Für beliebige $y, z \in K$ sei $h(t) := f(z + t(y-z))$. Als Zusammensetzung differenzierbarer Funktionen ist h differenzierbar für $t \in [0,1]$, und aus der Kettenregel folgt
$$h'(t) = Df(z + t(y-z))(y-z).$$
Mit dieser Rechnung können wir jetzt folgendermaßen abschätzen:

$$\begin{aligned}\|f(y) - f(z) - Df(z)(y-z)\| &= \|h(1) - h(0) - h'(0)\| = \left\| \int_0^1 (h'(t) - h'(0))\, dt \right\| \leq \\ &\leq \int_0^1 \|h'(t) - h'(0)\|\, dt \leq \int_0^1 \|Df(z + t(y-z)) - Df(z)\|\, \|y-z\|\, dt \leq \\ &\leq \int_0^1 \lambda t \|z-y\|^2\, dt = \tfrac{\lambda}{2}\|z-y\|^2.\end{aligned}$$

□

Fortsetzung des Beweises von Theorem 5.2.1.1:
Aus diesem Resultat erhalten wir sofort

$$\|x_{k+1} - x_k\| \leq \tfrac{\beta\gamma}{2}\|x_k - x_{k-1}\|^2,$$

woraus wir mittels Voraussetzungen (b) und (f) und Induktion sofort

$$\|x_{k+1} - x_k\| \leq \alpha h^{2^k-1} \tag{5.2}$$

ableiten können. Mit Hilfe der Dreiecksungleichung folgern wir dann weiter

$$\begin{aligned}\|x_{k+1} - x_0\| &\leq \|x_{k+1} - x_k\| + \|x_k - x_{k-1}\| + \cdots + \|x_1 - x_0\| \leq \\ &\leq \alpha(1 + h + h^3 + h^7 + \cdots + h^{2^k-1}) < \tfrac{\alpha}{1-h} = r,\end{aligned}$$

was $x_{k+1} \in B_r(x_0)$ und damit Behauptung 1 impliziert.

Wegen Gleichung (5.2) und der folgenden Abschätzung

$$\begin{aligned}\|x_{m+1} - x_n\| &\leq \|x_{m+1} - x_m\| + \cdots + \|x_{n+1} - x_n\| \leq \\ &\leq \alpha h^{2^n-1}(1 + h^{2^n} + h^{2^n 2} + \cdots + h^{2^n m-n}) < \\ &< \tfrac{\alpha h^{2^n-1}}{1-h^{2^n}} < \varepsilon\end{aligned} \tag{5.3}$$

ist die Folge der x_k eine Cauchyfolge, welche wegen der Vollständigkeit von $\mathbb{R}^n$ gegen einen Wert ξ konvergiert. Da nach 1. alle Glieder der Folge in $B_r(x_0)$ liegen, liegt der Grenzwert $\xi \in \overline{B_r(x_0)}$.

Gleichung (5.3) impliziert auch Behauptung 3, indem man den Grenzübergang $m \to \infty$ betrachtet:

$$\lim_{m\to\infty} \|x_m - x_n\| = \|\xi - x_n\| \leq \tfrac{\alpha h^{2^n-1}}{1-h^{2^n}}.$$

Was noch zu zeigen bleibt ist, daß ξ tatsächlich Nullstelle von f ist. Wegen (d) haben wir

$$\begin{aligned}\|Df(x_k) - Df(x_0)\| \leq \gamma\|x_k - x_0\| < \gamma r \quad \text{und} \\ \|Df(x_k)\| \leq \gamma r + \|Df(x_0)\| := L,\end{aligned}$$

weil außerdem $x_k \in B_r(x_0)$ liegt. Formen wir noch die Definition der Iteration um, so erhalten wir

$$\|f(x_k)\| = \| - Df(x_k)(x_{k+1} - x_k)\| \leq \|Df(x_k)\|\,\|x_{k+1} - x_k\| \leq L\|x_{k+1} - x_k\|.$$

Weil $(x_k)_k$ eine Cauchyfolge ist, folgt

$$\lim_{k\to\infty} \|f(x_k)\| = 0,$$

und wegen der Stetigkeit von f in U haben wir $\lim_{k\to\infty} \|f(x_k)\| = \|f(\xi)\|$, was beweist, daß ξ Nullstelle von f ist. □

Mit etwas verstärkten Voraussetzungen kann man sogar noch zeigen, daß ξ die einzige Nullstelle in $B_r(x_0)$ ist.

In Anwendungen werden die Voraussetzungen oft nicht geprüft sondern man geht einfach davon aus, daß die Folge konvergiert. Daß bei dieser Art vorzugehen jedoch Vorsicht geboten ist, werden wir im nächsten Abschnitt erkennen. Zuvor jedoch noch ein Beispiel zur Illustration.

Beispiel 5.2.1.3. *Sei das Nullstellenproblem $f(x) = 0$ mit*

$$f(x) = \begin{pmatrix} x_1^2 - 4x_1x_2 + x_3^2 - \frac{x_2x_3}{2} \\ \sin(\pi x_1^2) + 2x_2 - 3x_3 + 7 \\ \cosh(x_3 - x_2 - 1) - x_1 \end{pmatrix}$$

gegeben. Um das Newtonverfahren zur Lösung des Problems zu verwenden, müssen wir zuerst die Jacobimatrix von f bestimmen:

$$Df(x) = \begin{pmatrix} 2x_1 - 4x_2 & -4x_1 - \frac{x_3}{2} & -\frac{x_2}{2} + 2x_3 \\ 2\pi\cos(\pi x_1^2) & 2 & -3 \\ -1 & \sinh(1+x_2-x_3) & -\sinh(1+x_2-x_3) \end{pmatrix}.$$

Ausgehend vom Startwert $x = (0.5, 12, 10)$ bestimmen wir Schrittweise die folgenden Punkte:

k	x_k			$\|f(x_k)\|$
0	0.5000000000	12.00000000	10.00000000	18.93450000
1	1.199626781	12.60562314	11.49084434	4.457012910
2	0.8343344427	7.410425222	7.120878107	1.720078006
3	0.9548245031	6.534573843	6.840092039	0.5226925059
4	0.958977746	5.167866923	5.861886381	0.3234079524
5	0.9815951629	4.402683337	5.307732421	0.1664317682
6	0.9965529932	4.05646574	5.045203193	$4.456496668 \cdot 10^{-2}$
7	0.9999366525	4.000457763	5.000450925	$1.520846802 \cdot 10^{-3}$
8	1.0000000000	3.999999841	4.999999898	$2.205649753 \cdot 10^{-7}$
9	1.0000000000	4.000000000	5.000000000	

Ab Schritt 5 befindet sich das Verfahren in dem Bereich, in dem es quadratisch konvergiert, wie man aus der Größe der Fehler deutlich ablesen kann.

Einen Nachteil des allgemeinen Newton–Verfahrens kann man aufspüren, wenn man versucht, das Nullstellenproblem mit anderen Startwerten zu lösen. Versucht man es etwa mit dem — näher an $(1,4,5)$ liegenden — Startwert $x_0 = (0,4,9)$, so erhält man eine Folge, die sich zuerst erratisch hin und her bewegt und dann bei $k = 84$ am Punkt $x_{84} = (-93.03541212, -15.71451993, 64.76081246)$ den Algorithmus mit einem Überlauffehler während der Berechnung von $f_3(x_{84}) \approx 1.639337246353882 \cdot 10^{34}$ abbricht. Das allgemeine Newton–Verfahren ist nicht global konvergent.

Möchte man Arbeit einsparen, so kann man das allgemeine Newton–Verfahren so abändern, daß man das Update der Jacobi–Matrix von f unterläßt und anstatt dessen immer mit der Jacobi–Matrix am Startwert rechnet. Ist der Startwert gut gewählt, dann ist $Df(x_0)$ eine passable Näherung für $Df(x_k)$. Dieses *vereinfachte Newton–Verfahren* verwendet also als Iterationsfolge die Iterationsfunktion

$$\Phi(x) = x - (Df(x_0))^{-1} f(x).$$

Dieses Vorgehen hat den Vorteil, daß man die n^2 Werte der Funktionalmatrix nur einmal berechnen muß und ebenso die *LR*–Zerlegung zur Lösung des linearen Gleichungssystems. Geht man so vor, dann erhält man den folgenden Algorithmus.

Vereinfachtes Newton–Verfahren

Wähle x_0 und Toleranz ε
$x = x_0$
Berechne $A = Df(x_0)$
$v = f(x)$
while $|v| \geq \varepsilon$ **do**
    Löse $Aw = v$
    $x = x - w$
    $v = f(x)$
done

Für diesen Algorithmus ist der Hauptaufwand in jedem Iterationsschritt die Lösung des linearen Gleichungssystemes, das bei bereits vorliegender *LR*–Zerlegung $O(n^2)$ elementare Rechenschritte benötigt. Die Anwendbarkeit dieses Verfahrens ist jedoch durch die beiden Fakten beschränkt, daß die Konvergenzordnung nur **linear** ist und daß der Startwert wirklich nahe an der Nullstelle liegen muß. Ist das nicht der Fall, so ist $Df(x_0)$ keine gute Näherung für $Df(x_k)$ und das Verfahren konvergiert überhaupt nicht.

Beispiel 5.2.1.4. *Wir betrachten wieder die Funktion f aus Beispiel 5.2.1.3. Diesmal wählen wir als Startwert den Punkt $x_0 = (0.9815951629, 4.402683337, 5.307732421)$, der auch in Schritt 5 des allgemeinen Newton–Verfahrens aufgetreten ist. Führt man das vereinfachte Newton–Verfahren durch, so konvergiert es in 15 Schritten:*

k	x_k			$\|f(x_k)\|$
0	0.9815951629	4.402683337	5.307732421	0.1664317682
1	0.9965529932	4.056465741	5.045203193	$4.456496647 \cdot 10^{-2}$
2	0.9991609589	4.010956729	5.009184449	$1.410488679 \cdot 10^{-2}$
3	0.9998415503	4.001368106	5.001278591	$3.561033034 \cdot 10^{-3}$
4	0.9999844468	3.999897697	4.999971781	$6.575133491 \cdot 10^{-4}$
5	1.000003445	3.999867673	4.999905554	$5.65119618 \cdot 10^{-5}$
6	1.000002496	3.999949242	4.999960884	$1.810916676 \cdot 10^{-5}$
7	1.00000086	3.999986531	4.999989135	$1.143892528 \cdot 10^{-5}$
8	1.000000212	3.999997434	4.999997812	$3.793227186 \cdot 10^{-6}$
9	1.000000036	3.999999756	4.999999752	$9.069735509 \cdot 10^{-7}$
10	1.0000000020	4.000000060	5.000000033	$1.487105113 \cdot 10^{-7}$
11	0.9999999986	4.000000042	5.000000030	$6.222409192 \cdot 10^{-9}$
12	0.9999999992	4.000000014	5.000000011	$7.002019409 \cdot 10^{-9}$
13	0.9999999998	4.000000003	5.000000003	$3.413833529 \cdot 10^{-9}$
14	0.9999999999	4.000000001	5.000000001	$1.028712258 \cdot 10^{-9}$
15	1.0000000000	4.000000000	5.000000000	

Man kann deutlich das lineare Konvergenzverhalten erkennen.

5.2.2 Ein modifiziertes Newton–Verfahren

Das allgemeine Newton–Verfahren hat leider zwei große Nachteile. Zum ersten ist der Aufwand in jedem einzelnen Iterationsschritt sehr hoch ($O(n^3)$), zum anderen ist das Verfahren in den meisten Fällen wirklich nur lokal konvergent. Das bedeutet aber, daß man einen guten Startwert finden muß, denn andernfalls konvergiert das Verfahren nicht gegen eine Nullstelle (siehe Beispiel 5.2.1.3). Dieses Verhalten tritt auch schon in eindimensionalen Beispielen auf.

Beispiel 5.2.2.1. *Wir versuchen das Nullstellenproblem $f(x) = 0$ mit*

$$f(x) = \arctan(x)$$

zu lösen. Die Iterationsfolge des allgemeinen Newton–Verfahrens ist gegeben durch die Beziehung

$$x_{k+1} = x_k - (1 + x_k^2)\arctan(x_k).$$

Wählt man den Startwert jedoch so, daß

$$\arctan(|x_0|) \geq \frac{2|x_0|}{1 + x_0^2}$$

gilt, so divergiert die Folge der $|x_k|$, denn es folgt

$$\lim_{k \to \infty} |x_k| = \infty.$$

In diesem Abschnitt wollen wir Methoden entwickeln, die *global konvergieren*. Sie werden hauptsächlich darin bestehen, in jedem Schritt *Suchrichtungen* s_k und *Schrittweiten* λ_k zu bestimmen und damit die Folge

$$x_{k+1} = x_k - \lambda_k s_k$$

zu konstruieren. Die Schrittweiten werden dabei so gewählt, daß die Funktion $h(x) = \|f(x)\|^2$ streng monoton fällt und damit die x_k gegen ein Minimum von h konvergieren. Diese Vorgangsweise zur Konstruktion einer Folge heißt auch *Liniensuche*.

Im Falle des modifizierten Newton–Verfahrens wird als Suchrichtung die Newton–Richtung $(Df(x_k))^{-1}f(x_k)$ gewählt. Zur Bestimmung der Schrittweite müssen wir noch ein wenig über Minimierungsverfahren ohne Nebenbedingungen herleiten.

Einfache Minimierungsverfahren

Um ein Verfahren mit Liniensuche vernünftig durchführen zu können, muß man eine Möglichkeit finden, s_k und λ_k so zu wählen, daß die Funktion h auf der Folge der x_k monoton fällt. Ideal wäre es natürlich, wenn man z.B. λ_k immer so wählen könnte, daß h auf dem Strahl $x_k - \mu s_k$ minimiert wird. Das sei auch die erste Idee für einen Algorithmus. Bevor wir den Algorithmus formulieren, müssen wir allerdings noch eine kurze Definition einschieben, die Menge der „sinnvollen Suchrichtungen“.

Definition 5.2.2.2. *Sei*

$$D(\gamma, x) := \{y \in \mathbb{R}^n \mid \|y\|_2 = 1 \text{ mit } \nabla h(x)y \geq \gamma \|\nabla h(x)\|_2\}$$

definiert für $\gamma \in]0,1]$. *Diese Menge beschreibt alle jene Richtungen, die einen nicht zu großen (spitzen) Winkel mit dem Gradienten* $\nabla h(x)$ *von h an x bilden.*

Exakte Liniensuche

Wähle einen Startwert x_0 und eine Toleranz ε,
Wähle eine Zahlenfolge $(\gamma_k)_k$ mit $\inf_k \gamma_k > 0$,
Wähle eine Zahlenfolge $(\sigma_k)_k$ mit $\inf_k \sigma_k > 0$,
$k = 0$
while $\|\nabla h(x_k)\| > \varepsilon$ **do**
  Wähle $s_k \in D(\gamma_k, x_k)$,
  Bestimme $\lambda_k \in I_k := [0, \sigma_k \|\nabla h(x_k)\|_2]$ so, daß
    $h(x_k - \lambda_k s_k) = \min_{\mu \in I_k} h(x_k - \mu s_k)$.
  $x_{k+1} = x_k - \lambda_k s_k$
  $k = k + 1$
done

Dieses Verfahren konvergiert gemäß dem folgenden Resultat:

Proposition 5.2.2.3. *Seien* $x_0 \in \mathbb{R}^n$ *und* $h : \mathbb{R}^n \to \mathbb{R}$ *eine Funktion mit folgenden Eigenschaften:*

(a) Die Menge $K := \{x \in \mathbb{R}^n | h(x) \leq h(x_0)\}$ *ist kompakt;*

(b) h ist auf einer Umgebung von K stetig differenzierbar.

Dann gilt für die Folge $(x_k)_k$, *die durch den Algorithmus 5.2.2 bestimmt wird:*

1. $x_k \in K$ *für alle k, und daher besitzt* $\{x_k\}$ *mindestens einen Häufungspunkt* $\bar{x}$ *in K.*
2. *Jeder Häufungspunkt* $\bar{x}$ *von* $(x_k)_k$ *ist stationärer Punkt von h, d.h.* $\nabla h(\bar{x}) = 0$.

Beweis. 1. Nach Konstruktion ist h auf den Folgengliedern monoton fallend. Daher ist $x_k \in K$ für alle k. Der Rest folgt aus der Kompaktheit von K.

2. Sei $\bar{x}$ ein Häufungspunkt von $(x_k)_k$, der nicht stationärer Punkt von h ist, also mit $\nabla h(\bar{x}) \neq 0$. O.B.d.A. können wir annehmen, daß $\lim_{k\to\infty} x_k = \bar{x}$ gilt. Seien $\gamma := \inf_k \gamma_k$, $\sigma := \inf_k \sigma_k$.

   Behauptung: Es gibt eine Umgebung $U(\bar{x})$ und eine Zahl $\lambda > 0$ mit

   $$h(x-\mu s) \leq h(x) - \mu\tfrac{\gamma}{4}\|\nabla h(\bar{x})\|_2.$$

   für alle $x \in U(\bar{x})$ und $s \in D(\gamma, x)$ und $0 \leq \mu \leq \lambda$.

   Beweis der Behauptung. Aus der Stetigkeit von $\nabla h(x)$ auf einer Umgebung $V(\bar{x})$ von K folgt, weil $\nabla h(x) \neq 0$, daß die Menge

   $$U_1(\bar{x}) := \{x \in V(\bar{x}) \,|\, \|\nabla h(x) - \nabla h(\bar{x})\|_2 \leq \tfrac{\gamma}{4}\|\nabla h(\bar{x})\|\}$$

   eine Umgebung von $\bar{x}$ ist. Ebenso zeigt man, daß

   $$U_2(\bar{x}) := \left\{x \in K \,|\, D(\gamma, x) \subseteq D(\tfrac{\gamma}{2}, \bar{x})\right\}$$

   eine Umgebung von $\bar{x}$ ist. Der Schnitt zweier Umgebungen ist wieder Umgebung, und weil $\mathbb{R}^n$ ein metrischer Raum ist, gibt es $\lambda > 0$ mit

   $$\overline{B_{2\lambda}(\bar{x})} \subseteq U_1(\bar{x}) \cap U_2(\bar{x}).$$

   Mit diesen Definitionen können wir

   $$U(\bar{x}) := \overline{B_\lambda(\bar{x})}$$

   setzen. Dann existiert für $x \in U(\bar{x})$ und $0 \leq \mu \leq \lambda$, $s \in D(\gamma, x)$ ein $\theta \in (0,1)$ mit

   $$\begin{aligned} h(x) - h(x-\mu s) &= \mu \nabla h(x - \theta\mu s)s = \\ &= \mu\Big(\big(\nabla h(x-\theta\mu s) - \nabla h(\bar{x})\big)s + \nabla h(\bar{x})s\Big). \end{aligned}$$

   Nach Konstruktion gilt für $x \in U(\bar{x})$, μ, θ, s wie oben auch $x - \mu s, x - \theta\mu s \in U_1 \cap U_2$, und daher

   $$\begin{aligned} h(x) - h(x-\mu s) &\geq -\tfrac{\mu\gamma}{4}\|\nabla h(\bar{x})\|_2 + \mu \nabla h(\bar{x})s \geq \\ &\geq -\tfrac{\mu\gamma}{4}\|\nabla h(\bar{x})\|_2 + \tfrac{\mu\gamma}{2}\|\nabla h(\bar{x})\|_2 = \\ &= \tfrac{\mu\gamma}{4}\|\nabla h(\bar{x})\|_2. \end{aligned}$$

□

Fortsetzung des Beweises von Proposition 5.2.2.3 Wegen $\lim_{k\to\infty} x_k = \bar{x}$ und $\nabla h(\bar{x}) \neq 0$ gibt es ein $N \in \mathbb{N}$, sodaß für alle $n \geq N$ folgt

a) $x_n \in U(\bar{x})$,
b) $\|\nabla h(x_n)\|_2 \geq \frac{1}{2}\|\nabla h(\bar{x})\|_2$.

Wenn wir jetzt $\Lambda := \min_{\mu\in[0,\Lambda]} h(x_n - \mu s_n)$ und $\varepsilon := \Lambda\frac{\gamma}{4}\|\nabla h(\bar{x})\|_2$ setzen, dann gilt wegen $\sigma_n \geq \sigma$ die Inklusion $[0,\Lambda] \subseteq [0, \sigma_n\|\nabla h(x_n)\|_2]$ für alle $n \geq N$. Nach Definition von x_{n+1} wissen wir, daß

$$h(x_{n+1}) \leq h(x_n) - \tfrac{\Lambda\gamma}{4}\|\nabla h(\bar{x})\|_2 = h(x_k) - \varepsilon$$

für alle $n \geq N$. Nun ist aber $\varepsilon > 0$, und daher folgt sofort $\lim_{k\to\infty} h(x_k) = -\infty$, was im Widerspruch zu $h(x_k) \geq h(x_{k+1}) \geq h(\bar{x})$ steht. Daher gilt $\nabla h(\bar{x}) = 0$.

□

Unglücklicherweise ist der Aufwand für Algorithmus 5.2.2 unverhältnismäßig hoch. Man muß nämlich in jedem Schritt zur Berechnung von x_{k+1} das globale Minimum der Funktion

$$\mu \mapsto h(x_k - \mu s_k)$$

auf $[0, \sigma_k \|\nabla h(x_k)\|_2]$ bestimmen, was man im allgemeinen nur näherungsweise, iterativ, mit bedeutendem Rechenaufwand kann. Doch man kann das Verfahren dadurch verbessern, daß man die Minimumsuche auf einen endlichen Prozeß reduziert. Der so entstehende Algorithmus wird *Armijo Liniensuche* (*Armijo–line search*) genannt.

Armijo Liniensuche

Wähle einen Startwert x_0 und eine Toleranz ε,
Wähle eine Zahlenfolge $(\gamma_k)_k$ mit $\inf_k \gamma_k > 0$,
Wähle eine Zahlenfolge $(\sigma_k)_k$ mit $\inf_k \sigma_k > 0$,
$k = 0$
while $h(x_k) > \varepsilon$ **do**
    Wähle $s_k \in D(\gamma_k, x_k)$,
    $\rho_k = \sigma_k \|\nabla h(x_k)\|_2$
    $h_k(\mu) = h(x_k - \mu s_k)$
    Bestimme die kleinste Zahl $j \geq 0$ so, daß
        $h_k(\rho_k 2^{-j}) \leq h_k(0) - \rho_k 2^{-j} \frac{\gamma_k}{4} \|\nabla h(x_k)\|_2$,
    Bestimme $i \in \{0, 1, \ldots, j\}$ so daß
        $h_k(\rho_k 2^{-i}) = \min_{1 \leq n \leq j} h_k(\rho_k 2^{-n})$,
    $\lambda_k = \rho_k 2^{-i}$,
    $x_{k+1} = x_k - \lambda_k s_k$,
    $k = k + 1$
done

In diesem Verfahren wird der Punkt x_{k+1} in endlich vielen Rechenschritten konstruiert. Meist beschränkt man sich sogar darauf, eine obere Schranke M für das zu suchende j anzugeben und das Verfahren zu terminieren, wenn $j \geq M$ und damit $2^{-j} \leq 2^{-M}$ wird.

Ohne diese Modifikation gilt das folgende Resultat, das im Prinzip eine Kopie von Proposition 5.2.2.3 ist.

Proposition 5.2.2.4. *Unter den Voraussetzungen von Proposition 5.2.2.3 gelten auch für die von Algorithmus 5.2.2 gelieferten Folgen $(x_k)_k$ die dort angegebenen Aussagen.*

Beweis. Eine Verfeinerung der Abschätzungen im Beweis von Proposition 5.2.2.3. Der vollständige Beweis kann etwa in [Stoer 1994a, 5.4.1] gefunden werden. □

Das modifizierte Newton–Verfahren

Zur Lösung des Nullstellenproblems $f(x) = 0$ setzen wir wie angedeutet $h(x) = \|f(x)\|_2^2$ und wenden Algorithmus 5.2.2 an. Was noch bleibt, ist eine gute Wahl der Suchrichtungen s_k und der γ_k und σ_k.

Im modifizierten Newton–Verfahren wählt man als Suchrichtungen s_k die Newton–Richtungen

$$s_k := \frac{d_k}{\|d_k\|_2}, \quad d_k := Df(x_k)^{-1} f(x_k),$$

falls $Df(x_k)$ regulär und $f(x) \neq 0$ ist. Damit in diesem Fall $s_k \in D(\gamma_k, x_k)$ liegt muß

$$\gamma_k \in \left]0, \frac{1}{\kappa_2(Df(x_k))}\right]$$

gelten wie die folgende Rechnung zeigt.

Setzen wir $x = x_k$ und $s = s_k$, so erhalten wir wegen $h(x) = \|f(x)\|_2^2$

$$\nabla h(x) = 2 f(x)^\top Df(x)$$

und damit die beiden Abschätzungen

$$\|f(x)^\top Df(x)\| \leq \|Df(x)\| \, \|f(x)\|,$$
$$\|Df(x)^{-1} f(x)\| \leq \|Df(x)^{-1}\| \, \|f(x)\|.$$

Daraus leitet man sofort die Beziehung

$$\frac{\nabla h(x)s}{\|\nabla h(x)\|} = \frac{f(x)^\top Df(x)Df(x)^{-1}f(x)}{\|Df(x)^{-1}f(x)\|\,\|f(x)^\top Df(x)\|} \geq \frac{1}{\kappa_2(Df(x))} > 0$$

her. Es gilt also für alle γ_k mit $0 < \gamma_k \leq 1/\kappa_2(Df(x_k))$, daß $s_k \in D(\gamma_k, x_k)$ liegt. Wenn also $Df(x)$ regulär ist, ist x genau dann stationärer Punkt von h (d.h. $\nabla h(x) = 0$), wenn x Nullstelle von f ist.

Aus diesen Ergebnissen können wir nun einen neuen Algorithmus konstruieren, ein *modifiziertes Newton–Verfahren*.

---

modifiziertes Newton–Verfahren

Wähle einen Startwert x_0 und eine Toleranz ε
Bestimme maximale Iterationsanzahlen K und J
$k = 0$
while $f(x_k) \geq \varepsilon$ **and** $k < K$ **do**
  if $f(x_k) = 0$ **then**
    stop
  endif
  Berechne $Df(x_k)$
  Löse $Df(x_k)d_k = f(x_k)$
  Berechne $\gamma_k = 1/\kappa_2(Df(x_k))$
  $h_k(\tau) = \|f(x_k - \tau d_k)\|_2^2$
  $z_k = \frac{1}{4}\gamma_k \|d_k\|_2 \|\nabla h(x_k)\|_2$
  $j = 0$
  while $h_k(2^{-j}) > h_k(0) - 2^{-j} z_k$ **do**
    j=j+1
  done
  Bestimme λ_k, sodaß $h(x_k - \lambda_k d_k) = \min\limits_{0 \leq i \leq j} h_k(2^{-i})$
  $x_{k+1} = x_k - \lambda_k d_k$
done

Wegen Proposition 5.2.2.4 können wir folgendes Resultat über die Konvergenz des modifizierten Newton–Verfahrens beweisen.

Theorem 5.2.2.5. *Wir betrachten eine Funktion $f : \mathbb{R}^n \to \mathbb{R}^n$ und einen Punkt x_0 und setzen wieder $h(x) = \|f(x)\|_2^2$. Gelten die Voraussetzungen*

(a) Die Menge $K := \{x | h(x) \leq h(x_0)\}$ ist kompakt;

(b) f ist auf einer Umgebung von K stetig differenzierbar;

(c) Für alle $x \in K$ existiert $Df(x)^{-1}$,

so ist die durch Algorithmus 5.2.2 gebildete Folge $(x_k)_k$ wohldefiniert und sie erfüllt

1. *$x_k \in K$ für alle $k \in \mathbb{N}$, und $(x_k)_k$ besitzt mindestens einen Häufungspunkt $\bar{x} \in K$.*
2. *Jeder Häufungspunkt $\bar{x}$ von $(x_k)_k$ ist Nullstelle von f.*

Beweis. Algorithmus 5.2.2 definiert eine monoton fallende Folge $h(x_k)$, daher gilt $x_k \in K$. O.B.d.A sei $f(x_k) \neq 0$ für alle k, da sonst der Algorithmus an einer Nullstelle terminiert. Wegen Voraussetzung (c) sind d_k und γ_k wohldefiniert, wenn x_k definiert ist, und es gilt nach Konstruktion $s_k \in D(\gamma_k, x_k)$. Die Existenz von

j und damit die Wohldefiniertheit von x_{k+1} folgt aus Proposition 5.2.2.4. Auch der Rest der Aussagen dieses Satzes folgt aus Proposition 5.2.2.4, falls wir noch zeigen können, daß

$$\inf_k \gamma_k > 0, \quad \inf_k \sigma_k > 0.$$

Weil $Df(x)^{-1}$ wegen Voraussetzungen (b) und (c) auf der kompakten Menge K stetig ist, ist auch $\kappa_2(Df(x))$ stetig. Daher existiert

$$\gamma := \frac{1}{\max_{x\in K} \kappa_2(Df(x))},$$

und wegen der Voraussetzung $f(x_k) \neq 0$ für alle k folgt

$$\inf_k \gamma_k \geq \gamma > 0.$$

Für die Abschätzung der σ_k benutzen wir

$$\|d_k\| = \|Df(x_k)^{-1} f(x_k)\| \geq \frac{1}{\|Df(x_k)\|} \|f(x_k)\|$$
$$\|\nabla h(x_k)\| \leq 2\,\|Df(x_k)\|\,\|f(x_k)\|,$$

woraus wir, wegen der Beschränktheit nach oben von $\|Df(x)\|$ auf K, sofort

$$\sigma_k \geq \frac{1}{2\,\|Df(x_k)\|^2} \geq \sigma > 0$$

erhalten. Daher folgen alle Aussagen aus Proposition 5.2.2.4. □

Der Nachteil des modifizierten Newton–Verfahrens ist der Aufwand. In jedem Schritt muß wie im allgemeinen Newton–Verfahren die Funktionalmatrix $Df(x_k)$ berechnet werden, gefolgt von der Lösung eines linearen Gleichungssystems mit der berechneten Matrix. Zusätzlich muß auch noch die Konditionszahl $\kappa_2(Df(x_k))$ bestimmt werden. Aus den Beweisen kann man jedoch ableiten, daß anstelle von $\gamma_k = 1/\kappa_2(Df(x_k))$ auch eine kleinere untere Schranke $0 < \gamma \leq \gamma_k$ verwendet werden kann. In praktischen Anwendungen wird daher meist in der inneren **while** –Schleife die Abbruchbedingung $h_k(2^{-j}) > h_k(0) - 2^{-j} z_k$ durch die vereinfachte Bedingung $h_k(2^{-j}) \geq h_k(0)$ ersetzt. Das erspart einem zwar die Berechnung von $\kappa_2(Df(x_k))$, doch kann diese Vereinfachung in manchen Fällen die Konvergenz des Verfahrens zerstören. Dieses vereinfachte Verfahren (wie auch manchmal das nicht vereinfachte) wird mitunter auch *gedämpftes Newton–Verfahren* genannt.

Ein wesentlicher Vorteil des Verfahrens liegt darin begründet, daß es in einer genügend kleinen Umgebung der Nullstelle automatisch $\lambda_k = 1$ wählt und damit zum gewöhnlichen Newton–Verfahren „mutiert“ und daher **lokal quadratisch** konvergiert.

Theorem 5.2.2.6. *Mit den Voraussetzungen von Theorem 5.2.2.5 existiert eine Umgebung $U(\bar{x})$ von $\bar{x}$, in der das modifizierte Newton–Verfahren (Algorithmus 5.2.2) quadratisch konvergiert.*

Beweis. Aus $\lim_{k\to\infty} x_k = \bar{x}$ und $f(\bar{x}) = 0$ folgt nämlich die Existenz einer Umgebung $V(\bar{x})$, mit der Eigenschaft, daß für alle Folgenglieder von $(z_k)_k$, der Folge die vom allgemeinen Newton–Verfahren erzeugt wird, die Abschätzungen

$$\|z_{k+1} - \bar{x}\|_2 \leq a\|z_k - \bar{x}\|_2^2$$
$$32a^2 \kappa_2(Df(\bar{x}))^2 \|z_k - \bar{x}\|_2^2 \leq 1$$

gelten. Beachtet man noch

$$\frac{\|x-\bar{x}\|_2}{\|(Df(\bar{x}))^{-1}\|_2} \leq \|Df(\bar{x})(x-\bar{x})\|_2 \leq \|Df(\bar{x})\|_2\,\|x-\bar{x}\|_2$$
$$f(x) = 0 + Df(\bar{x})(x-\bar{x}) + o(\|x-\bar{x}\|_2^2),$$

die untere Beziehung folgt dabei aus dem Satz von Taylor, so sieht man, daß es eine weitere Umgebung $W(\bar{x})$ gibt mit

$$\tfrac{1}{4}\|(Df(\bar{x}))^{-1}\|_2^{-2}\|x-\bar{x}\|_2^2 \le h(x) \le 4\|Df(\bar{x})\|_2^2\,\|x-\bar{x}\|_2^2$$

für alle $x \in W$.

Wählt man nun eine Umgebung $U(\bar{x}) \subseteq V(\bar{x}) \cap W(\bar{x})$ und ein k_0 mit $x_k \in U(\bar{x})$ für alle $k \ge k_0$, so erhält man folgende Ungleichung:

$$h(x_{k+1}) \le 4\|Df(\bar{x})\|_2^2\,\|x_{k+1}-\bar{x}\|_2^2 \le 16a^2\kappa_2(Df(\bar{x}))^2\|x_k-\bar{x}\|_2^2 h(x_k) \le (1-\tfrac{1}{2})h(x_k) = \tfrac{1}{2}h(x_k).$$

Aus der Definition des gedämpften Newton–Verfahrens erhält man die Ungleichung

$$\gamma_k\|d_k\|_2\,\|Df(x_k)\|_2 \le 2\gamma_k\|(Df(x_k))^{-1}\|_2\,\|Df(x_k)\|_2 h(x_k) = 2h(x_k)$$

und damit die Abschätzung

$$h(x_{k+1}) \le \tfrac{1}{2}h(x_k) \le h(x_k) - \tfrac{\gamma_k}{4}\|d_k\|_2\,\|Df(x_k)\|_2.$$

Aus diesem Grund wird für alle $k \ge k_0$ im Algorithmus $j = 0$ und daher $\lambda_k = 1$ gewählt. Erreicht die Folge also die Umgebung $U(\bar{x})$, so ist spätestens von diesem Zeitpunkt an das gedämpfte Newton–Verfahren mit dem allgemeinen Newton–Verfahren identisch und konvergiert daher lokal quadratisch. □

Quasi–Newton–Verfahren

Zur praktischen Durchführung eines gedämpften Newton–Verfahrens benötigt man leider noch weitere Untersuchungen. Zum ersten ist der Aufwand sehr hoch, weil in jedem Iterationsschritt die Funktionalmatrix $Df(x_k)$ berechnet und das lineare Gleichungssystem $Df(x_k)d = f(x_k)$ gelöst werden muß. Zum zweiten steht für hinreichend komplizierte Funktionen f die Funktionalmatrix überhaupt nicht mehr zur Verfügung. Das passiert zum Beispiel dann, wenn f Lösung einer Differentialgleichung ist und das Lösungsverfahren nur Funktionswerte $f(x)$ liefert, obwohl aus der Theorie mitunter bekannt ist, daß f stetig differenzierbar ist.

Im Prinzip gibt es zwei Möglichkeiten, diese Probleme zu umgehen. Die erste ist, auf eine Schätzung der Funktionalmatrix zurückzugreifen. Dies führt direkt zum Rang–1 Verfahren von Broyden und zu darauf aufbauenden Rang–2 Verfahren, sowie zu anderen quasi–Newton–Verfahren. Die andere Methode führt zum Newtonschen Einzelschrittverfahren und der Methode der Überrelaxation, die in Abschnitt 5.2.3 vorgestellt wird.

Die einfachste Variante zur Berechnung eines Schätzwertes Δf von Df ist, alle Differentialquotienten $\partial f_i/\partial x_j$ durch entsprechende Differenzenquotienten anzunähern.

$$\Delta_k f_i(x) := \frac{f_i(x_1,\ldots,x_{k-1},x_k+h_k,x_{k+1},\ldots,x_n) - f_i(x_1,\ldots,x_{k-1},x_k,x_{k+1},\ldots,x_n)}{h_k}.$$

Man kann die gesamte Matrix $\Delta f := (\Delta_k)$ auf diese Weise mit zwei zusätzlichen Vektoroperationen und n weiteren Funktionsauswertungen bestimmen:

$$\begin{aligned}\Delta f &:= (\Delta_1 f,\ldots,\Delta_n f)\\ \Delta_k f(x) &:= \frac{f(x_1,\ldots,x_{k-1},x_k+h_k,x_{k+1},\ldots,x_n) - f(x_1,\ldots,x_{k-1},x_k,x_{k+1},\ldots,x_n)}{h_k} =\\ &= \frac{f(x+h_k e_k)-f(x)}{h_k}.\end{aligned}$$

(Alternativ kann man als numerisch stabilere Variante auch den zentralen Differenzenquotienten verwenden

$$\Delta_k f(x) = \frac{f(x+h_k e_k) - f(x-h_k e_k)}{2h_k},$$

wobei diese Berechnungsmethode allerdings $2n$ zusätzliche Funktionsauswertungen benötigt.)

Die richtige Wahl der h_k ist wesentlich, denn sind sie zu groß, dann ist Δf eine schlechte Approximation von Df, sind sie andererseits zu klein, dann tritt bei der Berechnung von $f(x+h_k e_k) - f(x)$ Auslöschung auf, was wiederum die Güte der Näherung beeinträchtigt. Üblicherweise wählt man die h_k so, daß

$$|h_k| \, \|\Delta_k f(x)\| \approx \sqrt{\widetilde{\mathsf{eps}}} \|f(x)\|$$

gilt. $\widetilde{\mathsf{eps}}$ ist dabei die Genauigkeit, mit der man annimmt, $f(x)$ berechnen zu können (oft gilt $\widetilde{\mathsf{eps}} \approx \mathsf{eps}$). Wählt man h_k auf diese Weise, so stimmen bei Rechnung auf s Stellen $f(x)$ und $f(x+h_k e_k)$ in etwa $s/2$ Stellen überein. Bei der Berechnung der Differenz werden also $s/2$ Stellen ausgelöscht. Die bleibenden $s/2$ Stellen Rechengenauigkeit werden als akzeptabel angesehen.

Unglücklicherweise bedeuten aber für viele Funktionen selbst die n zusätzlichen Auswertungen in jedem Iterationsschritt des gedämpften Newton–Verfahrens zu viel an Aufwand. Das folgende algebraische Resultat weist einen Weg, eine Näherung $\Delta f(x_k)$ in weniger Rechenschritten zu bestimmen.

Proposition 5.2.2.7. *Seien $A, B \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$; die affine Abbildung $F : \mathbb{R}^n \to \mathbb{R}^n$ sei gegeben als $F(u) := Au + b$. Für zwei Vektoren x und y seien p und q definiert als*

$$p := x - y, \quad q := F(x) - F(y) = Ap.$$

Dann gelten für die Matrix $\tilde{B}$

$$\tilde{B} := B + \tfrac{1}{\langle p,p\rangle}(q - Bp)p^\top$$

die Abschätzung

$$\|\tilde{B} - A\|_2 \le \|B - A\|_2$$

und die Gleichung

$$\tilde{B}p = Ap = q.$$

Beweis. Die Gleichung $\tilde{B}p = Ap$ folgt direkt aus der Definition:

$$\tilde{B}p = Bp + \tfrac{1}{\langle p,p\rangle}(Ap - Bp)p^\top p = Bp + Ap - Bp = Ap.$$

Sei u mit $\|u\|_2 = 1$ beliebig. Dann können wir u zerlegen als $u = \alpha p + v$ mit $\alpha = \langle u, p\rangle / \langle p, p\rangle$. Dann steht v orthogonal auf p und nach dem Satz von Pythagoras gilt $\|v\|_2 \le 1$. Damit gilt

$$\begin{aligned}\|(\tilde{B} - A)u\|_2 = \|(\tilde{B} - A)v\|_2 = \|(B - A)v\|_2 \le \\ \le \|B - A\|_2 \, \|v\|_2 \le \|B - A\|_2,\end{aligned}$$

woraus

$$\|\tilde{B} - A\|_2 = \sup_{\|u\|_2 = 1} \|(\tilde{B} - A)u\|_2 \le \|B - A\|_2.$$

□

Aus Proposition 5.2.2.7 kann man ablesen, daß die (konstante) Funktionalmatrix einer affinen Funktion F durch $\tilde{B}$ wenigstens ebenso gut approximiert wird wie durch die Matrix B. Außerdem stimmen $\tilde{B}$ und DF darin überein, daß sie den Vektor p in denselben Vektor q abbilden. Wegen des Satzes von Taylor kann man innerhalb eines kleinen Bereiches, insbesondere in einer kleinen Umgebung einer Nullstelle $\bar{x}$, jede nichtlineare Funktion $f : \mathbb{R}^n \to \mathbb{R}$ durch eine affine Funktion approximieren:

$$f(x) \approx f(x') + Df(x')(x - x').$$

Diese Überlegungen folgend konstruiert man den folgenden Algorithmus, das *Rang–1–Verfahren von Broyden*.

Der Grund für die neuerliche Berechnung von B_k im Falle $\lambda_k < \frac{1}{2}$ ist sinnvoll, um zu große Auslöschungseffekte zu vermeiden.

Rang–1–Verfahren von Broyden

Wähle einen Startwert x_0 und eine Toleranz ε
Bestimme $B_0 = \Delta f(x_0)$
$k = 0$
while $f(x_k) \geq \varepsilon$ **do**
    Löse $B_k d_k = f(x_k)$
    $h_k(\tau) = \|f(x_k - \tau d_k)\|_2^2$
    $j = 0$
    while $h_k(2^{-j}) > h_k(0)$ **do**
        j=j+1
    done
    Bestimme λ_k, sodaß $h(x_k - \lambda_k d_k) = \min\limits_{0 \leq i \leq j} h_k(2^{-i})$
    $x_{k+1} = x_k - \lambda_k d_k$
    if $\lambda_k < \frac{1}{2}$ **then**
      $B_{k+1} = \Delta f(x_k)$
    else
      $p_k = x_{k+1} - x_k$
      $q_k = f(x_{k+1}) - f(x_k)$
      $B_{k+1} = B_k + 1/\langle p_k, p_k \rangle (q_k - B_k p_k) p_k^\top$
    endif
done

Theorem 5.2.2.8 (Broyden, Dennis, Moré (1973))**.** *Existiert für eine Nullstelle $\bar{x}$ eine Umgebung U mit den Eigenschaften*

- *$Df(x)$ exisitiert für $x \in U$ und ist stetig,*
- *$\|Df(x) - Df(\bar{x})\|_2 \leq \Lambda \|x - \bar{x}\|_2$ für $x \in U$,*
- *$Df(\bar{x})^{-1}$ existiert,*

und wählt das Verfahren $\lambda_k = 1$ wenn $x_k \in U$, so exisitiert k_0, sodaß alle B_k regulär sind für $k \geq k_0$, und wenn $\|x_{k_0} - \bar{x}\|_2$ und $\|B_{k_0} - Df(\bar{x})\|_2$ klein genug sind, so gilt für $k \geq k_0$

$$\lim_{k \to \infty} \frac{\|x_{k+1} - \bar{x}\|_2}{\|x_k - \bar{x}\|_2} = 0,$$

wenn $x_k \neq \bar{x}$ für alle k. Das heißt, das Verfahren konvergiert superlinear *gegen $\bar{x}$.*

Beweis. Siehe [Broyden et al. 1970]. □

Algorithmus 5.2.2 besitzt zwei große Vorteile. Er benötigt erstens keine Ableitungen. Die Jacobi–Matrix wird durch Bildung der Differenzenquotienten geschätzt. Außerdem benötigt die Lösung des linearen Gleichungssystemes $B_k d_k = f(x_k)$ bei vorliegender *LR*–Zerlegung lediglich $O(n^2)$ elementare Rechenoperationen. Die Bestimmung der *LR*–Zerlegung $L_{k+1} R_{k+1} = P_{k+1} B_{k+1}$ aus derjenigen für B_k ist ebenfalls mit Aufwand $O(n^2)$ möglich, da sich B_{k+1} von B_k nur durch eine Rang–1 Matrix unterscheidet. Das Verfahren, mit dessen Hilfe man die Zerlegung bestimmen kann, nennt man auch Rang–1 Update. Der Aufwand für jeden Iterationsschritt mit $\lambda_k \geq \frac{1}{2}$ beträgt also nur $O(n^2)$, im Gegensatz zu Algorithmus 5.2.2.

Minimierungsprobleme ohne Nebenbedingungen

Für reine Minimierungsprobleme kann man den Algorithmus ähnlich konstruieren. Man kann ihn sogar etwas beschleunigen, indem man beachtet, daß die Iterationsvorschrift auf der Beziehung

$$x_{k+1} = x_k - \lambda_k \nabla^{-2} h(x_k) \nabla h(x_k)$$

beruht; hierbei ist $h : \mathbb{R}^n \to \mathbb{R}$ die zu minimierende Funktion. Die Matrix $\nabla^2 h(x_k)$ der zweiten Ableitungen, die *Hesse–Matrix*, ist symmetrisch und an einem nicht-degenerierten Minimum positiv definit. Es liegt daher nahe, zur Approximation von $\nabla^2 h(x_k)$ positiv definite Matrizen H_k zu verwenden. In diesem Fall ist darauf zu achten, daß die Transformation $H_k \to H_{k+1}$ Symmetrie und positive Definitheit erhält. Im allgemeinen verwendet man ein Rang–2 Update

$$\begin{aligned} H_{k+\frac{1}{2}} &:= H_k + \alpha_k u_k u_k^\top, \quad \alpha_k > 0 \\ H_{k+1} &:= H_{k+\frac{1}{2}} - \beta_k v_k v_k^\top, \quad \beta_k > 0 \end{aligned}$$

mit geeigneten Konstanten α_k, β_k und Vektoren u_k und v_k. Zur Lösung der linearen Gleichungssysteme $H_k d_k = \nabla h(x_k)$ verwendet man wegen der positiven Definitheit am besten Cholesky–Faktorisierungen. Wie beim Rang–1 Verfahren von Broyden kann man aus der Zerlegung von H_k mit Aufwand $O(n^2)$ die Faktorisierung von H_{k+1} bestimmen. Ein solches *Rang–2 Verfahren* wird unter anderem in [Gill et al. 1974] beschrieben. Die im folgenden vorgestellte Klasse von Algorithmen basiert auf der Arbeit [Oren, Luenberger 1974], die Darstellung hält sich an [Stoer 1994a, 5.11].

Betrachten wir das Minimierungsproblem

$$\min_{x \in \mathbb{R}^n} h(x)$$

für $h \in C^2(\mathbb{R}^n)$. Abkürzend führen wir

$$g(x) := \nabla h(x)$$

für den Gradienten und

$$H(x) := \left(\frac{\partial^2 h}{\partial x^i \partial x^j} \right), \quad i, j = 1, \ldots, n,$$

für die Hesse–Matrix ein.

Analog zu den Nullstellenverfahren verwendet man Iterationsverfahren, entsprechend dem Schema

$$x_{k+1} = x_k - \lambda_k s_k,$$

wobei λ_k mittels Armijo–Liniensuche bestimmt wird. Dabei wird die Suchrichtung s_k als Abstiegsrichtung gewählt. Setzen wir $\varphi_k(\lambda) = h(x_k - \lambda s_k)$, so verlangen wir

$$\varphi_k'(0) = -\langle g_k, s_k \rangle < 0,$$

mit $g_k = g(x_k)$.

Nachdem jedes lokale Minimum von $h(x)$ Nullstelle von $g(x)$ sein muß, kann man jedes Nullstellenverfahren zur Lösung des Minimierungsproblems verwenden. Im (gedämpften) Newton–Verfahren, dem meistverwendeten Nullstellenalgorithmus, wählt man als Suchrichtung

$$s_k = H(x_k)^{-1} g_k.$$

Um in jedem Iterationsschritt die sehr aufwendige Berechnung der Hesse–Matrix $H(x)$ zu vermeiden, muß diese geschätzt und durch eine leichter zu bestimmende Matrix H_k^{-1} approximiert werden. Bei diesen *Quasi–Newton–Verfahren* wählt man

$$s_k = H_k g_k.$$

Genauer spricht man nur dann von einem Quasi–Newton–Verfahren, wenn H_k die *Quasi–Newton–Gleichung*

$$g_{k+1} - g_k = H_{k+1}^{-1}(x_{k+1} - x_k)$$

erfüllt. Diese Bedingung folgt aus der Beziehung

$$g_{k+1} - g_k = H(x_{k+1})(x_{k+1} - x_k) + O(\|x_{k+1} - x_k\|^2)$$

für die Hesse–Matrix von h. Wie schon weiter oben angedeutet, ist als Zusatzforderung sinnvoll, für H_k positive Definitheit zu verlangen. Das impliziert auch, daß s_k automatisch eine Abstiegsrichtung ist:

$$-\langle g_k, s_k\rangle = -g_k^\top H_k g_k < 0.$$

In [Oren, Luenberger 1974] wurde ein Rang–2 Update von H_k definiert, das allen Anforderungen genügt. Mit den Vektoren

$$p_k := x_{k+1} - x_k, \quad q_k := g_{k+1} - g_k$$

und frei wählbaren Parametern

$$\gamma_k > 0, \quad \theta_k \geq 0$$

definiert man

$$H_{k+1} := \Psi(\gamma_k, \theta_k, H_k, p_k, q_k)$$

mit der Iterationsfunktion

$$\Psi(\gamma,\theta,H,p,q) := \gamma H + \left(1 + \gamma\theta\frac{q^\top Hq}{p^\top q}\right)\frac{pp^\top}{p^\top q} - \gamma\frac{1-\theta}{q^\top Hq}Hq\cdot q^\top H - \\ - \frac{\gamma\theta}{p^\top q}(pq^\top H + Hqp^\top),$$

für $p^\top q \neq 0$ und $q^\top Hq \neq 0$. Die Matrizen H_{k+1} und H_k unterscheiden sich voneinander durch eine Matrix, die maximal Rang 2 hat.

Diese Untersuchungen führen zu einer Klasse von Minimierungsverfahren, die man wegen der Erfinder auch Oren–Luenberger–Klasse nennt.

Für verschiedene Wahlen der Parameter γ_k und θ_k erhält man folgende Spezialfälle. Das BFGS–Verfahren ist eines der meist verwendeten.

DFP–Verfahren Das Verfahren von Davidson, Fletscher und Powell erhält man für $\gamma_k \equiv 1$ und $\theta \equiv 0$.

BFGS–Verfahren Dieses oft verwendete Verfahren stammt von Broyden, Fletcher, Goldfab und Shanno. Man setzt hier $\gamma_k \equiv 1$ und $\theta \equiv 1$.

Symmetrisches Rang–1 Verfahren Dieses ist eine Verallgemeinerung des Rang–1 Verfahrens von Broyden. Man setzt $\gamma_k \equiv 1$ und $\theta_k = p_k^\top q_k/(p_k^\top q_k - p_k^\top H_k q_k)$. Hier kann allerdings passieren, daß $\theta_k < 0$ wird. Dann verliert H_{k+1} die Eigenschaft, positiv definit zu sein.

Diese Verfahren sind vorwiegend durch praktische Überlegungen begründet. Mathematische Gründe führen zu etwas anderen Wahlen, doch die Resultate sind nicht signifikant besser.

1. Will man den Quotienten $\kappa_2(H_{k+1})/\kappa_2(H_k)$ minimieren, so führt das zum folgenden Rezept: Mit

   $$\alpha := p_k^\top H_k^{-1} p_k, \quad \beta := p_k^\top q_k, \quad \sigma := q_k^\top H_k q_k$$

   wählt man

   $$(\gamma_k, \theta_k) = \begin{cases} \left(\frac{\alpha}{\beta}, 0\right) & \text{falls } \frac{\alpha}{\beta} \leq 1, \\ \left(\frac{\beta}{\sigma}, 1\right) & \text{falls } \frac{\beta}{\sigma} \geq 1, \\ \left(1, \frac{\beta(\alpha-\beta)}{\alpha\sigma-\beta^2}\right) & \text{falls } \frac{\beta}{\sigma} \leq 1 \leq \frac{\alpha}{\beta}. \end{cases}$$

Minimierung nach Oren und Luenberger

Wähle einen Startwert x_0 und eine Toleranz ε
Setze $H_0 = \mathbb{I}$ oder wähle eine andere positiv definite Matrix
$k = 0$
while $g(x_k) \geq \varepsilon$ **do**
    Bestimme $s_k = H_k g(x_k)$
    $h_k(\tau) = h(x_k - \tau s_k)$
    $j = 0$
    while $h_k(2^{-j}) > h_k(0)$ **do**
        j=j+1
    done
    Bestimme λ_k, sodaß $h(x_k - \lambda_k s_k) = \min\limits_{0 \leq i \leq j} h_k(2^{-i})$
    $x_{k+1} = x_k - \lambda_k s_k$
    Wähle $\gamma_k > 0$ und $\theta_k \geq 0$
    $p_k = x_{k+1} - x_k$
    $q_k = g(x_{k+1}) - g(x_k)$
    $H_{k+1} = \Psi(\gamma_k, \theta_k, H_k, p_k, q_k)$
done

2. Davidson hat gezeigt, daß für $\gamma_k \equiv 1$ gerade die Wahl

$$\theta_k = \begin{cases} \frac{\beta(\alpha-\beta)}{\alpha\sigma-\beta^2} & \text{falls } \beta \leq \frac{2\alpha\sigma}{\alpha+\sigma}, \\ \frac{\beta}{\beta-\sigma} & \text{sonst} \end{cases}$$

den Quotienten $\lambda_{\max}/\lambda_{\min}$ des größten und kleinsten Eigenwertes des nachstehenden allgemeinen Eigenwertproblems minimiert: Bestimme $\lambda \in \mathbb{C}$ und $y \neq 0$ mit $H_{k+1}y = \lambda H_k y$ und $\det(H_k^{-1}H_{k+1} - \lambda\mathbb{I}) = 0$.

Theorem 5.2.2.9. *Ein Minimierungsverfahren der Oren–Luenberger–Klasse hat folgende Eigenschaften:*

1. *Falls für ein $k \geq 0$ sowohl H_k positiv definit ist als auch $g_k \neq 0$ gilt, so ist die Matrix $H_{k+1} = \Psi(\gamma_k, \theta_k, H_k, p_k, q_k)$ wohldefiniert und wieder positiv definit. Ferner erfüllt H_{k+1} die Quasi–Newton–Gleichung.*

2. *Für eine quadratische Funktion $h(x) = \frac{1}{2}x^\top Ax + b^\top x + c$ mit positiv definiter Matrix A liefert das Verfahren in höchstens n Schritten das Minimum, soferne anstelle von Armijo–Liniensuche exakte Liniensuche verwendet wird.*

3. *Ist $H(\bar{x})$ am Minimum positiv definit, und ist $H(x)$ an $\bar{x}$ Lipschitz–stetig, so konvergiert die Methode lokal superlinear (in Spezialfällen kann sogar quadratische Konvergenz bewiesen werden).*

Beweis. Der Beweis kann in [Stoer 1994a, 5.11] und den dort zitierten Arbeiten nachgelesen werden. □

Beispiel 5.2.2.10. *Das Beispiel stammt aus [Stoer 1994a, 5.11] und bringt einen Vergleich zwischen dem DFP–Verfahren, dem BFGS–Verfahren und dem Gradienten–Verfahren ($s_k := g(x_k)$). Die zu minimierende Funktion sei*

$$h(x,y) := 100\big(y^2(3-x) - x^2(3+x)\big)^2 + \frac{(2+x)^2}{1+(2+x)^2}.$$

Das Minimum liegt bei

$$(\bar{x},\bar{y}) = (-2, 0.8942719099\ldots), \quad h(\bar{x},\bar{y}) = 0,$$

und als Startwerte für jedes der Verfahren werden

$$x_0 := 0.1, \quad y_0 := 4.2, \quad H_0 = \mathbb{I}$$

gewählt. Bei einer Rechengenauigkeit von $\varepsilon = 10^{-11}$ *erhält man die Ergebnisse*

	BFGS	*DFP*	*Gradientenverf.*
N	*54*	*47*	*201*
F	*374*	*568*	*1248*
ε	$\leq 10^{-11}$	$\leq 10^{-11}$	*0.7*

N bedeutet die Anzahl der Iterationsschritte und F die Anzahl der Funktionsauswertungen. $\varepsilon := \|g(x_N, y_N)\|_2$ *ist die erreichte Endgenauigkeit. Das BFGS–Verfahren ist die Methode mit der größten Effizienz, während das DFP–Verfahren nur unwesentlich schlechter ist. Beide schlagen jedenfalls das Gradientenverfahren um Längen.*

5.2.3 Methode der Überrelaxation — SOR–Newton–Verfahren

Für sehr große Probleme ist mitunter sogar die erste Berechnung von $\Delta f(x_0)$ zu viel, bzw. ist die Berechnung des Rang–1 Updates zu aufwendig. Manchmal ist auch das Abspeichern der Matrix B_k im Rang–1 Verfahren von Broyden das Problem.

In diesem Fall borgt man sich eine Idee aus der numerischen linearen Algebra. Möchte man das lineare Gleichungssystem $Ax = b$ lösen und erfüllt die Matrix $A_{ii} \neq 0$ (das ist keine wesentliche Einschränkung), so kann man die Gleichungen getrennt betrachten

$$A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n = b_i$$

und die i–te Gleichung nach x_i auflösen:

$$x_i = \tfrac{1}{A_{ii}}(b_i - A_{i1}x_1 - \cdots - A_{i,i-1}x_{i-1} - A_{i,i+1}x_{i+1} - \cdots - A_{in}x_n)$$

Das *lineare Einzelschrittverfahren*, ein linear konvergentes Lösungsverfahren für lineare Gleichungssysteme, berechnet ausgehend von einem Startvektor $x^{(0)}$ iterativ aus $x^{(k)}$ den Vektor $x^{(k+1)}$ gemäß der Iterationsvorschrift

$$x_i^{(k+1)} = \tfrac{1}{A_{ii}}(b_i - A_{i1}x_1^{(k+1)} - \cdots - A_{i,i-1}x_{i-1}^{(k+1)} - A_{i,i+1}x_{i+1}^{(k)} - \cdots - A_{in}x_n^{(k)}).$$

Man verwendet also in jedem Schritt jeweils die neueste Information. Der entstehende Algorithmus läßt sich folgendermaßen formulieren:

Lineares Einzelschrittverfahren

Wähle eine Toleranz ε
Wähle einen Startvektor $x^{(0)}$
$k = 0$
while $\|Ax^{(k)} - b\|_2 > \varepsilon$ **do**
  for $i = 1$ **to** n **do**
    $x_i^{(k+1)} = \tfrac{1}{A_{ii}}(b_i - A_{i1}x_1^{(k+1)} - \cdots - A_{i,i-1}x_{i-1}^{(k+1)} - A_{i,i+1}x_{i+1}^{(k)} - \cdots - A_{in}x_n^{(k)})$
  done
  $k = k + 1$
done

Wollen wir nun das nichtlineare Nullstellenproblem $f(x) = 0$ mit $f : \mathbb{R}^n \to \mathbb{R}^n$ lösen, so zerlegen wir das Problem ebenfalls in die einzelnen eindimensionalen Gleichungen

$$f_i(x_1, \ldots, x_n) = 0.$$

In Analogie zur Forderung $A_{ii} \neq 0$ von oben verlangen wir, daß

$$\frac{\partial f_i(x_1,\ldots,x_n)}{\partial x_i} \not\equiv 0$$

gilt. Das bedeutet eigentlich nur, daß f_i von x_i abhängt und kann durch Umsortieren der Gleichungen üblicherweise erreicht werden. Der einzige Unterschied zum linearen Einzelschrittverfahren besteht nun darin, daß die Gleichung nicht so leicht nach x_i aufgelöst werden kann. Aus der impliziten Gleichung wird x_i mit Hilfe eines eindimensionalen (gedämpften) Newton–Verfahrens bestimmt. Es entsteht der folgende Algorithmus:

Nichtlineares Einzelschrittverfahren

Wähle Toleranzen ε_1 und ε_2
Wähle einen Startvektor $x^{(0)}$
$k = 0$
$s = \infty$
while $s \geq \varepsilon_1$ **do**
  for $i = 1$ **to** n **do**
    $s = 0$
    $\xi = x_i^{(k)}$
    while $f_i(x_1^{(k+1)},\ldots,x_{i-1}^{(k+1)},\xi,x_{i+1}^{(k)},\ldots,x_n^{(k)}) \geq \varepsilon_2$ **do**

$$\Delta\xi = \frac{f_i(x_1^{(k+1)},\ldots,x_{i-1}^{(k+1)},\xi,x_{i+1}^{(k)},\ldots,x_n^{(k)})}{\frac{\partial f_i(x_1^{(k+1)},\ldots,x_{i-1}^{(k+1)},\xi,x_{i+1}^{(k)},\ldots,x_n^{(k)})}{\partial x_i}}$$

      $\xi = \xi - \Delta\xi$
    done
    $s = s + |\xi - x_i^{(k)}|$
    $x_i^{(k+1)} = \xi$
  done
  $k = k + 1$
done

Das nichtlineare Einzelschrittverfahren kann dadurch beschleunigt werden, daß das innere eindimensionale Newton–Verfahren nicht bis zur Konvergenz ausgeführt wird. Der in der inneren Iteration berechnete Wert ist ohnehin nur eine Näherung, und selbst wenn man die i–te Gleichung exakt löst, konvergiert das Verfahren nur linear. Daher führt man nur einen Schritt des quadratisch konvergenten Newton–Verfahrens aus, und erhält ein weiteres linear konvergentes Verfahren, das *Newtonsche Einzelschrittverfahren*.

Schließlich läßt sich die Konvergenz dieses Verfahrens noch deutlich verbessern, indem man die Korrektur der i–ten Komponente $\Delta\xi$ mit einem konstanten, geeignet gewählten *Relaxationsfaktor* $\omega \in]0,2[$ multipliziert. Das so entstehende SOR–Newton–Verfahren (SOR steht für successive overrelaxation) lautet somit

Besonders geeignet ist das SOR–Newton–Verfahren zur Lösung von sehr großen nichtlinearen Gleichungssystemen, bei denen die i–te Gleichung nur von sehr wenigen Variablen x_k abhängt. Im Gegensatz zum gedämpften Newton–Verfahren muß man nicht die gesamte Jacobi–Matrix $Df(x^{(k)})$ bestimmen sondern nur deren n Diagonalelemente. Den Relaxationsfaktor ω bestimmt man üblicherweise durch Raten oder Probieren. In wichtigen Spezialfällen, etwa für die Gleichungen, die bei der numerischen Behandlung von nichtlinearen partiellen Differentialgleichungen mit Differenzenmethoden oder der Methode der finiten Elemente (siehe Kapitel 9) auftreten, existieren Resultate über die optimale Wahl des Relaxationsparameters zur Erzielung schnellstmöglicher Konvergenz.

Newtonsches Einzelschrittverfahren

Wähle eine Toleranz ε
Wähle einen Startvektor $x^{(0)}$
$k = 0$
$s = \infty$
while $s \geq \varepsilon$ **do**
    for $i = 1$ **to** n **do**
        $s = 0$
        $\xi = x_i^{(k)}$
        $\Delta\xi = \dfrac{f_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, \xi, x_{i+1}^{(k)}, \ldots, x_n^{(k)})}{\frac{\partial f_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, \xi, x_{i+1}^{(k)}, \ldots, x_n^{(k)})}{\partial x_i}}$
        $\xi = \xi - \Delta\xi$
        $s = s + |\xi - x_i^{(k)}|$
        $x_i^{(k+1)} = \xi$
    done
    $k = k + 1$
done

SOR–Newton–Verfahren

Wähle eine Toleranz ε
Wähle einen Startvektor $x^{(0)}$
Wähle einen Relaxationsfaktor $\omega \in]0,2[$
$k = 0$
$s = \infty$
while $s \geq \varepsilon$ **do**
    for $i = 1$ **to** n **do**
        $s = 0$
        $\xi = x_i^{(k)}$
        $\Delta\xi = \omega \dfrac{f_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, \xi, x_{i+1}^{(k)}, \ldots, x_n^{(k)})}{\frac{\partial f_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, \xi, x_{i+1}^{(k)}, \ldots, x_n^{(k)})}{\partial x_i}}$
        $\xi = \xi - \Delta\xi$
        $s = s + |\xi - x_i^{(k)}|$
        $x_i^{(k+1)} = \xi$
    done
    $k = k + 1$
done

6 Interpolation II: Mehrdimensionaler Fall

Die Verallgemeinerung der Interpolation vom skalaren Fall ins Mehrdimensionale erfordert mehr als die bloße Einführung zusätzlicher Variablen. Die Interpolation durch mehrdimensionale Polynome ist nicht mehr so einfach möglich wie Polynominterpolation im Eindimensionalen.

Die Menge aller Polynome $\mathbb{K}[x_1,\ldots,x_n]$ in n Variablen ist ein unendlich dimensionaler Vektorraum. Der *Grad eines Monoms*

$$x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$$

definiert durch das n–Tupel $\alpha = (\alpha_1,\ldots,\alpha_n)$ nichtnegativer ganzer Zahlen ist definiert als

$$\deg x^\alpha := \sum_{i=1}^{n} \alpha_i.$$

Der *Grad eines Polynoms* in n Variablen

$$p(x) = \sum_{\alpha\in A} a_\alpha x^\alpha$$

ist das Maximum aller Monomgrade. Der Raum $\mathbb{K}^r[x_1,\ldots,x_n]$ aller multivariaten Polynome, die höchstens Grad r aufweisen, ist ein Teilraum von $\mathbb{K}[x_1,\ldots,x_n]$ mit Dimension

$$d_n^r := \dim \mathbb{K}^r[x_1,\ldots,x_n] = \binom{n+r}{n}.$$

In der folgenden Tabelle ist d_n^r für verschiedene Werte von r und n berechnet:

n	$r=1$	$r=2$	$r=3$	$r=5$	$r=10$	$r=20$
1	2	3	4	6	11	21
2	3	6	10	21	66	231
3	4	10	20	56	286	1 771
5	6	21	56	252	3 003	53 130
10	11	66	286	3 003	184 756	30 045 015
20	21	231	1 771	53 130	30 045 015	137 846 528 820

Man sieht unschwer, daß die Dimension bereits für kleine Werte von n und r sehr groß wird. Möchte man ein Interpolationsproblem in einem der Räume $\mathbb{K}^r[x_1,\ldots,x_n]$ lösen, so muß die Anzahl der Interpolationspunkte mit der Dimension des Raumes übereinstimmen, um eine eindeutige Lösung zu implizieren. Ist nun die Anzahl der Interpolationspunkte nicht mit einer der Zahlen d_n^r identisch, so muß der Raum $K^r[x_1,\ldots,x_n]$ für ein bestimmtes r weiter eingeschränkt werden; meist ist keine natürliche Wahl für eine Einschränkung abzusehen.

Andererseits, selbst wenn die Anzahl der Punkte mit d_n^r für ein r übereinstimmt, wie im folgenden Beispiel, können Probleme auftreten. Sei $n=2$, $r=1$, und seien die Punkte

$$y^{(0)} = (0,0), \quad y^{(1)} = (1,1), \quad y^{(2)} = (2,2)$$

gegeben. Die zulässigen Interpolationspolynome haben die Form

$$f(x) = a_0 + a_1 x_1 + a_2 x_2,$$

und wir wollen das Interpolationsproblem $f(y_i) = f_i$ für ebenfalls gegebene Werte f_0, f_1, f_2 lösen. Nachdem f eine affine Funktion ist, gilt

$$f(x) = f_0 + p(x)$$

für eine lineare Funktion p, die die beiden Gleichungen $p(i,i) = f_i - f_0$ für $i = 1,2$ erfüllen muß. Weil p aber linear ist, folgt

$$f_2 - f_0 = p(2,2) = 2p(1,1) = 2(f_1 - f_0)$$

und damit

$$f_2 - 2f_1 + f_0 = 0.$$

Erfüllen die Werte f_i diese Gleichung nicht, so hat das Interpolationsproblem keine Lösung. Stimmt die Gleichung, so interpolieren auch alle Polynome der Form

$$\tilde{f}(x) = f(x) + \lambda(x_1 - x_2)$$

mit beliebigem $\lambda \in \mathbb{K}$ die gegebenen Daten.

Wir können also die folgende Schlußfolgerung ziehen: *Im Mehrdimensionalen ist das Polynominterpolationsproblem weder immer lösbar noch ist die Lösung im Fall ihrer Existenz immer eindeutig.*

Aus diesem Grund ist mehrdimensionale Polynominterpolation nicht sehr bedeutend, jedenfalls im Vergleich mit dem skalaren Fall.

Im multivariaten Fall verwendet man aus diversen Gründen fast ausschließlich Spline–ähnliche Interpolationsverfahren oder radiale Basisfunktionen (RBF). Die Anwendungen mehrdimensionaler Interpolation reichen von graphischer Darstellung (Fonts — interpolierende Kurven, photorealistische Darstellung — interpolierende Flächen, CAD) bis zur Modellbildung komplizierter oder schwer zu berechnender Funktionen oder von Funktionen mit unbekanntem analytischen Ausdruck (z.B. Lösung einer Differentialgleichung) aus einigen Funktionswerten, etwa zum Zweck der Optimierung.

In den folgenden Abschnitten werden wir die Grundlagen der mehrdimensionalen Interpolation durch Splines zuerst im Fall von Kurven im $\mathbb{R}^n$ (Abschnitt 6.1), der einfachsten Verallgemeinerung der univariaten Interpolation, dann im Fall der Flächen im $\mathbb{R}^n$ (Abschnitt 6.2), ein Spezialfall ist dabei die Interpolation einer Funktion in zwei Variablen, und schließlich allgemein im Höherdimensionalen (Abschnitt 6.3) entwickeln. Ein Abschnitt (6.5) über radiale Basisfunktionen wird das Kapitel beenden.

Der letzte Abschnitt des Kapitels beschäftigt sich schließlich mit Triangulierungen, einer wesentlichen Vorarbeit nicht nur zur Berechnung mehrdimensionaler Interpolationsfunktionen sondern auch zur Lösung gewöhnlicher und partieller Differentialgleichungen, wie in den Kapiteln **??** und 9 besprochen.

Ein Großteil der hier dargestellten Fakten ist [Risler 1992] entnommen.

6.1 Interpolierende Kurven

Die einfachste Verallgemeinerung der univariaten Interpolation ist die Lösung des folgenden Problems: Es seien $r+1$ Punkte $P_0, \ldots, P_r$ im $\mathbb{R}^s$ gegeben und $r+1$ verschiedene reelle Zahlen $t_0, \ldots, t_r$. Wir suchen dann aus einer vorgegebenen Klasse von Kurven $\gamma \in C^k(\mathbb{R}, \mathbb{R}^s)$ jene, die das Interpolationsproblem

$$\gamma(t_i) = P_i$$

lösen.

Wir wollen hier die verbreitetste Methode zur Kurveninterpolation beleuchten, die der Spline–Kurven. Wie im skalaren Fall setzt man die Kurve γ aus einfach zu berechnenden Stücken zusammen, die an den Übergangspunkten bestimmte Stetigkeitsbedingungen erfüllen. Die in Kapitel **??** Abschnitt **??** hergeleiteten Grundlagen über B–Splines werden dabei eine große Rolle spielen.

6.1.1 Grundlagen

Bevor wir das Interpolationsproblem angehen können, müssen wir noch einige grundlegende mathematische Fakten aufarbeiten. Das beinhaltet zum einen einiges aus der Theorie der Kurven im $\mathbb{R}^s$ zum anderen die algebraisch wichtigsten Fakten über die Bernstein–Polynome.

Kurven im $\mathbb{R}^n$

Die Aussagen in diesem Abschnitt können in den meisten Analysis Bücher nachgelesen werden, etwa in [Heuser 1986/2, 161, XXI].

In vielen mathematischen Modellen (und anderswo) geht es darum, Teilmengen des $\mathbb{R}^n$ zu untersuchen, die sich durch Gleichungen der Form

$$x_1 = \gamma_1(t), \ldots, x_n = \gamma_n(t)$$

mit $t \in [a,b] \subseteq \mathbb{R}$ ($[a,b]$ kann dabei auch ein unbeschränktes Intervall sein) beschreiben lassen. Faßt man die Funktionen γ_i zu einer Funktion $\gamma : [a,b] \to \mathbb{R}^n$ zusammen, kann man die Gleichungen auch schreiben als

$$x = \gamma(t), \quad t \in [a,b].$$

In manchen Fällen benötigt man die Menge aller Punkte, die dieser Gleichung genügen, also die Menge

$$\Gamma := \{\gamma(t) \mid t \in [a,b]\}.$$

In anderen Fällen ist es interessant, nicht nur Γ zu beschreiben sondern auch die Parametrisierung γ der Menge Γ zu untersuchen, etwa wenn die Bahnkurve einer Rakete berechnet werden soll. Dann interpretiert man den Parameter t als die Flugzeit und den Punkt $\gamma(t)$ als die Position im Raum zum Zeitpunkt t.

Beispiel 6.1.1.1. *Die Abbildung* $\gamma : [0, 2\pi[\to \mathbb{R}^2$

$$\gamma(t) = \big(\cos t, \sin t\big)$$

beschreibt den Einheitskreis S^1 *im* $\mathbb{R}^2$ *(also* $\Gamma = S^1$*). Aber auch für die Abbildung*

$$\tilde{\gamma}(t) = \big(\cos(-2t), \sin(-2t)\big)$$

gilt $\tilde{\Gamma} = S^1$.

Ein wesentlicher Unterschied zwischen den beiden Darstellungen ist allerdings, daß im ersten Fall jeder Punkt von S^1 *nur einmal getroffen wird, im zweiten Fall wird jeder Punkt des Einheitskreises zweimal durchlaufen. Betrachtet man die Reihenfolge, in der die Punkte erreicht werden, so sieht man, daß der Durchlauf bei der Parametrisierung* γ *gegen den Uhrzeigersinn erfolgt, während er bei* $\tilde{\gamma}$ *mit dem Uhrzeigersinn geht.*

Am vorangegangenen Beispiel erkennt man, daß bei der Untersuchung von Kurven genau unterschieden werden muß zwischen der Menge Γ und der Parametrisierung γ. Viele der folgenden Definitionen und Resultate sind nicht an Teilmengen des $\mathbb{R}^n$ gebunden. Die weitaus meisten Aussagen lassen sich auf metrische Räume oder noch allgemeinere topologische Räume verallgemeinern.

Definition 6.1.1.2. *Sei* $[a,b] \subseteq \mathbb{R}$ *ein nichtleeres Intervall. Eine stetige Abbildung* $\gamma : [a,b] \to \mathbb{R}^n$ *heißt ein* Weg in $\mathbb{R}^n$. *Unter einem* Bogen in $\mathbb{R}^n$ *versteht man die zu einem Weg* γ *gehörende Punktemenge* $\Gamma := \{\gamma(t) \,|\, t \in [a,b]\}$. *Die Gleichung* $x = \gamma(t)$ *wird* Parameterdarstellung von Γ *genannt, t heißt dabei der Parameter.* $\gamma(a)$ *nennt man den* Anfangspunkt, $\gamma(b)$ *den* Endpunkt *des Weges* γ. *Man sagt auch, der Weg* γ *verbindet die Punkte* $\gamma(a)$ *und* $\gamma(b)$.

Eine wichtige Beobachtung ist, daß man bei der Beschreibung eines Bogens das *Parameterintervall* $[a,b]$ beliebig wählen kann. Wäre etwa $[c,d]$ ein anderes Intervall, so kann man $[c,d]$ mittels der C^∞–Funktion

$$\varphi(t) := \frac{ad - bc}{d - c} + \frac{b - a}{d - c} t$$

bijektiv auf $[a,b]$ abbilden. Der Weg $\gamma \circ \varphi : [c,d] \to \mathbb{R}^n$ hat dann denselben Bogen Γ wie der Weg $\gamma : [a,b] \to \mathbb{R}^n$.

Definition 6.1.1.3. *1. Dem Weg $\gamma : [a,b] \to \mathbb{R}^n$ ordnet man den* inversen Weg

$$\gamma^-(t) := \gamma(a+b-t), \quad t \in [a,b]$$

zu, dessen Bogen Γ^- mit Γ übereinstimmt. Der Unterschied ist, daß bei γ^- der Bogen in umgekehrter Richtung durchlaufen wird, Endpunkt und Anfangspunkt tauschen ihre Rollen.

2. *Sind zwei Wege $\gamma_i : [a_i,b_i] \to \mathbb{R}^n$, $i = 1,2$ gegeben, und haben wir $\gamma_1(b_1) = \gamma_2(a_2)$, so definieren wir den* zusammengesetzten Weg *oder die* Summe der Wege γ_i *als $\gamma := \gamma_1 \oplus \gamma_2 : [a_1, b_1 - a_2 + b_2] \to \mathbb{R}^n$ mit*

$$\gamma(t) = \begin{cases} \gamma_1(t) & t \in [a_1,b_1] \\ \gamma_2(t-b_1+a_2) & t \in [b_1, b_1-a_2+b_2]. \end{cases}$$

Man „hängt also die beiden Wege hintereinander". Der Bogen $\Gamma := \Gamma_1 \oplus \Gamma_2$ heißt die Summe der Bögen Γ_i.

Die Definition 6.1.1.2 ist für die Anschauung leider viel zu allgemein. Im Jahr 1890 überraschte Peano die mathematische Welt mit dem Beispiel eines Weges $[0,1] \to \mathbb{R}^2$, dessen Bogen $\Gamma = [0,1]^2$ das *gesamte Einheitsquadrat* ist. Solche flächenfüllende Bögen werden seitdem als *Peanobögen* oder *Peanokurven* bezeichnet.

Aus diesem Grund muß man die Begriffe Bogen und Weg etwas einschränken. Dazu benötigen wir unter anderem den Begriff der Weg– oder Bogenlänge.

Definition 6.1.1.4. *Ein Weg $\gamma : [a,b] \to \mathbb{R}^n$ heißt* rektifizierbar, *wenn es eine Konstante M gibt, so daß für alle Zerlegungen $Z := \{t_0, \ldots, t_k\}$ des Intervalls $[a,b]$ immer*

$$L(\gamma, Z) := \sum_{j=1}^{k} \|\gamma(t_j) - \gamma(t_{j-1})\|_2 \leq M$$

gilt. Die Zahl

$$L(\gamma) := \sup_{Z \in \mathcal{Z}} L(\gamma, Z),$$

wobei das Supremum über die Menge $\mathcal{Z}$ aller Zerlegungen von $[a,b]$ gebildet wird, heißt dann die (Weg–)Länge von γ.

Es gilt, daß ein Weg $\gamma : [a,b] \to \mathbb{R}^n$ genau dann rektifizierbar ist, falls jede Komponentenfunktion $\gamma_i : [a,b] \to \mathbb{R}$ von beschränkter Variation ist.

Sind γ_1, γ_2 zwei rektifizierbare Wege, so ist deren Summe $\gamma_1 \oplus \gamma_2$ ebenfalls rektifizierbar, und es gilt

$$L(\gamma_1 \oplus \gamma_2) = L(\gamma_1) + L(\gamma_2).$$

Die Weglänge verhält sich also additiv.

Definition 6.1.1.5. *Ist $\gamma : [a,b] \to \mathbb{R}^n$ ein rektifizierbarer Weg, und bezeichnen wir für jedes Teilintervall $[c,d] \subseteq [a,b]$ mit $\gamma_{[c,d]}$ den (ebenfalls rektifizierbaren) Teilweg von γ, der definiert ist durch $\gamma_{[c,d]} := \gamma|_{[c,d]}$, so können wir die Funktion s definieren durch*

$$s(t) := \begin{cases} 0 & \text{für } t = a, \\ L(\gamma_{[a,t]}) & \text{für } t \in]a,b]. \end{cases}$$

Die Funktion s wird auch als Weglängenfunktion (Bogenlängenfunktion) bezeichnet. Sie ist monoton wachsend und stetig $[a,b] \to \mathbb{R}$.

Definition 6.1.1.6. *Ist ein rektifizierbarer Weg $\gamma : [a,b] \to \mathbb{R}^n$ injektiv, so ist die Funktion s sogar streng monoton wachsend, und γ ist auf keinem Teilintervall $[c,d]$ von $[a,b]$ konstant. Ein Weg dieser Art wird als* Jordanweg *bezeichnet.*

Unglücklicherweise ist auch der Begriff des Jordanweges nicht speziell genug, um diverse unerwünschte Pathologien auszuschließen. Man kann zwar zeigen, daß keine Peanokurve ein Jordanweg ist, doch der Mathematiker Helge von Koch konnte 1906 einige hochexotische Jordanbögen konstruieren, die fraktale Eigenschaften aufweisen (unter anderem nur Parameterdarstellungen besitzen, die nirgends differenzierbar sind), etwa die berühmte Kochsche Schneeflocke. Für unsere speziellen Zwecke wird es ausreichen, Glattheitsvoraussetzungen zu machen.

Definition 6.1.1.7. *Ist $\gamma : [a,b] \to \mathbb{R}^n$ nicht nur stetig sondern sogar stetig differenzierbar (also C^1), so können wir an t_0 den Vektor $T_{t_0}\gamma := \dot{\gamma}(t_0) := \frac{d}{dt}\gamma(t_0) \in \mathbb{R}^n$ betrachten. Dieser Vektor heißt der* Tangentialvektor *von γ bei t_0, falls $T_{t_0}\gamma \neq 0$ gilt.*

Ist der Weg an allen Punkten bis auf endlich viele stetig differenzierbar, so heißt er stückweise stetig differenzierbar.

Proposition 6.1.1.8. *Ist der Weg $\gamma : [a,b] \to \mathbb{R}^n$ stetig differenzierbar, dann ist er auch rektifizierbar, und für alle $t \in [a,b]$ gilt*

$$s(t) = \int_a^t \|\dot{\gamma}(y)\|_2 \, dy.$$

Ist γ stückweise stetig differenzierbar, so ist er ebenfalls rektifizierbar, und seine Weglänge ist die Summe der Weglängen seiner stetig differenzierbaren Teile.

Ein stetig differenzierbarer Weg $\gamma : [a,b] \to \mathbb{R}^n$ heißt regulär, *falls für alle $t \in [a,b]$ gilt $T_t\gamma \neq 0$.*

Im folgenden wollen wir die Begriffe Bogen, Weg und Parameterdarstellung näher untersuchen.

Sei $\gamma : [a,b] \to \mathbb{R}^n$ ein Weg, Γ der dazu gehörige Bogen. Ist $\varphi : [c,d] \to [a,b]$ eine stetige bijektive Abbildung, so ist Γ auch der Bogen, der vom Weg $\gamma \circ \varphi : [c,d] \to \mathbb{R}^n$ definiert wird, und die Gleichung

$$x = \gamma \circ \varphi(u), \quad u \in [c,d]$$

ist eine andere Parameterdarstellung von Γ. Die Abbildung φ heißt eine *Parameterwechselabbildung* oder eine *Umparametrisierung von* Γ. Gelten weiters $\varphi(c) = a$ und $\varphi(d) = b$ und φ monoton, so heißt φ *orientierungserhaltend*.

Ist $\gamma : [a,b] \to \mathbb{R}^n$ ein regulärer Weg, so wollen wir nur reguläre Umparametrisierungen zulassen; das sind Parameterwechsel φ, die C^1 sind und $\varphi'(u) > 0$ für alle $u \in [c,d]$ erfüllen.

Definition 6.1.1.9. *Ein Bogen Γ heißt* Jordanbogen, *falls es einen Jordanweg $\gamma : [a,b] \to \mathbb{R}^n$ gibt, dessen Bogen mit Γ übereinstimmt. Wir sagen dann auch γ sei eine* Jordandarstellung *von Γ.*

Proposition 6.1.1.10.

1. *Seien $\gamma_1 : [a_1,b_1] \to \mathbb{R}^n$ und $\gamma_2 : [a_2,b_2] \to \mathbb{R}^n$ zwei Jordandarstellungen desselben Jordanbogens Γ. Dann gibt es eine orientierungserhaltende Umparametrisierung $\varphi : [a_2,b_2] \to [a_1,b_1]$ mit*

$$\gamma_2 = \gamma_1 \circ \varphi \quad \textit{oder} \quad \gamma_2 = \gamma_1^- \circ \varphi.$$

2. *Ist ψ eine stetige und streng monoton wachsende Abbildung, die $[a_1,b_1]$ bijektiv auf $[a_3,b_3]$ abbildet, so wird durch*

$$\gamma_3 = \gamma_1 \circ \psi$$

eine Jordandarstellung von Γ definiert.

3. *γ_1^- ist eine Jordandarstellung von Γ.*

Man erhält also alle Jordandarstellungen des Jordanbogens Γ aus einer beliebigen frei gewählten Jordandarstellung $\gamma : [a,b] \to \mathbb{R}^n$ durch orientierungserhaltende Umparametrisierungen ω:

$$\gamma \circ \omega \quad \textit{oder} \quad \gamma \circ \omega^-.$$

Legt man noch fest, welcher Punkt Anfangspunkt und welcher Punkt Endpunkt ist, so erhält man alle Jordandarstellungen mit dem Anfangspunkt $\gamma(a)$ und dem Endpunkt $\gamma(b)$ aus γ und orientierungserhaltenden Umparametrisierungen ω durch Zusammensetzung:

$$\gamma \circ \omega.$$

Proposition 6.1.1.11. *$\gamma_i : [a_i, b_i] \to \mathbb{R}^n$ ($i = 1, 2$) seien zwei Wege, und es existiere eine streng monotone Umparametrisierung $\omega : [a_2, b_2] \to [a_1, b_1]$ mit*

$$\gamma_2 = \gamma_1 \circ \omega.$$

Dann sind entweder beide γ_i rektifizierbar oder beide Wege sind nicht rektifizierbar, und im Fall gleichzeitiger Rektifizierbarkeit stimmen ihre Weglängen überein.

Im Lichte der vorausgegangenen Propositionen sehen wir, daß die Länge eines Weges eine Zahl ist, die invariant unter streng monotonen (also insbesondere unter orientierungserhaltenden) Parameterwechseln ist. Daher ist sie eigentlich eine Eigenschaft des Bogens.

Definition 6.1.1.12. *Ein Jordanbogen heißt* rektifizierbar*, falls er eine rektifizierbare Jordandarstellung γ besitzt. In diesem Fall sind alle Jordandarstellungen von Γ rektifizierbar, und alle besitzen dieselbe Weglänge $L(\gamma)$. Diese Zahl nennen wir dann die* Bogenlänge *von Γ, und wir bezeichnen sie mit $L(\Gamma)$.*

Definition 6.1.1.13. *Ein Weg $\gamma : [a, b] \to \mathbb{R}^n$ heißt geschlossen, falls $\gamma(a) = \gamma(b)$ gilt. Ein Bogen Γ heißt eine* Jordankurve*, falls er von einem geschlossenen Weg erzeugt wird, der auf $[a, b[$ injektiv ist. Einen derartigen Weg nennt man eine* Jordandarstellung *der Jordankurve Γ. Eine Jordankurve heißt rektifizierbar, falls sie eine rektifizierbare Jordandarstellung besitzt. Ist die Jordankurve Γ rektifizierbar, stimmen die Weglängen aller ihrer Jordandarstellungen überein, und die gemeinsame Länge heißt der* Umfang *von Γ.*

Daß der Begriff „Umfang“ vernünftig gewählt ist, kann man dem folgenden berühmten (und sehr schwer zu beweisenden Satz) entnehmen.

Theorem 6.1.1.14 (Jordanscher Kurvensatz)**.** *Jede Jordankurve $\Gamma \subseteq \mathbb{R}^2$ zerlegt $\mathbb{R}^2$ in zwei Gebiete, die von ihr berandet werden. Genauer gilt*

$$\mathbb{R}^2 \setminus \Gamma = G_1 \cup G_2,$$

wobei die G_i Gebiete sind mit $\partial G_1 = \partial G_2 = \Gamma$. Genau eines dieser Gebiete ist beschränkt, es wird das Innere von Γ *genannt. Das andere Gebiet heißt dem entsprechend das* Äußere von Γ.

Nachdem nun zu einem gegebenen Jordanbogen (einer gegebenen Jordankurve) eine ganze Klasse (mehr oder weniger äquivalenter) Parameterdarstellungen gehört, bleibt die Frage, ob es eine „beste“ (natürliche) Darstellung gibt. Für rektifizierbare Bögen ist das in einem gewissen Sinn in der Tat der Fall.

Für einen rektifizierbaren Jordanweg $\gamma : [a, b] \to \mathbb{R}^n$ ist die Bogenlängenfunktion $s : [a, b] \to \mathbb{R}$ eine streng monoton wachsende Funktion, die $[a, b]$ bijektiv auf das Intervall $[0, L(\gamma)]$ abbildet. Daher besitzt sie auch eine Umkehrfunktion $\tilde{s} : [0, L(\gamma)] \to [a, b]$. $\tilde{s}$ ist ebenfalls eine streng monoton wachsende Funktion und daher eine orientierungserhaltende Umparametrisierung. Damit ist $\gamma \circ \tilde{s} : [0, L(\gamma)] \to \mathbb{R}^n$ eine Jordandarstellung von Γ, die Bogenlängenparametrisierung. Diese Darstellung ist wegen der Unabhängigkeit der Bogenlänge von der Parametrisierung γ ebenfalls unabhängig von γ, und ihr definierendes Intervall läßt sich auch schreiben als $[0, L(\Gamma)]$.

Theorem 6.1.1.15. *Γ sei ein rektifizierbarer Jordanbogen (eine rektifizierbare Jordankurve). Dann besitzt Γ eine (rektifizierbare) Jordandarstellung $\gamma_\Gamma : [0, L(\Gamma)] \to \mathbb{R}^n$ mit der Bogenlänge als Parameter. Besitzt überdies Γ eine reguläre Jordandarstellung, so ist γ_Γ stetig differenzierbar und regulär, und für alle $s \in [0, L(\Gamma)]$ gilt*

$$\left\| \frac{d\gamma_\Gamma(s)}{ds} \right\|_2 = 1.$$

Bis jetzt ist der Begriff *Kurve*, der in der Überschrift des Abschnittes vorgekommen ist, noch nicht gefallen. Das hat seinen Grund, denn in der Literatur werden abwechselnd γ und Γ als Kurve bezeichnet. Manchmal wird auch keine explizite Unterscheidung zwischen den Begriffen Bogen und Weg gemacht. Was wir eigentlich unter einer Kurve verstehen, ist ein Bogen zusammen mit einer regulären Parametrisierung, wobei es uns aber nicht darauf ankommen soll, welche Parametrisierung wir genau wählen. Wir sind sozusagen an parametrisierten Bögen modulo Reparametrisierungen interessiert. Wenn wir in den folgenden Abschnitten von Kurven sprechen, werden wir darunter parametrisierte Wege verstehen, wobei wir sinnvolle Parametrisierungen verwenden werden.

Definition 6.1.1.16 (informell)**.** *Eine (geschlossene)* Kurve *ist ein regulärer Jordanbogen (eine reguläre Jordankurve)* Γ*, das ist ein Jordanbogen (eine Jordankurve) mit regulärer Bogenlängenparametrisierung, zusammen mit einer fest gewählten Äquivalenzklasse von regulären Jordandarstellungen* $[\gamma]$ *bezüglich der Äquivalenzrelation* $\gamma_1 \sim \gamma_2$*, genau dann wenn es eine stetig differenzierbare orientierungserhaltende Umparametrisierung* ω *mit* $\gamma_2 = \gamma_1 \circ \omega$ *gibt.*

Oft werden wir auch einen Repräsentanten γ *stellvertretend für die Äquivalenzklasse* $[\gamma]$ *herausgreifen und ihn selbst als Kurve bezeichnen, doch es sei immer das Paar* $(\Gamma, [\gamma])$ *gemeint. Haben wir einen Repräsentanten* γ *gewählt, bezeichnen wir den Bogen* Γ *auch oft mit* γ^**, um eine eindeutige Zuordnung von Kurven, Parametrisierungen und Bögen zueinander vornehmen zu können.*

Bernstein–Polynome

Ein wesentlicher Spezialfall für alle im folgenden vorgestellten Konzepte basiert auf speziell gewählten B-Splines (siehe Definition **??** aus Kapitel **??**). Die Bernstein–Polynome, eigentlich viel früher als die B-Splines eingeführt, ergeben sich als Spezialfall für eine bestimmte Knotensequenz und die mit deren Hilfe definierten speziellen B-Spline–Kurven, die Bézier–Kurven, sie waren der ursprüngliche Beginn der Spline–Kurven.

Definition 6.1.1.17. *Sei* $[a,b] = [0,1]$*, und betrachten wir die spezielle Knotensequenz* $\tau_0 = (t_0, \ldots, t_{2k+1})$ *mit*

$$\begin{cases} t_0 = \cdots = t_k & = 0 \\ t_{k+1} = \cdots = t_{2k+1} & = 1. \end{cases}$$

Die zu dieser Knotensequenz gehörenden B–Splines $B_i^k(x) := B_{i,k,\tau_0}(x)$ *für* $0 \le i \le k$ *werden* Bernstein–Polynome *genannt. Sie sind Polynome vom Grad k auf* $[0,1]$*, die nach Kapitel* **??** *Definition* **??** *die Beziehung*

$$B_i^k(x) = xB_i^{k-1}(x) + (1-x)B_{i+1}^{k-1}(x)$$

erfüllen. Wegen des B-Spline–Basis–Satzes (Kapitel **??** *Theorem* **??***) bilden die* B_i^k *eine Basis des Vektorraums aller Polynome (auf* $[0,1]$*), die* Bernstein–Basis*.*

Aus den Eigenschaften für B–Splines (Kapitel **??** Proposition **??**) und einem einfachen Induktionsbeweis folgen die nachstehenden Ergebnisse und die explizite Darstellung der Bernstein–Polynome

Proposition 6.1.1.18. *Für die Bernstein–Polynome* $B_i^k(x)$ *gilt*

1. $B_i^k(x) = \binom{k}{i}(1-x)^{k-i}x^i$ *für* $0 \le i \le k$*,*
2. *Die* B_i^k *für* $0 \le i \le k$ *bilden eine Basis für den Vektorraum* $\mathbb{R}^k[x]$ *aller Polynome höchstens k–ten Grades,*
3. $B_i^k(x) \ge 0$ *für* $x \in [0,1]$*,*
4. $\sum_{i=0}^{k} B_i^k(x) \equiv 1$*,*
5. $\frac{d}{dx}B_i^k(x) = k(B_{i-1}^{k-1}(x) - B_i^{k-1}(x))$*,*
6. $B_i^k(x) = B_{k-i}^k(1-x)$ *für* $x \in [0,1]$*.*

Beweis. Die explizite Darstellung folgt aus der Rekursionsformel für die Bernstein–Polynome aus Definition 6.1.1.17 und der Beziehung

$$\binom{k}{i} = \binom{k-1}{i} + \binom{k-1}{i-1}.$$

Die Formel für die Ableitung folgt aus der Darstellung, und alle anderen Behauptungen folgen aus Proposition **??** aus Kapitel **??**.

Die Partition der Eins Eigenschaft sieht man auch aus dem binomischen Lehrsatz

$$\sum_{i=0}^{k} \binom{k}{i} x^i (1-x)^{k-i} = (x+1-x)^k = 1.$$

□

Beispiel 6.1.1.19. *Die vier Elemente der Bernstein–Basis vom Grad 3 sind im folgenden aufgelistet:*

$$B_0^3(x) = (1-x)^3, \quad B_1^3(x) = 3(1-x)^2 x,$$
$$B_2^3(x) = 3(1-x)x^2, \quad B_3^3(x) = x^3.$$

Abbildung 6.1: Bernstein–Basis in Dimension 3

Bemerkung 6.1.1.20. *Für ein beliebiges Intervall $[a,b]$ werden oft in Analogie ebenfalls Bernstein–Polynome definiert mit der Knotensequenz $\tau_{[a,b]}$*

$$\begin{cases} t_0 = \cdots = t_k & = a \\ t_{k+1} = \cdots = t_{2k+1} & = b. \end{cases}$$

Der Zusammenhang zu den oben definierten Bernstein–Polynomen auf $[0,1]$ ist

$$B_{i,k,\tau_{[a,b]}}(x) = B_i^k(\tfrac{x-a}{b-a}).$$

6.1.2 B-Spline–Kurven

Ausgehend von den B-Splines, die wir in Kapitel **??** kennengelernt haben, definieren wir zunächst B-Spline–Kurven als Glättungen eines vorgegebenen Polygons, des *Kontrollpolygons*. Dann lösen wir das Interpolationsproblem wie für Funktionen durch Lösung eines linearen Gleichungssystemes.

Im Gegensatz zu den Interpolationsfunktionen muß man dabei die Reihenfolge der zu interpolierenden Punkte beachten. Während im $\mathbb{R}^1$ eine Totalordnung existiert, die die Reihenfolge der Interpolationspunkte festlegt, kann bei Kurveninterpolation die Reihenfolge nicht allein aus der Position der Punkte abgeleitet werden (siehe Abbildung 6.2).

Abbildung 6.2: Zwei verschiedene Kurven durch dieselben Interpolationspunkte

Grundlagen

Seien im folgenden $P_0,\dots,P_{n-1}$ Punkte im $\mathbb{R}^s$. Mit $P=[P_0,\dots,P_{n-1}]$ sei das von $P_0,\dots,P_{n-1}$ definierte Polygon bezeichnet.

Definition 6.1.2.1. *Die* B-Spline–Kurve *assoziiert zum* Kontrollpolygon *P ist die Kurve γ parametrisiert durch*

$$\gamma(t)=\sum_{i=0}^{n-1} P_i B_{i,k,\tau}(t), \quad t\in[a,b] \tag{6.1}$$

für eine vorgegebene B-Spline–Basis $B_{i,k,\tau}$ mit n Elementen auf dem Intervall $[a,b]$.

Im speziellen Fall der Bernstein–Polynome B_i^k definiert das Kontrollpolygon P (in diesem Fall gilt $n=k+1$) die Bézier–Kurve

$$B(t)=\sum_{i=0}^{k} P_i B_i^k(t)$$

assoziiert zu P.

Ein typisches Beispiel für eine Spline–Kurve ist in Abbildung 6.3 dargestellt.

Eine Zusammenfassung der wichtigsten Eigenschaften von B-Spline–Kurven ist in der folgenden Proposition enthalten. Einige dieser Eigenschaften kann man auch durch genaue Betrachtung von Abbildung 6.3 erahnen.

Proposition 6.1.2.2. *1. γ geht im allgemeinen nicht durch die Eckpunkte P_i des Kontrollpolygons. Es gilt allerdings, falls $t_0=\dots=t_k=a$ und $t_n=\dots=t_{n+k}=b$ sind, $\gamma(a)=P_0$ und $\gamma(b)=P_{n-1}$. In diesem Fall verläuft γ tangential zum Kontrollpolygon in den Endpunkten P_0 und P_{n-1}.*

2. γ^* liegt in der konvexen Hülle der Punkte $P_0,\dots,P_{n-1}$. Genauer liegt $\gamma(t)$ in der konvexen Hülle der Punkte $P_{i-k},\dots,P_i$ für $t_i\le t<t_{i+1}$.

3. Sind die Knoten t_i $(k+1\le i\le n-1)$ einfach, so ist γ eine C^{k-1}–Kurve aufgebaut aus n parametrisierten polynomialen Bögen vom Grad k.

4. Die Form von γ^* ist invariant unter affinen Abbildungen des $\mathbb{R}^s$: Für jede solche Abbildung h bilden die Punkte $h(P_i)$ das Kontrollpolygon der Kurve $h(\gamma(t))$.

Abbildung 6.3: B-Spline–Kurve

5. *$\gamma(t)$ regularisiert das Kontrollpolygon P im folgenden Sinn: Für jede Hyperebene H, die transversal zu γ^* liegt, ist die Anzahl der Schnittpunkte $|H \cap \gamma^*|$ höchstens so groß wie die Anzahl der Schnittpunkte von H und P.*

$$|H \cap \gamma^*| \leq |H \cap P|$$

Beweis. Die meisten Behauptungen folgen offensichtlich aus den Eigenschaften für B-Splines aus Kapitel **??** (Proposition **??**).

Für die Aussagen über die konvexe Hülle verwendet man

$$\sum_{i=0}^{n-1} B_{i,k,\tau}(x) \equiv 1$$

$$B_{i,k,\tau} \geq 0$$

auf $[a,b]$. Daher ist die definierende Gleichung (6.1) eine konvexe Linearkombination. Die genauere Aussage folgt aus der Trägereigenschaft der B-Splines $B_{i,k,\tau}(x) = 0$ für $i \leq j-k-1$ und $i \geq j+1$, falls $t_i \leq t < t_{i+1}$. □

Bemerkung 6.1.2.3. *Eine der wichtigsten Eigenschaften von Spline–Kurven ist ihr* lokales Verhalten.

1. *Ist Y der Punkt der Kurve γ für den Parameterwert t_0, so hängt die Position von Y nur von höchstens $k+1$ Punkten P_i ab, weil für $t_j \leq t_0 < t_{j+1}$*

$$\gamma(t_0) = \sum_{i=0}^{n-1} P_i B_{i,k,\tau}(t_0) = \sum_{i=j-k}^{j} P_i B_{i,k,\tau}(t_0)$$

   gilt.

2. *In analoger Weise beeinflußt die Position des Kontrollpunktes P_j nur jene Punkte $\gamma(t)$ für $t \in [t_j, t_{j+k+1}]$.*

Diese Eigenschaft bedingt, daß bei Verschiebung eines Kontrollpunktes P_j nur ein geringer Teil der Kurve γ neu berechnet werden muß.

Bemerkung 6.1.2.4. *Möchte man geschlossene Kurven darstellen, so wählt man am günstigsten eine gleichförmige Knotensequenz, etwa $t_i = i \in \mathbb{Z}$. Für $r \in \mathbb{Z}$ gilt dann $B_{i,k,\tau}(x) = B_{i+r,k,\tau}(x+r)$; die B-Splines sind also ganzzahlige Translate eines „Muster–Splines“. Setzt man dann*

$$\gamma(t) = \sum_{i=-\infty}^{\infty} P_i B_{i,k,\tau}(t),$$

wobei man $P_{i+kn} = P_i$ für $0 \leq i \leq n-1$ und $k \in \mathbb{Z}$ definiert, so erhält man eine Parametrisierung γ, die $\gamma(t+kn) = \gamma(t)$ erfüllt. Eine solche geschlossene Spline–Kurve ist in Abbildung 6.4 dargestellt.

Abbildung 6.4: Geschlossene Spline–Kurve

Algorithmen

Wie in Kapitel **??** gibt es auch zur Verwendung von B-Spline–Kurven eine Reihe wichtiger Algorithmen. Besonderes Augenmerk sei dabei auf den Auswertungsalgorithmus 6.1.2 und den Subunterteilungsalgorithmus 6.1.2 gerichtet, die auch für die Bildschirmdarstellung durch Näherung mit einem Polygonzug eine zentrale Rolle spielen.

Im folgenden seien immer

$$\gamma(t) = \sum_{i=0}^{n-1} P_i B_{i,k,\tau}(t)$$

eine gegebene B-Spline–Kurve und

$$B(t) = \sum_{i=0}^{k} P_i B_i^k(t)$$

eine Bézier–Kurve.

Auswertung einer B-Spline–Kurve.

Suche j mit $t_j \leq t < t_{j+1}$
$P_i^{(0)} = P_i$ für $j-k \leq i \leq j$
for $r = 0$ **to** $k-1$ **do**
  for $i = j-k+r+1$ **to** j **do**
    $$P_i^{(r+1)} = \frac{(t-t_i)P_i^{(r)} + (t_{i+k-r}-t)P_{i-1}^{(r)}}{t_{i+k-r}-t_i}$$
  done
done
$\gamma(t) = P_j^{(k)}$

Es gilt mit der Notation von Kapitel **??**

$$P_i^{(r+1)} = \frac{(t-t_i)P_i^{(r)} + (t_{i+k-r}-t)P_{i-1}^{(r)}}{t_{i+k-r}-t_i} = \omega_{i,k}(t)P_i^{(r)} + (1-\omega_{i,k}(t))P_{i-1}^{(r)},$$

wobei

$$\omega_{i,k}(t) = \begin{cases} \frac{t-t_i}{t_{i+k}-t_i} & \text{wenn } t_i < t_{i+k}, \\ 0 & \text{sonst.} \end{cases}$$

Dieser Algorithmus wird in der Literatur auch „De Boor–Cox"–Algorithmus genannt. Es gilt im übrigen, daß die Kurve γ bei $P_j^{(k)}$ tangential zur Strecke $\overline{P_j^{(k-1)}P_{j-1}^{(k-1)}}$ verläuft. Das ist z.B. eine Möglichkeit, den Tangentialvektor $\gamma'(t)$ zu bestimmen.

Im Fall von Bézier–Kurven vereinfacht sich der Auswertungsalgorithmus auf eine einfache Linearkombination. Auch der Name ist anders: „De Casteljau"–Algorithmus.

Auswertung einer Bézier–Kurve

$P_i^{(0)} = P_i$ für $0 \le i \le k-1$
for $r = 0$ **to** $k-1$ **do**
  for $i = r+1$ **to** k **do**
    $P_i^{(r+1)} = (1-t)P_{i-1}^{(r)} + tP_i^{(r)}$
  done
$B(t) = P_k^{(k)}$ **done**

Beispiel 6.1.2.5. *Für eine kubische B-Spline–Kurve mit Knotenfolge $t_0 = \cdots = t_3 = 0$, $t_4 = 1$, $t_5 = 2$, $t_6 = \cdots = t_9 = 3$ sei $\gamma(t) = \sum_{i=0}^{5} P_i B_{i,k,\tau}(t)$ definiert. Wenn wir Algorithmus 6.1.2 zur Auswertung an $t = 3/2$ verwenden, berechnen wir die Zwischenergebnisse*

$$P_2^{(1)} = \tfrac{1}{4}P_1 + \tfrac{3}{4}P_2, \quad P_3^{(1)} = \tfrac{1}{2}P_2 + \tfrac{1}{2}P_3, \quad P_4^{(1)} = \tfrac{3}{4}P_3 + \tfrac{1}{4}P_4,$$
$$P_3^{(2)} = \tfrac{1}{4}P_2^{(1)} + \tfrac{3}{4}P_3^{(1)}, \quad P_4^{(2)} = \tfrac{3}{4}P_3^{(1)} + \tfrac{1}{4}P_4^{(1)},$$
$$\gamma(\tfrac{3}{2}) = P_4^{(3)} = \tfrac{1}{2}P_3^{(2)} + \tfrac{1}{2}P_4^{(2)}.$$

In Abbildung 6.5 ist der Vorgang graphisch zusammengefaßt. Die gestrichelten Linien sind dabei die Verbindungsstrecken zwischen den Punkten $P_i^{(1)}$ bzw. $P_i^{(2)}$.

Wie im Fall von B-Splines kann man auch für B-Spline–Kurven die Splines durch Einfügung zusätzlicher Knoten verändern. Sei $\tilde{t}$ ein neuer Knoten, der aus der Knotensequenz τ die Sequenz $\tilde{\tau}$ macht. Dann erhält man eine neue B-Spline–Kurve

$$\tilde{\gamma}(t) = \sum_{i=0}^{n} \tilde{P}_i B_{i,k,\tilde{\tau}}(t)$$

mit

$$\tilde{P}_i = \begin{cases} P_i & \text{wenn } t_{i+k} \le \tilde{t}, \\ \omega_{i,k}(\tilde{t})P_i + (1-\omega_{i,k}(\tilde{t}))P_{i-1} & \text{wenn } t_i < \tilde{t} < t_{i+k}, \\ P_{i-1} & \text{wenn } \tilde{t} \le t_i. \end{cases}$$

Sei σ ein Knoten der Vielfachheit $k+1$. Dann wissen wir aus Kapitel **??**, daß für einen B-Spline $B_{i,k,\tau}(\sigma) = 1$ gilt. In diesem Fall verschwinden dann alle anderen Splines in σ. Für eine B-Spline–Kurve hat das den Effekt, daß $\gamma(\sigma)$ ein Punkt P_j des Kontrollpolygons ist. Das ist der Ausgangspunkt für den Subunterteilungsalgorithmus für B-Spline–Kurven.

Abbildung 6.5: Evaluation einer B-Spline–Kurve

Man fügt einen Knoten σ in der Mitte der B-Spline–Kurve $(k+1)$–mal ein. Auf diese Weise erhält man ein neues Kontrollpolygon $\tilde{P}_i$, das bei $\gamma(\sigma)$ die Kurve berührt. Sei $\tilde{P}_i$ dieser Berührungspunkt. Dann haben alle B-Splines $B_{j,k,\tilde{\tau}}$ mit $j < i$ Träger $[a,\sigma]$ und alle mit $j > i$ haben Träger $[\sigma,b]$. Eine Hälfte des Splines $B_{i,k,\tilde{\tau}}$ hat ebenfalls Träger links von σ und die andere Hälfte hat Träger rechts von σ. Aus diesen Betrachtungen erkennt man, daß die beiden Kurvenhälften $\gamma_1(t) := \gamma(t)$ für $t \in [a,\sigma]$ und $\gamma_2(t) := \gamma(t)$ für $t \in [\sigma,b]$ unabhängig von einander sind bis auf den Berührungspunkt $\tilde{P}_i$.

Auf diese Weise kann man die Spline–Kurve in zwei Teilkurven unterteilen, die jeweils ein eigenes Kontrollpolygon besitzen, dessen Eckpunkte vom Unterteilungsalgorithmus mitgeliefert werden. Verwendet man den Subunterteilungsalgorithmus wiederholte Male, so konvergieren die Kontrollpolygone der Teilkurven schnell gegen den Bogenzug γ^* der B-Spline–Kurve. Nach kurzer Zeit sind sie, legt man die Auflösung des Bildschirms oder Druckers zugrunde, von γ^* nicht mehr zu unterscheiden und daher geeignet, anstelle von γ zur Darstellung herangezogen zu werden.

Beginnen wir zuerst mit der Untersuchung der Bézier–Kurven. In diesem Fall läßt sich der Algorithmus in besonders kurzer Form zusammenfassen. Üblicherweise wählt man $\sigma = \frac{1}{2}$, um die Berechnung so einfach wie möglich zu gestalten.

Subunterteilungsalgorithmus für Bézier–Kurven

$P_i^{(0)} = P_i$ für $0 \le i \le k$
for $j = 0$ **to** $k-1$ **do**
  for $i = j+1$ **to** k **do**
    $P_i^{(j+1)} = \frac{1}{2}(P_{i-1}^{(j)} + P_i^{(j)})$
  done
done
$B(\frac{1}{2}) = P_k^{(k)}$
$B_1(t) = \sum_{i=0}^{k} P_i^{(i)} \hat{B}_i^k(t)$ für $t \in [0, \frac{1}{2}]$
$B_2(t) = \sum_{i=0}^{k} P_k^{(i)} \tilde{B}_i^k(t)$ für $t \in [\frac{1}{2}, 1]$
$\hat{B}_i^k(t) = B_i^k(2t)$; $\tilde{B}_i^k(t) = B_i^k(2t-1)$

Wenn man statt $\hat{B}_i^k$ und $\tilde{B}_i^k$ die üblichen Bernstein–Polynome einsetzt, erhält man zwei Kurven B_1 und

B_2, die jeweils auf $[0,1]$ definiert sind und deren Bögen B_1^* und B_2^* zusammen den Bogen B^* ergeben. Man beachte, daß der Algorithmus auch die Kontrollpolygone für B_1 und B_2 mitliefert: $P_1 = [P_0^{(0)} P_1^{(1)} \dots P_k^{(k)}]$ bzw. $P_2 = [P_k^{(k)} P_k^{(k-1)} \dots P_k^{(0)}]$. Für den Fall $k = 3$ sind alle Punkte in Abbildung 6.6 dargestellt.

Abbildung 6.6: Subunterteilungsalgorithmus für Bézier–Kurven

Wir können den Subunterteilungsalgorithmus iterieren und auf die Teilstücke B_1 und B_2 anwenden, um die ursprünglichen Bézier–Kurve in 4 und später in 2^r Stücke zu unterteilen. Die dabei entstehenden Bögen haben Kontrollpolygonzüge, die wir mit Π_r^i für $i = 0, \dots, 2^r - 1$ bezeichnen wollen. Diese Polygonzüge haben zusammen $2^r k + 1$ Ecken, unter denen $2^r + 1$ Punkte der ursprünglichen Kurve sind und zwar die Punkte $B(p/2^r)$ für $p = 0, \dots, 2^r$.

Proposition 6.1.2.6. *Die Folge von Polygonzügen Π_n^k konvergiert für $n \to \infty$ gleichmäßig gegen den Bogen B der Bézier–Kurve $B(t)$ mit Konvergenzgeschwindigkeit $\lambda/2^n$, wobei λ eine von n unabhängige Konstante bezeichnet.*

Der Subunterteilungsalgorithmus erzeugt also die Eckpunkte einer Folge von Polygonzügen $\Pi_n := \bigcup_{i=0}^{2^n-1} \Pi_n^i$, die mit *exponentieller Geschwindigkeit* gegen die Bézier–Kurve B konvergiert. Beachtet man die nur endliche Auflösung von Bildschirmen und Druckern, dann erkennt man, daß nach wenigen Subunterteilungsschritten der Polygonzug Π_n von B^* nicht mehr zu unterscheiden ist. Auf diese Weise kann man Bézier–Kurven schnell zeichnen.

Für allgemeine B-Spline–Kurven ist das Verfahren etwas komplizierter. Seien τ und $\overline{\tau}$ zwei Knotensequenzen für B-Splines, und sei $\overline{\tau}$ feiner, d.h. $\tau \subseteq \overline{\tau}$. Dann sind die B-Splines $B_{i,k,\tau}$ ausdrückbar in den $B_{i,k,\overline{\tau}}$:

$$B_{i,k,\tau}(t) = \sum_j \alpha_{i,k}(j) B_{j,k,\overline{\tau}}(t).$$

Wenn wir die Kurve

$$\gamma(t) = \sum_{i=0}^{n-1} P_i B_{i,k,\tau}(t)$$

in den $B_{i,k,\overline{\tau}}$ ausdrücken wollen,

$$\gamma(t) = \sum_{j=0}^{m-1} Q_j B_{j,k,\overline{\tau}}(t),$$

dann können wir die Q_j mit Hilfe folgender Gleichung berechnen:

$$Q_j = \sum_i \alpha_{i,k}(j) P_i.$$

Oslo–Algorithmus

$$\alpha_{i,0}(j) = \begin{cases} 1 & \text{wenn } t_i \le \bar{t}_j < t_{i+1} \\ 0 & \text{sonst} \end{cases}$$

for $m = 1$ **to** k **do**
  for $i = 0$ **to** $n-1$ **do**
    if $\bar{t}_j < \bar{t}_{j+m}$ **then**
      $\alpha_{i,m}(j) = \omega_{i,m}(\bar{t}_{j+m})\alpha_{i,m-1}(j) + (1-\omega_{i+1,m}(\bar{t}_{j+m}))\alpha_{i+1,m-1}(j)$
    else
      $\alpha_{i,m}(j) = \omega_{i,m}(\bar{t}_{j+m})\alpha_{i,m-1}(j+1) + (1-\omega_{i+1,m}(\bar{t}_{j+m}))\alpha_{i+1,m-1}(j+1)$
    endif
  done
done

Mit Hilfe des „Oslo“–Algorithmus kann man die $\alpha_{i,k}(j)$ bestimmen.

Der Algorithmus benötigt zur Bestimmung jedes einzelnen Q_j die Berechnung von $\frac{k(k+1)}{2}$ Linearkombinationen.

Proposition 6.1.2.7. *Verwendet man den Oslo–Algorithmus n–mal, wobei man in jedem Schritt in der Mitte aller Intervalle $[t_i, t_{i+1}]$ zusätzliche Knoten einfügt, und bezeichnet man die entstehenden Kontrollpolygone mit Π_n, so konvergiert die Folge der Polygone gegen den Bogen γ^* der B-Spline–Kurve. Die Konvergenzordnung ist $O(1/2^n)$, also exponentiell.*

Bemerkung 6.1.2.8. *Auch für die in Bemerkung 6.1.2.4 eingeführten geschlossenen Spline–Kurven gibt es einen Subunterteilungsalgorithmus. Er kann z.B. in [Risler 1992, 2.3.f] nachgeschlagen werden.*

6.1.3 Verbindung von Kurven

In Anwendungen kommt es häufig vor, daß verschiedene Kurvenstücke möglichst glatt aneinandergefügt werden müssen. Dabei kommt es vor allem darauf an, daß das Bild stimmt, d.h. der entstehende *Kurvenbogen* glatt aussieht.

Seien im folgenden $\gamma^{(1)}$ und $\gamma^{(2)}$ (bzw. $B^{(1)}$ und $B^{(2)}$) mit

$$\gamma^{(i)}(t) := \sum_{j=0}^{n_i-1} P_j^{(i)} B_{j,k_i,\tau^{(i)}}(t)$$

zwei B-Spline–Kurven definiert auf den Intervallen $[a^{(i)}, b^{(i)}]$, $i = 1,2$ (bzw. Bézier–Kurven beide definiert auf $[0,1]$). Wir betrachten wie in Abschnitt 6.1.1 die zusammengesetzte Kurve $\gamma := \gamma^{(1)} \oplus \gamma^{(2)}$ definiert auf dem Intervall $[a^{(1)}, b^{(1)} + b^{(2)} - a^{(2)}]$

$$\gamma(t) = \begin{cases} \gamma^{(1)}(t) & t \in [a^{(1)}, b^{(1)}], \\ \gamma^{(2)}(t - b^{(1)} + a^{(2)}) & t \in [b^{(1)}, b^{(1)} + b^{(2)} - a^{(2)}]. \end{cases}$$

Im folgenden wollen wir untersuchen, welche Glattheitseigenschaften die Parameterdarstellung γ am Stoßpunkt $\gamma(b^{(1)})$ besitzt.

Damit γ an $b^{(1)}$ stetig ist, also C^0 und damit ein Weg, genügt die einfache Voraussetzung $P_{n_1-1}^{(1)} = P_0^{(2)}$, also der Endpunkt von $\gamma^{(1)}$ muß mit dem Anfangspunkt von $\gamma^{(2)}$ zusammenfallen. Für höhere Glattheitseigenschaften ist eine genauere Untersuchung notwendig.

Wir werden nur den Fall der Verbindung zweier Bézier–Kurven näher betrachten, da für beliebige B-Spline–Kurven die Formeln sehr komplex werden.

Seien $B^{(1)}$ und $B^{(2)}$ zwei Bézier–Kurven

$$B^{(j)}(t) = \sum_{i=0}^{k} P_i^{(j)} B_i^k(t).$$

Wir können ohne Beschränkung der Allgemeinheit annehmen, daß der Grad beider Kurven gleich ist. Ist dies nicht der Fall, können wir den Grad der Kurve niedrigeren Grades gemäß dem folgenden Algorithmus steigern.

Gradsteigerung einer Bézier–Kurve

$Q_0 = P_0$
for $i = 1$ **to** k **do**
$\quad Q_i = \frac{iP_{i-1}+(k-i+1)P_i}{k+1}$
done
$Q_{k+1} = P_k$
$B(t) := \sum_{i=0}^{k+1} Q_i B_i^{k+1}(t)$

Sei der Differenzenoperator Δ wie folgt definiert:

$$\Delta P_i = P_{i+1} - P_i$$
$$\Delta^r P_i = \Delta(\Delta^{r-1} P_i) = \sum_{j=0}^{r} (-1)^{r-j} \binom{r}{j} P_{i+j}$$
$$\Delta^0 P_i = P_i$$

Mit dieser Notation gilt für die Ableitung einer Bézier–Kurve

$$\frac{d^r}{dt^r} B(t) = \frac{k!}{(k-r)!} \sum_{i=0}^{k-r} \Delta^k(P_i) B_i^{k-r}(t).$$

Bedenken wir, daß bei C^ℓ–Glattheit am Verbindungspunkt alle Ableitungen bis zur Ordnung ℓ übereinstimmen müssen, so können wir aus obigen Gleichungen die Glattheitsbedingungen ablesen.

Für C^0 benötigt man

$$P_k^{(1)} = P_0^{(2)}$$

und für C^ℓ muß zusätzlich zu den Bedingungen für $C^{\ell-1}$ noch

$$\Delta^\ell P_{k-\ell}^{(1)} = \Delta^\ell P_0^{(2)}$$

gelten.

Nun ist aber C^ℓ–Glattheit etwas mehr als man in Wirklichkeit benötigt. Für einen glatten Bogenzug γ^* ist C^ℓ–Glattheit nicht notwendig, da nur *eine* reguläre Parametrisierung von γ^* existieren muß, die C^ℓ ist. Das impliziert nicht, daß die durch die Bernstein–Polynome vorgegebene Parametrisierung ebenfalls C^ℓ ist. In der algorithmischen Geometrie hat sich für solche Kurven ein Begriff eingebürgert:

Definition 6.1.3.1. *Seien $r \in \mathbb{N}$ und $\gamma^{(1)}(t)$ ($t \in [0,1]$) und $\gamma^{(2)}(t)$ ($t \in [1,2]$) Parametrisierungen zweier regulärer C^r–Kurven im $\mathbb{R}^s$ mit $\gamma^{(1)}(1) = \gamma^{(2)}(1)$. Ist $1 \le i \le r$ eine weitere ganze Zahl, so sagt man, daß $\gamma^{(1)}$ und $\gamma^{(2)}$ eine G^i–*Verbindung *am Punkt $\gamma^{(1)}(1) = \gamma^{(2)}(1)$ haben, wenn es eine reguläre orientierungserhaltende Parametertransformation $t = \varphi(u)$ der Klasse C^i gibt mit $\varphi'(t) > 0$, $\varphi(0) = 0$ und $\varphi(1) = 1$, sodaß die Kurve $\gamma^{(1)}(\varphi(u))$ eine C^i–Verbindung mit der Kurve $\gamma^{(2)}$ bei $u = t = 1$ hat.*

Das ist äquivalent zu der Tatsache, daß es eine Reparametrisierung der zusammengesetzten Kurve $\gamma^{(1)} \oplus \gamma^{(2)}$ gibt, die C^i ist an $\gamma^{(1)}(1)$.

Klarerweise ist G^0 gleichbedeutend mit C^0, doch ab G^1 unterscheidet sich G^i von C^i.
Für den Fall der Verbindung zweier Bézier–Kurven wollen wir die Begriffe G^1 und G^2 genauer betrachten.

G^1 Es besteht eine G^1–Verbindung, wenn die Tangentialvektoren der beiden Kurven an der Berührungsstelle parallel sind (C^1 würde bedeuten, daß sie zusätzlich noch gleich lang sind). Es muß daher eine Konstante $\beta_1 > 0$ existieren mit $\beta_1 B^{(1)'}(1) = B^{(2)'}(1)$. Die Verbindung ist C^1, falls $\beta_1 = 1$.

G^2 Für G^2–Glattheit müssen zusätzlich zu den G^1–Bedingungen die Radii der Krümmungskreise im Verbindungspunkt übereinstimmen. Das läßt sich in die folgende Bedingung übersetzen: Es existiere eine Konstante β_2 mit

$$B^{(2)''}(1) = \beta_2 B^{(1)'}(1) + \beta_1^2 B^{(1)''}(1).$$

Eine C^2–Verbindung liegt vor, falls $\beta_1 = 1$ und $\beta_2 = 0$ gelten.

Traditionell heißen β_1 der *Bias–Parameter* und β_2 der *Spannungsparameter*.

Für Bézier–Kurven gilt, daß die Verbindung G^1 ist, wenn eine Konstante $\beta_1 > 0$ existiert mit

$$\beta_1 \Delta P_{k-1}^{(1)} = \Delta P_0^{(2)}.$$

G^2–Glattheit liegt vor, wenn es zusätzlich eine Konstante β_2 gibt mit

$$\beta_2 \Delta P_{k-1}^{(1)} + \beta_1^2 \Delta^2 P_{k-2}^{(1)} = \Delta^2 P_0^{(2)}.$$

6.1.4 Rationale Spline–Kurven

Eine B-Spline–Kurve glättet den vorgegebenen Polygonzug $[P_0 \dots P_{n-1}]$ an allen Stellen in gleichem Maße. Dadurch lassen sich mitunter stärkere Krümmungen nur durch eine große Anzahl an Punkten darstellen. Um diesem Manko abzuhelfen, kann man die Definition der B-Spline–Kurven ein wenig verallgemeinern und die Kurven nicht aus stückweise polynomialen Bögen zusammensetzen sondern stattdessen stückweise rationale Bögen verwenden. Dieses Verfahren führt zu den rationalen B-Spline–Kurven bzw. den rationalen Bézier–Kurven.

Definition 6.1.4.1. *Seien $P_0, \dots, P_{n-1}$ wieder n Punkte im $\mathbb{R}^s$, und $B_{i,k,\tau}$ eine n–elementige B-Spline–Basis auf dem Intervall $[a,b]$. Seien weiters n reelle Zahlen $w_0, \dots, w_{n-1}$ gegeben, die nicht alle gleich 0 sind. Dann ist die zu diesen Daten gehörende* rationale B-Spline–Kurve *γ definiert als*

$$\gamma(t) = \frac{\sum_{i=0}^{n-1} P_i w_i B_{i,k,\tau}(t)}{\sum_{i=0}^{n-1} w_i B_{i,k,\tau}(t)}.$$

Die w_i heißen die zu γ assoziierten Gewichte.
Wir führen zusätzlich noch die folgende Notation ein

$$\psi_i(t) = \frac{w_i B_{i,k,\tau}(t)}{\sum_{i=0}^{n-1} w_i B_{i,k,\tau}(t)},$$

denn mit dieser Definition läßt sich γ kürzer schreiben als

$$\gamma(t) = \sum_{i=0}^{n-1} P_i \psi_i(t).$$

Wählt man als B-Spline–Basis die Bernstein–Polynome, so erhält man eine rationale Bézier–Kurve*:*

$$B(t) = \frac{\sum_{i=0}^{k} P_i w_i B_i^k(t)}{\sum_{i=0}^{k} w_i B_i^k(t)}.$$

Die rationalen B-Spline–Kurven sind eine Verallgemeinerung der üblichen B-Spline–Kurven, haben aber ähnliche Eigenschaften wie das folgende Resultat zeigt.

Proposition 6.1.4.2. *1. Wenn alle $w_i = 1$ sind, erhalten wir die üblichen B-Spline–Kurven, weil*

$$\sum_{i=0}^{n-1} B_{i,k,\tau}(t) \equiv 1$$

gilt.

2. Für $w_i > 0$ verschwindet der Nenner von $\gamma(t)$ nirgends. Das ist der häufigste Fall.

3. Es gilt

$$\sum_{i=0}^{n-1} \psi_i(t) \equiv 1,$$

und $\operatorname{supp} \psi_i = \operatorname{supp} B_{i,k,\tau}$. *Daher gelten für rationale B-Spline–Kurven dieselben Eigenschaften bezüglich Lokalität und konvexen Hüllen wie für B-Spline–Kurven.*

Die Auswirkungen einiger Parameterwahlen sind in Abbildung 6.7 dargestellt. Die gekrümmte durchgezogene Linie ist der übliche B–Spline, die gestrichelte Linie ist mit den Gewichten $w = (0.3, 4, 7, 0.2)$, die strichpunktierte Linie aus der Wahl $w = (0.3, 50, 100, 0.2)$ entstanden. Man sieht deutlich, wie die hohen Gewichte die Kurve zum Polygonzug hin ziehen. Man kann sich die w_i als Spannung von Gummibändern vorstellen, die die Spline–Kurve mit dem Kontrollpolygon verbinden. Erhöht man die Spannung, dann nähert sich der Spline dem Polygon.

Abbildung 6.7: rationale B-Spline–Kurven

6.1.5 Das Interpolationsproblem

Nach all diesen Vorbereitungen können wir endlich das Problem behandeln, von dem wir ursprünglich ausgegangen sind, das Interpolationsproblem.

Gegeben seien n Punkte $Q_0, \ldots, Q_{n-1}$ im $\mathbb{R}^s$; wir suchen eine B-Spline–Kurve $\gamma(t)$, die in der richtigen Reihenfolge durch alle Punkte Q_i geht. Wir suchen also eine Kurve γ und reelle Zahlen u_i mit $u_i < u_{i+1}$, sodaß

$$\gamma(u_i) = Q_i, \quad i = 0, \ldots, n-1$$

erfüllt ist.

Der erste Schritt zur Lösung des Problems ist die Wahl einer Knotensequenz $\tau = (t_i)_{i=0}^{n+k}$. Hat man die zu dieser Knotensequenz gehörenden B-Splines $B_{i,k,\tau}$ gebildet, so bleibt noch übrig, ein Kontrollpolygon $[P_0 \dots P_{n-1}]$ zu finden, sodaß

$$\gamma(t) = \sum_{i=0}^{n-1} P_i B_{i,k,\tau}(t)$$

die Interpolationsbedingungen erfüllt. Dieses Kontrollpolygon läßt sich aus dem linearen Gleichungssystem

$$Q_j = \sum_{i=0}^{n-1} P_i B_{i,k,\tau}(u_j)$$

berechnen. Die Lösbarkeit dieses Gleichungssystemes untersucht der folgende Satz.

Theorem 6.1.5.1. *Die Matrix $N = \left(B_{j,k,\tau}(u_i)\right)_{i,j}$ ist regulär genau dann, wenn alle Diagonalelemente ungleich Null sind, d.h. wenn $B_{j,k,\tau}(u_j) \neq 0$ für $0 \leq j \leq n-1$ gilt. Das ist äquivalent zu der Forderung $t_i < u_i < t_{i+k+1}$ für $0 \leq i \leq n-1$.*

Beweis. Es wird der Beweis nur in groben Zügen wiedergegeben. Im ganzen Beweis wird die Notation $B_i := B_{i,k,\tau}$ verwendet.

Die Bedingung ist notwendig für die Regularität: Angenommen, es existiert ein i mit $B_i(u_i) = 0$. Dann gilt $u_i \leq t_i$ (und $t_i < t_{i+k}$, falls $u_i = t_i$). In diesem Fall enthält jede der ersten $i+1$ Zeilen von N höchstens i Terme, die ungleich Null sind, nämlich $B_0(u_r), \dots, B_{i-1}(u_r)$ in Zeile r, weil $B_j(u_r) = 0$ für $j \geq i$ gilt. Daher sind diese $i+1$ Zeilen linear abhängig, woraus die Singularität von N folgt.

Daß die Bedingung auch hinreichend ist, ist schwieriger zu zeigen. Sei

$$B(t) = \sum_{j=0}^{n-1} \lambda_j B_j(t).$$

Dann ist $B(u_i) = 0$ für $0 \leq i \leq n-1$ eine Relation zwischen den Spalten von N. Zu zeigen ist, daß aus der Annahme $B_i(u_i) \neq 0$ für $i = 0, \dots, n-1$ folgt, daß alle $\lambda_i = 0$ sind. Dann sind die Spalten von N linear unabhängig und N ist regulär.

Aus den Eigenschaften der B-Splines kann man folgendes Lemma leicht herleiten:

Lemma 6.1.5.2.

1. *Wenn $i \geq k$ und $t_i < t_{i+1}$ gilt, dann sind die B-Splines $B_{i-k}(t), \dots, B_i(t)$ eingeschränkt auf das Intervall $[t_i, t_{i+1}[$ linear unabhängig.*

2. *Gilt $0 \leq i < k$ und ist $t_i < t_{i+1}$, dann sind die B-Splines $B_0(t), \dots, B_i(t)$ linear unabhängig, eingeschränkt auf das Intervall $[t_i, t_{i+1}[$.*

Dieses Resultat erlaubt zu folgern, daß im Falle $B(t) \equiv 0$ für $t \in [t_i, t_{i+1}[$ $(t_i < t_{i+1})$ automatisch $\lambda_{i-k} = \dots = \lambda_i = 0$ folgt. Wenn außerdem für alle Intervalle $[t_j, t_{j+k+1}[$ ein i existiert mit $j \leq i \leq j+k+1$, $t_i < t_{i+1}$ und $B(t)|_{[t_i, t_{i+1}[} \equiv 0$, dann gilt schon $B(t) \equiv 0$, was den Beweis beenden würde.

Aufgrund dieser Fakten können wir annehmen, daß ein Intervall $I = [t_r, t_s[$ existiert mit folgenden Eigenschaften:

(a) Es gilt in keinem Teilintervall $[t_i, t_{i+1}[$ $(t_i < t_{i+1})$ von $[t_r, t_s[$, daß $B(t) \equiv 0$;

(b) $s \geq r+k+1$;

(c) I ist maximal mit diesen Eigenschaften.

Setzen wir nun

$$\tilde{B}(t) = \sum_{i=r-k}^{s-1} \lambda_i B_i(t).$$

Aus Eigenschaft (c) folgt $B(t) \equiv 0$ auf den Intervallen $[t_{r-1}, t_r[$ und $[t_s, t_{s+1}[$. Aus Lemma 6.1.5.2 folgt daher $\lambda_{r-k} = \cdots = \lambda_{r-1} = 0$ und $\lambda_{s-k} = \cdots = \lambda_s = 0$, was wiederum

$$\tilde{B}(t) = \sum_{i=r}^{s-k-1} \lambda_i B_i(t)$$

impliziert. Aus den Trägereigenschaften für B-Splines folgt zusätzlich $B(t) = \tilde{B}(t)$ für $t \in [t_r, t_s[$.

Wegen der Hypothese $B_i(u_i) \neq 0$ haben wir

$$t_r < u_r < \cdots < u_{s-k-1} < t_s,$$

und daher hat die Funktion $\tilde{B}(t)$ mindestens $s-k-r$ verschiedene Nullstellen im Intervall $[t_r, t_s[$ (weil $B(u_i) = \tilde{B}(u_i) = 0$ gilt laut Annahme), und $\tilde{B}(t)$ ist nicht identisch Null auf irgend einem Teilintervall $[t_i, t_{i+1}[$.

Verwendet man jetzt noch, daß B-Splines stückweise Polynome sind, so kann man Nullstellen zählen und findet, daß solch ein $\tilde{B}$ nicht existieren kann. Daher ist $I = \emptyset$ und $B(t) \equiv 0$. Darum sind alle $\lambda_i = 0$, und die Spalten von N sind linear unabhängig; also ist N invertierbar. □

Die Matrix N hat noch weitere wichtige Eigenschaften, die ihre numerische Bearbeitung sehr einfach machen.

Proposition 6.1.5.3. *1. N ist eine Bandmatrix mit Bandbreite $k+1$. Falls man kubische Bézier–Kurven verwendet, ist N eine Tridiagonalmatrix.*

2. N ist total positiv. Das heißt man kann N stabil ohne Pivotsuche LR–zerlegen. Bei der LR–Zerlegung von N tritt damit keine Bandverbreiterung auf.

Üblicherweise wählt man

$$u_i = \frac{t_{i+1} + \cdots + t_{i+k}}{k},$$

denn diese Wahl impliziert $B_{i,k,\tau}(u_i) \neq 0$ und damit die eindeutige stabile Lösbarkeit des Interpolationsproblemes. In manchen Fällen (z.B. in Anwendungen aus der Physik) stehen die Interpolationswerte u_i allerdings schon im Vorhinein fest. Dann hat man meist das Problem, die Knotenpunkte richtig zu wählen. In diesem Fall verwendet man am besten die Formeln

$$\begin{cases} t_0 = \cdots = t_k = u_0 \\ t_{i+k} = \frac{u_i + \cdots + u_{i+k-1}}{k} & \text{für } 1 \leq i \leq n-k-1 \\ t_n = \cdots = t_{n+k} = u_{n-1}. \end{cases}$$

6.2 Interpolierende Flächen

Möchte man mehrdimensionale Funktionen interpolieren, ist es geschickter, gleich einen allgemeineren Fall zu betrachten: Wir werden versuchen, eine r–dimensionale Fläche im $\mathbb{R}^s$ zu interpolieren. Eine Funktion $f : \mathbb{R}^r \to \mathbb{R}$ kann man auch als r–dimensionale Fläche im $\mathbb{R}^{r+1}$ betrachten.

Für den Beginn wollen wir den Fall $r = 2$ und s beliebig betrachten, also Flächen im $\mathbb{R}^s$ untersuchen. Die meisten Resultate, die wir dabei erhalten, können dann mit mäßigem Aufwand verallgemeinert werden (siehe Abschnitt 6.3).

Wir werden ähnlich vorgehen wie im vorangehenden Abschnitt: Zuerst definieren wir Spline–Patches, dann untersuchen wir deren Eigenschaften und Anwendbarkeit.

6.2.1 Grundlagen

Wie im Fall der Kurven benötigen wir auch in diesem Abschnitt einiges Wissen aus der Analysis, das wir hier kurz wiederholen.

Flächen im $\mathbb{R}^n$

Ähnlich wie im Abschnitt über Kurven (6.1.1) gilt es auch bei deren mehrdimensionaler Verallgemeinerung zwischen den Punktmengen und deren Parametrisierungen zu unterscheiden.

Auch die hier genannten Ergebnisse und Definitionen können so oder in ähnlicher Form in jedem Analysis–Buch, etwa [Heuser 1986/2, XXIV f.], nachgelesen werden. Um Schwierigkeiten zu vermeiden, werden wir die Definitionen nicht in größter Allgemeinheit geben. In Anwendungen werden kaum allgemeinere als die hier definierten Flächen benötigt werden.

Definition 6.2.1.1. *Sei $B \subseteq \mathbb{R}^p$ eine zusammenhängende, beschränkte und abgeschlossene Teilmenge, die Jordan–meßbar ist (ihre charakteristische Funktion χ_B ist Riemann–integrierbar). Im folgenden werden wir so eine Menge einen* Parameterbereich *nennen.*

Unter einer (p–dimensionalen) Flächenparametrisierung *mit dem Parameterbereich B verstehen wir die injektive Einschränkung $\Phi|_B$ einer C^1–Abbildung $\Phi : M \to \mathbb{R}^n$ auf B; dabei ist M ein offenes Gebiet, das B enthält. Die Bildmenge $F := \Phi(B)$ wird ein (p–dimensionales)* Flächenstück *genannt; die Gleichung*

$$r = \Phi(u), \quad u \in B$$

heißt Parameterdarstellung *von F.*

Zwischen Flächenparametrisierung und Flächenstück besteht ein analoger Zusammenhang wie zwischen Weg und Bogen (lediglich die Namen klingen holpriger).

Definition 6.2.1.2. *Der* Inhalt einer parametrisierten Fläche Φ *ist definiert durch das Integral*

$$I(\Phi) := \int_B \mathrm{vol}(P_\Phi(u))\, du,$$

wobei $\mathrm{vol}(P_\Phi(u))$ *das p–dimensionale Volumen des Parallelepipeds $P_\Phi(u)$ ist, das von den p Vektoren*

$$\frac{\partial \Phi(u_1, \ldots, u_p)}{\partial u_i}, \quad i = 1, \ldots, p$$

aufgespannt wird.

Sind $p = 2$ und $n = 3$, so kann man den Inhalt auch einfacher berechnen als

$$I(\Phi) = \int_B |\nu(u_1, u_2)|\, d(u_1, u_2),$$

wobei $\nu(u_1, u_2)$ der Normalenvektor von Φ im Flächenpunkt $\Phi(u_1, u_2)$ ist:

$$\nu(u_1, u_2) = \frac{\partial \Phi}{\partial u_1}(u_1, u_2) \times \frac{\partial \Phi}{\partial u_2}(u_1, u_2).$$

Wie für Bögen wollen wir eigentlich einem Flächenstück F einen Inhalt (den Flächeninhalt) zuordnen. Wie in der Theorie der Kurven müssen wir uns zuerst um die verschiedenen Möglichkeiten kümmern, ein bestimmtes Flächenstück F zu parametrisieren.

Sei B' eine weitere Jordan–meßbare Teilmenge und M' ein Gebiet im $\mathbb{R}^p$, das B' enthält. Eine bijektive C^1–Abbildung $g : M' \to M$ heißt *Parametertransformation*, falls g bijektiv von B' auf B ist und die Funktionalmatrix

$$Jg = \left(\frac{\partial g(u)}{\partial u} \right)$$

an jedem Punkt $u \in M'$ regulär ist. Die Parametertransformation g heißt *orientierungserhaltend*, falls die Determinante von Jg, die Funktionaldeterminante $\det Jg$ überall positiv ist.

Ist $g : M' \to M$ eine Parametertransformation von B' auf B, und ist Φ eine Flächenparametrisierung mit Parameterbereich B, so ist $\Phi \circ g$ eine Flächenparametrisierung mit Parameterbereich B', die dasselbe Flächenstück F wie Φ definiert.

Mit Hilfe der Kettenregel sieht man sofort, daß

$$\mathrm{vol}(P_{\Phi\circ g}(u)) = \mathrm{vol}(P_\Phi(g(u))) \cdot |\det Jg(u)|$$

gilt. Aus der Transformationsregel für Mehrfachintegrale folgt dann

$$I(\Phi\circ g) = \int_{B'} \mathrm{vol}(P_{\Phi\circ g}(u))\,du = \int_B \mathrm{vol}(P_\Phi(v))\,dv = I(\Phi).$$

Der Inhalt einer parametrisierten Fläche ist also invariant unter Umparametrisierung. Daher ist es gerechtfertigt vom *Flächeninhalt* des Flächenstücks $F = \Phi(B)$ zu sprechen.

Definition 6.2.1.3. *Ist F ein (p–dimensionales) Flächenstück und $\Phi : B \to \mathbb{R}^n$ eine Parametrisierung. Dann ist der (p-dimensionale)* Flächeninhalt *von F definiert als*

$$I(F) := I(\Phi).$$

Definition 6.2.1.4 (informell)**.** *Unter einer (p–dimensionalen)* Fläche *ist ein p–dimensionales Flächenstück zusammen mit einer Äquivalenzklasse $[\Phi]$ von Flächenparametrisierungen zu verstehen. Dabei ist die Äquivalenzrelation, bezüglich der Klassen gebildet werden folgendermaßen definiert: $\Phi \sim \Psi$ genau dann wenn es eine orientierungserhaltende Umparametrisierung g gibt mit $\Phi = \Psi \circ g$.*

Oft werden wir auch einen Repräsentanten Φ aus der Äquivalenzklasse $[\Phi]$ auswählen und als Fläche bezeichnen, doch es ist immer das Paar $(F, [\Phi])$ gemeint. Haben wir einen Repräsentanten Φ gewählt, werden wir das zugehörige Flächenstück statt mit F manchmal auch mit Φ^* bezeichnen, wieder um eindeutige Zuordnungen besser vornehmen zu können.

Die Verallgemeinerung der Kurveninterpolation in höhere Dimensionen kann zwei grundverschiedene Richtungen nehmen. Dabei muß man nämlich beachten, daß der Parameterbereich B einer Fläche ebenso wie die Fläche selbst geeignet approximiert werden muß.

Der eine Zugang schränkt die Form des Parameterbereiches auf achsenparallele Quader ein, ein Rechteck für $p = 2$. Diesen achsenparallelen Quader kann man dann durch parallele Hyperebenen in kleinere Subquader zerlegen. Für $p = 2$ sind verschiedene mögliche Unterteilungen in Abbildung 6.8 dargestellt. Die Stütz–

(a) B (b) B

Abbildung 6.8: (a) äquidistante Unterteilung, (b) allgemeinere Unterteilung

oder die Interpolationspunkte liegen dann auf den Eckpunkten der Subquader. Die Spline–Flächen, die bei einem solchen Zugang verwendet werden, sind aus Tensorprodukt–Patches zusammengesetzt und werden in Abschnitt 6.2.2 behandelt.

Die andere Methode Flächen zusammenzusetzen funktioniert folgendermaßen: Man approximiert den (allgemeiner gestalteten) Parameterbereich durch eine Vereinigung von Simplices (Hypertetraedern) oder allgemeineren Polyedern. Auf jedem dieser Simplices, Dreiecken für $p = 2$, definiert man ein Flächenstück,

Abbildung 6.9: Triangulierung

einen Bézier– oder B-Spline–Patch. Aus vielen solchen Patches setzt man dann die zu modellierende (interpolierende) Fläche zusammen. Ein Beispiel für eine Triangulierung sehen wir in Abbildung 6.9; die Theorie der Triangulierungen finden wir in Abschnitt 6.4, die Konstruktion mehrdimensionaler Flächenstücke wird in den Abschnitten 6.2.3, 6.3.2 und 6.3.3 behandelt.

Zu Beginn wollen wir der Einfachheit halber alle Entwicklungen für $p = 2$ durchführen, also für Flächen, der bei weitem häufigsten Anwendung.

6.2.2 Tensorprodukt–Patches

Wählt man als Parameterbereich für die Flächenstücke achsenparallele Rechtecke, erhält man die folgende einfache Verallgemeinerung der eindimensionalen B-Spline–Kurven.

Nehmen wir an, der Parameterbereich sei das Rechteck $[a,b] \times [c,d]$. Auf $[a,b]$ bzw. $[c,d]$ sei je eine Knotensequenz $\tau = (t_0, \ldots, t_{n+k})$ bzw. $\upsilon = (y_0, \ldots, y_{m+l})$ gegeben, die die B-Splines $B_{i,k,\tau}$ bzw. $B_{j,l,\upsilon}$ bestimmen.

Definition 6.2.2.1. *Mit obiger Notation, seien P_{ij}, $0 \le i \le n-1$ und $0 \le j \le m-1$, Punkte in $\mathbb{R}^s$. Ein* Tensorprodukt–B-Spline–Patch *ist eine Fläche der Form*

$$S(u,v) := \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} P_{ij} B_{i,k,\tau}(u) B_{j,l,\upsilon}(v), \quad (u,v) \in [a,b] \times [c,d].$$

Wir schreiben auch manchmal

$$\begin{aligned} S(u,v) &= \sum_{i=0}^{n-1} P_i(v) B_{i,k,\tau}(u) = \\ &= \sum_{j=0}^{m-1} P^j(u) B_{j,l,\upsilon}(v), \quad \textit{mit} \\ P_i(v) &= \sum_{j=0}^{m-1} P_{ij} B_{j,l,\upsilon}(v), \\ P^j(u) &= \sum_{i=0}^{n-1} P_{ij} B_{i,k,\tau}(u). \end{aligned}$$

Einige Eigenschaften der Tensorprodukt–B-Splines werden durch die zweite Schreibweise besonders augenfällig.

Proposition 6.2.2.2. *Die Einschränkung eines Tensorprodukt–B-Splines auf eine achsenparallele Linie $v = v_0$ (oder $u = u_0$) ist eine B-Spline–Kurve mit dem von den Punkten $P_i(v_0)$ ($P^j(u_0)$) erzeugten Kontrollpolygon.*

Beweis. Offensichtlich aus der Definition. □

Jede Koordinate der Funktion $S(u,v)$ hat den Totalgrad $l+k$. Das bedeutet im üblichen Fall $l = k = 3$ haben die Koordinaten von $S(u,v)$ Totalgrad 6.

Der Name Tensorprodukt–B-Spline–Patch stammt daher, daß der Raum, der von den $B_{i,k,\tau}(u)B_{j,l,\upsilon}(v)$ aufgespannt wird, der Raum $\mathbb{P}_{k,\tau} \otimes \mathbb{P}_{l,\upsilon}$ ist, das Tensorprodukt der endlich dimensionalen Vektorräume $\mathbb{P}_{k,\tau}$ und $\mathbb{P}_{l,\upsilon}$.

Der größte Vorteil der Tensorprodukt–Patches ist, daß man alle Algorithmen, die für den eindimensionalen Fall entwickelt wurden, wiederverwenden kann.

Möchte man etwa $S(u_0,v_0)$ bestimmen, so verwendet man Algorithmus 6.1.2, um $P_i(v_0)$ für die relevanten i ($i = r-k,\ldots,r$, falls $u_0 \in [u_r, u_{r+1}]$) auszurechnen. Danach wendet man erneut Algorithmus 6.1.2 an, um $S(u_0,v_0)$ zu berechnen. In ähnlicher Weise kann man den Subunterteilungsalgorithmus 6.1.2 oder den Oslo–Algorithmus 6.1.2 verwenden, um Approximationen an die B-Spline–Fläche durch Polyeder zu finden.

Interpolation

Das Interpolationsproblem läßt sich für Tensorprodukt–Patches dann lösen, wenn auch die zu interpolierenden Punkte auf einem rechteckigen Raster vorgegeben sind. Seien $Q_{\lambda\mu}$, $0 \le \lambda \le n-1$ und $0 \le \mu \le m-1$, Punkte des $\mathbb{R}^s$. Das Interpolationsproblem besteht nun darin, einen Tensorprodukt–B-Spline–Patch zu finden, der folgenden Interpolationsbedingungen genügt:

$$S(\alpha_\lambda, \beta_\mu) = Q_{\lambda\mu}, \quad \text{für } 0 \le \lambda \le n-1 \text{ und } 0 \le \mu \le m-1.$$

Die α_λ und β_μ kommen dabei aus der Anwendung oder werden ähnlich gewählt wie in Abschnitt 6.1.5.

Definiert man die beiden Matrizen $A = \left(B_{i,k,\tau}(\alpha_\lambda)\right)_{i,\lambda}$ und $B = \left(B_{j,l,\upsilon}(\beta_\mu)\right)_{j,\mu}$, dann werden die Interpolationsbedingungen zur Gleichung

$$Q = AP^\top B,$$

wobei $Q = (Q_{\lambda\mu})$ und $P = (P_{ij})$ die Matrizen sind, die aus den gegebenen bzw. gesuchten Punkte gebildet werden. Diese Gleichung läßt sich genau dann lösen, wenn die Matrizen A und B invertierbar sind. Gemäß Theorem 6.1.5.1 ist das genau dann der Fall, wenn $B_{i,k,\tau}(\alpha_i) \neq 0$ und $B_{j,l,\upsilon}(\beta_j) \neq 0$ gelten.

Tensorprodukt–Patches werden sehr oft verwendet, weil sie leicht zu berechnen sind und man bereits implementierte eindimensionale Algorithmen leicht anpassen kann. Leider haben sie aber auch einige unangenehme Nachteile:

- Sie sind nur gut verwendbar, wenn man sich auf rechteckige Parameterbereiche einschränkt.
- Es gibt zwei privilegierte Familien von Kurven auf einem Tensorprodukt–B-Spline–Patch, die durch $u = \text{const}$ bzw. $v = \text{const}$ definiert sind. Für diese Kurven sind viele Eigenschaften bekannt. Die Kurven in alle anderen Richtungen, etwa $u + v = \text{const}$ sind viel schwieriger zu kontrollieren.
- Selbst im häufigsten, dem bikubischen Fall ($k = l = 3$) ist jede Koordinate vom Totalgrad 6 in u und v. Das bedeutet, daß die implizite Gleichung, die einen Teil dieses Tensorprodukt–Patches bestimmt $F(x,y,z) = 0$, eine algebraische Gleichung vom Grad 18 ist. Der Schnitt zweier solcher Flächen ist dann eine algebraische Kurve vom Grad 324, was formalen Rechnungen schnell einen Riegel vorschiebt.

Beispiel 6.2.2.3. *Ein Beispiel für einen Tensorprodukt–B-Spline–Patch mit $k = l = 3$ ist in Abbildung 6.10 zu sehen.*

Abbildung 6.10: Tensorprodukt–B-Spline–Patch

6.2.3 Bézier–Patches

Die am Ende des vorigen Abschnitts aufgezählten Nachteile führen dazu, einen unterschiedlichen Zugang zu B-Spline–Flächen zu wählen. In ihm sind die Parameterbereiche der Flächenteile Dreiecke, was es auch ermöglicht, durch Triangulierung (siehe Abschnitt 6.4) sehr allgemeine Parameterbereiche zu verwenden.

Für $p = 2$ werden wir nur die Verallgemeinerungen der Bézier–Kurven besprechen. Auf die höherdimensionalen Varianten allgemeiner B-Splines gehen wir erst in Abschnitt 6.3 ein, wenn wir über den allgemeinsten Fall, beliebiges p, sprechen.

Mehrdimensionale Bernstein–Polynome

Wie im Eindimensionalen sind die Grundlage für die Definition von Bézier–Patches die Bernstein–Polynome.

Sei $\Delta \subset \mathbb{R}^2$ ein Dreieck mit den Ecken A, B und C. Mit (r,s,t) bezeichnen wir die *baryzentrischen Koordinaten* von Punkten $P \in \mathbb{R}^2$ bezüglich dieser Ecken. Diese Zahlen haben die folgenden Eigenschaften, die wir aus der Linearen Algebra wiederholen wollen.

- $r + s + t = 1$,
- P ist im Inneren von Δ genau dann, wenn $r > 0$, $s > 0$ und $t > 0$,

- r, s und t geben das Verhältnis der Flächen der Dreiecke *PBC*, *PAC* und *PAB* zum Flächeninhalt von Δ an.

Definition 6.2.3.1. *Sei Δ der Einheitssimplex des $\mathbb{R}^2$ (das Dreieck, mit den Ecken $(0,0)$, $(1,0)$ und $(0,1)$), und sei $n \in \mathbb{N}$ beliebig. Dann sind die (zweidimensionalen)* Bernstein–Polynome *vom Grad n definiert durch*

$$B^n_{i,j,k} = \frac{n!}{i!\,j!\,k!} r^i s^j t^k, \quad \text{mit } i+j+k = n.$$

Proposition 6.2.3.2. *1. Es gibt $\frac{(n+1)(n+2)}{2}$ Bernstein–Polynome vom Grad $\leq n$.*

2. *Die Funktionen $B^n_{i,j,k}(r,s,1-r-s)$ bilden eine Basis für den Raum aller Polynome in r und s vom Totalgrad $\leq n$. Diese Basis heißt auch* Bernstein–Basis.

3. *Es gilt die Induktionsformel*

$$B^n_{i,j,k} = rB^{n-1}_{i-1,j,k} + sB^{n-1}_{i,j-1,k} + tB^{n-1}_{i,j,k-1}.$$

4. *Wir haben*

$$\sum_{i+j+k=n} B^n_{i,j,k}(r,s,1-r-s) \equiv 1.$$

5. *Ist $P \in \Delta^0$ im Inneren der Menge Δ mit den Koordinaten (r,s,t), so ist $B^n_{i,j,k} > 0$.*

6. *Man kann das Polynom $B^n_{i,j,k}$ auch schreiben als*

$$B^n_{i,j,k} = \tfrac{1}{n+1}\left((i+1)B^{n+1}_{i+1,j,k} + (j+1)B^{n+1}_{i,j+1,k} + (k+1)B^{n+1}_{i,j,k+1}\right),$$

also den Grad von $B^n_{i,j,k}$ „erhöhen“.

Beweis. 1. Ein leichtes Abzählargument.

2. Es genügt, die lineare Unabhängigkeit zu zeigen. Diese ist aber klar, da der Term kleinsten Grades in $r^i s^j (1-r-s)^k$ gerade das Monom $r^i s^j$ ist.

3. Diese Induktionsformel folgt aus der Induktionsformel für Multinomialkoeffizienten:

$$\frac{n!}{i!\,j!\,k!} = \frac{(n-1)!}{(i-1)!\,(j-1)!\,(k-1)!}\left(\frac{1}{jk} + \frac{1}{ik} + \frac{1}{ij}\right).$$

4. Eine Anwendung des multinomischen Lehrsatzes liefert

$$\sum_{i+j+k=n} B^n_{i,j,k}(r,s,1-r-s) = \sum_{i+j+k=n} \frac{n!}{i!\,j!\,k!} r^i s^j t^k = \\ = (r+s+t)^n = 1^n = 1.$$

5. Klar aus der Definition, da r, s und t alle > 0 sind.

6. Das Resultat folgt auch aus der Rekursionsformel für Multinomialkoeffizienten.

□

Man bemerke noch, daß die Definition der Bernsteinpolynome durch baryzentrische Koordinaten die Polynome eigentlich unabhängig vom Einheitssimplex Δ macht. Man kann durch genau dieselbe Definition die Bernsteinpolynome über einem beliebigen nicht entarteten Dreieck $\tilde{\Delta} \subset \mathbb{R}^2$ erklären.

Beispiel 6.2.3.3. *Für $n = 3$ besteht die Menge der Bernstein–Polynome aus*

$$\begin{array}{ccccccc} & & & s^3 & & & \\ & & 3s^2t & & 3rs^2 & & \\ & 3st^2 & & 6rst & & 3r^2s & \\ t^3 & & 3rt^2 & & 3r^2t & & r^3 \end{array}$$

Dreiecks–Bézier–Patches

Mit Hilfe der Bernstein–Polynome kann man als nächstes 2–dimensionale Bézier–Patches einführen:

Definition 6.2.3.4. *Sei $\Delta \subset \mathbb{R}^2$ ein Dreieck, und seien Punkte $P_{i,j,k}$ mit $i+j+k=n$ im $\mathbb{R}^s$ gegeben. Ein (Dreiecks–) Bézier–Patch ist die Fläche $B : \Delta \to \mathbb{R}^s$, die definiert ist durch*

$$B(r,s,t) = \sum_{i+j+k=n} P_{i,j,k} B^n_{i,j,k}(r,s,t).$$

Beispiel 6.2.3.5. *Für $n = 3$ sieht ein Bézier–Patch etwa wie in Abbildung 6.11 aus. In dieser Darstellung ist auch eine Triangulierung des Kontrollpolyeders angedeutet. Die Punkte $P_{i,j,k}$ sind hervorgehoben.*

Abbildung 6.11: Dreiecks–Bézier–Patch

Die Eigenschaften der Bézier–Patches sind Verallgemeinerungen der Eigenschaften von Bézier–Kurven:

Proposition 6.2.3.6. *1. Der Patch $B(r,s,t)$ geht durch die Punkte $P_{n,0,0}$, $P_{0,n,0}$ und $P_{0,0,n}$.*

2. *Der Rand des Bézier–Patches wird von drei Bézier–Kurven gebildet. Die Kontrollpolygone der j–ten Bézier–Kurve sind all jene P_{i_1,i_2,i_3}, bei denen $i_j = 0$ gilt. Zum Beispiel ist das Kontrollpolygon der ersten Randkurve $(P_{0,0,n}, P_{0,1,n-1}, \ldots, P_{0,n-1,1}, P_{0,0,n})$.*

3. *Der Patch ist innerhalb der konvexen Hülle der Punkte $P_{i,j,k}$.*

4. *Die Tangentialebenen an den Patch in den Endpunkten $P_{n,0,0}$, $P_{0,n,0}$ und $P_{0,0,n}$ werden durch die zwei jeweils benachbarten Punkte aufgespannt. Z.B. geht die Tangentialebene im Punkt $P_{n,0,0}$ durch die Punkte $(P_{n,0,0}, P_{n-1,1,0}, P_{n-1,0,1})$.*

5. *Das Bild unter B einer beliebigen geraden Linie ist eine Kurve vom Grad n.*

Beweis. 1. Das ist klar wegen

$$B^n_{n,0,0}(1,0,0) = B^n_{0,n,0}(0,1,0) = B^n_{0,0,n}(0,0,1) = 1.$$

2. Die Funktionen $B^n_{0,j,l}(0,s,1-s)$ sind genau die eindimensionalen Bernstein–Polynome.

3. Die Linearkombination $\sum_{i+j+k} P_{i,j,k} B^n_{i,j,k}(r,s,t)$ ist wegen der Eigenschaften der Bernstein–Polynome eine konvexe Kombination.

4. Das folgt aus den Eigenschaften für Bézier–Kurven.

5. Durch Nachrechnen.

□

Die letzte Eigenschaft des Satzes ist, wie bereits erwähnt, falsch für Tensorprodukt–Patches. Sie macht, nicht zuletzt, Dreiecks–Bézier–Patches besonders verwendbar.

Einer Verallgemeinerung bedarf es auch, will man einen Dreiecks–Bézier–Patch auswerten:

Auswertung eines Bézier–Patches (De Casteljau Algorithmus)

$P^{(0)}_{i,j,k} = P_{i,j,k}$
for $\ell = 1$ **to** n **do**
$\quad P^{(\ell)}_{i,j,k} = rP^{(\ell-1)}_{i+1,j,k} + sP^{(\ell-1)}_{i,j+1,k} + tP^{(\ell-1)}_{i,j,k+1}$ für $i+j+k = n-\ell$
done
$B(r,s,t) = P^{(n)}_{0,0,0}$.

Wie im Eindimensionalen liefert der Auswertungsalgorithmus eine Möglichkeit, durch Knoteneinfügung den Bézier–Patch zu unterteilen und damit durch einen Polyeder zu approximieren.

Subunterteilungsalgorithmus

1. Verwende Algorithmus 6.2.3 für $(r,s,t) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

2. Der Patch ist geteilt in 3 Teilpatches mit den Kontrollpolyedern

$$\{P^{(i)}_{0,n-i,0}, P^{(i)}_{0,0,n-i}, P^{(0)}_{0,i,n-i} \mid i = 0,\dots,3\} \quad \text{bzw.}$$
$$\{P^{(i)}_{n-i,0,0}, P^{(i)}_{0,0,n-i}, P^{(0)}_{i,0,n-i} \mid i = 0,\dots,3\} \quad \text{bzw.}$$
$$\{P^{(i)}_{0,n-i,0}, P^{(i)}_{n-i,0,0}, P^{(0)}_{i,n-i,0} \mid i = 0,\dots,3\}$$

Wir können den Subunterteilungsalgorithmus iterieren und auf die drei Teilpatches anwenden, um den ursprünglichen Bézier–Patch in 9 und später in 3^r Stücke zu unterteilen. Die dabei entstehenden Flächenstücke haben Kontrollpolyeder, die wir mit Π^i_r für $i = 0,\dots,3^r-1$ bezeichnen wollen. Diese Polyeder haben zusammen $3^r k(k+1)/2+1$ Ecken, unter denen 3^r+1 Punkte des ursprünglichen Flächenstücks sind, und zwar die Punkte $B(r/3^r, s/3^r, t/3^r)$ für $r = 0,\dots,3^r$, $s = 0,\dots,3^r$, $t = 0,\dots,3^r$ und $r+s+t = 1$.

Proposition 6.2.3.7. *Die Folge von Polyedern Π^k_n konvergiert für $n \to \infty$ gleichmäßig gegen das Flächenstück B^* des Bézier–Patches $B(r,s,t)$ mit Konvergenzgeschwindigkeit $\lambda/3^n$, wobei λ eine von n unabhängigen Konstante bezeichnet.*

Der Subunterteilungsalgorithmus erzeugt also die Eckpunkte einer Folge von Polyedern $\Pi_n := \bigcup_{i=0}^{3^n-1} \Pi^i_n$, die mit *exponentieller Geschwindigkeit* gegen den Bézier–Patch B konvergiert. Beachtet man die nur endliche Auflösung von Bildschirmen und Druckern, dann erkennt man, daß nach wenigen Subunterteilungsschritten der Polyeder Π_n von B^* nicht mehr zu unterscheiden ist. Auf diese Weise kann man Bézier–Patches schnell zeichnen und schattieren, da man sie durch eine Vereinigung von Dreiecken approximiert.

Möchte man kompliziertere Flächen aus einer Triangulierung des Parameterbereichs und einer Vereinigung von Dreiecks–Bézier–Patches, deren definierende Dreiecke genau die Elemente der Triangulierung sind, darstellen, so muß man die Glattheit der Verbindung zweier Bézier–Patches untersuchen.

Seien also zwei Bézier–Patches gegeben mit Parameterdreiecken Δ bzw. Δ'. O.B.d.A. können wir annehmen, daß Δ und Δ' eine Kante gemeinsam haben, genau die Kante mit baryzentrischen Koordinaten $r = 0$

bzw. $r' = 0$. $B(r,s,t)$ und $B'(r',s',t')$ seien die beiden Bézier–Patches, und weil man Bernstein–Polynome vom Grad n auf Bernstein–Polynome vom Grad $n+1$ umrechnen kann wegen Proposition 6.2.3.2, können wir annehmen, daß der Grad der beiden Bézier–Patches gleich ist.

Damit die Verbindung der beiden Patches C^0 ist, müssen die Kurven $B(0,s,1-s)$ und $B'(0,s',1-s')$ übereinstimmen, woraus $P_{0,j,k} = P'_{0,j,k}$ für $j+k=n$ folgt.

Für C^1–Glattheit müssen zusätzlich zu dieser Bedingung noch alle Tangentialvektoren an jedem Punkt der Berührungskurve übereinstimmen. Es ist nicht schwer zu zeigen, daß das genau dann der Fall ist, wenn alle Dreiecke der Form $(P_{0,j,n-j}, P_{0,j+1,n-j-1}, P_{1,j,n-j-1})$ koplanar zu den entsprechenden Dreiecken $(P'_{0,j,n-j}, P'_{0,j+1,n-j-1}, P'_{1,j,n-j-1})$ sind.

Möchte man nur, daß die Ebenen, die von den Tangentialvektoren aufgespannt werden, die Tangentialebenen, in jedem Punkt übereinstimmen, begnügt man sich also mit G^1–Glattheit, so hat man etwas größere Freiheit in der Wahl der Punkte, doch die Formeln sind so komplex, daß wir sie hier nicht herleiten wollen.

Es gibt übrigens noch eine Verallgemeinerung der Bézier–Patches, die ähnlich den rationalen Kurven definiert wird (siehe Abschnitt 6.1.4):

Definition 6.2.3.8. *Ein* dreieckiger rationaler Bézier–Patch *vom Grad n zu den Punkten $P_{i,j,k}$ und den Gewichten $w_{i,j,k}$ ist definiert als das Bild des Einheitssimplex Δ unter der Abbildung*

$$R(r,s,t) = \frac{P(r,s,t)}{Q(r,s,t)}$$

mit

$$P(r,s,t) = \sum_{i+j+k=n} P_{i,j,k} w_{i,j,k} B^n_{i,j,k}(r,s,t)$$
$$Q(r,s,t) = \sum_{i+j+k=n} w_{i,j,k} B^n_{i,j,k}(r,s,t).$$

Es gelten wieder ähnliche Eigenschaften wie für Bézier–Patches und rationale Spline–Kurven. Zum ersten liegt der Patch in der konvexen Hülle der $P_{i,j,k}$, zum anderen wirken die Gewichte wieder wie Gummibänder. Je höher das Gewicht $w_{i,j,k}$ desto stärker wird der rationale Patch zum Punkt $P_{i,j,k}$ hingezogen. Wählt man alle $w_{i,j,k} = 1$, so erhält man einen üblichen Dreiecks–Bézier–Patch.

Ein rationaler Bézier–Patch, der sich vom Bézier–Patch aus Abbildung 6.11 nur durch anders gewählte Gewichte unterscheidet, ist in Abbildung 6.12 dargestellt.

Abbildung 6.12: rationaler Bézier–Patch

6.3 Interpolation in höheren Dimensionen

Die meisten Methoden, die wir in den vorhergehenden Abschnitten für $p = 2$ entwickelt haben, lassen sich ohne große Probleme auf $p > 2$ verallgemeinern.

6.3.1 Tensorprodukt–Splines

Die Grundlage für die einfachste Variante sind wieder Flächenstücke, die mehrdimensionale Quader als Parameterbereiche haben, welche wiederum achsenparallel unterteilt werden.

Definition 6.3.1.1. *Sei $I = \prod_{i=1}^{p} [a_i, b_i] \subset \mathbb{R}^p$ ein Quader. Ein auf I definierter (p–dimensionaler)* Tensorprodukt–B-Spline–Patch *ist eine Fläche der Form*

$$B(u) = \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_p=0}^{n_p-1} P_{i_1,\ldots,i_p} \prod_{j=1}^{p} B_{i_j,k_j,\tau_j}(u_j)$$

für Knotensequenzen τ_k, den zugehörigen auf $[a_k, b_k]$ definierten B-Splines und einer Familie von Punkten $P_{i_1,\ldots,i_p} \in \mathbb{R}^s$.

Man kann mehrdimensionale Tensorprodukt–B-Splines wie im Fall $p = 2$ rekursiv in B-Spline–Kurven mit variablen Koeffizienten verwandeln:

$$B(u) = \sum_{i_1=0}^{n_1-1} P_{i_1}(u_2, \ldots, u_p) B_{i_1,k_1,\tau_1}(u_1),$$

wobei

$$P_{i_1}(u_2, \ldots, u_p) = \sum_{i_2=0}^{n_2-1} \cdots \sum_{i_p=0}^{n_p-1} P_{i_1,\ldots,i_p} \prod_{j=2}^{p} B_{i_j,k_j,\tau_j}(u_j)$$

gesetzt wird. Aus dieser Darstellung kann man ablesen, daß die Flächen mit $u_1 = \text{const}$ (oder $u_i = \text{const}$ für i beliebig) $(p-1)$–dimensionale Tensorprodukt–B-Spline–Patches sind. Nimmt man $p-1$ der u_i konstant an, so erhält man eine B-Spline–Kurve.

Wie im Fall $p = 2$ kann man alle Algorithmen des eindimensionalen Falles iterativ anwenden und kann damit die Software, die für Kurven entwickelt wurde, wiederverwenden.

Ähnlich zum Zweidimensionalen sind jedoch auch die Nachteile hochdimensionaler Tensorprodukt–Patches die schwierig zu kontrollierenden Kurven und niedriger dimensionalen Teilflächen in andere als achsenparallele Richtungen, die Beschränkung auf rechteckige Parameterbereiche und der hohe Polynomgrad der zur Beschreibung der Koordinaten und von Schnittflächen benötigt wird.

6.3.2 Polyedersplines

Die stärkste Verallgemeinerung der B-Spline–Kurven ins Mehrdimensionale benötigt weder Quader noch Tetraeder als Parameterbereiche sondern startet mit mehrdimensionalen Polyedern.

Definition 6.3.2.1. *Seien $X_0, \ldots, X_n$ Punkte im $\mathbb{R}^s$, und sei* $\text{conv}(X_0, \ldots, X_n)$ *ihre konvexe Hülle (die kleinste konvexe Menge, die alle Punkte enthält).*

Im folgenden werden wir immer annehmen, daß $\text{vol}(\text{conv}(X_0, \ldots, X_n)) > 0$ gilt, woraus $n \geq s$ folgt.

Definition 6.3.2.2. *Sei $\Delta_n \subset \mathbb{R}^{n+1}$ der n–dimensionale Standardsimplex. Der B-Spline assoziiert zu den Punkten $X_0, \ldots, X_n$ ist die eindeutige Funktion $B(x|X_0, \ldots, X_n)$ auf $\mathbb{R}^s$, die*

$$\int_{\mathbb{R}^s} f(x) B(x|X_0, \ldots, X_n)\, dx = n! \int_{\Delta_n} f(t_0 X_0 + \cdots + t_n X_n)\, dt_1 \wedge \cdots \wedge dt_n$$

für alle $f \in C_c^\infty(\mathbb{R}^s)$ (Funktionen mit kompaktem Träger) erfüllt.

Durch diese Beziehung wird $B(x|X_0, \ldots, X_n)$ als Distribution *erklärt, und es ist nicht trivial zu zeigen, daß für* $\text{vol}(\text{conv}(X_0, \ldots, X_n)) > 0$ *diese Distribution in Wahrheit eine Funktion auf $\mathbb{R}^s$ ist.*

Das Integral auf der rechten Seite der Definition bedeutet, daß über alle konvexen Linearkombinationen der X_i integriert wird, also über die konvexe Hülle $\mathrm{conv}(X_0,\ldots,X_n)$ der X_i.

Definition 6.3.2.3. *Sei $\pi : \mathbb{R}^n \to \mathbb{R}^s$ eine affine Abbildung, und sei $P \subset \mathbb{R}^n$ ein kompakter Polyeder. Der zu P und π assoziierte* polyedrische Spline *(*Polyederspline*) ist die Funktion $B_P(x) : \mathbb{R}^s \to \mathbb{R}^n$, die für alle Funktionen $f \in C_c^\infty(\mathbb{R}^s)$*

$$\int_{\mathbb{R}^s} f(x)B_P(x)\,dx = \int_P f\circ\pi\, dt_1 \wedge \cdots \wedge dt_n$$

erfüllt.

Wie die B-Splines werden auch die polyedrischen Splines zuerst als Distributionen eingeführt. Sie sind genau dann Funktionen, wenn $\pi(P)$ den Vektorraum $\mathbb{R}^s$ erzeugt.

Polyedrische Splines sind Verallgemeinerungen der mehrdimensionalen B-Splines aus Definition 6.3.2.2. Ist nämlich $P = \Delta_n'$ die Projektion von Δ_n in den $\mathbb{R}^n$, so gilt

$$B(x|X_0,\ldots,X_n) = n!B_{\Delta_n'}(x).$$

Proposition 6.3.2.4. *1. Der Träger von $B_P(x)$ ist $\pi(P)$, speziell ist der Träger von $B(x|X_0,\ldots,X_n)$ die konvexe Hülle der X_i.*

2. *Ist π surjektiv, so gilt $B_P(x) = \mathrm{vol}_{n-s}(\pi^{-1}(x)\cap P)$.*

3. *Gilt $n = s$ so berechnet sich $B(x|X_0,\ldots,X_n)$ einfach als*

$$B(x|X_0,\ldots,X_n) = \frac{\chi_{\mathrm{conv}(X_0,\ldots,X_n)}(x)}{\mathrm{vol}(\mathrm{conv}(X_0,\ldots,X_n))}.$$

4. *Die Funktion $B(x|X_0,\ldots,X_n)$ ist stückweise polynomial vom Grad $\le n-s$. Sie ist eine C^{d-2} Funktion, wenn d die kleinste Kardinalität der Teilmengen $Y \subset X = \{X_0,\ldots,X_n\}$ ist, für die $X \setminus Y$ den Raum $\mathbb{R}^s$ nicht affin aufspannt.*

5. *Sei $x \in \mathrm{conv}(X_0,\ldots,X_n)$ und seien λ_i seine baryzentrischen Koordinaten ($x = \sum\lambda_i X_i$, $\sum\lambda_i = 1$). Dann gilt*

$$B(x|X_0,\ldots,X_n) = \frac{n}{n-s}\sum_{j=0}^{n}\lambda_j B(x|X_0,\ldots,\widehat{X_j},\ldots,X_n),$$

wobei $\widehat{X_j}$ bedeute, daß X_j ausgelassen wird. Dies ist die Induktionsformel für höherdimensionale B-Splines. Mit ihrer Hilfe lassen sich mehrdimensionale B-Splines auf eindimensionale zurückführen.

Beweis. [Risler 1992, 4.8] □

Zwei Spezialfälle von polyedrischen Splines sind besonders interessant. Sie werden in den Abschnitten 6.3.3 und 6.3.4 behandelt.

6.3.3 Boxsplines

Hat man achsenparallele Unterteilungen, möchte man aber nicht die Nachteile von Tensorprodukt–Patches in Kauf nehmen, so kann man stattdessen polyedrische Splines verwenden mit $P = I$, dem Einheitswürfel im $\mathbb{R}^n$.

Sei $X = (X_0,\ldots,X_n)$ eine Sequenz von $n+1$ Punkten im $\mathbb{R}^s$, die die folgenden Bedingungen erfüllt:

1. $X_0 = 0$, und $X_i \in \mathbb{Z}^n$,

2. Die X_i erzeugen $\mathbb{R}^s$.

Definition 6.3.3.1. *Sei $\{e_1,\dots,e_n\}$ die kanonische Basis des $\mathbb{R}^n$, $\pi:\mathbb{R}^n\to\mathbb{R}^s$ die lineare Abbildung definiert durch $\pi(e_i)=X_i$, $i=1,\dots,n$. Der Polyeder P sei der Einheitswürfel $P:=[0,1[^n\subset\mathbb{R}^n$.*

Der polyedrische Spline $B_P(x)$ assoziiert zu P und π heißt dann der zu X assoziierte Boxspline *und wird mit $M(x|X)$ bezeichnet.*

Proposition 6.3.3.2. *1. Die Funktion $M(x|X)$ ist ein stückweises Polynom vom Grad $\le n-s$ und sie ist C^{d-1}, wenn $d+1$ die kleinste Kardinalität einer Menge $Y\subset X$ ist, sodaß $X\setminus Y$ nicht $\mathbb{R}^s$ erzeugt.*

2. Es gilt wieder eine Induktionsformel

$$M(x|X)=\frac{1}{n-s}\sum_{i=1}^{n}\Big(t_iM(x|X\setminus X_i)+(1-t_i)M(x-X_i|X\setminus X_i)\Big).$$

3. Sei $\mathcal{S}_X$ die Menge der Funktionen

$$f(x)=\sum_{\alpha\in\mathbb{Z}^s}\lambda_\alpha M(x-\alpha|X),\quad \lambda_\alpha\in\mathbb{R}.$$

Ist $\mathbb{R}^d[x]$ der Vektorraum der mehrdimensionalen Polynome vom Totalgrad $\le d$ (d wie oben), so gilt $\mathbb{R}^d[x]\subset\mathcal{S}_X$. Das bedeutet, daß man jede Funktion $f\in C^\infty(\mathbb{R}^s)\cap L^p(\mathbb{R}^s)$ bis zur Ordnung h^{d+1} durch Funktionen der Form $g(x/h)$ mit $g\in\mathcal{S}_X$ approximieren kann.

4. Die Elemente $M(x-\alpha|X)$ mit $\alpha\in\mathbb{Z}^s$ heißen stark linear unabhängig, *falls aus $\sum_{\alpha\in\mathbb{Z}^s}\lambda_\alpha M(x-\alpha|X)=0$ sofort $\lambda_\alpha=0$ folgt.*

Die $M(x-\alpha|X)$ sind genau dann stark linear unabhängig, falls jede s-elementige Teilmenge von X, die eine Basis von $\mathbb{R}^s$ bildet, auch eine Basis von $\mathbb{Z}^s$ als abelsche Gruppe bildet. Diese Bedingung ist äquivalent zu der Forderung, daß für jede freie Familie $(X_{i_1},\dots,X_{i_s})\subset X$ gilt $|\det(X_{i_j})|=1$.

Hat man eine stark linear unabhängige Familie $M(x-\alpha|X)$, so normiert man die M so um, daß zusätzlich

$$\sum_{\alpha\in\mathbb{Z}^s}M(x-\alpha|X)=1$$

gilt, die M also eine Partition der Eins bilden. Die oft verwendeten Boxsplines erfüllen diese Bedingung.

6.3.4 mehrdimensionale Bézier–Patches

Ein weiterer Spezialfall der polyedrischen Splines kann auch als direkte Verallgemeinerung der Dreiecks–Bézier–Patches angesehen werden.

Definition 6.3.4.1. *Sei $\tilde\Delta_p$ der p–dimensionale volle Einheitssimplex, und seien $(r_0,\dots,r_p)$ die baryzentrischen Koordinaten des Punktes $X\in\tilde\Delta_p$. Dann sind die p–dimensionalen* Bernstein–Polynome *vom Grad n definiert als*

$$B^n_{i_0,\dots,i_p}(x):=\frac{n!}{i_0!\dots i_p!}r_0^{i_0}\dots r_p^{i_p},\quad i_0+\dots+i_p=n.$$

Die mehrdimensionalen Bernstein–Polynome sind genau die polyedrischen B–Splines zu $X_0=0$, $X_i=e_i$, wobei

$$B^n_{i_0,\dots,i_p}(x)=B(x|(X_0)^{i_0+1},\dots,(X_p)^{i_p+1})$$

gilt. $(X_j)^{i_j+1}$ bedeutet dabei, daß in der Menge X der definierenden Punkte X_j mit Vielfachheit i_j+1 vorkommt.

Sei T_p ein p–dimensionaler Hypertetraeder (affines Bild des Einheitssimplex). Ein p–dimensionaler Bézier–Patch *assoziiert zu den Punkten $P_{i_0,\dots,i_p}\in\mathbb{R}^s$ mit $i_0+\dots+i_p=n$ ist eine Fläche der Form*

$$B(x)=\sum_{i_0+\dots+i_p=n}P_{i_0,\dots,i_p}B^n_{i_0,\dots,i_p}(x).$$

Alle Resultate für Dreiecks–Bézier–Patches lassen sich sinngemäß auf p–dimensionale Bézier–Patches verallgemeinern. Möchte man komplizierte p–dimensionale Flächen durch Bézier–Patches darstellen, so muß man den Parameterbereich simplizial approximieren (triangulieren) und auf jedem Tetraeder der Triangulierung einen Bézier–Patch definieren. Die Glattheit der Verbindung zweier Bézier–Patches führt auf analoge Beziehungen wie im Fall $p = 2$, doch sind die Ausdrücke noch komplexer und werden daher hier nicht näher ausgeführt.

6.4 Triangulierungen

Die Approximation von Flächen durch Dreiecke (Simplices) ist Grundlage für viele numerische Verfahren. Abgesehen von mehrdimensionaler Interpolation oder Approximation basieren auch manche Verfahren zur Lösung von Differentialgleichungen auf Triangulierungen.

Im Gegensatz zur algebraischen Topologie sind Triangulierungen in der numerischen Mathematik wirklich als Mengen von Simplices (affinen nicht homöomorphen Bildern des Standardsimplex) verstanden.

Wir werden in diesem Abschnitt nur Triangulierungen behandeln, die zu einer Menge von Punkten assoziiert sind. Das ist keine große Einschränkung, da man in einem durch Dreiecke zu approximierenden Bereich nur Punkte, der Anwendung entsprechend gut verteilt, zu wählen braucht.

Generell sind die Triangulierungen, die zu einer vorgegebenen Menge von Punkten assoziiert sind, nicht eindeutig bestimmt. In numerischen Anwendungen ist es jedoch zumeist vernünftig, als Zusatzbedingung zu fordern, daß die Innenwinkel der Dreiecke möglichst groß sind, daß also keine zu spitzen Winkel (schleifende Schnitte zwischen den Kanten) auftreten.

6.4.1 Voronoï–Diagramm

Seien P_i, $i = 1,\dots,n$ Punkte in $\mathbb{R}^k$. Wir wollen versuchen, die Lage dieser Punkte zu untersuchen.

Definition 6.4.1.1. *Die Punkte $\{P_i | i = 1,\dots,n\}$ sind in* allgemeiner Lage*, wenn keine $k+2$ Punkte auf einer $(k-1)$–Sphäre liegen.*

Der Begriff „in allgemeiner Lage“ kann in der Mathematik viele verschiedene Bedeutungen haben. Manchmal verlangt man, daß es nicht $k+1$ Punkte gibt, durch die eine Hyperebene geht. Manchmal verlangt man auch, daß die Koordinaten nicht rational abhängig sind. In allgemeiner Lage zu sein bedeutet nur, daß sie nicht in spezieller Lage sind, also in einer Lage, die Eigenschaften hat, die durch kleine Änderungen zerstört werden. Für $k = 2$ bedeutet in unserem Fall in allgemeiner Lage zu sein, daß keine vier Punkte auf einem Kreis liegen.

Ab nun nehmen wir im Rest des Kapitels an, daß sich die Punkte P_i in allgemeiner Lage befinden.

Definition 6.4.1.2. *Sei $H(P_i,P_j)$ der abgeschlossene Halbraum, der P_i enthält und dessen Rand die Streckensymmetriehyperebene der Strecke $\overline{P_iP_j}$ ist (also die Normalebene zur Strecke $\overline{P_iP_j}$, die durch den Halbierungspunkt eben dieser Strecke geht).*

Sei

$$V(P_i) = V_i = \bigcap_{j\neq i} H(P_i,P_j).$$

V_i ist die Menge der Punkte $x \in \mathbb{R}^k$ mit $d(x,P_i) \leq d(x,P_j)$ für alle $j \neq i$.

Die Menge, die die V_i und deren paarweise Durchschnitte enthält, heißt das Voronoï–Diagramm *assoziiert zu den Punkten P_i.*

Das Voronoï–Diagramm einiger Punkte der Ebene ist in Abbildung 6.13 dargestellt. Dabei stellt der gestrichelte Polygonzug den Rand der konvexen Hülle $\mathrm{conv}(P_0,\dots,P_n)$ dar.

Proposition 6.4.1.3. *1. V_i ist ein konvexes Polytop der Dimension k, das P_i in seinem Inneren enthält.*

2. Jede Ecke von V_i ist in $k+1$ Polytopen V_j und in $k+1$ Kanten der Dimension 1.

Abbildung 6.13: Voronoï–Diagramm

3. *Das Polytop V_i ist genau dann unbeschränkt, wenn P_i Ecke des Randes der konvexen Hülle* $\operatorname{conv}(P_0,\ldots,P_n)$ *der P_i ist.*

4. *Ist P_s einer der P_i am nächsten liegenden Punkte (unter allen P_j mit $j \neq i$), so ist eine Seitenfläche von V_i in der Streckensymmetriehyperebene von $\overline{P_iP_s}$ enthalten.*

5. *$\bigcup_{i=1}^n V_i = \mathbb{R}^k$, $V_i^\circ \cap V_j^\circ = \emptyset$ für $i \neq j$, wobei V_i° das Innere von V_i bezeichnet.*

Beweis. 1. Als Schnitt endlich vieler konvexer Halbräume hat V_i offensichtlich die behaupteten Eigenschaften.

2. Sei $F_{i,j} = V_i \cap V_j$, und sei S eine Ecke des Voronoï–Diagramms. Dann gibt es r Seitenflächen $F_{i,j}$ der Dimension $n-1$, sodaß $S = \bigcap F_{i,j}$. Natürlich ist $r \geq k$, weil S eine Ecke ist (die Kodimension von S ist k, und der Schnitt von p Hyperebenen, die jeweils Kodimension 1 besitzen, hat Kodimension p). O.B.d.A. können wir annehmen, daß die Punkte P_i so numeriert sind, daß diese r Flächen gerade $F_{0,1}, F_{1,2}, \ldots, F_{r-1,r}$ sind.

   Nachdem jedes $F_{i,j}$ gerade die Menge aller Punkte $x \in \mathbb{R}^k$ ist mit

$$d(x,P_i) = d(x,P_j) \leq d(x,P_s), \quad \text{für } 0 \leq s \leq n,$$

   und weil S der Schnitt aller r Hyperebenen ist, gilt

$$d(S,P_0) = d(S,P_1) = \cdots = d(S,P_r) < d(S,P_s), \quad \text{für alle } s > r.$$

   S ist daher der Mittelpunkt einer Sphäre, die durch die Punkte $P_0,\ldots,P_r$ geht und keinen weiteren Punkt P_s enthält. Die Annahme, daß die P_i in allgemeiner Lage sind, impliziert $r+1 < k+2$ und daher gilt $r = k$.

   Daher ist

$$S \in V_0 \cap \cdots \cap V_k, \quad S \notin V_s \quad \text{für } s > k.$$

   Außerdem gehen alle jene Kanten s_j durch S, die von der Form

$$s_j = \bigcap_{\substack{i=0 \\ j \neq i}}^{k} V_i$$

   sind. Von diesen gibt es ebenfalls $k+1$ verschiedene.

3. Wir beweisen die Äquivalenz der beiden Aussagen
   (a) V_i ist beschränkt
   (b) P_i ist nicht im Rand ∂C der konvexen Hülle $C := \mathrm{conv}(P_0, \ldots, P_n)$.

   Vor dem Beweis bemerke man, daß ein Punkt P genau dann in ∂C liegt, wenn es eine Hyperebene H durch P gibt, sodaß alle P_i auf derselben Seite von H liegen.

   (a)$\Longrightarrow$(b) Ist H eine Hyperebene durch P_i und D ein Halbstrahl durch P_i, der nicht in H enthalten ist (siehe Abbildung 6.14). Nachdem V_i beschränkt ist, trifft D den Rand von V_i in einem Punkt

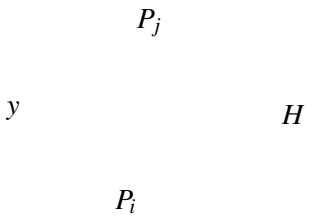


Abbildung 6.14:

   $y \in F_{i,j} = V_i \cap V_j$ für ein j. Daher ist y in der Streckensymmetriehyperebene von $\overline{P_iP_j}$, woraus folgt, daß P_i auf derselben Seite von H wie D ist. Daher kann H keine Hyperebene sein, für die alle P_k auf einer Seite liegen. Daher ist $P_i \notin \partial C$.

   (b)$\Longrightarrow$(a) Sei $P_i \notin \partial C$ und sei wiederum D ein Halbstrahl, der durch P_i geht. Um zu zeigen, daß V_i beschränkt ist, müssen wir zeigen, daß $D \not\subset V_i$. Sei H die Hyperebene, die durch P_i geht und orthogonal auf D steht. Weil P_i nicht in ∂C ist, existiert ein Punkt P_j auf derselben Seite wie D. Der Punkt y aus Abbildung 6.14 liegt dann auf $D \cap V_j$, und daher gilt $D \not\subset V_i$.

4. Sei y der Mittelpunkt von $\overline{P_iP_s}$. Dann gilt

$$d(y,P_i) = d(y,P_s) \leq d(y,P_r), \quad \text{für alle } r,$$

   und daher gilt $y \in V_i \cap V_s$.

5. Das ist klar.

□

Die Triangulierungen, die sich als besonders nützlich für numerische Anwendungen erweisen werden, die Delauney–Triangulierungen, sind in gewisser Weise dual zu den Voronoï–Diagrammen. Diese Tatsache wird in Abschnitt 6.4.3 gezeigt.

6.4.2 Allgemeine Triangulierungen

Sei (T_i) eine endliche Familie von abgeschlossenen k–Simplices, $E = \bigcup T_i$.

Definition 6.4.2.1. *Die T_i sind eine* Triangulierung *von E, wenn $T_i^\circ \cap T_j^\circ = \emptyset$ für $i \neq j$, und für jedes Paar (i,j) mit $i \neq j$ ist der Durchschnitt $T_i \cap T_j$ ein Randsimplex von T_i und T_j.*

Abbildung 6.15 zeigt einige nicht erlaubte Konfigurationen im Fall $k = 2$.

Mit Hilfe von Triangulierungen kann man viele interessante topologische Fakten beweisen. Wir wollen hier jedoch nicht näher darauf eingehen.

Definition 6.4.2.2. *Sei $P = (P_i)$, $i = 0, \ldots, n$ eine Familie von Punkten im $\mathbb{R}^k$. Eine Triangulierung T der konvexen Hülle* $\mathrm{conv}(P_0, \ldots, P_n)$ *heißt* assoziiert zu den Punkten P_i, *wenn die folgenden Bedingungen erfüllt sind:*

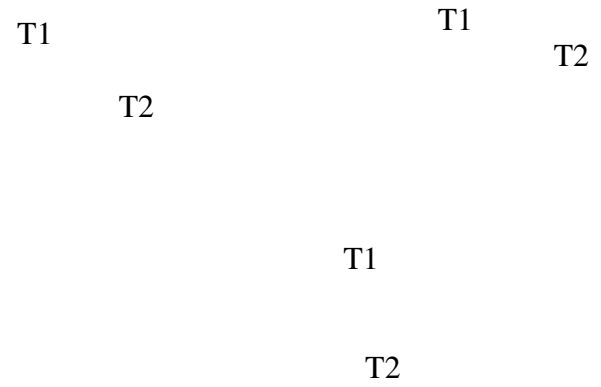


Abbildung 6.15: Nicht erlaubte Triangulierungen

(a) Die Eckenmenge jedes Simplex aus T ist eine Teilmenge von P.

(b) Jeder Punkt P_i ist Ecke mindestens eines Simplex aus T.

Proposition 6.4.2.3. *Sei $k = 2$, und sei T eine Familie von Dreiecken, deren Ecken nur aus der Punktmenge $P = (P_i)$ gewählt werden. In diesem Fall ist T genau dann eine Triangulierung assoziiert zu den Punkten P_i, wenn die folgenden Bedingungen erfüllt sind:*

1. *Jedes Dreieck $P_iP_jP_k$ von T enthält keinen Punkt P_l in seinem Inneren.*
2. *Ist P_iP_j eine Kante der konvexen Hülle $\mathrm{conv}(P_0,\ldots,P_n)$ der P_i, so ist sie Kante genau eines der Dreiecke in T.*
3. *Ist P_iP_j Kante eines Dreiecks von T und keine Kante der konvexen Hülle, so gehört P_iP_j zu genau zwei Dreiecken von T.*
4. *Jeder Punkt P_i ist Ecke mindestens eines Dreiecks von T.*

Beweis. Offensichtlich. □

Proposition 6.4.2.3 läßt sich auf einfache Weise auf den Fall $k > 2$ verallgemeinern, indem man die Worte Dreieck und Kante durch k–Simplex bzw. $(k-1)$–Teilsimplex (Seitenfläche) ersetzt und die Anzahl der Punkte entsprechend anpaßt.

Die einzigen Triangulierungen, die uns interessieren, sind jene, die für numerische Anwendungen besonders verwendbar sind. Diese Delaunay–Triangulierungen werden im nächsten Abschnitt definiert und konstruiert. Sie werden die Eigenschaft haben, daß die Innenwinkel der Dreiecke in gewissem Sinn maximal sind. Sie sind speziell im Fall $k = 2$ besonders brauchbar. Für $k > 2$ können auch bei Delaunay–Triangulierungen „spitze“ Simplices auftreten. Daher wurden für höhere Dimensionen einige spezielle Triangulierungsalgorithmen konstruiert, die jedoch weniger hübsche mathematische Eigenschaften aufweisen und die meist von einer groben Anfangstriangulierung abhängen, die per Hand vorgenommen werden muß.

6.4.3 Delaunay–Triangulierung

In diesem Abschnitt werden wir zu gegebener Eckenmenge $P = (P_i)$ eine spezielle assoziierte Triangulierung konstruieren, die *Delaunay–Triangulierung*.

Definition 6.4.3.1. *Die* Delaunay–Triangulierung *der konvexen Hülle einer gegebenen Punktemenge P ist eine Menge $\mathscr{D}$ von k–Simplices, deren Ecken nur aus Punkten in P bestehen, und die dual ist zum Voronoï–Diagramm der P_i im folgenden Sinn:*

P_iP_j ist eine Kante eines Simplices in $\mathscr{D}$ genau dann, wenn $V_i \cap V_j \neq \emptyset$.
$P_iP_jP_k$ ist eine Seitenfläche der Dimension 2 genau dann, wenn $V_i \cap V_j \cap V_k \neq \emptyset$.
usw...

In Abbildung 6.16 sind eine Delaunay–Triangulierung und das zugehörige Voronoï–Diagramm dargestellt.

Abbildung 6.16: Delaunay–Triangulierung

Proposition 6.4.3.2. *Bei obiger Notation betrachten wir den Fall $n \geq k$, und die Punkte P_i seien in allgemeiner Lage in $\mathbb{R}^k$. Dann ist die Delaunay–Triangulierung eine Triangulierung, die zu den P_i assoziiert ist. Die Delaunay–Triangulierung ist unter allen zu den P_i assoziierten Triangulierungen durch die folgende Eigenschaft charakterisiert: Jede $(k-1)$–Sphäre, die einem Simplex in $\mathscr{D}$ umschrieben ist, enthält keinen Punkt P_ℓ in ihrem Inneren.*

Beweis. Wir werden den Beweis nur im Fall $k = 2$ durchführen, da die Verallgemeinerung für $k > 2$ nur Notations–Schwierigkeiten mit sich bringt.

Jede Ecke des Voronoï–Diagramms V ist der Durchschnitt von 3 der V_i, und daher ist wirklich jedes Element von $\mathscr{D}$ ein Dreieck. Zuerst müssen wir zeigen, daß $\mathscr{D}$ wirklich eine Triangulierung ist, die zu den P_i assoziiert ist.

Sei dazu $\Delta = P_iP_jP_k$ ein Dreieck in $\mathscr{D}$. Dieses Dreieck entspricht laut Definition einer Ecke des Voronoï–Diagramms, für die $\{S\} = V_i \cap V_j \cap V_k$ gilt. S hat dann von allen drei Ecken gleichen Abstand und ist daher der Umkreismittelpunkt von Δ. Es gilt daher $d(S,P_i) = d(S,P_j) = d(S,P_k) < d(S,P_\ell)$ für $\ell \neq i,j,k$. Daher enthält der Umkreis des Dreiecks (und damit das Dreieck selbst) keinen weiteren Punkt P_ℓ.

Sei P_iP_j eine Kante des Rands der konvexen Hülle $\mathrm{conv}(P_0,\ldots,P_n)$ der P_i. Dann sind V_i und V_j unbeschränkt (Proposition 6.4.1.3) Ist $n \geq 3$, so ist $V_i \cap V_j$ ein Halbstrahl, der in der Streckensymmetrale von $\overline{P_iP_j}$ liegt und daher genau eine Ecke S des Voronoï–Diagramms enthält. Deshalb existiert genau ein Dreieck in $\mathscr{D}$, das P_iP_j enthält, weil nach Dualität die Dreiecke von $\mathscr{D}$, die P_iP_j enthalten, den Ecken von V entsprechen, die auf $V_i \cap V_j$ liegen.

Liegt P_iP_j nicht auf dem Rand der konvexen Hülle, dann ist V_i oder V_j beschränkt, also wird $V_i \cap V_j$ von zwei Ecken von V begrenzt. Nach Dualität existieren also genau zwei Dreiecke, die P_iP_j enthalten.

Sei $P_i \in P$ beliebig. V_i ist nichtleer, also gibt es j mit $V_i \cap V_j \neq \emptyset$, weil $n \geq k$ gilt und $\mathbb{R}^2 = \bigcup V_i$. P_iP_j ist daher nach Definition Kante eines Dreiecks in $\mathscr{D}$, das P_i (und P_j) als Ecken enthält.

Aus der Charakterisierung von Triangulierungen (Proposition 6.4.2.3) folgt, daß $\mathscr{D}$ eine zu den P_i assoziierte Triangulierung ist.

Von der zweiten Behauptung haben wir die eine Richtung schon bewiesen. Sind umgekehrt für eine beliebige Triangulierung T drei Punkte P_i, P_j und P_k gegeben, sodaß der Umkreis des Dreiecks $P_iP_jP_k$ keinen weiteren Punkt P_ℓ enthält, dann ist $S = V_i \cap V_j \cap V_k$ nach Definition eine Ecke von V, und daher ist $P_iP_jP_k$ in $\mathscr{D}$. Erfüllt also T die Bedingung, dann ist jedes Dreieck von T auch Dreieck von $\mathscr{D}$. □

Wir wollen im einfachsten Fall $k = 2$, $n = 4$ die Delaunay–Triangulierung studieren. Im folgenden werden wir sehen, daß dieser Fall in gewissem Sinn schon der allgemeinste ist.

Lemma 6.4.3.3. *Seien A, B, C und D vier Punkte in allgemeiner Lage im $\mathbb{R}^2$, die ein konvexes Viereck bilden. Dann existieren zwei Triangulierungen von* conv(A,B,C,D)*, die Delaunay–Triangulierung $\mathscr{D}$ und eine weitere Triangulierung T. Der Übergang von $\mathscr{D}$ zu T erfolgt durch „Austausch der Diagonalen" (siehe Abbildung 6.17).*

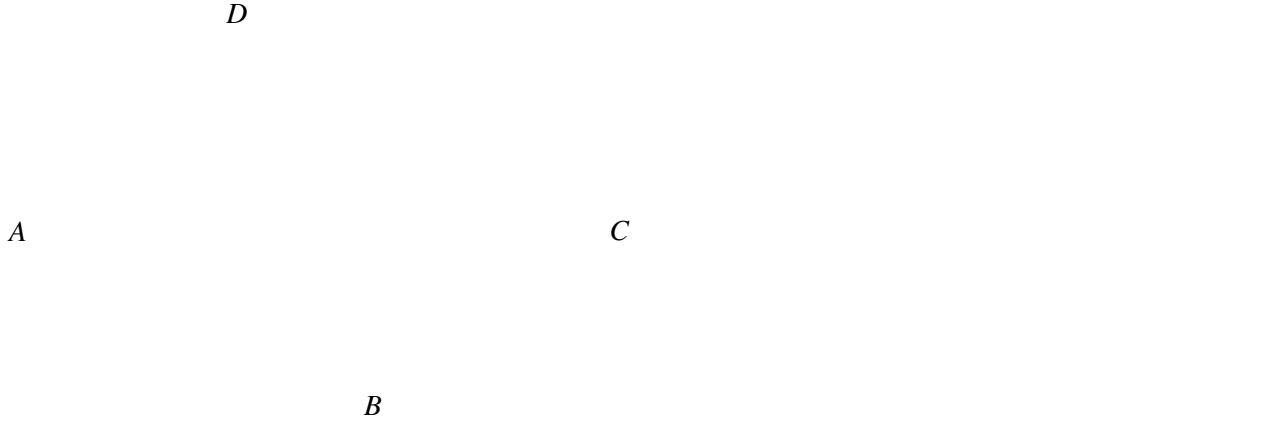


Abbildung 6.17: $\mathscr{D}$ (durchgezogen) und T (strichliert)

Die Triangulierung $\mathscr{D}$ ist diejenige, die die untere Schranke der Innenwinkel der Dreiecke maximiert.

Beweis. Wie $\mathscr{D}$ und T aussehen ist klar. Die Maximalitätseigenschaft folgt aus elementaren Formeln der ebenen Geometrie und der Umkreiseigenschaft der Delaunay–Triangulierung. □

Definition 6.4.3.4. *Seien $P = (P_i)$ wieder $n+1$ Punkte in allgemeiner Lage in der Ebene. Eine Triangulierung, die zu den P_i assoziiert ist, heißt* lokal Delaunay*, wenn die Einschränkung auf jedes Paar von Dreiecken, die eine gemeinsame Kante haben, eine Delaunay–Triangulierung ist.*

Das folgende Resultat zeigt, daß man Delaunay–Triangulierungen lokal charakterisieren kann.

Proposition 6.4.3.5. *Sei T eine zu den Punkten P_i (in allgemeiner Lage) assoziierte Triangulierung. Dann ist T genau dann die Delaunay–Triangulierung, wenn T lokal Delaunay ist.*

Beweis. Daß eine Delaunay–Triangulierung auch lokal Delaunay ist, ist klar. Beweisen wir also die Umkehrung. Sei T eine lokale Delaunay–Triangulierung, die nicht gleich $\mathscr{D}$ ist. Dann ist die Menge E aller Dreiecke von T, die nicht in $\mathscr{D}$ sind, nichtleer. Sei Δ ein Dreieck, das eine Kante hat, die zu keinem anderen Dreieck von E gehört (so ein Dreieck existiert klarerweise immer). Diese Kante ist dann einem Dreieck von T und einem Dreieck von $\mathscr{D}$ gemeinsam (das folgt aus den Fakten, daß T und $\mathscr{D}$ Triangulierungen sind und daß Kanten auf dem Rand der konvexen Hülle der P_i zu genau einem Dreieck gehören).

Sei o.B.d.A. P_1P_2 eine gemeinsame Kante von einem Dreieck in T und einem Dreieck in $\mathscr{D}$. Sei $P_1P_2P_3$ das Dreieck in T und nehmen wir an, daß dieses Dreieck nicht in $\mathscr{D}$ liegt.

Sei R der Eckpunkt eines Dreiecks P_1P_2R in $\mathscr{D}$, der auf derselben Seite von P_1P_2 liegt wie P_3. R liegt dann im Umkreis von $P_1P_2P_3$, weil P_3 nicht im Umkreis von P_1P_2R liegen kann ($\mathscr{D}$ ist die Delaunay–Triangulierung!).

Nachdem aber T eine Triangulierung ist assoziiert zu den P_i ist, kann R nicht im Dreieck $P_1P_2P_3$ liegen. O.B.d.A. sind R und P_2 auf derselben Seite der Kante P_1P_3. $P_4 \neq P_2$ sei ein weiterer Punkt, sodaß $P_1P_3P_4$ ein Dreieck von T ist. Dieser muß existieren, weil P_1P_3 nicht zum Rand der konvexen Hülle der P_i gehört und daher in T genau zwei Dreiecke existieren, die P_1P_3 enthalten. P_4 ist nicht im Inneren des Umkreises von $P_1P_2P_3$, weil T lokal Delaunay ist. Daher ist $R \neq P_4$ und R ist nicht im Dreieck $P_1P_3P_4$.

Jetzt haben wir dieselbe Situation wie zuvor erreicht bloß haben wir das Dreieck $P_1P_2P_3$ durch $P_1P_3P_4$ ersetzt. Auf dieselbe Weise wie zuvor können wir einen Punkt P_5 finden, der $\neq R$ ist, so daß $P_3P_4P_5$ ein Dreieck von T ist, in dem R nicht liegt.

Führen wir diesen Prozeß beliebig oft durch, so können wir schließen, weil nur endlich viele Punkte P_i existieren, daß $R \notin P$ gilt, was ein Widerspruch zur Annahme ist, daß $\mathscr{D}$ die Delaunay–Triangulierung der P_i ist. □

Aus Lemma 6.4.3.3 und Proposition 6.4.3.5 folgt, daß $\mathscr{D}$ das Maximum der unteren Schranke der Innenwinkel aller Dreiecke erreicht. Es treten also bei der Delaunay–Triangulierung die wenigsten schleifenden (damit numerisch instabilen) Schnitte auf.

Die Eigenschaft, das Maximum der unteren Schranke der Innenwinkel zu erreichen, ist *nicht charakteristisch* für die Delaunay–Triangulierung. Es kann auch andere Triangulierungen geben, die diese Eigenschaft aufweisen. Umgekehrt erreicht $\mathscr{D}$ im allgemeinen auch nicht das Minimum der oberen Schranke der Innenwinkel.

In [Hermeline 1982] wurde folgender Algorithmus angegeben, wie man die Delaunay–Triangulierung (nicht nur in $\mathbb{R}^2$) einer Menge von Punkten P_i konstruieren kann.

Konstruktion der Delaunay–Triangulierung

$\mathscr{D} = \{P_0P_1P_2\}$
for $\ell = 3$ **to** n **do**
  if $P_\ell \in \text{conv}(P_0,\ldots,P_{\ell-1})$ **then**
    $\Sigma := \{$Dreiecke von $\mathscr{D}$, deren Umkreis P_ℓ enthält$\}$
    $K := \{$Kanten, die zu genau einem Dreieck in Σ, gehören$\}$
    $\mathscr{D} = (\mathscr{D} \setminus \Sigma) \cup \bigcup_{e \in K} \{P_\ell e\}$
  else
    if P_ℓ ist in keinem Umkreis eines Dreiecks von $\mathscr{D}$ **then**
      $K := \{$Kanten von $\text{conv}(P_0,\ldots,P_{\ell-1})$, die P_ℓ von $P_0,\ldots,P_{\ell-1}$ trennen$\}$
      $\mathscr{D} = \mathscr{D} \cup \bigcup_{e \in K} \{P_\ell e\}$
    else
      $\Sigma := \{$Dreiecke von $\mathscr{D}$, deren Umkreis P_ℓ enthält$\}$
      $K := \{$Kanten von $\text{conv}(P_0,\ldots,P_{\ell-1})$, die nicht zu Dreiecken in Σ gehören
          und P_ℓ von den P_i trennen$\}$
      $B := \{$Kanten, die zu genau einem Dreieck in Σ gehören
          und P_ℓ nicht von den anderen P_i trennen$\}$
      $\mathscr{D} = (\mathscr{D} \setminus \Sigma) \cup \bigcup_{e \in K \cup B} \{P_\ell e\}$
    endif
  endif
$\ell = \ell + 1$
done

Am Ende enthält $\mathscr{D}$ die Dreiecke der Delaunay–Triangulierung der Punkte P_i. Im Fall $k > 2$ setzt man im ersten Schritt $\mathscr{D} = \{P_0P_1 \ldots P_k\}$ und beginnt die Schleife bei $\ell = k+1$. Im weiteren Algorithmus muß man

nur mehr die Begriffe Dreieck, Kante und Umkreis durch k–Simplex, $(k-1)$–Seitensimplex und umschriebene $(k-1)$–Sphäre ersetzen.

Im Fall $k = 2$ hat der Algorithmus Komplexität $O(n^2)$. Das ist nicht ganz optimal, denn es existiert ein Algorithmus, der das Problem in $O(n \log n)$ Schritten löst. Unglücklicherweise ist dieser schnellere Algorithmus sehr kompliziert und läßt sich nicht in höhere Dimensionen verallgemeinern. $O(n \log n)$ ist das schnellste, was man erwarten kann, da man mit Hilfe einer Delaunay–Triangulierung das Voronoï–Diagramm und damit die konvexe Hülle von n Punkten in der Ebene konstruieren kann. Kann man auf der anderen Seite die konvexe Hülle von n Punkten in der Ebene konstruieren, kann man auch Zahlen sortieren, was bekanntermaßen nicht schneller als in $O(n \log n)$ Schritten machbar ist.

6.5 Radiale Basisfunktionen

Eine weitere Verallgemeinerung der Polynominterpolation, die in allen Dimensionen brauchbar ist, und besonders bei Oberflächendarstellungen, die später verzerrt werden sollen, eingesetzt wird, sind die *radialen Basisfunktionen*.

6.6 Software

Software zur mehrdimensionalen Interpolation ist in den größeren Bibliotheken enthalten.

Tensorprodukt–Splines

Diese kann man in $\mathbb{R}^2$ bzw. $\mathbb{R}^3$ mit Hilfe der Programme `IMSL/MATH-LIBRARY/bs2in` (`IMSL/MATH-LIBRARY/bs3in`) bzw. `NAG/e01daf` bestimmen. Auch Auswertungsfunktionen stehen dort zur Verfügung.

Andere Interpolationsmethoden:

Für mehrdimensionale polyedrische Splines und Bézier–Patches existiert Software in einigen CAD–Programmen. Diese Software ist leider nicht frei verfügbar und muß in Lizenz zugekauft werden.

Es existieren jedoch verwandte Verfahren, die stückweise polynomial interpolieren und Triangulierungen verwenden, etwa `IMSL/MATH-LIBRARY/surf`, `NETLIB/TOMS/526` oder auch `NAG/e01saf`.

7 Statistik, Stochastik

In diesem Abschnitt wollen wir einige Grundlagen aufarbeiten für ein Prinzip, das in der numerischen Mathematik große Tradition besitzt:

> Ist ein Problem zu kompliziert bzw. zu rechenintensiv, um es zu lösen, erfinde einen Algorithmus, der auf stochastischen Prinzipien beruht und das Problem mit hoher Wahrscheinlichkeit löst.

Solche *stochastische Algorithmen* haben meist den Vorteil, daß sie in viel kürzerer Zeit gute Ergebnisse liefern als *deterministische Methoden*. Auf der anderen Seite besitzen sie jedoch den Nachteil, daß Konvergenzaussagen und Fehlerschranken nur mit hoher Wahrscheinlichkeit gelten, und man daher diese stochastischen Verfahren einige Male anwenden muß, um Vergleiche anzustellen.

Werden die Anwendungsprobleme so kompliziert, daß selbst die Entwicklung stochastischer Algorithmen zu aufwendig wird, kann man versuchen einige Aussagen über das Modell dadurch zu erhalten, indem man es im Computer simuliert. Dabei wird das mathematische Modell nicht gelöst und es existieren auch keine Aussagen über Fehlerschranken, doch ähnlich einem physikalischen Experiment kann man durch die Simulation einige Eigenschaften des Modells herausfinden (meist genügt das dem Anwender).

Für alle stochastischen Verfahren benötigt man *Zufallszahlen*, also Folgen von Zahlen, die durch einen Zufallsprozeß erzeugt werden, welcher bestimmtes statistisches Verhalten aufweist. Nach einer kurzen Wiederholung der grundlegenden Begriffe der Wahrscheinlichkeitstheorie werden wir uns mit der Erzeugung und dem Testen von Zufallszahlen beschäftigen.

Viele Resultate über die Theorie der Zufallszahlengeneratoren sind aus [Knuth 1969, II, Chapter 3] entnommen.

7.1 Grundlagen

Eine große Anzahl natürlicher, ökonomischer oder technischer Vorgänge sind vom Zufall beeinflußt. Um auch über zufallsbeeinflußte Zusammenhänge quantitative Aussagen zu machen, zeigt es sich, daß man eine große Zahl solcher Vorgänge unter gleichbleibenden Bedingungen beobachten muß.

7.1.1 Wahrscheinlichkeitsraum

Definition 7.1.1.1. *Ein* Zufallsexperiment *oder ein* Versuch *ist ein Vorgang, bei dem verschiedene Ausgänge möglich sind ohne daß man im vorhinein sagen kann, welches Resultat eintreten wird. Ein bedeutsamer Gegenstand der Untersuchungen ist die Menge der möglichen einander ausschließenden Ausgänge eines Versuchs, die Menge E der* elementaren Ereignisse (Elementarereignisse).

Definition 7.1.1.2. *Für einen vorgegebenen Versuch sei E die Menge der elementaren Ereignisse. Jede Teilmenge $A \subseteq E$ heißt ein* Ereignis. *Ein Ereignis A tritt genau dann ein, wenn eines der Elementarereignisse, aus denen A besteht, eintritt. Die speziellen Teilmengen E und $\emptyset$ heißen das* sichere *bzw.* unmögliche Ereignis.

Zwei Ereignisse A_1 und A_2 heißen unvereinbar, *falls $A_1 \cap A_2 = \emptyset$.*

Ist A ein Ereignis, so heißt $\bar{A} = E \setminus A$ das zu A komplementäre Ereignis.

Sei A_i, $i \in I$ eine Menge von Ereignissen, so nennt man die Ereignisse

$$\bigcup_{i \in I} A_i$$

die Summe der Ereignisse A_i *und*

$$\bigcap_{i\in I} A_i$$

das Produkt der Ereignisse A_i.

Um Wahrscheinlichkeitstheorie zu treiben, schränkt man sich auf eine Teilmenge aller möglichen Ereignisse Σ ein. Man verlangt, daß Σ eine σ–Algebra bildet. Im folgenden sei $\mathscr{P}E$ die Potenzmenge von E.

Definition 7.1.1.3. *Sei E eine Menge und $\Sigma \subseteq \mathscr{P}E$. Σ heißt σ–Algebra, wenn*

1. $E \in \Sigma$,
2. *Für jedes $A \in \Sigma$ liegt auch $\bar{A} \in \Sigma$,*
3. *Sei I eine endliche oder abzählbar unendliche Indexmenge, und seien $A_i \in \Sigma$, $i \in I$. Dann gilt*

$$\bigcup_{i\in I} A_i \in \Sigma.$$

Aus den Voraussetzungen folgt unter anderem sofort, daß $\emptyset \in \Sigma$ und daß auch der Durchschnitt höchstens abzählbar vieler Elemente von Σ wieder in Σ liegt.

Definition 7.1.1.4. *Ist E die Menge der elementaren Ereignisse zu einem Versuch. Die σ–Algebra Σ von Ereignissen heißt die Menge der* zufälligen Ereignisse.

Eine in der Wahrscheinlichkeitstheorie wichtige σ–Algebra ist die σ–Algebra der Borelmengen, die kleinste σ–Algebra, die alle Intervalle (Quader) im $\mathbb{R}^n$ enthält.

Jedem zufälligen Ereignis möchte man ein Maß für die Wahrscheinlichkeit des Eintretens zuordnen. Wie man dieses Maß wählen muß, hat Kolmogorov in seinen berühmten Axiomen zusammengefaßt:

Definition 7.1.1.5 (Die Kolmogorovschen Axiome)**.** *Gegeben sei eine Funktion $\mathbb{P} : \Sigma \to [0,1]$, das Wahrscheinlichkeitsmaß, die die folgenden Eigenschaften hat:*

1. $\mathbb{P}(E) = 1$,
2. *Sind A_i, $i \in I$ höchstens abzählbar unendlich viele paarweise unvereinbare Ereignisse, d.h. $A_j \cap A_k = \emptyset$. Dann gilt*

$$\mathbb{P}(\bigcup_{i\in I} A_i) = \sum_{i\in I} \mathbb{P}(A_i).$$

Das Wahrscheinlichkeitsmaß ist also σ–additiv.

Insbesondere folgen aus diesen Definitionen $\mathbb{P}(\emptyset) = 0$ und $\mathbb{P}(\bar{A}) = 1 - \mathbb{P}(A)$. Für beliebige, nicht paarweise unvereinbare Ereignisse A_i gilt immer noch

$$\mathbb{P}(\bigcup_{i\in I} A_i) \leq \sum_{i\in I} \mathbb{P}(A_i).$$

Definition 7.1.1.6. *Eine Menge E zusammen mit einer σ–Algebra A von zufälligen Ereignissen und einem Wahrscheinlichkeitsmaß $\mathbb{P}$ heißt* Wahrscheinlichkeitsraum *und wird mit dem Tripel $(E, A, \mathbb{P})$ bezeichnet.*

Beispiel 7.1.1.7. *Ein Beispiel für einen Wahrscheinlichkeitsraum erhält man, indem man $E = [0,1]^n \subseteq \mathbb{R}^n$ betrachtet. Sei $B(E)$ die σ–Algebra der Borelmengen in E, und sei $\tilde{\lambda}$ das Wahrscheinlichkeitsmaß, das jedem Teilwürfel $A = \prod_i [a_i, b_i]$ die Wahrscheinlichkeit*

$$\tilde{\lambda}(A) = \prod_{i=1}^{n} (b_i - a_i)$$

zuweist.

Fügt man zu $B(E)$ alle jene Mengen hinzu, die Teilmengen von $\tilde{\lambda}$–Nullmengen (Mengen $X \subset E$ mit $\tilde{\lambda}(X) = 0$) sind, so erzeugen diese im Verein mit $B(E)$ wieder eine σ–Algebra $\mathscr{L}(E)$, die Menge der Lebesgue–meßbaren Teilmengen von E. Setzt man das Maß $\tilde{\lambda}$ auf $\mathscr{L}(E)$ fort, indem man allen Teilmengen von Nullmengen das Maß 0 zuweist, so erhält man wieder ein Wahrscheinlichkeitsmaß λ, das Lebesgue–Maß. Der Maßraum $(E, \mathscr{L}(E), \lambda)$ ist ein bedeutender Raum in der Mathematik und wird in Kapitel 8 eine zentrale Rolle spielen.

Meist ändert sich die Wahrscheinlichkeit für das Eintreten eines Ereignisses A, wenn bekannt ist, daß ein anderes Ereignis B bereits eingetreten ist. Diese geänderte Wahrscheinlichkeit wird mit $\mathbb{P}(A|B)$ bezeichnet und heißt die *bedingte Wahrscheinlichkeit für A unter der Bedingung B*.

Allgemein definiert man den Ausdruck $\mathbb{P}(A|B)$ als

$$\mathbb{P}(A|B) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}, \quad \mathbb{P}(B) \neq 0.$$

Definition 7.1.1.8. *Zwei Ereignisse A und B heißen* voneinander unabhängig*, wenn $\mathbb{P}(A|B) = \mathbb{P}(A)$ gilt.*

Formt man die Formeln für die bedingte Wahrscheinlichkeit um, so erhält man das bekannte Multiplikationstheorem: Für zwei unabhängige Ereignisse A und B gilt

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B).$$

Definition 7.1.1.9. *n Ereignisse $A_1, \ldots, A_n$ heißen* insgesamt unabhängig*, falls für jedes $m \leq n$ und jedes m–Tupel $i_1, \ldots, i_m$ mit $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ gilt:*

$$\mathbb{P}(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_m}) = \mathbb{P}(A_{i_1})\mathbb{P}(A_{i_2}) \ldots \mathbb{P}(A_{i_m}).$$

Sie heißen paarweise unabhängig*, falls für alle i und j mit $i \neq j$*

$$\mathbb{P}(A_i \cap A_j) = \mathbb{P}(A_i)\mathbb{P}(A_j)$$

erfüllt ist.

Aus insgesamter Unabhängigkeit folgt paarweise Unabhängigkeit aber *nicht umgekehrt*!

Proposition 7.1.1.10 (Satz von der totalen Wahrscheinlichkeit)**.** *Es seien A_i, $i = 1, \ldots, n$ paarweise unvereinbare zufällige Ereignisse mit*

$$E = \bigcup_{i=1}^{n} A_i.$$

Für ein beliebiges weiteres Ereignis B gilt dann

$$\mathbb{P}(B) = \sum_{i=1}^{n} \mathbb{P}(A_i)\mathbb{P}(B|A_i).$$

Proposition 7.1.1.11 (Formel von Bayes)**.** *Unter den Voraussetzungen von Proposition 7.1.1.10 gilt*

$$\mathbb{P}(A_i|B) = \frac{\mathbb{P}(A_i)\mathbb{P}(B|A_i)}{\sum_{j=1}^{n} \mathbb{P}(A_j)\mathbb{P}(B|A_j)}.$$

7.1.2 Verteilungen

Eine reelle Variable, die abhängig vom eintretenden Ereignis eines Versuchs verschiedene Werte annimmt heißt *Zufallsgröße* oder *Zufallsvariable*.

Definition 7.1.2.1. *Sei X eine Zufallsvariable. Die* Verteilungsfunktion $F(x)$ *der Zufallsgröße X ist definiert als die Funktion*

$$F(x) = \mathbb{P}(X < x).$$

Eine Zufallsvariable wird durch ihre Verteilungsfunktion vollständig charakterisiert. Mit Hilfe dieser Funktion kann sofort angegeben werden, wann eine Zufallsvariable in ein vorgegebenes Intervall $[a,b[$ fällt:

$$\mathbb{P}(a \leq X < b) = F(b) - F(a).$$

Proposition 7.1.2.2. *Die Verteilungsfunktion $F(x)$ einer reellen Zufallsvariablen hat folgende Eigenschaften:*

1. $\lim_{x\to+\infty} F(x) = 1$, $\lim_{x\to-\infty} F(x) = 0$,
2. $F(x)$ *ist monoton steigend,*
3. $F(x)$ *ist linksseitig stetig.*

Obwohl Verteilungsfunktionen eine recht komplizierte Struktur aufweisen können, sind für die Praxis vor allem zwei Typen von Verteilungsfunktionen relevant: stetige Verteilungen und Treppenfunktionen.

Diskrete Zufallsvariable

Definition 7.1.2.3. *Eine Zufallsvariable X heißt* diskret, *wenn* $|E| \leq \aleph_0$.
Ihre Verteilungsfunktion ist charakterisiert durch die Werte

$$p_i := \mathbb{P}(X = x_i), \quad \text{für } x_i \in E \text{ und } i = 1,2,\ldots,$$

und es gilt

$$F(x) = \sum_{x_i<x} p_i,$$

F ist also eine Treppenfunktion. Die p_i erfüllen natürlich $p_i > 0$ und

$$\sum_i p_i = 1.$$

Die folgenden Verteilungen sind die wichtigsten Beispiele für diskrete Verteilungsfunktionen.

Binomialverteilung Ein Versuch werde n–mal wiederholt, die Resultate der Wiederholungen seien voneinander unabhängig, und in jedem Versuch solle ein Ereignis A eintreten oder nicht eintreten können. Das Eintreten von A im Einzelversuch erfolge mit Wahrscheinlichkeit p.

X sei die Anzahl wie oft A in diesen n Versuchen eingetroffen ist. E ist in diesem Fall gleich $\{0,\ldots,n\}$. Für die Wahrscheinlichkeiten $p_k = \mathbb{P}(X = k)$ gilt:

$$p_k = \binom{n}{k} p^k q^{n-k}, \quad q = 1-p, \quad k = 0,\ldots,n.$$

Eine Zufallsvariable heißt (n,p)–*binomialverteilt*, wenn sie die Werte $\{0,\ldots,n\}$ mit den oben beschriebenen Wahrscheinlichkeiten p_k annimmt.

Hypergeometrische Verteilung Wenn aus einer Urne, in der N Kugeln (M weiße und $N-M$ schwarze) liegen, ohne Zurücklegen n Kugeln gezogen werden, so ist die Wahrscheinlichkeit dafür, daß darunter k weiße sind

$$p_k = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}, \quad k = 0, \ldots, n.$$

Eine Zufallsvariable X, die ganzzahlige Werte von 0 bis n annehmen kann, heißt (N,M,n)*–hypergeometrisch verteilt*, wenn sie den Wert i mit Wahrscheinlichkeit p_i annimmt.

Ist $N \gg n$, so kann die hypergeometrische Verteilung durch eine Binomialverteilung mit $p = M/N$ approximiert werden.

Poissonverteilung Geht es darum, etwa Pannenhäufigkeiten abzuschätzen, tritt meist die Poissonverteilung auf. Eine Zufallsvariable X heißt λ*–poissonverteilt*, wenn sie alle Werte $k \in \mathbb{N}_0$ annimmt mit Wahrscheinlichkeit

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, \ldots$$

Die Poissonverteilung ist auch eine gute Näherung für die Binomialverteilung, wenn n sehr groß ist. In diesem Fall muß der Parameter $\lambda = np$ gewählt werden.

(diskrete) Gleichverteilung Eine Zufallsvariable X heißt *(diskret) gleichverteilt* auf der Menge $\{0, \ldots, d-1\}$, wenn jedes Element dieser Menge mit Wahrscheinlichkeit $p = 1/d$ angenommen wird.

Stetige Zufallsvariable

Definition 7.1.2.4. *Eine Zufallsvariable X heißt* stetig*, wenn die Verteilungsfunktion F die Form*

$$F(x) = \int_{-\infty}^{x} f(t)\,dt$$

hat. Die dabei auftretende Funktion f heißt die Dichte der Verteilung*. Für diese Dichtefunktion gilt*

$$\int_{-\infty}^{+\infty} f(t)\,dt = 1.$$

Für stetige Verteilungsfunktionen gilt $\mathbb{P}(X = a) = 0$ für jedes $a \in \mathbb{R}$.

Die wichtigsten stetigen Verteilungsfunktionen sind:

Gleichverteilung Eine Zufallsvariable X ist *gleichverteilt* auf dem Intervall $[a,b]$, wenn sie die Dichtefunktion

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a,b], \\ 0 & \text{sonst} \end{cases}$$

besitzt.

Normalverteilung Die bekannteste Verteilung ist die Gauß–Verteilung oder Normalverteilung. Eine Zufallsgröße X ist $N(\mu,\sigma^2)$–normalverteilt, falls ihre Dichtefunktion die Gestalt

$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

hat.

Um die Werte der Verteilungsfunktion zu bestimmen, transformiert man durch Substitution auf die Form

$$\operatorname{erf}(y) = \int_{-\infty}^{y} \varphi(t)\,dt$$

mit

$$\varphi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}.$$

erf heißt die Gaußsche Fehlerfunktion und ist seit vielen Jahrzehnten tabelliert. Die Bedeutung der Normalverteilung liegt vor allem im zentralen Grenzwertsatz 7.1.4.6 begründet.

Exponentialverteilung Bei Lebensdauerabschätzungen tritt häufig die *Exponentialverteilung* mit folgender Dichte auf:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0. \end{cases}$$

Der Parameter λ muß dabei größer als Null gewählt werden.

7.1.3 Statistische Größen

Definition 7.1.3.1. *Sei X eine (diskrete oder stetige) Zufallsgröße mit Verteilungsfunktion F. Die Zahl*

$$\alpha_k := \int_{-\infty}^{+\infty} x^k \, dF(x)$$

heißt das k–te Moment *der Verteilung F. Löst man in den beiden Spezialfällen (diskrete bzw. stetige Verteilungen) das obige Stieltjes–Integral auf, so findet man die folgenden Formeln*

$$\alpha_k = \sum_{x_i \in E} p_i x_i^k$$

$$\alpha_k = \int_{-\infty}^{+\infty} x^k f(x) \, dx.$$

Von besonderer Bedeutung ist das erste Moment α_1. Es wird mit $\mathbb{E}X$ bezeichnet und heißt der Erwartungswert *von X.*

Proposition 7.1.3.2. *Für den Erwartungswert einer Zufallsvariablen gelten folgende Rechenregeln:*

1. $\mathbb{E}a = a$ *für jede Konstante a,*
2. $\mathbb{E}(X_1 + X_2) = \mathbb{E}(X_1) + \mathbb{E}(X_2)$,

Beispiel 7.1.3.3. *Der Erwartungswert der vorgestellten Verteilungen ist jeweils:*

(n,p)**–Binomialverteilung** $\mathbb{E}X = np$,

(N,M,n)**–Hypergeometrische Verteilung** $\mathbb{E}X = n\frac{M}{N}$,

λ**–Poissonverteilung** $\mathbb{E}X = \lambda$,

diskrete Gleichverteilung auf $\{0,\ldots,d-1\}$ $\mathbb{E}X = \frac{d-1}{2}$,

Gleichverteilung auf (a,b) $\mathbb{E}X = \frac{a+b}{2}$,

$N(\mu,\sigma^2)$**–Normalverteilung** $\mathbb{E}X = \mu$,

λ**–Exponentialverteilung** $\mathbb{E}X = \frac{1}{\lambda}$.

Definition 7.1.3.4. *Sei X eine (diskrete oder stetige) Zufallsgröße mit Verteilungsfunktion F. Die Zahl*

$$\beta_k := \int_{-\infty}^{+\infty} (x - \mathbb{E}X)^k \, dF(x)$$

*heißt das k–*te zentrale Moment *der Verteilung F. Löst man wieder in den Spezialfällen das Stieltjes–Integral auf, so findet man*

$$\beta_k = \sum_{x_i \in E} p_i (x_i - \mathbb{E}X)^k$$
$$\beta_k = \int_{-\infty}^{+\infty} (x - \mathbb{E}X)^k f(x)\, dx.$$

Häufig verwendet wird das zweite zentrale Moment β_2. Es wird mit $\mathbb{V}X$ bezeichnet und heißt die Varianz *von X. Die Quadratwurzel aus dieser Zahl heißt die* Standardabweichung $\sigma = \sqrt{\mathbb{V}X}$ *von X.*

Beispiel 7.1.3.5. *Für die vorgestellten Verteilungsfunktionen lassen sich natürlich auch die Varianzen berechnen:*

(n,p)**–Binomialverteilung** $\mathbb{V}X = np(1-p)$,

(N,M,n)**–Hypergeometrische Verteilung** $\mathbb{V}X = n\frac{M}{N}\frac{N-n}{N-1}\left(1-\frac{M}{N}\right)$,

λ**–Poissonverteilung** $\mathbb{V}X = \lambda$,

diskrete Gleichverteilung auf $\{0,\ldots,d-1\}$ $\mathbb{V}X = \frac{d^2-1}{12}$,

Gleichverteilung auf (a,b) $\mathbb{V}X = \frac{(b-a)^2}{12}$,

$N(\mu,\sigma^2)$**–Normalverteilung** $\mathbb{V}X = \sigma^2$,

λ**–Exponentialverteilung** $\mathbb{V}X = \frac{1}{\lambda^2}$.

Alle Definitionen kann man auf vektorwertige Zufallsvariablen verallgemeinern. Dabei muß nur die Definition der Verteilungsfunktion geändert werden zu:

$$F(x_1,\ldots,x_n) = \mathbb{P}(X_1 < x_1,\ldots,X_n < x_n).$$

Alle Summen und Integrale werden zu *n*–fach iterierten Integralen.

7.1.4 Grenzwertsätze

Definition 7.1.4.1. *Eine Folge $(X_n)_n$ von Zufallsvariablen heißt* konvergent in Wahrscheinlichkeit *oder* konvergent mit Wahrscheinlichkeit 1 *gegen die Zufallsgröße X, wenn für alle $\varepsilon > 0$ gilt:* $\lim_{n\to\infty} \mathbb{P}(|X_n - X| < \varepsilon) = 1$.

Die Folge heißt fast sicher konvergent, *falls* $\mathbb{P}(\lim_{n\to\infty} X_n = X) = 1$ *gilt.*

Definition 7.1.4.2. *Man sagt, eine Folge von Zufallszahlen genügt dem* schwachen Gesetz der großen Zahlen, *falls die Folge*

$$Z_n = \frac{1}{n}\sum_{k=1}^{n} X_k - \frac{1}{n}\sum_{k=1}^{n} \mathbb{E}X_k$$

mit Wahrscheinlichkeit 1 *gegen* 0 *konvergiert.*

Man sagt sie genügt dem starken Gesetz der großen Zahlen, *falls Z_n fast sicher gegen* 0 *konvergiert.*

Theorem 7.1.4.3 (Satz von Tschebyscheff)**.** *Sind die Varianzen der Glieder einer Folge* paarweise unabhängiger *Zufallsvariablen gleichmäßig beschränkt ($\mathbb{V}X_k \le C$ für alle k), so genügt die Folge dem schwachen Gesetz der großen Zahlen.*

Theorem 7.1.4.4 (Satz von Kolmogorov)**.** *Ist für die Folge* $(X_n)_n$ *von* (insgesamt) unabhängigen *Zufallsvariablen die Bedingung*

$$\sum_{n=1}^{\infty} \frac{\mathbb{V}X_n}{n^2} < \infty$$

erfüllt, so genügt sie dem starken Gesetz der großen Zahlen.

Theorem 7.1.4.5 (Lokaler Grenzwertsatz)**.** *Ist die Zufallsvariable* X_n *binomialverteilt mit den Parametern* (n,p)*. Ist* $p_k^{(n)}$ *die Wahrscheinlichkeit für das Eintreffen von k bei der Zufallsvariable* X_n*, so gilt*

$$\lim_{n\to\infty} \frac{p_k^{(n)}}{\frac{1}{\sqrt{2\pi}\sqrt{np(1-p)}} e^{-x^2/2}} = 1$$

für

$$x = \frac{k-np}{\sqrt{np(1-p)}}.$$

Anders ausgedrückt ist eine binomialverteilte Zufallsgröße asymptotisch normalverteilt mit den Parametern $\mu = np$ *und* $\sigma^2 = np(1-p)$*.*

In Abbildung 7.1 sind eine $(27, \frac{1}{3})$*–Binomialverteilung und eine* $N(9,6)$*–Normalverteilung dargestellt.*

Abbildung 7.1: Binomial– und Normalverteilung

.

Theorem 7.1.4.6 (Zentraler Grenzwertsatz)**.** *Sei* $(X_n)_n$ *eine Folge* unabhängiger *Zufallsgrößen. Wir definieren die Folge*

$$Z_n = \frac{\sum_{i=1}^n (X_i - \mathbb{E}X_i)}{\sqrt{\sum_{i=1}^n \mathbb{V}X_i}}$$

der normierten, zentrierten Summen. $\Phi_n(x)$ *seien die Verteilungsfunktionen der* Z_n*,* $F_k(x)$ *die der* X_k*.*

Genau dann, wenn die Lindebergsche Bedingung

$$\lim_{n\to\infty} \frac{1}{C_n^2} \sum_{k=1}^n \int_{|x-\mathbb{E}X_k|>\varepsilon C_n} (x-\mathbb{E}X_k)^2 \, dF_k(x) = 0, \quad \forall \varepsilon > 0,$$

mit $C_n^2 = \sum_{i=1}^n \mathbb{V}X_i$ *erfüllt ist, gilt*

$$\lim_{n\to\infty} \Phi_n(x) = \operatorname{erf}(x).$$

Ist also eine Zufallsgröße die Summe einer sehr großen Anzahl unabhängiger Terme, von denen jeder nur einen kleinen Beitrag liefert, so ist diese Zufallsgröße annähernd normalverteilt.

Die Lindebergsche Bedingung ist übrigens jedenfalls dann erfüllt, wenn alle X_k dieselbe Verteilung $F_k(x) = F(x)$ besitzen, und wenn diese endlichen Erwartungswert und endliche Varianz aufweist.

7.2 Zufallszahlen

Zufallszahlen, also zufällig gewählte Zahlen, werden nicht nur in mathematischen Anwendungen an wichtiger Stelle benötigt.

Simulation Um Simulationen umfangreicher Phänomene im Computer realistisch zu gestalten, benötigt man Zufallszahlen. Ob dabei physikalische, chemische oder Wirtschaftsprozesse simuliert werden, spielt keine Rolle.

Sampling Bei Modellen, in denen stochastische Phänomene vorkommen (etwa optimale Kaufstrategien für Aktien), kann man oft nicht alle möglichen Fälle untersuchen. In diesem Zusammenhang kann es hilfreich sein, gute zufällig zusammengestellte Testbeispiele zu untersuchen.

Numerische Mathematik Die berühmten Monte-Carlo–Methoden, um verschiedene komplexe numerische Probleme zu bearbeiten, allen voran die Lösung mehrdimensionaler Integrale, basieren auf Zufallszahlen; und gerade in diesen mathematischen Anwendungen ist die *Güte* der verwendeten Zufallszahlen entscheidend für die Brauchbarkeit der Ergebnisse. Neben Monte-Carlo–Integration existieren noch andere Anwendungen wie „Simulated Annealing“ oder „Threshold Accepting“ zur globalen Optimierung.

Programmierung und Testen Die Effizienz und Stabilität von Computerprogrammen kann unter anderem mit Daten getestet werden, die auf zufällige Art und Weise bestimmt werden. Ein berühmtes Programm, das die Absturzsicherheit von UNIX–Betriebssystemen testet, ist `crashme`, das zufällige Befehlsfolgen erzeugt.

Entscheidungsfindung Wie werden wohl die Noten für die Numerik 2–Prüfung bestimmt? Spaß beiseite: In der Spieltheorie gibt es Resultate, daß in manchen Situationen zufällig getroffene Entscheidungen zu optimaler Strategie führen.

Entspannung Würfeln, Kartenspielen, Roulette, Lotto und andere Spiele, die auf Zufall beruhen, sind heutzutage zu einer Industrie geworden, die jedes Jahr Billionen Schilling umsetzt.

Nachdem wir jetzt eine lange Liste von Anwendungen für Zufallszahlen angegeben haben, ist es an der Zeit darüber zu reden, was Zufallszahlen sind. In gewisser Weise existiert so etwas wie eine Zufallszahl nicht. Ist etwa 42 eine Zufallszahl? Was soll diese Frage überhaupt bedeuten?

Wenn wir von Zufallszahlen sprechen, meinen wir *Folgen unabhängiger* Zufallszahlen mit vorgegebener *Verteilung*. Die vage Bedeutung davon ist, daß wir die Folgenglieder unabhängig von einander durch einen Zufallsprozeß erhalten haben. Unabhängig bedeutet dabei, daß jede einzelne Zahl absolut nichts mit den anderen Gliedern der Folge zu tun hat. Vorgegebene Verteilung bedeutet, daß die Wahrscheinlichkeit, daß eine bestimmte Zahl in einen gegebenen Bereich fällt, durch eine Wahrscheinlichkeitsverteilung beschrieben wird.

Die wichtigsten, weil am häufigsten auftretenden, Zufallszahlen sind *gleichverteilte Zufallszahlen*, also Folgen von Zufallszahlen bezüglich einer Gleichverteilung. Diese werden wir in Abschnitt 7.2.1 untersuchen. Anders verteilte Zufallsfolgen können meist ohne großen Aufwand aus gleichverteilten berechnet werden, wie wir in Abschnitt 7.2.3 sehen werden.

Die ersten Wissenschafter, die Zufallszahlen benötigten, zogen numerierte Bälle aus Urnen, würfelten oder deckten Karten auf. L. H. C. Tippet veröffentlichte 1927 eine Tabelle, die über 40 000 Zufallsziffern enthielt. Er hatte sie zufällig aus Finanzberichten entnommen.

Seit dieser Zeit wurde eine Anzahl spezieller Maschinen konstruiert, die auf mechanischem Weg Zufallszahlen erzeugen sollten. Die erste solche Maschine wurde von M. G. Kendall und B. Babington-Smith 1939 gebaut und erzeugte eine Tabelle von 100000 Zufallsziffern. Eine sehr lange in weiten Bereichen verwendete Tabelle von einer Million Zufallsziffern publizierte die RAND Corporation im Jahr 1955; sie war ebenfalls auf mechanischem Weg erzeugt worden.

Andere berühmte Maschinen, die Zufallszahlen erzeugen, sind ERNIE, eine Maschine, die seit Jahrzehnten die Zahlen der British Premium Savings Bonds Lottery erzeugt, und die Maschine, die jeden Mittwoch und Sonntag die Gewinnzahlen des „Lotto 6 aus 45“ der Österreichischen Lotterie bestimmt.

Alle diese Verfahren sind jedoch unbrauchbar, wenn man in Computerprogrammen Zufallszahlen benötigt. Darum ist bald nach Einführung der Computer nach effizienten Algorithmen zur Zufallszahlenerzeugung geforscht worden. Die so erzeugten Zahlenreihen sind nicht wirklich zufällig erzeugt, sondern werden auf deterministischem Weg bestimmt. Daher sind sie nicht eigentlich Zufallszahlen, was dazu geführt hat, daß diese Zahlenreihen mitunter als *Pseudozufallszahlen* bezeichnet werden.

Hat man eine Reihe von Zufallszahlen mit einem Algorithmus erzeugt, müssen Methoden erfunden werden, um zu testen ob die Zahlenreihe „zufällig genug“ erscheint oder ob sich Regelmäßigkeiten erkennen lassen. Diese statistischen Tests werden in Abschnitt 7.2.4 näher erläutert.

7.2.1 Gleichverteilte Zufallszahlen

Bevor wir uns in Abschnitt 7.2.2 der Erzeugung gleichverteilter Zufallszahlen zuwenden, wollen wir einige Eigenschaften solcher Zahlenfolgen untersuchen.

Angenommen wir wollten Folgen von Zufallsziffern generieren, also Folgen auf der Menge $\{0,1,\ldots,9\}$ gleichverteilter Zahlen. In einer solchen Folge tritt an jeder Stelle jede der Ziffern mit Wahrscheinlichkeit $\frac{1}{10}$ auf. Man kann also erwarten, daß in einer Folge von einer Million Zufallszahlen jede der Ziffern etwa 100000 Mal auftritt. Doch eine „echte“ Zufallsfolge (etwa durch Würfeln bestimmt) wird kaum genau 100000 Nullen, 100000 Einsen,... enthalten, die Wahrscheinlichkeit dafür ist sogar ausgesprochen gering; eine *Folge* solcher Folgen wird im Mittel diese Eigenschaft haben.

Jede einzelne Folge von Ziffern ist genauso wahrscheinlich wie jede andere, genauso wahrscheinlich wie die Folge, die aus einer Million Nullen besteht. Haben wir einen echten Zufallsprozeß, und haben wir bereits 999999 Nullen in unserer Folge, so ist immer noch die Wahrscheinlichkeit, daß auch an der millionsten Stelle eine Null steht genau $\frac{1}{10}$.

Eine gute abstrakte Definition, was zufällig bedeutet, ist schwer zu geben, doch wir werden es versuchen nachdem wir zuerst die historisch ersten Methoden beleuchtet haben, Zufallszahlen auf algorithmischem Weg zu erzeugen.

1946 hat John von Neumann, wer sonst, sich den ersten Zugang zur Erzeugung gleichverteilter ganzer Zufallszahlen im Bereich $[0,10^{10}-1]$ überlegt, die *Quadratmitten–Methode*. Er verwendete das Quadrat der vorangegangenen Zufallszahl und entnahm die mittleren Ziffern zur Definition der nächsten Zahl:

$$5772156649 \longrightarrow 33317792380594909291 \longrightarrow 7923805949$$

Unglücklicherweise ist diese Methode eine schlechte Quelle für Zufallszahlen. Die Gefahr ist, daß die Folge die Tendenz hat, in kurze Zyklen sich wiederholender Zahlen zu degenerieren. Null ist offensichtlich ein möglicher Zyklus der Länge 1 und $6100 \longrightarrow 2100 \longrightarrow 4100 \longrightarrow 8100$ ist ein möglicher Zyklus der Länge 4.

Zufallszahlen erzeugende Algorithmen der Form

$$X_{n+1} = f(X_n)$$

generieren zwangsläufig für jeden möglichen Startwert X_0 eine zyklische Folge, wenn der Zahlenbereich für die X_i endlich ist. Dies stört jedoch nicht, falls die Zykluslänge viel größer ist als die Anzahl der benötigten Zufallszahlen und die Folge „zufällig genug“ aussieht.

Bevor wir uns mit der algorithmischen Erzeugung von Zufallszahlen beschäftigen, wollen wir noch eine kurze Diskussion über mögliche Definitionen des Begriffs „Zufallszahl“ führen, eine Abhandlung, die ebenfalls [Knuth 1969, II, 3.5] entnommen ist.

Die mathematische Theorie hat lange Zeit wohlwissend vermieden, eine quantitative Definition für den Begriff „zufällig“ zu geben. Erst nach der Mitte des zwanzigsten Jahrhunderts haben Mathematiker versucht, den Begriff zu fassen.

> *D. H. Lehmer (1951):* „Eine Zufallsfolge ist ein vager Begriff, der die Idee einer Folge faßt, in der jedes Glied unvorhersagbar für den Uninformierten ist und deren Zahlen eine Reihe statistischer Tests bestehen, die einerseits aus traditionellen Gründen gewählt andererseits durch die Anwendungen bestimmt werden.“

> *J. N. Franklin (1962):* „Eine Folge von Zahlen $U_i \in [0,1[$ ist zufällig, wenn sie jede Eigenschaft hat, die jede unendliche Folge unabhängiger Stichproben von auf $[0,1[$ gleichverteilten Zufallsvariablen hat.“

Dabei verallgemeinert Franklins Aussage Lehmers Definition dahingehend, daß die Folge *alle* denkbaren statistischen Tests besteht. Diese nicht ganz präzise Definition hat in einer vernünftigen Interpretation leider die Konsequenz, daß so etwas wie eine Zufallsfolge im Sinn von Franklin nicht existiert. Daher werden wir Lehmers Definition mathematisch exakter fassen und etwas verschärfen.

Eine Klasse von Folgen, die im folgenden Kapitel 8 über mehrdimensionale Integration zentrale Bedeutung haben wird, ist auch Grundlage für die Diskussion des Begriffs Zufallsfolge.

Im folgenden sei $S := (U_n)_n$ eine Folge von Zahlen aus $[0,1[$.

Definition 7.2.1.1. 1. *Die Folge S heißt* gleichverteilt*, wenn für alle Paare von Zahlen $u, v \in [0,1[$ mit $v > u$ gilt*

$$\lim_{n\to\infty} \frac{A(n, \in [u,v])}{n} = v - u,$$

wobei $A(n, \in M)$ die Anzahl der Folgenglieder $u_0, \ldots, u_{n-1}$ ist, die in M liegen:

$$A(n, \in M) := |\{u_i \in S \,|\, i \in \{0, \ldots, n-1\} \wedge u_i \in M\}|.$$

2. *Sei $Q(n)$ eine Aussage über die ganze Zahl n und die Folge S. Wir sagen $\mathbb{P}(Q_n) = \lambda$, wenn $\lim_{n\to\infty} A(n, Q(n))/n = \lambda$, wobei wieder $A(n, Q(n))$ die Anzahl der ganzen Zahlen ist, sodaß $Q(j)$ erfüllt für $0 \le j < n$.*

Aus zwei gleichverteilten Folgen V_i und W_i kann man leicht eine Folge $U_i = (\frac{1}{2}V_0, \frac{1}{2} + \frac{1}{2}W_0, \frac{1}{2}V_1, \frac{1}{2} + \frac{1}{2}W_1, \ldots)$ definieren, die auch gleichverteilt ist aber die offensichtliche Eigenschaft hat, daß alle geraden Folgenglieder $\le \frac{1}{2}$ und alle ungeraden Folgenglieder $\ge \frac{1}{2}$ sind, eine Folge also, die man keinesfalls als Zufallsfolge bezeichnen würde.

Darum betrachtet man folgende natürliche Verallgemeinerung gleichverteilter Folgen:

Definition 7.2.1.2. *Eine Folge S heißt k–*verteilt*, falls*

$$\mathbb{P}(u_1 \le U_n < v_1, \ldots, u_k \le U_{n+k-1} < v_k) = (v_1 - u_1) \ldots (v_k - u_k)$$

für beliebige Wahlen reeller Zahlen $u_i, v_i \in [0,1[$ mit $u_i < v_i$.

Eine gleichverteilte Folge ist 1–verteilt, und klarerweise ist jede k–verteilte Folge auch $(k-1)$–verteilt, da man etwa $u_1 = 0$ und $v_1 = 1$ wählen kann. Daher können wir zu jeder Folge das größte k zu finden versuchen, für das die Folge k–verteilt ist.

Definition 7.2.1.3. *Eine Folge S heißt ∞–*verteilt*, wenn sie k–verteilt ist für jede natürliche Zahl k.*

Ähnlich kann man vorgehen, wenn man statt $[0,1[$–Folgen S Folgen X von ganzen Zahlen betrachtet. Eine Folge $X=(X_n)_n$ ganzer Zahlen heißt b–adische Folge, falls alle Folgenglieder $X_n \in \{0,\dots,b-1\}$ liegen. Eine b–adische Zahl ist eine geordnete endliche Sequenz ganzer Zahlen $x_1x_2\dots x_k$ mit $x_i \in \{0,\dots,b-1\}$ (identifizierbar mit der Darstellung in Basis b).

Definition 7.2.1.4. *Eine b–adische Folge heißt k–*verteilt*, wenn*

$$\mathbb{P}(X_nX_{n+1}\dots X_{n+k-1}=x_1x_2\dots x_k)=\tfrac{1}{b^k}$$

*für alle b–adischen Zahlen $x_1x_2\dots x_k$ gilt. Sie heißt ∞–*verteilt*, wenn sie k–verteilt ist für jede natürliche Zahl k.*

Ist $S=(U_i)$ eine k–verteilte $[0,1[$–Folge, dann ist für jedes $b\in\mathbb{N}$ die Folge $X_i=\lfloor bU_i\rfloor$ eine k–verteilte b–adische Folge. Es gilt genauer, daß eine $[0,1[$–Folge $(U_n)_n$ genau dann k–verteilt ist, wenn die b–adische Folge $(X_n)_n$ auch k–verteilt ist für jedes $b\in\mathbb{N}$.

Die Frage ist: „Sind ∞–verteilte Folgen Zufallsfolgen?“ Vielleicht, doch zuerst wollen wir einige wichtige Eigenschaften k–verteilter Folgen auflisten.

Theorem 7.2.1.5. *Sei $S=(U_n)_n$ eine k–verteilte $[0,1[$–Folge, und sei $f:\mathbb{R}^k\to\mathbb{R}$ Riemann–integrierbar. Dann gilt*

$$\lim_{n\to\infty}\tfrac{1}{n}\sum_{0\le j<n} f(U_j,U_{j+1},\dots,U_{j+k-1})=\int_0^1\dots\int_0^1 f(x_1,\dots,x_k)\,dx_1\dots dx_k.$$

Bezüglich einiger Tests, die in Abschnitt 7.2.4 besprochen werden, kann man theoretische Aussagen machen.

Proposition 7.2.1.6. *Eine k–verteilte $[0,1[$–Folge erfüllt den Permutationstest von Ordnung k (Abschnitt 7.2.4).*

Auch der Serien–Korrelationstest (Abschnitt 7.2.4) ist erfüllt:

Proposition 7.2.1.7. *Für eine $(k+1)$–verteilte Folge geht der Korrelationskoeffizient zwischen U_n und U_{n+k} gegen Null:*

$$\lim_{n\to\infty}\frac{\frac{1}{n}\sum U_jU_{j+k}-\left(\frac{1}{n}\sum U_j\right)\left(\frac{1}{n}\sum U_{j+k}\right)}{\sqrt{\left(\frac{1}{n}\sum U_j^2-\left(\frac{1}{n}\sum U_j\right)^2\right)\left(\frac{1}{n}\sum U_{j+k}^2-\left(\frac{1}{n}\sum U_{j+k}\right)^2\right)}}=0.$$

Eine interessante Frage für die Untersuchung von Zufallsfolgen ist natürlich, was für Eigenschaften Teilfolgen der Form $(U_{mi})_i$ haben.

Definition 7.2.1.8. *Eine $[0,1[$–Folge $(U_n)_n$ heißt (m,k)–verteilt, falls*

$$\mathbb{P}(u_1\le U_{mn+j}<v_1,u_2\le U_{mn+j+1}<v_2,\dots,u_k\le U_{mn+j+k-1}<v_k)=(v_1-u_1)\dots(v_k-u_k)$$

für alle Wahlen von reellen Zahlen $u_r,v_r\in[0,1]$ mit $u_r<v_r$ und alle ganzen Zahlen j mit $0\le j<m$.

Natürlich ist jede (m,k)–verteilte Folge $(m,k-1)$–verteilt. Es gilt klarerweise auch, daß jede (m,k)–verteilte Folge (d,k)–verteilt ist für jeden Teiler d von m.

Analog kann man b–adische (m,k)–verteilte Folgen einführen. Für beide Folgenkonzepte gilt das folgende Resultat:

Theorem 7.2.1.9 (Niven, Zuckerman)**.** *Eine ∞–verteilte Folge ist (m,k)–verteilt für alle ganzen Zahlen m und k.*

Das ist erstaunlich und etwas unerwartet. Es könnte als Hinweis verstanden werden, daß ∞–verteilte Folgen genau das sind, was wir suchen. Die Existenz solcher Folgen wird durch den folgenden Satz gewährleistet:

Theorem 7.2.1.10 (Franklin). *Die* $[0,1[$*–Folge* $(U_n)_n$ *mit*

$$U_n = (\theta^n \mod 1)$$

ist ∞*–verteilt für fast jede reelle Zahl* $\theta > 1$*. Es ist notwendig, daß* θ *eine transzendente Zahl ist.*

Sind nun ∞–verteilte Folgen Zufallsfolgen? Sind sie das, was wir uns darunter vorstellen?

Nun, verpaßt man dem Raum aller $[0,1[$–Folgen irgendein vernünftiges Wahrscheinlichkeitsmaß, dann ist eine Folge ∞–verteilt mit Wahrscheinlichkeit 1. Nachdem eine Zufallsfolge mindestens die Eigenschaft haben sollte, ∞–verteilt zu sein, wollen wir versuchen, eher noch stärkere Eigenschaften zu verlangen.

Untersuchen wir zuerst die Definition von Franklin vom Beginn dieses Abschnitts und versuchen wir eine mathematische Formulierung:

> *Franklin (mathematisch):* Eine $[0,1[$–Folge $(U_n)_n$ heißt „zufällig“ wenn folgendes gilt: Ist P eine Eigenschaft, sodaß $P(V_n)$ mit Wahrscheinlichkeit 1 gilt für eine Folge $(V_n)_n$ unabhängiger Zufallsexperimente für $[0,1[$–gleichverteilte Zufallsvariable, dann gilt $P(U_n)$.

Sei $x \in [0,1[$, und sei $P_x(V_n)$ die Eigenschaft, daß alle Elemente von $(V_n)_n$ verschieden von x sind. Dann ist P_x wahr mit Wahrscheinlichkeit 1, daher ist laut Franklins Definition keine Folge zufällig, die x enthält, also existiert keine einzige zufällige Folge im Sinne Franklins.

Das Problem mit ∞–verteilten Folgen ist die folgende: Gegeben eine Zufallsfolge, dann sollte auch die Teilfolge $(U_{n^2})_n$ eine Zufallsfolge sein. Dies ist jedoch für ∞–verteilte Folgen im allgemeinen nicht der Fall. Man kann nämlich eine beliebige ∞–verteilte Folge $(V_n)_n$ wählen und die Folge $(U_n)_n$ wie folgt bilden: $U_i = V_i$ für $i \neq n^2$ für alle n und $U_{n^2} = 0$ für n beliebig. Diese Folge ist wieder ∞–verteilt, weil die Konvergenzeigenschaft der Zählfunktion nicht verändert wird.

Man möchte aber auf Zufälligkeitseigenschaften für Teilfolgen nicht verzichten. Leider kann man nicht fordern, daß jede Teilfolge wieder ∞–verteilt ist, da man beweisen kann, daß keine solche Folge existiert. Auf der anderen Seite hat Lehmer gefordert, daß „jedes Folgenglied für den Uninformierten unvorhersagbar“ sein soll. Es soll also keinen Algorithmus geben, mit dem man aus den bekannten Folgengliedern das nächste bestimmen kann.

Definition 7.2.1.11. 1. *Eine* Teilsequenzregel $\mathfrak{R}$ *ist eine unendliche Folge von Funktionen* $(f_n(x_1,\ldots,x_n))_n$ *so, daß die n–te Funktion von n Variablen abhängt und daß der Wert jeder Funktion* 0 *oder* 1 *ist. Eine Teilsequenzregel* $\mathfrak{R}$ *definiert eine Teilsequenz* $(U_n)|_{\mathfrak{R}}$ *folgendermaßen: Das k–te Glied der Folge* $(U_n)_n$ *ist in der Teilsequenz, wenn* $f_k(U_0,\ldots,U_{k-1}) = 1$ *gilt. Man bemerke, daß Teilsequenzen einer Folge, die mit einer Teilsequenzregel erzeugt werden, keine Teilfolgen sein müssen, da sie nicht unendlich viele Elemente zu enthalten brauchen.*

2. *Eine Teilsequenzregel heißt* berechenbar, *falls es einen effektiven Algorithmus (einen endlichen Algorithmus) gibt, der den Wert von* $f_n(x_1,\ldots,x_n)$ *berechnet, wenn n,* $x_1,\ldots,x_n$ *gegeben sind.*

Leider gibt es Probleme, berechenbare Teilfolgenregeln anzugeben für $[0,1[$–Folgen. So ist es etwa nicht leicht mit einem endlichen Algorithmus festzustellen, ob eine gegebene Zahl $x > \frac{1}{3}$ ist, wenn z.B. eine Dezimalentwicklung von x vorliegt. Aus diesem Grund schränken wir uns auf b–adische Folgen ein.

Definition 7.2.1.12. *Eine b–adische Folge* $(X_n)_n$ *heißt* teilfolgen–∞–verteilt, *falls jede Teilfolge, die mit Hilfe einer berechenbaren Teilsequenzregel erzeugt wird,* 1*–verteilt ist.*

Daraus folgt unter anderem, daß $(X_n)_n$ und alle Teilfolgen, die man mit Hilfe berechenbarer Teilsequenzregeln erzeugen kann, sogar ∞–verteilt sind, weil das mit Hilfe endlicher Algorithmen überprüfbar ist.

Definition 7.2.1.13. *Eine b–adische Folge* $(X_n)_n$ *heißt* zufällig, *falls für jeden effektiven Algorithmus, der eine Folge verschiedener natürlicher Zahlen* $(s_n)_n$ *als Funktion von n und den Zahlen* $X_{s_0},\ldots,X_{s_{n-1}}$ *erzeugt, die Teilfolge* $(X_{s_n})_n$ *teilfolgen–*∞*–verteilt ist.*

Eine $[0,1[$*–Folge* $(U_n)_n$ *heißt* zufällig, *falls die Folge* $X_n = \lfloor bU_n \rfloor$ *zufällig ist für jede natürliche Zahl* $b \geq 2$.

Es ist bewiesen worden, daß zufällige Folgen im Sinn der obigen Definition wirklich existieren. Andererseits kann man zeigen, daß es keinen effektiven Algorithmus geben kann, der eine Zufallsfolge erzeugt.

Dies ist jedoch kein Problem, da man für Anwendungen immer nur endliche Teilsequenzen von Zufallsfolgen benötigt, und für endliche Sequenzen macht es keinen Sinn über Zufälligkeit zu sprechen. Wenn man weiters bedenkt, daß in einer 100000–verteilten Folge z.B. eine Teilsequenz existieren *muß*, die 100000 Nullen hintereinander enthält, kann man sich vorstellen, daß Zufallsfolgen nicht das sind, was man eigentlich benötigt. In den folgenden Abschnitten werden wir versuchen, Folgen von Zahlen zu konstruieren, die den wichtigsten statistischen Tests genügen und in den Anwendungen einen guten Ersatz für Zufallsfolgen bieten. Der mathematische Begriff für solche Folgen ist *Folge von Pseudozufallszahlen*. Die Algorithmen zur Erzeugung solcher Sequenzen heißen meist *Zufallszahlengeneratoren*.

7.2.2 Zufallszahlengeneratoren

Es gibt eine ganze Reihe von Methoden, mit denen man Pseudozufallszahlen erzeugen kann. Eines dieser Verfahren ist jedoch wegen seiner Einfachheit ungleich weiter verbreitet als alle anderen, die *lineare Kongruenzenmethode*. Erst in den letzten Jahren, als durch die Größe der Anwendungen der Rahmen der linearen Kongruenzenmethode mitunter gesprengt wird, beginnt man vermehrt andere Verfahren einzusetzen.

Im letzten Unterabschnitt werden wir schließlich Verfahren kennenlernen, die mäßige Zufallszahlengeneratoren mit wenig Aufwand verbessern können. Das benötigt man unter Umständen dann, wenn die in Softwarebibliotheken zur Verfügung gestellten Zufallszahlen schlechte Qualität haben.

Die lineare Kongruenzenmethode

Dieses Verfahren wurde von Lehmer 1948 eingeführt. Um sie zu formulieren, beginnen wir mit 4 „magischen“ natürlichen Zahlen:

$$\begin{array}{rlrcl} X_0, & \text{Startwert} & X_0 & \geq & 0, \\ a, & \text{Multiplikator} & a & \geq & 0, \\ c, & \text{Inkrement} & c & \geq & 0, \\ m, & \text{Modulus} & m & > & X_0, \quad m > a, \quad m > c. \end{array}$$

Die Folge von Pseudozufallszahlen $(X_n)_n$ erhält man durch die Vorschrift

$$X_{n+1} = (aX_n + c) \mod m, \quad n \geq 0.$$

Die so erzeugte Folge heißt *lineare Kongruenzenfolge*.

Beispiel 7.2.2.1. *Die Folge für $X_0 = a = c = 7$, $m = 10$ ist*

$$7,6,9,0,7,6,9,0,\ldots$$

Das vorangehende Beispiel zeigt, daß eine lineare Kongruenzenfolge nicht immer zufällig aussieht. Man sieht auch, daß die Folge einen Zyklus aufweist, was daraus folgt, daß die lineare Kongruenzenmethode eine Vorschrift der Form $X_{n+1} = f(X_n)$ über einem endlichen Zahlenbereich ist. Genauer ist die Zykluslänge (im obigen Fall 4) höchstens m. Diese Zykluslänge kann etwa durch die Wahl $a = c = 1$ auch erreicht werden, doch die entstehende Vorschrift $X_{n+1} = (X_n + 1) \mod m$ erzeugt alles andere als eine zufällig aussehende Folge.

Der folgende Satz gibt an, wann die Periodenlänge noch maximal ist:

Theorem 7.2.2.2. *Die lineare Kongruenzenmethode hat Zykluslänge m genau dann wenn*

1. *c und m sind teilerfremd;*
2. *$b = a - 1$ ist ein Vielfaches jeder Primzahl p, die m teilt;*
3. *b ist ein Vielfaches von 4, wenn m ein Vielfaches von 4 ist.*

Aus diesem Theorem kann man auch entnehmen, welche Zahlen für m in Frage kommen. Primzahlen lassen nur die Wahl $a = c = 1$ zu, was nicht zu guten Folgen führt. Man benötigt im Gegenteil Zahlen, die in ihrer Primfaktorenzerlegung Faktoren aufweisen, die von der Form p^k für $k > 1$ sind. Im Computerbereich wählt man für m meist $2^e - 1$ für geeignetes e, da sich die Kongruenzen für so gestaltetes m relativ rasch bestimmen lassen. In den letzten Jahren ist man davon etwas abgekommen, da die schnellen Hardwareimplementationen für Integerrechnung auch kompliziertere Restklassen schnell berechnen können. Heute wählt man ein m, das möglichst nah an der größten darstellbaren ganzen Zahl ist, und eine brauchbare Primfaktorenzerlegung aufweist.

Andere Methoden

Die *quadratische Kongruenzenmethode* versucht, Zufallsfolgen zu erzeugen, die bessere Eigenschaften aufweisen als die linearen Kongruenzenverfahren. Bei Versuchen, solche Verfahren zu erzeugen, muß man sehr vorsichtig sein. Es ist nämlich notwendig, immer theoretische Untersuchungen über die Qualität der erzeugten Zahlen anzustellen, da man sonst (große!) Gefahr läuft, unbrauchbare Verfahren zu erfinden, die Zyklen der Länge 1 und ähnliche Pathologien aufweisen.

Die Vorschrift für die quadratische Kongruenzenmethode ist:

$$X_{n+1} = (dX_n^2 + aX_n + c) \mod m,$$

wobei X_0, a, c und m wie für die lineare Kongruenzenmethode gewählt werden, und $d < m$ sein sollte. Für quadratische Kongruenzenverfahren gilt der folgende Satz:

Theorem 7.2.2.3. *Eine quadratische Kongruenzenmethode hat genau dann maximale Periodenlänge m, wenn*

1. *c und m sind teilerfremd;*
2. *d und $a-1$ sind beide Vielfache jeder ungeraden Primzahl p, die m teilt;*
3. *d ist gerade und $d \equiv a-1(4)$, wenn $m \equiv 0(4)$;*
   $d \equiv a-1(2)$, wenn $m \equiv 0(2)$;
4. *entweder $d \equiv 0(9)$ oder $a \equiv 1(9)$ und $cd \equiv 6(9)$, wenn $m \equiv 0(9)$.*

Wenn die Zykluslänge m für die Anwendung nicht ausreicht, dann muß man zwangsläufig das Gebiet der Verfahren der Form $X_{n+1} = f(X_n)$ verlassen. Eine mögliche Technik, um den Zyklus zu verlängern, ist f nicht nur von X_n sondern auch von X_{n-1} oder noch mehr Folgengliedern abhängig zu machen. Das einfachste Beispiel für ein derartiges Verfahren ist die Fibonacci–Folge

$$X_{n+1} = (X_n + X_{n-1}) \mod m$$

für geeignet gewähltes m. Dieser Algorithmus gibt für viele Startwerte X_0, X_1 Zykluslängen, die größer als m sind, doch wird im allgemeinen weder die maximale Zykluslänge m^2 erreicht noch sind die entstehenden Folgen befriedigend zufällig. Besser geeignet sind Generatoren der Form

$$X_{n+1} = (X_n + X_{n-k}) \mod m,$$

wenn k günstig gewählt wird. Erfunden wurden sie von Green, Smith und Klem 1959.

Allgemeiner kann man, wenn $m = p$ eine Primzahl ist, Kongruenzenfolgen verwenden, die große Zykluslängen und eine hervorragende „Zufälligkeit“ aufweisen. Aus der Theorie endlicher Körper (Algebra) folgt, daß eine rekursiv wie folgt definierte Folge

$$X_{n+1} = (a_1 X_{n-1} + \cdots + a_k X_{n-k}) \mod p$$

Zykluslänge $p^k - 1$ hat, wenn man die Multiplikatoren a_i geeignet wählt. Eine gründliche Untersuchung zeigt, daß das genau dann der Fall ist, wenn das aus den a_i gebildete Polynom

$$f(x) = x^k - a_1 x^{k-1} - \cdots - a_k$$

ein primitives Polynom modulo p ist, das heißt es besitzt eine Nullstelle, die ein primitives Element des Körpers mit p^k Elementen ($GF(p^k)$) ist. Leider ist das Auffinden solcher a_i für große m und k nicht trivial, doch in Anwendungen, die eine exzessiv hohe Zahl von Zufallszahlen benötigen, lohnt sich der Aufwand.

Aus einer etwas allgemeineren Untersuchung haben Mitchell und Moore 1959 die Sequenz

$$X_n = (X_{n-24} + X_{n-55}) \mod m$$

für gerades m definiert. Sind nicht alle Startwerte $X_0, \ldots, X_{54}$ gerade, so hat die so definierte Folge eine extrem lange Periode. Ist $m = 2^e$, so hat die Sequenz Periodenlänge $2^f(2^{55} - 1)$ für ein f mit $0 \leq f < e$.

Verbesserung von Zufallszahlen

Hat man ein oder zwei Quellen von Zufallszahlen, die die statistischen Tests nicht befriedigend erfüllen, so kann man einen der folgenden Algorithmen anwenden, um eine neue Folge von Pseudozufallszahlen zu erzeugen, die deutlich bessere Zufallseigenschaften aufweist.

Im ersten Fall nehmen wir an, wir hätten zwei Pseudozufallsfolgen $(X_n)_n$ und $(Y_n)_n$. m sei der Modulus der Y–Folge. Die Zykluslänge der so erzeugten Folge von Pseudozufallszahlen ist im allgemeinen das kleinste

---

Randomisieren durch Mischen 1

Wähle k (etwa $= 100$)
for $i = 0$ **to** $k - 1$ **do**
    $V[i] = X_i$
done
$r = k$
$s = 0$
while 1 **do**
    $X = X_r$
    $Y = Y_s$
    $j = \lfloor kY/m \rfloor$
    Gib $V[j]$ als nächste Pseudozufallszahl aus
    $V[j] = X$
    $r = r + 1$
    $s = s + 1$
done

---

gemeinsame Vielfache der Zykluslängen von $(X_n)_n$ und $(Y_n)_n$. Die Qualität der Pseudozufallszahlen ist fast immer besser als die der beiden Ausgangsfolgen. Nur wenn die Folgen $(X_n)_n$ und $(Y_n)_n$ stark abhängig sind, kann der Algorithmus versagen und eine Folge von Zahlen mit schlechten Eigenschaften erzeugen.

Ein noch besseres Verfahren, das man sogar anwenden kann, wenn man nur eine vorgegebene Folge $(X_n)_n$ hat, wurde 1976 von Carter Bays und S. D. Durham angegeben. In diesem Verfahren sei m der Modulus von $(X_n)_n$:

Dieses Verfahren verschlechtert die Zufälligkeitseigenschaften niemals, im vielen Fällen kann auch die Zykluslänge auf m^k gesteigert werden.

Randomisieren durch Mischen 2

Wähle k (etwa $= 100$)
for $i = 0$ **to** $k-1$ **do**
    $V[i] = X_i$
done
$r = k$
$Y = X_r$
while 1 **do**
    $j = \lfloor kY/m \rfloor$
    $Y = V[j]$
    Gib Y als nächste Pseudozufallszahl aus
    $V[j] = X_{r+1}$
    $r = r+1$
done

7.2.3 Nicht-gleichverteilte Zufallszahlen

Nachdem wir einige Methoden kennengelernt haben, mit denen man Folgen gleichverteilter Zufallszahlen erzeugen kann, werfen wir einen kurzen Blick auf Verfahren, mit deren Hilfe man Folgen erzeugen kann, die scheinbar aus Prozessen entstanden sind, die anderen als gleichverteilten Zufallsvariablen entsprechen.

Allgemeine Methoden für stetige Verteilungen

Sei X eine Zufallsvariable mit Verteilungsfunktion F. Dann gilt natürlich, daß F monoton wächst. Ist F sogar streng monoton steigend, dann existiert die inverse Funktion F^{-1}. Nachdem $\lim_{x\to-\infty} F(x) = 0$ und $\lim_{x\to-\infty} F(x) = 1$ gelten, bildet F das Intervall $]0,1[$ auf $]-\infty,+\infty[$ ab. (Ist F nicht streng monoton wachsend auf ganz $\mathbb{R}$ sondern bildet es irgendein Intervall $[a,b]$ bijektiv auf $[0,1]$ ab, so kann man analog vorgehen).

Ist U eine $[0,1]$–gleichverteilte Zufallsvariable, so ist $X = F^{-1}(U)$ eine F–verteilte Zufallsvariable. Verwenden wir also eines der obigen Verfahren, das eine Folge von gleichverteilten Pseudozufallszahlen in $[0,1]$ erzeugt, so kann man mit der Definition

$$X_n = F^{-1}(U_n)$$

eine Folge von Pseudozufallszahlen erzeugen, die F–verteilt erscheint.

Leider ist die Auswertung von F^{-1} für viele Verteilungsfunktionen zu aufwendig, weshalb für die gebräuchlichsten Verteilungen bessere Verfahren erfunden wurden.

Die Normalverteilung

Die Verteilungsfunktion der $N(0,1)$–Normalverteilung ist

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt.$$

Sie hat Erwartungswert 0 und Standardabweichung 1.

Die drei folgenden Verfahren sind geordnet aufsteigend nach Komplexität und Verwendbarkeit.

Der Algorithmus berechnet nie extrem große Werte X, doch die Wahrscheinlichkeit, daß man bei einem Zufallsexperiment Werte findet, die größer sind als die von Teichroews Verfahren erzeugbaren, ist kleiner als $1/50000$. Das Verfahren ist leicht zu implementieren, aber es ist nur approximativ. Anwendungen, die eine extrem gute Qualität voraussetzen, sollten daher eine der anderen Methoden vorziehen.

Der Algorithmus erzeugt aus zwei Zufallszahlen U_1 und V_1 zwei Zufallszahlen X_1 und X_2. Die erste Schleife, die dazu dient U_1 und U_2 zu finden, wird im Schnitt 1.27 Mal ausgeführt mit Standardabweichung

Methode von Teichroew

$a_1 = 3.949846138$
$a_3 = 0.252408784$
$a_5 = 0.076542912$
$a_7 = 0.008355968$
$a_9 = 0.029899776$
Erzeuge 12 unabhängige gleichverteilte Zahlen $U_1, \ldots, U_{12}$
$R = (U_1 + \cdots + U_{12} - 6)/4$
$X = ((((a_9R^2 + a_7)R^2 + a_5)R^2 + a_3)R^2 + a_1)R$

Die Polarmethode

$S = 2$
while $S \geq 1$ **do**
Erzeuge zwei unabhängige Pseudozufallszahlen (gleichverteilt) U_1, U_2
$V_1 = 2U_1 - 1$
$V_2 = 2U_2 - 1$
$S = V_1^2 + V_2^2$
done
$X_1 = V_1\sqrt{\frac{-2\log S}{S}}$
$X_2 = V_2\sqrt{\frac{-2\log S}{S}}$

0.587. Die Polarmethode ist ein sehr genaues Verfahren, doch die Berechnung der Quadratwurzel und des Logarithmus macht es recht langsam.

Das folgende Verfahren benötigt einige Hilfstabellen, die wir vor der eigentlichen Beschreibung angeben wollen.

$$\begin{aligned}
&A[0] = A[1] = A[2] = 0, \quad A[3] = A[4] = \tfrac{1}{4},\\
&A[5] = A[6] = \tfrac{1}{2}, \quad A[7] = \tfrac{3}{4}, \quad A[8] = 1, \quad A[9] = \tfrac{5}{4},
\end{aligned}$$

$$\begin{aligned}
&B[40] = B[41] = B[42] = \tfrac{1}{4}, \quad B[43] = \tfrac{1}{2}, \quad B[44] = B[45] = B[46] = \tfrac{3}{4},\\
&B[47] = 1, \quad B[48] = B[49] = \tfrac{3}{2}, \quad B[50] = \tfrac{7}{4}, \quad B[51] = 2,
\end{aligned}$$

$$\begin{aligned}
&C[208] = 0, \quad C[209] = \tfrac{1}{4}, \quad C[210] = C[211] = \tfrac{1}{2}, \quad C[212] = C[213] = \tfrac{3}{4},\\
&C[214] = C[215] = C[216] = 1, \quad C[217] = C[218] = C[219] = \tfrac{3}{2},\\
&C[220] = C[221] = \tfrac{7}{4}, \quad C[222] = C[223] = \tfrac{9}{4}, \quad C[224] = \tfrac{5}{2},
\end{aligned}$$

$$\begin{aligned}
S[j] &= (j-1)/4, \quad 1 \le j \le 13\\
P[j] &= p_1 + \cdots + p_{12} + (p_{13} + p_{25}) + \cdots + (p_{12+j} + p_{24+j}), \quad 1 \le j \le 12\\
P[13] &= 1\\
Q[j] &= P[j] - p_{24+j}, \quad 1 \le j \le 12\\
D[j] &= a_j / b_j\\
E[j] &= \frac{2}{\pi} \frac{e^{(-j/4)^2/2}}{b_j p_{j+24}}, \quad 1 \le j \le 12,
\end{aligned}$$

mit den folgenden Definitionen:

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
$\frac{49}{256}$	$\frac{45}{256}$	$\frac{38}{256}$	$\frac{30}{256}$	$\frac{23}{256}$	$\frac{16}{256}$	$\frac{11}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{2}{256}$	$\frac{1}{256}$	$\frac{0}{256}$

und

$$\begin{aligned}
f(x) &= \sqrt{\frac{2}{\pi}} e^{-x^2/2}, \quad x \ge 0.\\
p_{j+12} &= \tfrac{1}{4} f(j/4) - p_j, \quad 1 \le j \le 12,\\
s_j &= \tfrac{j-1}{4}, \quad h = \tfrac{1}{4}\\
p_{j+24} &= \sqrt{\frac{2}{\pi}} \int_{s_j}^{s_j+h} (e^{-t^2/2} - e^{-(j/4)^2/2})\, dt\\
f_{j+24} &= \frac{1}{p_{j+24}} \sqrt{\frac{2}{\pi}} \left(e^{-x^2/2} - e^{-(j/4)^2/2} \right), \quad s_j \le x \le s_j + h\\
b_j &= \begin{cases} -h f'_{j+24}(s_j + h) & 1 \le j \le 4,\\ f_{j+24}(s_j) & 5 \le j \le 12 \end{cases}\\
a_j &= \begin{cases} f_{j+24}(s_j) & 1 \le j \le 4,\\ f_{j+24}(x_j) + (x_j - s_j) b_j / h & 5 \le j \le 12 \end{cases}
\end{aligned}$$

und x_j ist die Lösung der Gleichung $f'_{j+24}(x_j) = -b_j/h$.

Die Mantissenlänge t sollte mindestens 24 sein, um eine vernünftige Genauigkeit zu gewährleisten. Dieser

Marsaglia–MacLaren Methode

Erzeuge eine gleichverteilte Zufallszahl $U = .b_0 b_1 \dots b_t$
und ihre Binärentwicklung
$\psi = b_0$
if $b_1 b_2 b_3 b_4 < 10$ **then**
$X = A[b_1 b_2 b_3 b_4] + .00 b_5 b_6 \dots b_t$
elseif $b_1 b_2 b_3 b_4 b_5 b_6 < 52$ **then**
$X = B[b_1 b_2 b_3 b_4 b_5 b_6] + .00 b_7 b_8 \dots b_t$
elseif $b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 < 225$ **then**
$X = C[b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8] + .00 b_9 b_{10} \dots b_t$
else
$j = \min\{k \mid 1 \le k \le 13 \wedge .b_1 b_2 \dots b_t < P[j]\}$
if $j < 13$ **then**
if $.b_1 b_2 \dots b_t < Q[j]$ **then**
Erzeuge eine gleichverteilte Zufallszahl $U = .b_0 b_1 \dots b_t$
else
$V = +\infty$; $U = -\infty$
while $V > U + E[j](e^{-(X^2 - S[j+1]^2)/2})$ **and** $V > D[j]$ **do**
Erzeuge zwei unabhängige gleichverteilte Zufallszahlen U, V
if $U > V$ **then**
$M = U$; $U = V$; $V = M$
endif
$X = S[j] + \frac{1}{4}U$
done
endif
else
$X = 0$
while $X < 3$ **do**
$W = +\infty$
while $W \ge 1$ **do**
Erzeuge zwei unabhängige gleichverteilte Zufallszahlen U, V
$W = U^2 + V^2$
done
$T = \sqrt{(9 - 2\log W)/W}$
$X = U \cdot T$
if $X < 3$ **then**
$X = V \cdot T$
endif
done
endif
endif
if $\psi = 1$ **then**
$X = -X$
endif

Algorithmus ist sehr genau und sehr schnell. In nur 12% der Fälle führt der Algorithmus den ersten **else**–Teil aus und benötigt daher meist nur eine Zufallszahl zur Berechnung von X. Der Implementationsaufwand ist allerdings sehr hoch. Das Verfahren ist eine Anwendung der *Rectangle–Wedge–Tail*–Methode, die von Marsaglia und MacLaren entwickelt wurde. Sie kann auch auf andere stetige Verteilungen angewendet werden; dabei müssen im Prinzip nur neue Tabellen berechnet werden.

Ein Beweis für die Funktionstüchtigkeit der drei Methoden kann z.B. in [Knuth 1969, II, 3.4.1] gefunden werden.

Andere stetige Verteilungen

Wichtige andere Verteilungen, für die schnelle Berechnungsmethoden in [Knuth 1969, II, 3.4.1] zu finden sind, inkludieren die folgenden:

Exponentialverteilung Sie tritt bei „Ankunftszeit"–Problemen auf, und ihre Verteilungsfunktion ist

$$F(x) = 1 - e^{-x/\mu}, \quad x \geq 0.$$

χ^2–Verteilung Sie wird vor allem für statistische Tests benötigt. Die χ^2–Verteilung mit ν Freiheitsgraden heißt manchmal auch Gammaverteilung von Ordnung $\nu/2$:

$$F(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} \int_0^x t^{\nu/2-1} e^{-t/2}\, dt, \quad x \geq 0.$$

7.2.4 Testen von Pseudozufallszahlen

Hat man einen Algorithmus gefunden, der eine Folge von Pseudozufallszahlen erzeugt, muß man sicher gehen, daß die Zufallsfolge in den Anwendungen den Anforderungen gerecht wird. Die übliche Vorgangsweise ist, eine Reihe statistischer Tests zu verwenden, die für wirkliche Zufallsfolgen erfüllt sein müssen.

Wir beginnen mit zwei theoretischen Tests, dem χ^2–Test für diskrete Verteilungen und dem Kolmogorov–Smirnov–Test für stetige Verteilungsfunktionen.

Danach stellen wir noch eine Reihe empirischer Tests vor, die auf den theoretischen Tests aufbauen und häufig zur Bewertung von Zufallszahlengeneratoren herangezogen werden.

χ^2–Test

Dieser Test ist der verbreitetste aller statistischen Tests. Er beruht darauf, Häufigkeiten auftretender Ereignisse zu zählen.

Beispiel 7.2.4.1. *Seien zwei Würfel gegeben. Die folgende Tabelle zeigt die Wahrscheinlichkeiten bei gleichzeitigem Wurf, die Gesamtsumme s zu erzielen.*

$s =$	2	3	4	5	6	7	8	9	10	11	12
$p_s =$	$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{12}$	$\frac{1}{9}$	$\frac{5}{36}$	$\frac{1}{6}$	$\frac{5}{36}$	$\frac{1}{9}$	$\frac{1}{12}$	$\frac{1}{18}$	$\frac{1}{36}$

Nehmen wir an, wir machen drei Experimentserien von jeweils 144 Würfen und erhalten die folgenden Ergebnisse:

$s =$	2	3	4	5	6	7	8	9	10	11	12
$Y_s^{(0)} =$	2	4	10	12	22	29	21	15	14	9	6
$Y_s^{(1)} =$	4	10	10	13	20	18	18	11	13	14	13
$Y_s^{(2)} =$	3	7	11	15	19	24	21	17	13	9	5
$np_s =$	4	8	12	16	20	24	20	16	12	8	4

Hier sind die $Y_s^{(i)}$ die gezählten absoluten Häufigkeiten; der Wert in der letzten Zeile ist die erwartete Anzahl von Ereignissen s.

Es gibt 36^{144} *mögliche Sequenzen von* 144 *Würfen, die alle gleich wahrscheinlich sind, auch die Sequenz, die nur* 2*er enthält. Wollen wir herausfinden, ob die Würfel gezinkt sind, anhand der durchgeführten Experimente, können wir nur versuchen,* Wahrscheinlichkeitsaussagen *darüber zu machen. Wir können feststellen, wie wahrscheinlich bestimmte Häufigkeitsverteilungen sind.*

Dazu bildet man das gewichtete Mittel aus den quadrierten Abweichungen der Experimente von den Erwartungswerten, die χ^2–Statistik

$$\chi^2 = \sum_{s=1}^{k} \frac{(Y_s - np_s)^2}{np_s},$$

wobei k die Anzahl verschiedener möglicher Resultate der Experimente ist. Im Fall zweier Würfel ist $k = 11$. Beachtet man, daß immer die beiden Gleichungen

$$\sum_{i=1}^{k} Y_i = n, \qquad \sum_{i=1}^{k} p_i = 1$$

gelten, so kann man die Formel für die χ^2–Statistik umformen zu dem bekannten Ausdruck

$$\chi^2 = \frac{1}{n} \sum_{s=1}^{k} \frac{Y_s^2}{p_s} - n.$$

Die χ^2–Statistik ist χ^2–verteilt mit $k-1$ Freiheitsgraden. (Das $k-1$ läßt sich dadurch begründen, daß aus $Y_1, \ldots, Y_{k-1}$ der Wert von Y_k berechnet werden kann, also nicht alle Y_i wirklich unabhängig sind.)

Die Verteilung der χ^2–Statistik ist meist tabelliert. Die größeren Programmpakete wie Mathematica oder MAPLE enthalten jedoch Funktionen, mit deren Hilfe die Werte im Programm bestimmt werden können. Tabelle 7.1 enthält einen Auszug aus den üblichen Tabellen, der die wichtigsten Werte enthält. Eine ausführlichere Tabelle kann etwa in [Bronstein, Semendjajev 1989] gefunden werden.

	$p = 99\%$	$p = 95\%$	$p = 75\%$	$p = 50\%$	$p = 25\%$	$p = 5\%$	$p = 1\%$
$\nu = 1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu = 2$	0.00201	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu = 3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu = 4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu = 5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu = 6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu = 7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu = 8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu = 9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu = 10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu = 12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu = 15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu = 20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu = 30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu > 30$	näherungsweise $\nu + 2\sqrt{\nu}x_p + \frac{4}{3}x_p^2 - \frac{2}{3}$						
$x_p =$	-2.33	-1.64	-0.675	0	0.675	1.64	2.33

Tabelle 7.1: χ^2–Verteilung

Beispiel 7.2.4.2. *Bestimmen wir die Werte der* χ^2*–Statistik für die Experimente* $Y_s^{(i)}$ *von zuvor. Es gilt*

$$\chi^2(Y^{(0)}) = 7\tfrac{7}{48}, \quad \chi^2(Y^{(1)}) = 29\tfrac{59}{120}, \quad \chi^2(Y^{(2)}) = 1\tfrac{17}{120}.$$

Wir finden, daß $\chi^2(Y^{(1)})$ viel zu hoch ist; wenn wir in Tabelle 7.1 für $\nu = 10$ nachsehen, erkennen wir, daß in weniger als 1% der Fälle so hohe Werte für die χ^2–Verteilung auftreten. Im Gegensatz dazu ist $\chi^2(Y^{(2)})$ viel zu klein, denn wir können aus Tabelle 7.1 ebenfalls entnehmen, daß in mehr als 99% aller Fälle größere Werte als $1\frac{17}{120}$ vorkommen. Die Abweichungen von den erwarteten Häufigkeiten sind also viel zu gering, um Experiment $Y^{(2)}$ als zufällig betrachten zu können.

Ganz anders ist das Resultat für $Y^{(0)}$. Der Wert $7\frac{7}{48}$ fällt zwischen die Werte für 75% und 50%, ist daher weder als zu hoch noch als zu niedrig einzustufen.

Allgemein kann man sagen, daß Ergebnisse zwischen den Tabellenwerten für 25% und 75% ein positives Testergebnis bedeuten. Liegt der Wert in der Gegend der 5% und der 95% Spalte, so ist das Resultat verdächtig. Im Fall von Ergebnissen unterhalb von 1% oder jenseits von 99% muß der Test als fehlgeschlagen gewertet werden.

Grundsätzlich ist es günstig, denselben Test mehrmals für verschiedene Folgen von Zufallszahlen, die mit demselben Generator erzeugt wurden, durchzuführen. Treten ein Fehlschlag oder mehrere verdächtige Resultate auf, so kann man den Generator als unbrauchbar verwerfen.

Eine interessante Frage bleibt noch: Wie groß muß n gewählt werden? Man kann davon ausgehen, daß „je größer desto besser“ gilt. Eine Faustregel ist, daß n mindestens so groß sein sollte, daß $np_s > 5$ für alle möglichen Resultate gilt. Diese Bedingung ist in unserem obigen Beispiel verletzt, doch die Abweichungen sind so groß, daß sie trotzdem Aussagekraft besitzen.

Grundsätzlich wird man gezinkte Würfel erkennen, wenn man genug Experimente macht, n also groß wählt. Leider tendieren große Werte von n dazu, *lokal* nicht-zufälliges Verhalten zu verbergen, wie es etwa einige Zufallszahlengeneratoren aufweisen.

Ein weiterer Nachteil des χ^2–Tests ist, daß die Approximation sehr grob ist. Manchmal kann es dadurch passieren, daß Testergebnisse in die falsche Kategorie („bestanden“, „verdächtig“, „durchgefallen“) eingeordnet werden. Für sehr große n ist jedoch auch dies unwahrscheinlich.

Kolmogorov–Smirnov–Test

Hat man eine Zufallsvariable, die kontinuierliche Werte annehmen kann, ist der χ^2–Test nicht anwendbar. In diesem Fall muß man anders vorgehen.

Machen wir n unabhängige Experimente für die Zufallsvariable X und erhalten wir Resultate $X_1, \ldots, X_n$ so können wir die *empirische Verteilungsfunktion* $F_n(x)$ bilden:

$$F_n(x) = \frac{|\{j \in \{1,\ldots,n\} : X_j \leq x\}|}{n}.$$

Wächst n, so sollte $F_n(x)$ die Verteilung $F(x)$ von X immer besser approximieren.

Der Kolmogorov–Smirnov–Test kann verwendet werden, wenn $F(x)$ keine Sprungstellen besitzt. Er basiert auf der Differenz von $F(x)$ und $F_n(x)$. Ein schlechter Zufallszahlengenerator wird empirische Verteilungsfunktionen erzeugen, die $F(x)$ nur schlecht approximieren.

Um den Test durchzuführen, bilden wir die folgenden Statistiken:

$$K_n^+ = \sqrt{n} \max_{-\infty<x<+\infty} \big(F_n(x) - F(x)\big)$$
$$K_n^- = \sqrt{n} \max_{-\infty<x<+\infty} \big(F(x) - F_n(x)\big)$$

K_n^+ (K_n^-) mißt die Maximalabweichung an Punkten, an denen F_n größer (kleiner) als F ist. Wie beim χ^2–Test kann man K_n^+ und K_n^- in einer Tabelle (Tabelle 7.2) nachschlagen, um festzustellen, ob sie signifikant zu hoch oder zu klein sind. Im Gegensatz zum χ^2–Test sind für $K_n^\pm$ die Tabellenwerte keine groben Näherungen, die für große n gelten, sondern exakte Werte für alle n. Daher kann der Kolmogorov–Smirnov–Test für alle n zuverlässig verwendet werden.

Nachdem die Ausdrücke für K_n^+ und K_n^- nicht geeignet sind für Computerauswertungen, wollen wir einen Algorithmus angeben, mit dessen Hilfe man die K–Statistiken einfach bestimmen kann.

	$p = 99\%$	$p = 95\%$	$p = 75\%$	$p = 50\%$	$p = 25\%$	$p = 5\%$	$p = 1\%$
$n = 1$	0.01000	0.05000	0.2500	0.5000	0.7500	0.9500	0.9900
$n = 2$	0.01400	0.06749	0.2929	0.5176	0.7071	1.0980	1.2728
$n = 3$	0.01699	0.07919	0.3112	0.5147	0.7539	1.1017	1.3589
$n = 4$	0.01943	0.08789	0.3202	0.5210	0.7642	1.1304	1.3777
$n = 5$	0.02152	0.09471	0.3249	0.5245	0.7674	1.1392	1.4024
$n = 6$	0.02336	0.1002	0.3272	0.5319	0.7703	1.1463	1.4144
$n = 7$	0.02501	0.1048	0.3280	0.5364	0.7755	1.1537	1.4246
$n = 8$	0.02650	0.1086	0.3280	0.5392	0.7797	1.1586	1.4327
$n = 9$	0.02786	0.1119	0.3274	0.5411	0.7825	1.1624	1.4388
$n = 10$	0.02912	0.1147	0.3297	0.5426	0.7845	1.1658	1.4440
$n = 11$	0.03028	0.1172	0.3330	0.5439	0.7863	1.1688	1.4484
$n = 12$	0.03137	0.1193	0.3357	0.5453	0.7880	1.1714	1.4521
$n = 15$	0.03424	0.1244	0.3412	0.5500	0.7926	1.1773	1.4606
$n = 20$	0.03807	0.1298	0.3461	0.5547	0.7975	1.1839	1.4698
$n = 30$	0.04354	0.1351	0.3509	0.5605	0.8036	1.1916	1.4801
$n > 30$	0.07089 $-\frac{0.15}{\sqrt{n}}$	0.1601 $-\frac{0.14}{\sqrt{n}}$	0.3793 $-\frac{0.15}{\sqrt{n}}$	0.5887 $-\frac{0.15}{\sqrt{n}}$	0.8326 $-\frac{0.16}{\sqrt{n}}$	1.2239 $-\frac{0.17}{\sqrt{n}}$	1.5174 $-\frac{0.20}{\sqrt{n}}$

Tabelle 7.2: Kolmogorov–Smirnov–Verteilungen

Bestimmung der $K_n^{\pm}$

1. Bestimme die Experimente $X_1, \ldots, X_n$
2. Sortiere die X_i in aufsteigender Reihenfolge
3. Berechne die Statistiken

$$K_n^+ = \sqrt{n} \max_{1 \le j \le n} \left(\frac{j}{n} - F(X_j) \right)$$
$$K_n^- = \sqrt{n} \max_{1 \le j \le n} \left(F(X_j) - \frac{j-1}{n} \right)$$

Ähnlich wie für den χ^2–Test ist es notwendig, über eine geeignete Wahl von n zu sprechen. Möchte man nämlich nachweisen, daß die Verteilungsfunktion der Zufallsvariablen X nicht F sondern $G \neq F$ ist, muß man n sehr groß wählen.

Wie im χ^2–Test verbirgt dieses Vorgehen *lokal* nicht-zufälliges Verhalten, was mitunter störend sein kann. Möchte man lokal schlechtes Verhalten nachweisen, so muß man n klein wählen.

Eine günstige Vorgangsweise ist es, n mittelgroß, etwa $n = 1000$ zu wählen und eine größere Anzahl an Berechnungen von K_{1000}^+ auf verschiedenen Teilen der Zufallsfolge durchzuführen. Dies führt zu Werten

$$K_{1000}^+(1), \quad K_{1000}^+(2), \quad \ldots, \quad K_{1000}^+(r).$$

Dann kann man auf *diese Resultate* wieder einen Kolmogorov–Smirnov–Test anwenden, wenn man beobachtet, daß für große n die Verteilung von K_n^+ sehr gut von

$$F_\infty(x) = 1 - e^{-2x^2}, \quad x \ge 0$$

approximiert wird. Diese Methode wird beides *global* und *lokal* nicht-zufälliges Verhalten aufspüren.

Man kann natürlich auch den χ^2–Test mit dem Kolmogorov–Smirnov–Test verbinden. Diese Vorgehensweise ist auch viel genauer als die Einteilung in die Klassen („bestanden“, „verdächtig“ und „durchgefallen“). Man führt mehrere χ^2–Experimente durch und überprüft danach mit dem Kolmogorov–Smirnov–Test, ob sich die Ergebnisse χ^2–verteilt verhalten.

Auf Grundlage dieser beiden Tests kann man einige empirische Testmethoden verwenden, um die Qualität einer gleichverteilten Zufallsfolge zu testen. In den folgenden Abschnitten werden wir eine Folge $U = (U_n)_n$ reeller Zahlen aus dem Intervall $[0,1[$ untersuchen. Für manche Tests werden wir ganze Zahlen im Bereich $0,\ldots,d-1$ für eine geeignete Zahl d benötigen. In diesem Fall werden wir die Folge $Y = (Y_n)_n$ mit $Y_n = \lfloor dU_n \rfloor$ im Test verwenden.

Frequenz–Test

Mit diesem Test überprüft man, ob die Folge U gleichverteilt ist. Dazu verwendet man entweder den Kolmogorov–Smirnov–Test mit der Verteilungsfunktion

$$F(x) = \begin{cases} 0 & x < 0 \\ x & x \in [0,1] \\ 1 & x > 1, \end{cases}$$

oder man wählt eine geeignete Zahl d und verwendet den χ^2–Test mit den möglichen Ergebnissen $s = 0,\ldots,d-1$ und den Wahrscheinlichkeiten $p_s = 1/d$.

Serien–Test

Für eine Zufallsfolge sollen nicht nur die Frequenzen des Auftretens der verschiedenen Ereignisse „stimmen“. Wir wollen auch, daß Paare aufeinanderfolgender Zahlen unabhängig gleichverteilt sind. Um diesen Test durchzuführen, zählen wir wie oft jedes Paar $(q,r) = (Y_{2j}, Y_{2j+1})$ auftritt für $0 \le j < n$ und $0 \le q,r < d$. Auf diese $k = d^2$ Frequenzen wenden wir den χ^2–Test an mit Wahrscheinlichkeiten $p_s = 1/d^2$.

d wird dabei als bequeme, nicht allzu große ganze Zahl gewählt. Dabei muß bedacht werden, daß $n > 5d^2$ sein muß, um die Faustregel des χ^2–Tests zu erfüllen.

Natürlich kann man diesen Test auch allgemeiner für Tripel oder m–Tupel durchführen, doch die Anzahl der Kategorien und damit die Anzahl der Experimente wächst sehr schnell an, und daher muß d immer kleiner gewählt werden, so lange bis der Test nicht mehr aussagekräftig ist. Möchte man m–Tupel für $m > 2$ untersuchen, verwendet man am besten den Pokertest 7.2.4.

Gap–Test

Dieser Test überpruft die Länge von „Löchern“ zwischen Vorkommnissen von U_j in einem bestimmten Bereich. Sind α und β zwei reelle Zahlen mit $0 \le \alpha < \beta \le 1$, dann möchten wir die Längen von Teilfolgen $U_j, U_{j+1}, \ldots, U_{j+r}$ untersuchen, in denen $U_{j+r} \in [\alpha,\beta]$ liegt aber die anderen Us nicht. Diese Teilfolge von $r+1$ Zahlen entspricht dann einem Loch (*gap*) der Länge r.

Der folgende Algorithmus dient dazu, die Daten für den Gap–Test zu beschaffen:

Danach wird ein χ^2–Test angewendet auf $count[i]$, $i = 0,\ldots,t$, mit den Wahrscheinlichkeiten

$$p_i = p(1-p)^i, \quad i = 0,\ldots,t-1, \qquad p_t = (1-p)^t,$$

wobei $p = \beta - \alpha$ gesetzt wird. Die Werte für n und t sollten wieder so gewählt werden, daß die Werte von $count[i]$ größer als 5 erwartet werden.

Daten für den Gap–Test

Wähle t geeignet
$j = 0$
$s = 0$
$count[r] = 0$ für $0 \le r \le t$
while $s < n$ **do**
    $r = 0$
    while $U_j < \alpha$ **or** $U_j \ge \beta$ **do**
        $j = j + 1$
        $r = r + 1$
    done
    if $r \ge t$ **then**
      $count[t] = count[t] + 1$
    else
      $count[r] = count[r] + 1$
    endif
    $s = s + 1$
    $j = j + 1$
done

Pokertest

Der klassische Pokertest besteht darin, n Gruppen von 5 aufeinander folgenden ganzen Zahlen zu betrachten $(Y_{5j}, Y_{5j+1}, \ldots, Y_{5j+4})$ und die Häufigkeiten für das Auftreten der typischen Würfelpokerresultate zu bestimmen, wobei das Hauptaugenmerk auf die Muster

Alle verschieden:	*abcde*	Full House:	*aaabb*
Ein Paar:	*aabcd*	Poker:	*aaaab*
Zwei Paare:	*aabbc*	Grande:	*aaaaa*
Tripel:	*aaabc*		

gelegt wird. Danach wird ein χ^2–Test auf die Anzahl der Quintupel in jeder dieser Kategorien angewendet.

Eine etwas vereinfachte Version dieses Tests, der auf m–Tupel verallgemeinert werden kann, zählt nur die Anzahl *unterschiedlicher* Werte in jedem m–Tupel. Für $m = 5$ reduziert sich die Anzahl der Kategorien auf 5:

5 verschiedene	$\doteq$	alle verschieden
4 verschiedene	$\doteq$	ein Paar
3 verschiedene	$\doteq$	zwei Paare oder ein Tripel
2 verschiedene	$\doteq$	Full House oder Poker
1 verschiedenes	$\doteq$	Grande

Für diese Vereinfachung sind die Auftrittswahrscheinlichkeiten einfacher zu berechnen.

Im allgemeinen Fall ist die Wahrscheinlichkeit für das Auftreten von r verschiedenen Zahlen in einem m–Tupel

$$p_r = \frac{d(d-1)\cdots(d-r+1)}{d^m} \left\{ \begin{array}{c} m \\ r \end{array} \right\},$$

wobei die Stirlingzahlen zweiter Art definiert sind als

$$\begin{aligned}
\left\{ \begin{array}{c} n \\ 0 \end{array} \right\} &= 0, \\
\left\{ \begin{array}{c} 0 \\ n \end{array} \right\} &= \binom{0}{n}, \\
\left\{ \begin{array}{c} n \\ n \end{array} \right\} &= 1, \\
\left\{ \begin{array}{c} n \\ n-1 \end{array} \right\} &= \binom{n}{2}, \\
\left\{ \begin{array}{c} n \\ 1 \end{array} \right\} &= 1, \\
\left\{ \begin{array}{c} n \\ 2 \end{array} \right\} &= 2^{n-1} - 1, \\
\left\{ \begin{array}{c} n \\ m \end{array} \right\} &= m \left\{ \begin{array}{c} n-1 \\ m \end{array} \right\} + \left\{ \begin{array}{c} n-1 \\ m-1 \end{array} \right\}, \\
\left\{ \begin{array}{c} m \\ n \end{array} \right\} &= \tfrac{1}{n!}(-1)^n \sum_{k=0}^{n} \binom{n}{k} k^m (-1)^k.
\end{aligned}$$

Man gibt sich also m vor, zählt die Anzahl verschiedener Elemente in jedem m–Tupel und bestimmt für jede Anzahl r die Auftrittshäufigkeit. Danach führt man einen χ^2–Test mit den oben angegebenen Wahrscheinlichkeiten p_r durch.

Couponsammler–Test

„Wie lange brauche ich, um mein Fußball–WM–Album vollzukleben?“

Dieser Test verhält sich zum Pokertest etwa wie der Gap–Test zum Serien–Test. Wir betrachten die Folge Y und untersuchen die Länge r von Segmenten $Y_{j+1}, Y_{j+2}, \ldots, Y_{j+r}$, die wir benötigen, um einen kompletten Satz ganzer Zahlen von $0, \ldots, d-1$ zu erhalten.

Am Ende führt man für die berechneten Häufigkeiten $count[i]$ einen χ^2–Test durch, wobei man die Wahrscheinlichkeiten

$$p_i = \frac{d!}{d^i} \left\{ \begin{array}{c} i-1 \\ d-1 \end{array} \right\}, \quad d \le i < t, \qquad p_t = 1 - \frac{d!}{d^{t-1}} \left\{ \begin{array}{c} t-1 \\ d \end{array} \right\}$$

zugrunde legt.

Permutationstest

In diesem Test gibt man sich eine Länge t (nicht allzu groß!) vor und untersucht Teilsequenzen der Form $(U_{jt}, U_{jt+1}, \ldots, U_{jt+t-1})$. Die Elemente in jeder Gruppe können $t!$ viele verschiedene relative Ordnungen haben (für $t = 2$ gibt es die Möglichkeiten $U_{2j} < U_{2j+1}$ und $U_{2j+1} < U_{2j}$, für $t = 3$ gibt es schon 6 Möglichkeiten von durchgehend aufsteigend geordnet über Mischfälle bis durchgehend absteigend geordnet). Man zählt wieder die Häufigkeit jedes einzelnen Falls und verwendet, daß jeder dieser Fälle mit Wahrscheinlichkeit $1/t!$ auftritt.

Serien–Korrelationstest

Man berechne die Statistik

$$C = \frac{n\sum(U_j V_j) - (\sum U_j)(\sum V_j)}{\sqrt{\left(n\sum U_j^2 - (\sum U_j)^2\right)\left(n\sum V_j^2 - (\sum V_j)^2\right)}},$$

Daten für den Couponsammler–Test

```
j = 0
s = 0
count[i] = 0 für 0 ≤ i ≤ t
while s < n do
        q = 0
        r = 1
        occurs[i] = 0 für 0 ≤ i ≤ d − 1
        while q < d do
                while occurs[Y_j] do
                        j = j + 1
                        r = r + 1
                done
                occurs[Y_j] = 1
                q = q + 1
                j = j + 1
        done
        if r ≥ t then
          count[t] = count[t] + 1
        else
          count[r] = count[r] + 1
        endif
done
```

den *Serien–Korrelationskoeffizienten* für $V_j := U_{j+1}$. Er ist ein Maß für die Abhängigkeit von U_j und U_{j+1}. Er liegt immer zwischen 0 und 1. Aus theoretischen Überlegungen kann man zeigen, daß ein guter Wert für C im Intervall $[\mu_n - 2\sigma_n, \mu_n + 2\sigma_n]$ liegt mit

$$\mu_n = \frac{-1}{n-1}, \quad \sigma_n = \frac{1}{n-1}\sqrt{\frac{n(n-3)}{n+1}}, \quad n > 2.$$

Maximum von t–Test

Man untersucht die Folge $V_j = \max(U_{tj}, \ldots, U_{tj+t-1})$ mit Hilfe des Kolmogorov–Smirnov–Tests und der Verteilungsfunktion $F(x) = x^t$, $x \in [0,1]$.

Andere Tests

Es gibt noch eine Reihe anderer statistischer und theoretischer Tests, die man für die Untersuchung von Zufallszahlengeneratoren verwenden kann.

Der *Run–Test* bestimmt die Längen monoton wachsender (oder fallender) Teilsequenzen und untersucht diese Längen mit einer speziellen Statistik, die dem χ^2–Test ähnlich ist (siehe [Knuth 1969, II, 3.3.2.G]).

Der *Spektral–Test* ist ein Test, der vor allem für lineare Kongruenzenverfahren verwendet wird. Er wird heutzutage häufig dazu herangezogen, die Güte solcher Generatoren zu analysieren und ist dabei wahrscheinlich das wichtigste Hilfsmittel. Leider ist er algorithmisch und theoretisch aufwendig (er basiert auf einem ganzzahligen quadratischen Optimierungsproblem). Er kann aber in [Knuth 1969, II, 3.4] nachgelesen werden.

Wichtig ist noch zu bemerken, daß wenn in der Anwendung mit demselben Zufallsgenerator in demselben Algorithmus mehrere Folgen $(U_{s_1^1}, \ldots, U_{s_n^1}), \ldots, (U_{s_1^k}, \ldots, U_{s_n^k})$ erzeugt werden, die jede für sich eine

Folge unabhängiger Zufallszahlen sein soll. In diesem Fall ist es notwendig, alle Tests auch auf diese Teilfolgen anzuwenden, um sicher zu gehen, daß durch die Auswahl nicht zufällig Abhängigkeiten zwischen den Gliedern der Teilfolgen auftreten.

8 Integration II: Mehrdimensionaler Fall, Stochastisch

Die Berechnung mehrdimensionaler Integrale ist in weiten Bereichen der Anwendungen und auch innerhalb der numerischen Mathematik, etwa bei der Lösung von Differentialgleichungen, eine wichtige Aufgabe.

Betrachten wir das Integral

$$\int_B f(x)\,dx, \quad B \subseteq \mathbb{R}^s.$$

Die numerische Behandlung dieses Integrals unterscheidet sich vom skalaren Fall unter anderem dadurch, daß nicht nur der Integrand f sondern auch der Integrationsbereich B approximiert werden muß.

Historisch hat man zuerst mit der Berechnung zweidimensionaler Integrale begonnen, um Körpervolumina zu bestimmen. Ähnlich dem skalaren Fall war das Ziel, die Kantenlänge eines Würfels gleichen Volumens zu berechnen. Aus diesem Grund werden Verfahren zur numerischen Bestimmung höherdimensionaler Integrale immer noch *Kubaturmethoden* genannt.

8.1 Grundlagen

In diesem Kapitel werden wir versuchen, gute Näherungen für Integrale der Form

$$\int_B f(x)\,dx, \quad B \subseteq \mathbb{R}^s$$

für zusammenhängende Integrationsbereiche B zu finden.

Zwei Sätze der Analysis sind in diesem Zusammenhang wichtig:

Theorem 8.1.0.3 (Transformationsregel für Mehrfachintegrale)**.** *Sind B und B' zwei Integrationsbereiche, und sei $\varphi : B \to B'$ eine bijektive C^1–Abbildung. Ist $f : B' \to \mathbb{R}$ integrierbar. Dann gilt*

$$\int_{\varphi(B)} f(x)\,dx = \int_B f \circ \varphi(y) |\det J\varphi(y)|\,dy,$$

wobei $J\varphi$ die Jacobimatrix von φ ist.

Theorem 8.1.0.4 (Satz von Fubini)**.** *Hat der Integrationsbereich $B = B_1 \times B_2$ Produktstruktur, existieren*

$$\int_{B_1 \times B_2} f(x,y)\,d(x,y),$$

und

$$g(y) = \int_{B_1} f(x,y)\,dx$$

für alle $y \in B_2$, dann ist g auf B_2 integrierbar, und es gilt

$$\int_{B_1 \times B_2} f(x,y)\,d(x,y) = \int_{B_2} \left(\int_{B_1} f(x,y)\,dx \right) dy.$$

8.2 Direkte Kubaturmethoden

Um Kubaturverfahren zu entwickeln, die ähnlich funktionieren wie diejenigen im skalaren Fall kann man auf zwei verschiedene Arten vorgehen:

Unter Verwendung des Satzes von Fubini kann man, wenn $B = \prod_i [a_i, b_i]$ ein Quader im $\mathbb{R}^s$ ist, das Integral zurückführen auf eindimensionale Integrale:

$$\int_B f(x)\,dx = \int_{a_s}^{b_s} \dots \int_{a_1}^{b_1} f(x_1,\dots,x_s)\,dx_1 \dots dx_s.$$

Dann verwendet man skalare Quadraturmethoden zur Lösung dieser eindimensionalen Integrale. Die Steuerung des Approximationsfehlers wird dabei über die Genauigkeitsverbesserung in den eindimensionalen Integralen vorgenommen.

Die andere Möglichkeit ist, nach einer Approximation des Randes ∂B von B durch ein geschlossenes Polygon, B durch eine Triangulierung T anzunähern. Vorzugsweise verwendet man dazu eine Delaunay–Triangulierung, falls B konvex ist, oder eine Triangulierung, die lokal Delaunay ist, wenn B nicht konvex ist (siehe Kapitel 6, Abschnitt 6.4). Auf jedem Simplex S aus T wird die Funktion f durch einen Spline–Patch approximiert. Am einfachsten ist es, die Funktion linear anzunähern, also eine Hyperebene durch die Punkte $(e, f(e))$, wobei $e \in S$ die Ecken des Simplex S sind, zu legen. Um die Fehlergenauigkeit zu verbessern, kann man die Simplices S der Triangulierung subunterteilen, und eine feinere Triangulierung vornehmen. Diese Subunterteilung nimmt man meist *adaptiv* in jenen Simplices vor, in denen die Variation von f besonders groß ist.

Alle direkte Integrationsverfahren haben jedoch denselben Nachteil, der anhand der iterierten zusammengesetzten Trapezregel erläutert werden soll. Dazu sei $B = [0,1]^n$ und f beliebig. Für $n = 1$ liefert die Trapezregel die Approximation

$$\int_0^1 f(x)\,dx \approx \sum_{j=0}^{m} w_j f\left(\tfrac{j}{m}\right),$$

mit $w_0 = w_m = \frac{1}{2m}$ und $w_i = \frac{1}{m}$ für $i \neq 0, m$. Im mehrdimensionalen Fall $n > 2$ verwendet man den Satz von Fubini, um das Integral in iterierte eindimensionale Integrale zu verwandeln, welche dann mit Hilfe der Trapezregel approximiert werden:

$$\int_B f(x)\,dx = \int_0^1 \dots \int_0^1 f(x_1,\dots,x_s)\,dx_1 \dots dx_s \approx \sum_{j_1=0}^{m} \cdots \sum_{j_s=0}^{m} w_{j_1} \dots w_{j_s} f\left(\tfrac{j_1}{m},\dots,\tfrac{j_s}{m}\right). \tag{8.1}$$

Um den Aufwand abzuschätzen, muß man zählen wie viele Funktionsauswertungen die Approximation benötigt (Funktionsauswertungen sind die aufwendigste Operation, will man integrieren). Die Anzahl der Auswertungen beträgt $N = (m+1)^n$. Will man das mit der erzielten Genauigkeit vergleichen, muß man wie im skalaren Fall den Approximationsfehler analysieren.

Ist f eine C^2–Funktion, so weiß man für $s = 1$, daß der Integrationsfehlern $O(m^{-2})$ ist. Für $s > 2$ verbessert sich dieser Wert leider nicht, wie man für den Spezialfall $f(x_1,\dots,x_s) = g(x_1)$ leicht sehen kann. Der Approximationsfehler in Gleichung (8.1) ist also $O(m^{-2}) = O(N^{-2/s})$.

Mit steigender Dimension s schwindet also der Wert der Fehlerschranke $O(N^{-2/s})$ drastisch: Möchte man den Approximationsfehler auf $1/10$ reduzieren, muß man 10^s mal mehr Punkte berechnen.

Verwendet man anstelle der Trapezregel Triangulierungen, höhere Newton–Cotes–Formeln oder Gauß–Kronrod–Quadratur, so verändert sich die Fehlerschranke zu $O(N^{-k/s})$ für geeignetes $k \in \mathbb{N}$, welches nur vom Verfahren abhängt. Das ändert jedoch nichts an der Tatsache, daß die Güte der Approximation mit steigendem s exponentiell abnimmt. Dieses Problem der direkten Verfahren bezeichnet man mit „Fluch der Dimension“ oder „Dimensionsteufel“; es macht direkte Integrationsmethoden für Dimensionen $s \geq 5$ meist unbrauchbar.

Um diesen Dimensionsteufel zu bannen, muß man völlig andere Methoden entwickeln. Zuerst verwendet man einen stochastischen Zugang, um ein erstes Verfahren zu entwickeln, bei dem der Fehler unabhängig

von s abgeschätzt werden kann. Leider wird sich herausstellen, daß man dafür die Sicherheit der Konvergenz aufgeben muß (siehe Abschnitt 8.3).

Danach kostet es einige Anstrengung, den „herbeigerufenen Geist“ Wahrscheinlichkeitstheorie wieder loszuwerden, doch mit Mitteln der analytischen Zahlentheorie kann man nicht nur die Wahrscheinlichkeitsaussagen wieder deterministisch machen, sondern man kann auch die Fehlerabschätzungen radikal verbessern (siehe Abschnitt 8.4).

8.3 Monte-Carlo–Methoden

Ein *Monte-Carlo–Verfahren* besteht darin, eine zu berechnende Größe in einem stochastischen Modell umzuinterpretieren und danach durch Zufallssamples abzuschätzen.

Die berühmteste Monte-Carlo–Methode ist diejenige, die verwendet wird, um mehrdimensionale Integrale zu bestimmen, doch es existieren Monte-Carlo–Verfahren zur globalen Optimierung und zur Lösung stochastischer Differentialgleichungen.

Die Geschichte der Monte-Carlo–Methoden reicht zurück auf [Metropolis, Ulam, 1949], obwohl sie zuvor schon in geheimen Projekten des U.S. Verteidigungsministeriums genutzt worden war. Heute sind sie im Bereich der Integration beinahe vollständig durch die Quasi–Monte-Carlo–Verfahren verdrängt worden, die in Abschnitt 8.4 vorgestellt werden.

8.3.1 Monte Carlo–Integration

Im folgenden sei mit λ_s das Lebesgue–Maß auf $\mathbb{R}^s$ bezeichnet. Für Hyperquader $B = \prod_i [a_i, b_i]$ ist etwa $\lambda_s(B) = \prod_i (b_i - a_i)$.

Sei B ein Integrationsbereich mit $0 < \lambda_s(B) < \infty$. Wir betrachten den Wahrscheinlichkeitsraum $(B, \mathscr{L}(B), d\mu)$ mit dem Wahrscheinlichkeitsmaß $\mu = \lambda_s / \lambda_s(B)$. Für eine Funktion $f \in C(B)$ haben wir dann

$$\int_B f(x)\,dx = \lambda_s(B) \int_B f(x)\,d\mu(x) = \lambda_s(B)\mathbb{E}f(X),$$

wobei X eine auf B gleichverteilte Zufallsvariable ist. Das zu berechnende Integral kann also aufgefaßt werden als skalares Vielfaches des Erwartungswertes der Zufallsvariablen $f(X)$.

Das Problem, ein mehrdimensionales Integral zu berechnen, ist also darauf reduziert, den Erwartungswert einer Zufallsvariablen approximativ zu bestimmen (und das Volumen $\lambda_s(B)$ des Integrationsbereiches B zu berechnen).

Hat man eine beliebige Zufallsvariable $f(X)$ auf einem beliebigen Wahrscheinlichkeitsraum $(A, \mathscr{A}, \mu)$ gegeben, so kann man den *Monte-Carlo–Schätzwert* für den Erwartungswert $\mathbb{E}f(X)$ erhalten, indem man N unabhängige μ–verteilte Zufallszahlen $a_1, \ldots, a_N \in A$ bestimmt und

$$\mathbb{E}f(X) \approx \frac{1}{N} \sum_{i=1}^{N} f(a_i)$$

setzt. Nach dem Satz von Kolmogorov (Kapitel 7, Theorem 7.1.4.4), genügt die Folge der Summen dem starken Gesetz der großen Zahlen, und man erhält

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(a_i) = \mathbb{E}f(X)$$

fast sicher.

Für die wahrscheinlichkeitstheoretische Fehlerabschätzung benötigen wir zusätzlich zum Erwartungswert noch die Varianz $\mathbb{V}f(X)$:

$$\mathbb{V}f(X) = \int_A (f - \mathbb{E}f(X)) d\mu,$$

welche endlich ist, wenn $f \in L^2(\mu)$ liegt.

Proposition 8.3.1.1. *Ist $f \in L^2(\mu)$ dann gilt für alle $N \geq 1$ die Beziehung*

$$\int_A \dots \int_A \left(\frac{1}{N} \sum_{i=1}^{N} f(a_i) - \mathbb{E}f(X) \right) d\mu(a_1) \dots d\mu(a_N) = \frac{\mathbb{V}f(X)}{N}.$$

Beweis. Setzen wir $g = f - \mathbb{E}f(X)$, so ist $\int_A g\, d\mu = 0$ und

$$\frac{1}{N} \sum_{i=1}^{N} f(a_i) - \mathbb{E}f(X) = \frac{1}{N} \sum_{i=1}^{N} g(a_i).$$

Weiters haben wir

$$\begin{aligned}
&\int_A \dots \int_A \left(\frac{1}{N} \sum_{i=1}^{N} f(a_i) - \mathbb{E}f(X) \right) d\mu(a_1) \dots d\mu(a_N) = \\
&\quad = \int_A \dots \int_A \left(\frac{1}{N} \sum_{i=1}^{N} g(a_i) \right)^2 d\mu(a_1) \dots d\mu(a_N) = \\
&\quad = \frac{1}{N^2} \sum_{i=1}^{N} \int_A \dots \int_A g(a_i)^2\, d\mu(a_1) \dots d\mu(a_N) + \\
&\qquad + \frac{2}{N^2} \sum_{1 \leq i < j \leq N} \int_A \dots \int_A g(a_i) g(a_j)\, d\mu(a_1) \dots d\mu(a_N) = \\
&\quad = \frac{1}{N} \int_A g^2\, d\mu = \frac{\mathbb{V}f(X)}{N}.
\end{aligned}$$

□

Dieses Resultat impliziert, daß der absolute Approximationsfehler im Schnitt $\sigma(f)N^{-1/2}$ beträgt, wenn man $\sigma(f) = \sqrt{\mathbb{V}f(X)}$ setzt. Mit Hilfe des zentralen Grenzwertsatzes (Kapitel 7, Theorem 7.1.4.6) kann man genauer abschätzen:

$$\lim_{N \to \infty} \mathbb{P}\left(\frac{c_1 \sigma(f)}{\sqrt{N}} \leq \frac{1}{N} \sum_{i=1}^{N} f(a_i) - \mathbb{E}f(X) \leq \frac{c_2 \sigma(f)}{\sqrt{N}} \right) = \frac{1}{\sqrt{2\pi}} \int_{c_1}^{c_2} e^{-t^2/2}\, dt$$

für beliebige Konstanten c_1 und c_2. Man findet also, daß mit sehr großer Wahrscheinlichkeit der Approximationsfehler $O(N^{-1/2})$ ist, wobei die Konstante im O–Term von $\sigma(f)$ abhängt.

Verwendet man diese statistischen Überlegungen, so kann man den *Monte-Carlo–Schätzwert* für das mehrdimensionale Integral

$$\int_B f(x)\, dx \approx \frac{\lambda_s(B)}{N} \sum_{i=1}^{N} f(x_n)$$

angeben, wobei x_i, $i = 1, \dots, n$, auf B gleichverteilte Zufallszahlen sind. Der Absolutbetrag des Integrationsfehlers ist dann mit höchster Wahrscheinlichkeit $K\lambda_s(B)\sigma(f)N^{-1/2}$ für eine Konstante K. Im Schnitt ist $K = 1$.

Das bemerkenswerte Resultat ist, daß die Fehlerschranke **nicht** von der Dimension s abhängt. Aus diesem Grund sind Monte-Carlo–Methoden den direkten Integrationsmethoden überlegen für $n \geq 5$.

Leider ist der Monte-Carlo–Schätzwert für die Praxis noch unbrauchbar, weil die Berechnung von $\lambda_s(B)$ fast ebenso schwierig sein kann wie die Bestimmung des gesamten Integrals. Glücklicherweise läßt sich das Problem leicht umgehen. O.B.d.A. sei $B \subseteq I_s$, wobei $I_s = [0,1]^s$ der Einheitswürfel in $\mathbb{R}^s$ ist (Im Notfall führt man eine Schiebung und Reskalierung durch und wendet die Transformationsregel für Mehrfachintegrale an). In diesem Fall kann man schreiben

$$\int_B f(x)\, dx = \int_{I_s} f(x) \chi_B(x)\, dx,$$

wobei χ_B die charakteristische Funktion von B ist. Wird für das zweite Integral der Monte-Carlo–Schätzwert gebildet, erhält man den für die Praxis relevanten Monte-Carlo–Schätzwert

$$\int_B f(x)\,dx \approx \frac{1}{N} \sum_{\substack{i=1 \\ x_n \in B}}^{N} f(x_n),$$

wobei $x_1, \ldots, x_N$ gleichverteilte Zufallszahlen auf I_s sind. Auch in diesem Fall gilt trivialerweise die stochastische Fehlerschranke $O(N^{-1/2})$.

Diese Monte-Carlo–Methode erlaubt es also, den Dimensionsteufel auszutreiben. Unglücklicherweise hat das Verfahren einige nicht wegzudiskutierende Nachteile:

- Die Monte-Carlo–Methode bietet nur eine Fehlerschranke auf wahrscheinlichkeitstheoretischer Basis. Man kann niemals *sicher* sein, daß der Fehler wirklich so klein wie vermutet ist. Benötigt man in einer Anwendung sehr zuverlässige Resultate, darf man dieses Faktum nicht ignorieren.
- Die probabilistische Fehlerschranke $O(N^{-1/2})$ gilt schon, wenn man sehr schwache Voraussetzungen an die Regularität von f macht. Im skalaren Fall haben wir gesehen, daß Integrieren üblicherweise „einfacher“ wird, wenn f besonders glatt ist. Dies ist bei der Monte-Carlo–Methode *nicht* der Fall.
- Eine weitere fundamentale Schwierigkeit ist, daß sowohl das Konvergenzresultat als auch die Fehlerabschätzung stark verwenden, daß die x_i unabhängige gleichverteilte Zufallszahlen sind. Wenn wir uns daran erinnern, wie schwierig es ist solche Zufallszahlen zu berechnen (Kapitel 7, Abschnitt 7.2), ja überhaupt zu definieren was das ist, kann man sich vorstellen was das für die Methode bedeutet.

Bevor wir uns jedoch der Verbesserung der Methode widmen, soll noch ein Verfahren vorgestellt werden, mit dem die Fehlerschranke noch etwas verbessert werden kann.

Wir erinnern uns daran, daß die Fehlerschranke proportional zu $\sigma(f)N^{-1/2}$ war. Die Techniken zur *Varianzreduktion* transformieren f so, daß nach der Transformation die Varianz des Integranden kleiner ist.

Zerlegt man den Integrationsbereich $I_s = \bigcup_{i=1}^{k} A_i$ in k disjunkte Mengen mit $\lambda_s(A_i) > 0$, und berechnet man für $i = 1, \ldots, k$ Sequenzen unabhängiger auf A_i gleichverteilter Zufallszahlen $a_1^{(i)}, \ldots, a_{N_i}^{(i)}$, dann kann man den Schätzwert

$$\int_B f(x)\,dx = \sum_{i=1}^{k} \lambda_s(A_i) \int_{A_i} f(x)\,dx \approx \sum_{i=1}^{k} \frac{\lambda_s(A_i)}{N_i} \sum_{j=1}^{N_i} f(a_j^{(i)})$$

verwenden. Der mittlere Approximationsfehler ist dann

$$E_k := \sum_{i=1}^{k} \frac{\lambda_s(A_i)}{N_i} \int_{A_i} \left(f(y) - \frac{1}{\lambda_s(A_i)} \int_{A_i} f(x)\,dx \right)^2 dy,$$

und wir haben das Resultat:

Proposition 8.3.1.2. *Sind die Zahlen $N_i = \lambda(A_i)N$ für $i = 1, \ldots, k$ ganze Zahlen, dann gilt*

$$E_k \leq \frac{\mathbb{V}f(X)}{N}.$$

Beweis. Cauchy–Schwarzsche Ungleichung und rechnen. □

Die *Methode der antithetischen Variablen* kann ebenfalls benutzt werden: Iterativ für jede der s Variablen ersetzt man die Funktion f durch Funktionen g_i gemäß folgendem Schema: Ist f skalar, führt man eine Hilfsfunktion $g(x) := \frac{1}{2}(f(x) + f(1-x))$ ein. Dann gilt:

$$\int_0^1 f(x)\,dx = \int_0^1 g(x)\,dx$$

und für die Varianz von g stimmt folgendes Resultat:

Proposition 8.3.1.3. *Ist f eine monotone, stetige Funktion auf $[0,1]$ und g wie oben, so gilt*

$$\mathbb{V}g(X) \leq \tfrac{1}{2}\mathbb{V}f(X).$$

8.4 Quasi–Monte-Carlo–Methoden

Monte-Carlo–Methoden leiden an den zuvor erwähnten großen Nachteilen, bei denen zwei besonders hervorzuheben sind:

- Es gibt nur probabilistische Konvergenzaussagen und Fehlerschranken.
- Die Güte des Verfahrens ist extrem abhängig von der Güte der verwendeten Zufallszahlen.

Aus diesem Grund versucht man, sobald man ein Monte-Carlo–Verfahren zur Lösung eines Problems gefunden hat, die (Pseudo–)Zufallszahlen wieder aus der Methode zu entfernen und durch deterministisch konstruierte Zahlenfolgen zu ersetzen, sodaß die groben Eigenschaften des Verfahrens nicht verändert werden. Dabei untersucht man genau welche Eigenschaften der Zufallszahlen für das besonders gute Funktionieren der Monte-Carlo–Methode verantwortlich sind. Die probabilistischen Fehlerschranken nimmt man zum Anlaß, *Quasi–Zufallszahlen* zu erzeugen, die in den Fehlerabschätzungen signifikant besser als der Schnitt abschneiden. Auf diese Weise entfernt man nicht nur das Problem der Stochastik in den Konvergenzaussagen sondern man verbessert die Methode auch noch signifikant. Verfahren, die auf diese Weise erzeugt wurden, heißen auch *Quasi–Monte-Carlo–Methoden*.

Das bekannteste Beispiel einer Quasi–Monte-Carlo–Methode ist diejenige, die zur Berechnung mehrdimensionaler Integrale entwickelt wurde. Ihre Geschichte reicht zurück in die frühen 50er Jahre des zwanzigsten Jahrhunderts. Eine ausführliche Darstellung der Quasi–Monte-Carlo–Verfahren kann in [Niederreiter 1992] gefunden werden; aus diesem Buch sind auch große Teile der hier vorgestellten Fakten entnommen.

8.4.1 Quasi–Monte-Carlo–Integration

Die Quasi–Monte-Carlo–Methode zur Berechnung s–dimensionaler Integrale beginnt mit den gleichen Überlegungen wie die Monte-Carlo–Methode. Wir approximieren das Integral durch den *Quasi–Monte-Carlo–Schätzwert*

$$\int_B f(x)\,dx \approx \frac{1}{N}\sum_{\substack{i=1\\ x_i\in B}}^{N} f(x_i)$$

für eine geeignet gewählte Sequenz x_i, $i = 1,\dots,N$. Um die Darstellungen im weiteren zu vereinfachen, nehmen wir o.B.d.A. an, daß $B = I_s$ gilt.

Die Folge x_n muß so gewählt sein, daß

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} f(x_i) = \int_{I_s} f(x)\,dx$$

für eine vernünftig gewählte Klasse von Integranden f, etwa für alle stetigen, gilt.

Ist die Folge $(x_{1,1}, x_{1,2}, \dots, x_{1,s}, x_{2,1}, \dots)$ s–verteilt (siehe Kapitel 7, Definition 7.2.1.4), so ist nach Kapitel 7 Theorem 7.2.1.5 obige Forderung erfüllt. Eine Folge solcher s–dimensionaler Vektoren in I_s nennt man auch 1–*verteilt* oder *gleichverteilt* auf I_s.

Proposition 8.4.1.1. *Ist $(x_n)_n$ eine auf I_s gleichverteilte Folge. Dann sind dazu äquivalent:*

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\chi_J(x_i) = \lambda_s(J)$$

für alle Teilquader J von I_s, und

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} f(x_i) = \int_{I_s} f(x)\,dx$$

für alle auf I_s Riemann–integrierbaren Funktionen f.

Die Diskussion über gleichverteilte Zufallszahlen (Abschnitt 7.2.1) hat gezeigt, daß „zufällig“ und 1–(bzw. s–)verteilt verschiedene Konzepte sind. Es ist zwar jede Zufallsfolge auch gleichverteilt, doch die Umkehrung muß nicht gelten. Die einzige Eigenschaft der Zufallsfolge, die für die Konvergenz des Schätzwertes notwendig ist, ist die gleichmäßige Verteilung der Folgenglieder in I_s. Läßt man den Zufallsaspekt weg und beschränkt man sich auf gleichverteilte Folgen, so kann man wieder versuchen, eine Schranke für den Approximationsfehler des Quasi–Monte-Carlo–Schätzwerts herzuleiten:

Sei $P = (x_1, \ldots, x_N)$ eine Sequenz von Punkten aus I_s.

Definition 8.4.1.2. *Für eine beliebige Teilmenge $B \subseteq I_s$ sei*

$$A(P,B) := A(P, \in B) = \sum_{n=1}^{N} \chi_B(x_n)$$

die Zählfunktion*, die die Anzahl von Punkten in P, die in B liegen feststellt.*

Sei $\mathscr{B}$ eine Familie von Lebesgue–meßbaren Teilmengen von I_s. Die Diskrepanz *von P bezüglich $\mathscr{B}$ ist definiert als*

$$D_N(\mathscr{B},P) = \sup_{B\in\mathscr{B}} \left| \frac{A(P,B)}{N} - \lambda_s(B) \right|.$$

Es gilt immer $0 \le D_N(\mathscr{B},P) \le 1$. Für Speziallfälle von $\mathscr{B}$ erhalten wir die beiden wichtigsten Diskrepanzkonzepte:

Definition 8.4.1.3.

1. *Die* Sterndiskrepanz *$D_N^*(P)$ ist definiert als*

$$D_N^*(P) = D_N(\mathscr{J}^*, P),$$

   wobei $\mathscr{J}^*$ die Familie aller Teilquader von I_s der Form $\prod_{i=1}^{s} [0, u_i[$ ist.

2. *Die* (extreme) Diskrepanz *$D_N(P)$ ist definiert als*

$$D_N(P) = D_N(\mathscr{J}, P),$$

   wobei $\mathscr{J}$ die Familie aller Teilquader von I_s der Form $\prod_{i=1}^{s} [u_i, v_i[$ ist.

Proposition 8.4.1.4. *Für jede Punktsequenz P, die nur aus Punkten in I_s besteht, gilt*

$$D_N^*(P) \le D_N(P) \le 2^s D_N^*(P).$$

Beweis. Vollständige Induktion nach s und einfache Eigenschaften der Zählfunktion. □

Für $s = 1$ kann man explizite Formeln für die Diskrepanz und die Sterndiskrepanz einer Punktsequenz angeben:

Theorem 8.4.1.5 (Niederreiter, de Clerck)**.** *Gilt $0 \le x_1 < x_2 < \cdots < x_n \le 1$, dann sind*

$$D_N^*(P) = \frac{1}{2N} + \max_{1\le n\le N} \left| x_n - \frac{2n-1}{2N} \right|.$$

und

$$D_N(P) = \frac{1}{N} + \max_{1\le n\le N} \left(\frac{n}{N} - x_n \right) - \min_{1\le n\le N} \left(\frac{n}{N} - x_n \right).$$

Beweis. Siehe [Niederreiter 1992, 2.6, 2.7]. □

Sei S eine Folge von Elementen in I_s ($S \in I_s^{\mathbb{N}}$). Wir schreiben abkürzend

$$D_N(S) := D_N(P_N^S), \quad D_N^*(S) := D_N^*(P_N^S),$$

wobei P_N^S die Punktsequenz ist, die aus den ersten N Folgengliedern von S besteht. Mit dieser Schreibweise gilt der folgende zentrale Satz:

Theorem 8.4.1.6. *Sei S eine Folge von Zahlen aus I_s. Dann sind die folgenden Aussagen äquivalent:*

1. *S ist gleichverteilt auf I_s.*
2. *$\lim_{N\to\infty} D_N(S) = 0$.*
3. *$\lim_{N\to\infty} D_N^*(S) = 0$.*

Beweis. [Kuipers, Niederreiter 1974] □

Wollen wir also Folgen konstruieren, für die der Quasi–Monte-Carlo–Schätzwert gegen das Integral konvergiert für $N \to \infty$, so müssen diese Folgen $\lim_{N\to\infty} D_N(S) = 0$ erfüllen. Wir werden jedoch gleich sehen, daß die Diskrepanz auch im Zusammenhang mit der Fehlerabschätzung wichtig ist.

Für $s = 1$ gibt es folgendes klassisches Resultat

Proposition 8.4.1.7 (Koksma–Ungleichung)**.** *Hat $f : [0,1] \to \mathbb{R}$ beschränkte Variation $V(f)$ auf $[0,1]$, so gilt für jede Sequenz $x_1, \ldots, x_N \in [0,1]$*

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_0^1 f(x)\,dx \right| \leq V(f) D_N^*(x_1, \ldots, x_N).$$

Beweis. [Koksma 1942/43] □

Eine Abschätzung dieser Art ist genau was wir uns wünschen. Der Approximationsfehler für $s = 1$ hängt ab von der Diskrepanz der Punktsequenz, die in gewissem Maß die Verteilungseigenschaften der Sequenz mißt, und von der Totalvariation der Funktion f, eine Größe, die ein guter Ersatz für die Glattheit von f ist. Sie gibt an wie stark die Funktion f über I_s variiert. Eine genauere Untersuchung zeigt auch, daß diese Ungleichung das Beste ist, was man an Abschätzung erwarten kann, selbst wenn man $f \in C^\infty[0,1]$ voraussetzt.

Zu hoffen bleibt, daß es eine Verallgemeinerung dieser Ungleichung auf $s > 1$ gibt, die ähnlich verwendbar ist. Dazu müssen wir jedoch zuerst den Begriff „Totalvariation“ auf $s > 1$ verallgemeinern:

Definition 8.4.1.8. *Die* Variation von f im Sinn von Vitali *ist definiert als*

$$V^{(s)}(f) = \sup_{\mathscr{P}} \sum_{J\in\mathscr{P}} |\Delta(f,J)|,$$

wobei das Supremum über alle Partitionen $\mathscr{P}$ von I_s in Teilquader gebildet wird, und

$$\Delta(f,J) = \sum_{i_1=0}^{1} \cdots \sum_{i_s=0}^{1} (-1)^{\sum_{k=1}^{s} i_k} f(u_1^{(i_1)}, \ldots, u_s^{(i_s)})$$

für $J = \prod_{k=1}^{s} [u_k^{(0)}, u_k^{(1)}]$.

Ist f eine C^s–Funktion, so gilt

$$V^{(s)}(f) = \int_0^1 \ldots \int_0^1 \left| \frac{\partial^s f}{\partial u_1 \ldots \partial u_s} \right| du_1 \ldots du_s.$$

Für das zu 8.4.1.7 analoge Resultat muß jedoch auch die Variation entlang mancher Ränder von I_s herangezogen werden.

Definition 8.4.1.9. *Für $1 \leq k \leq s$ und $1 \leq i_1 < i_2 < \cdots < i_k \leq s$ sei $V^{(k)}(f; i_1, \ldots, i_k)$ die Variation im Sinn von Vitali von $f|_{S_{i_1,\ldots,i_k}}$, wobei*

$$S_{i_1,\ldots,i_k} := \{(u_1, \ldots, u_s) \in I_s \,|\, u_j = 1 \text{ für } j \neq i_1, \ldots, i_k\}.$$

Mit dieser Notation ist die Totalvariation von f *oder die* Variation im Sinn von Hardy und Krause

$$V(f) = \sum_{k=1}^{s} \sum_{1\leq i_1<\cdots<i_k\leq s} V^{(k)}(f; i_1, \ldots, i_k).$$

f heißt von beschränkter Variation*, wenn $V(f) < \infty$ gilt.*

Die Verallgemeinerung von Proposition 8.4.1.7 wurde von Hlawka bewiesen und lautet:

Theorem 8.4.1.10 (Koksma–Hlawka–Ungleichung)**.** *Ist f von beschränkter Variation $V(f)$ auf I_s, dann gilt für jede Sequenz $P = (x_1, \ldots, x_N)$*

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_{I_s} f(x)\, dx \right| \leq V(f) D_N^*(P).$$

Weiters existiert für jede solche Sequenz P und jedes $\varepsilon > 0$ eine C^∞–Funktion f mit $V(f) = 1$ und

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_{I_s} f(x)\, dx \right| > D_N^*(P) - \varepsilon.$$

Beweis. [Hlawka 1961] □

Diese Ungleichung ist das zentrale Resultat für die Theorie der Quasi–Monte-Carlo–Integration. Sie zeigt nämlich den Weg vor, den man einschlagen muß, um Punktfolgen zu konstruieren, für die der Quasi–Monte-Carlo–Schätzwert sehr gut ist. Diese Folgen sollten möglichst geringe Sterndiskrepanz aufweisen (und damit auch kleine Diskrepanz). Solche Folgen nennt man üblicherweise *Folgen geringer Diskrepanz*. Im folgenden Abschnitt wollen wir solche Folgen konstruieren.

8.4.2 Quasi–Zufallszahlen

Quasi–Zufallszahlen oder Zahlenfolgen geringer Diskrepanz kann man auf viele verschiedene Arten konstruieren. Die *Niederreiter*–Folgen gehören heute zu den besten bekannten Folgen dieser Art. Die genaue Konstruktionsmethode ist mathematisch aufwendig und kann daher hier nicht ausführlich behandelt werden. Sie kann jedoch bei Bedarf in [Niederreiter 1992, Chapter 4] nachgelesen werden.

Klassische Konstruktionen

Im eindimensionalen Fall ist es sehr einfach, das Minimum der Diskrepanzen D_N und D_N^* zu bestimmen. Wegen Theorem 8.4.1.5 gilt immer

$$D_N^*(x_1, \ldots, x_N) \geq \frac{1}{2N},$$

mit Gleichheit genau für

$$x_n = \frac{2n-1}{2N}.$$

Das Quasi–Monte-Carlo–Verfahren mit dieser Punktsequenz ist gerade

$$\int_0^1 f(x)\, dx \approx \frac{1}{N} \sum_{n=1}^{N} f\left(\frac{2n-1}{2N} \right),$$

die klassische Mittelpunktsregel für das Intervall $[0,1]$.

Während für Punktmengen P der Größe N ein Verhalten der Art $D_N(P) = O(N^{-1})$ möglich ist, existiert keine *Folge* S von Elementen von $[0,1]$ mit $D_N(S) = O(N^{-1})$. Im Gegenteil, es gibt das Phänomen der *Irregularität der Verteilung*, das besagt, daß $D_N(S)$ unendlich oft von größerer Ordnung ist.

Genauer existiert das Resultat [Schmidt 1972], welches besagt, daß eine Konstante c existiert mit

$$D_N(S) \geq c N^{-1} \log N$$

für unendlich viele N. Der beste bekannte Wert für c ist $c = 0.12$. Für die Sterndiskrepanz gilt daher

$$D_N^*(S) \geq 0.06 N^{-1} \log N$$

für unendlich viele N.

Wollen wir also Folgen geringer Diskrepanz erzeugen, können wir nichts besseres als

$$D_N^*(S) = O(N^{-1}\log N)$$

im eindimensionalen Fall erwarten. Die folgenden beiden Konstruktionen zeigen, daß diese Größenordnung tatsächlich erreicht werden kann.

Definition 8.4.2.1. *Sei n eine natürliche Zahl und $b \geq 2$ eine weitere natürliche Zahl. Dann gibt es eindeutige Zahlen $a_j(n) \in \mathbb{Z}_b$, $j = 0,1,\ldots$, sodaß $a_j(n) = 0$ für alle genügend großen j und*

$$n = \sum_{j=0}^{\infty} a_j(n) b^j.$$

Z_b ist dabei der Restklassenring modulo b, $\mathbb{Z}_b = \{0,\ldots,b-1\}$.

Mit obiger Notation ist die Inversionsfunktion in Basis b *definiert durch*

$$\Phi_b(n) = \sum_{j=0}^{\infty} a_j(n) b^{-j-1}.$$

$\Phi_b : \mathbb{N}_0 \to [0,1[$.

Definition 8.4.2.2. *Die* van der Corput–Folge in Basis b *ist die Folge $S_b = (x_n)_n$ mit*

$$x_n = \Phi_b(n)$$

für $n \geq 0$.

Die verallgemeinerte van der Corput–Folge in Basis b bzgl. σ *ist die Folge $S_b^\sigma = (x_n)_n$ mit*

$$x_n = \sum_{j=0}^{\infty} \sigma(a_j(n)) b^{-j-1}$$

für eine fixe Permutation σ der Menge $\{0,\ldots,d-1\}$.

Theorem 8.4.2.3 (Faure)**.** *Für die van der Corput–Folge S_b gilt*

$$\overline{\lim_{N\to\infty}} \frac{ND_N^*(S_b)}{\log N} = \overline{\lim_{N\to\infty}} \frac{ND_N(S_b)}{\log N} = \begin{cases} \frac{b^2}{4(b+1)\log b} & b \text{ gerade}, \\ \frac{b-1}{4\log b} & b \text{ ungerade}. \end{cases}$$

Alle van der Corput–Folgen erfüllen also $D_N(S_b) = O(N^{-1}\log N)$, und S_3 ist asymptotisch die beste van der Corput–Folge.

Die verallgemeinerten van der Corput–Folgen lassen sich einfach rekursiv generieren durch die Vorschrift

$$\begin{aligned} x_0 &= \frac{\sigma(0)}{b-1} \\ x_{bn+r} &= \tfrac{1}{b}(x_n + \sigma(r)), \quad n \geq 0, \quad 0 \leq r \leq b-1. \end{aligned}$$

Sie erfüllen alle $D_N(S_b^\sigma) = O(N^{-1}\log N)$.

Die besten bekannten verallgemeinerten van der Corput–Folgen wurden auch von Faure konstruiert. Die erste hat Basis $b = 36$ eine bestimmte Permutation von $\mathbb{Z}_{36}$ und erfüllt

$$\overline{\lim_{N\to\infty}} \frac{ND_N(S_b^\sigma)}{\log N} = \frac{23}{35\log 6} = 0.366\ldots$$

Für die zweite ist $b = 12$ und

$$\overline{\lim_{N\to\infty}} \frac{ND_N^*(S_b^\sigma)}{\log N} = \frac{1919}{3454\log 12} = 0.223\ldots$$

Eine andere Klasse von Folgen geringer Diskrepanz sind die *irrationalen Folgen modulo 1*. Für reelle Zahlen sei

$$\{u\} := u - \lfloor u \rfloor$$

die Zahl $u \mod 1$.

Definition 8.4.2.4. *Für eine irrationale Zahl z sei $S(z) = (x_n)_n$ mit*

$$x_n = \{nz\}, \quad \textit{für } n \geq 0.$$

die irrationale Folge modulo 1 zu z.

Theorem 8.4.2.5. *Haben wir eine Kettenbruchentwicklung*

$$z = [a_0; a_1, a_2, \ldots]$$

von z gegeben mit $\sum_{i=1}^m a_i = O(m)$, so gilt $D_N(S(z)) = O(N^{-1}\log N)$ für alle $N \geq 2$.
Insbesondere ist die Voraussetzung erfüllt, wenn z eine algebraische irrationale Zahl ist.

Für $s > 1$ können Folgen angegeben werden, die einfache Verallgemeinerungen des skalaren Falls sind:

Definition 8.4.2.6. *Sei $u = (u_1, \ldots, u_s) \in \mathbb{R}^s$. Der Teil von $u \mod 1$ ist definiert als*

$$\{u\} := (\{u_1\}, \ldots, \{u_s\}) \in I_s.$$

Seien $1, z_1, \ldots, z_s$ rational unabhängig, und setzen wir $z = (z_1, \ldots, z_n)$. Dann ist die Folge $S(z) = (x_n)_n$ mit

$$x_n = \{nz\}$$

eine irrationale Folge modulo 1 zu z.

Es gibt viele Untersuchungen über Folgen dieses Typs. Es ist bekannt, daß die irrationalen Folgen modulo 1 gleichverteilt sind. Schmidt hat bewiesen, daß für jedes $\varepsilon > 0$ gilt $D_N(S(z)) = O(N^{-1}(1+\log N)^{s+1+\varepsilon})$ für fast alle $z \in \mathbb{R}^s$. Es ist jedoch kein Vektor z für $s > 1$ bekannt, für den $D_N(S(z)) = O(N^{-1}(1+\log N)^{s+1})$ gilt. Niederreiter hat bewiesen, daß für algebraische Vektoren z die Diskrepanz $D_N(S(z)) = O(N^{-1+\varepsilon})$ erfüllt für alle $\varepsilon > 0$.

Bessere Resultate kann man jedoch für die Verallgemeinerung der van der Corput–Folgen beweisen.

Definition 8.4.2.7. *Eine* Halton–Folge zu den Basen $b_1, \ldots, b_s$ *ist eine Folge $S_{b_1,\ldots,b_s}$ mit den Folgengliedern*

$$x_n = (\Phi_{b_1}(n), \ldots, \Phi_{b_s}(n)).$$

Die Basen b_i sind dabei ganze Zahlen mit $b_i \geq 2$.

Folgendes Resultat besagt, daß Halton–Folgen Folgen mit geringer Diskrepanz sind.

Theorem 8.4.2.8. *Die Halton–Folge $S_{b_1,\ldots,b_s}$ mit paarweise teilerfremden Basen b_i erfüllt*

$$D_N^*(S_{b_1,\ldots,b_s}) < \frac{s}{N} + \frac{1}{N}\prod_{i=1}^s \left(\frac{b_i - 1}{2\log b_i}\log N + \frac{b_i + 1}{2}\right)$$

für alle $N \geq 1$.

Es gilt also

$$D_N^*(S_{b_1,\ldots,b_s}) = A(b_1,\ldots,b_s)N^{-1}(\log N)^s + O(N^{-1}(\log N)^{s-1}) = O(N^{-1}(\log N)^s)$$

mit

$$A(b_1,\ldots,b_s) = \prod_{i=1}^{s} \frac{b_i - 1}{2\log b_i}.$$

Die asymptotisch beste Halton–Folge ist jene mit $b_i = p_i$, wobei p_i die i–te Primzahl bezeichnet. Für diese Folge $S_{p;s}$ gilt $A_s := A(p_1,\ldots,p_s)$ ist der minimal mögliche Koeffizient des Führungsterms der asymptotischen Entwicklung.

Beweis. [Niederreiter 1992, 3.6] □

Das Verhalten $D_N(S) = O(N^{-1}(\log N)^s)$ ist das beste, das man für s–dimensionale Folgen erwarten kann. Beschränkt man sich auf Punktsequenzen der Größe N, so geht es noch ein wenig besser:

Definition 8.4.2.9. *Seien $s \geq 2$, $N \geq 1$, $b_i \geq 2$ für $i = 1,\ldots,s$. Die N–elementige* Hammersley–Punktmenge *$H_{b_1,\ldots,b_s}(N)$ in den Basen $b_1,\ldots,b_s$ ist definiert als*

$$x_n = \left(\frac{n}{N}, \Phi_{b_1}(n), \ldots, \Phi_{b_s}(n)\right) \in I_s,$$

für $n = 0,\ldots,N-1$.

Für die Hammersley–Punktmenge gilt

$$D_N^*(H_{b_1,\ldots,b_s}(N)) = O(N^{-1}(\log N)^{s-1}).$$

Der Koeffizient des führenden Terms ist wiederum $A(b_1,\ldots,b_s)$.

Die Verwendung von Hammersley–Punktmengen und Halton–Folgen in der Quasi–Monte-Carlo–Integration führt zu einer dramatischen Verbesserung der Fehlerschranke. Ist im Monte-Carlo–Fall der Approximationsfehler *im Schnitt* $O(N^{-1/2})$, so ist er bei der Quasi–Monte-Carlo–Integration bei der Verwendung von Folgen geringer Diskrepanz (etwa Halton–Folgen) *immer* $O(N^{-1}(\log N)^s)$!

Leider haben die Halton–Folgen einen unangenehmen Nachteil. Dieser hängt mit dem Koeffizienten A_s des Führungsterms der asymptotischen Entwicklung der bestmöglichen Folge zusammen. Aus dem Primzahlensatz folgt nämlich sofort, daß

$$\lim_{s\to\infty} \frac{\log A_s}{s\log s} = 1$$

gilt. $A_s = O(e^{s\log s})$ wächst also *superexponentiell* für $s \to \infty$. Dieser rasche Anstieg von A_s macht für große s die Fehlerschranken aus Theorem 8.4.2.8 praktisch unbrauchbar.

Die Theorie, die der Konstruktion besserer Folgen zugrunde liegt, ist die Theorie der (t,s)–Folgen. In [Niederreiter 1992, Chapter 4] wird eine spezielle $(0,s)$–Folge konstruiert, die *Niederreiter–Folge*, die nicht den Nachteil eines superexponentiell wachsenden Führungskoeffizienten hat.

Die zugrunde liegenden Definitionen seien hier aufgeführt:

Definition 8.4.2.10. 1. *Sei $s \geq 1$ und $b \geq 2$. Ein Teilquader E von I_s der Form*

$$E = \prod_{i=1}^{s} [a_i b^{-d_i}, (a_i+1)b^{-d_i}[$$

mit $a_i, d_i \in \mathbb{Z}$, $d_i \geq 0$ und $0 \leq a_i < b^{d_i}$ für $i = 1,\ldots,s$ heißt elementarer Quader in Basis b.

2. *Seien $0 \leq t \leq m$ ganze Zahlen. Ein (t,m,s)–*Netz in Basis b *ist eine Punktmenge P von b^m Punkten in I_s mit $A(P,E) = b^t$ für jeden elementaren Quader E mit $\lambda_s(E) = b^{t-m}$. Das ist gleichbedeutend mit dem Faktum, daß die Diskrepanz $D_{b^m}(\mathscr{E},P) = 0$ ist für die Familie $\mathscr{E}$ der elementaren Quader.*

3. *Sei $t \geq 0$ eine ganze Zahl. Eine Folge $S = (x_0, x_1, \dots)$ von Elementen von I_s heißt* (t,s)–Folge in Basis *b, wenn für alle $k \geq 0$ und $m > t$ die Punktmenge*

$$P_{k,m} := \{x_n \in S \,|\, kb^m \leq n < (k+1)b^m\}$$

ein (t,m,s)–Netz in Basis b ist.

Die van der Corput–Folgen in Basis b sind dann $(0,1)$–Folgen in Basis b.

Ist $t \leq u \leq m$, so ist nach Definition jedes (t,m,s)–Netz in Basis b automatisch auch (u,m,s)–Netz in Basis b, und jede (t,s)–Folge ist auch (u,s)–Folge. Daher ist klar, daß kleinere Werte von t stärkere Eigenschaften beschreiben.

Theorem 8.4.2.11. *Die Sterndiskrepanz eines (t,m,s)–Netzes P in Basis $b \geq 3$ erfüllt*

$$ND_N^*(P) \leq b^t \sum_{i=0}^{s-1} \binom{s-1}{t} \binom{m-t}{i} \left\lfloor \frac{b}{2} \right\rfloor^i.$$

Es gilt also

$$D_N^*(P) \leq \frac{1}{N} \frac{b^t}{(s-1)!} \left(\frac{\lfloor b/2 \rfloor}{\log b} \right)^{s-1} (\log N)^{s-1} + O(b^t N^{-1} (\log N)^{s-2}).$$

Die Sterndiskrepanz einer (t,s)–Folge S in Basis $b \geq 3$ erfüllt

$$D_N^*(S) \leq \frac{1}{N} \frac{1}{s!} b^t \frac{b-1}{2\lfloor b/2 \rfloor} \left(\frac{\lfloor b/2 \rfloor}{\log b} \right)^{s} (\log N)^{s} + O(b^t N^{-1} (\log N)^{s-1}).$$

(t,m,s)–Netze und (t,s)–Folgen sind also genau die Folgen geringer Diskrepanz, die wir suchen.

Die mit Mitteln der analytischen Zahlentheorie erzeugte Niederreiter–Folge N_s erfüllt

$$D_N^*(N_s) \leq C_s N^{-1} (\log N)^s + O(N^{-1} (\log N)^{s-1}).$$

Der Koeffizient des Führungsterms erfüllt

$$C_s < \frac{1}{s!} \left(\frac{s}{\log(2s)} \right)^s,$$

woraus man mit Hilfe der Stirlingschen Formel die Abschätzung

$$\overline{\lim_{s \to \infty}} \frac{\log C_s}{s \log\log s} \leq -1$$

herleiten kann. Der Führungskoeffizient der Niederreiterfolge ist also $C_s = O(e^{-s \log\log s})$ superexponentiell konvergent gegen 0 für $s \to \infty$. Die Fehlerschranken für den Quasi–Monte-Carlo–Schätzwert werden also immer **besser** für steigende Dimension! Ein Vergleich zwischen den Führungskoeffizienten der Halton–Folgen und der Niederreiter–Folge kann in Tabelle 8.1 gefunden werden.

s	A_s	C_s	s	A_s	C_s
			11	$3.37 \cdot 10^{3}$	$8.12 \cdot 10^{-5}$
2	$6.57 \cdot 10^{-1}$	$2.60 \cdot 10^{-1}$	12	$1.68 \cdot 10^{4}$	$5.60 \cdot 10^{-5}$
3	$8.16 \cdot 10^{-1}$	$1.26 \cdot 10^{-1}$	13	$9.06 \cdot 10^{4}$	$1.01 \cdot 10^{-5}$
4	$1.26 \cdot 10^{0}$	$8.58 \cdot 10^{-2}$	14	$5.06 \cdot 10^{5}$	$2.19 \cdot 10^{-5}$
5	$2.62 \cdot 10^{0}$	$2.47 \cdot 10^{-2}$	15	$3.02 \cdot 10^{6}$	$4.42 \cdot 10^{-6}$
6	$6.14 \cdot 10^{0}$	$1.86 \cdot 10^{-2}$	16	$1.98 \cdot 10^{7}$	$7.80 \cdot 10^{-7}$
7	$1.73 \cdot 10^{1}$	$4.11 \cdot 10^{-3}$	17	$1.41 \cdot 10^{8}$	$1.30 \cdot 10^{-7}$
8	$5.30 \cdot 10^{1}$	$2.99 \cdot 10^{-3}$	18	$1.03 \cdot 10^{9}$	$8.47 \cdot 10^{-8}$
9	$1.86 \cdot 10^{2}$	$6.05 \cdot 10^{-4}$	19	$8.06 \cdot 10^{9}$	$1.36 \cdot 10^{-8}$
10	$7.71 \cdot 10^{2}$	$4.28 \cdot 10^{-4}$	20	$6.62 \cdot 10^{10}$	$3.28 \cdot 10^{-8}$

Tabelle 8.1: Werte für A_s und C_s für $s = 2, \ldots, 20$

9 Numerik partieller Differentialgleichungen

9.1 Grundlagen

9.1.1 Lineare partielle Differentialgleichungen erster Ordnung

9.2 Lineare partielle Differentialgleichungen

9.2.1 Elliptische Randwertprobleme

Die Membrangleichungen

9.2.2 Parabolische Anfangsrandwertprobleme

Die Wärmeleitungsgleichung

9.2.3 Hyperbolische Anfangsrandwertprobleme

Die Wellengleichung

9.3 Nichtlineare partielle Differentialgleichungen

9.3.1 Elliptische Randwertprobleme

9.3.2 Parabolische Anfangsrandwertprobleme

Die Diffusionsgleichung

9.3.3 Hyperbolische Anfangsrandwertprobleme

Die Burgers Gleichung

Die Gleichungen zur Gasdynamkik

Die Navier–Stokes Gleichung

9.3.4 Typfreie Differentialgleichungen

9.3.5 Differentialgleichungen höherer Ordnung

.

Literaturverzeichnis

[Rump 1988] Rump, S.M., *Algorithm for verified inclusions—theory and practice*, in "Reliability in Computing" (Moore, R.E., ed.), Academic Press, San Diego, 1988.

[Hansen 1992] Hansen, E., *Global Optimization using Interval Analysis*, Monographs in Pure and Applied Mathematics, Marcel Dekker, New York, 1992.

[Überhuber 1995a] Überhuber, C., *Computernumerik 1*, Springer Verlag, Berlin Heidelberg New York, 1995.

[Überhuber 1995b] Überhuber, C., *Computernumerik 2*, Springer Verlag, Berlin Heidelberg New York, 1995.

[Goldberg 1991] Goldberg, D., *What every Computer Scientist should Know About Floating-Point Arithmetic*, ACM Computing Surveys **23** (1991), 17–27.

[Reichel, Zöchling 1990] Reichel, H.-C., Zöchling, J., *Tausend Gleichungen - und was nun? - Computertomographie als Einstieg in ein aktuelles Thema des Mathematikunterrichtes*, Didaktik der Mathematik **4** (1990), 245–270.

[Stoer 1994a] Stoer, J., *Numerische Mathematik 1*, Springer Verlag, Berlin Heidelberg New York, 1994.

[Stoer 1994b] Stoer, J., *Numerische Mathematik 2*, Springer Verlag, Berlin Heidelberg New York, 1994.

[Ören 1979] Ören, T.I., *Concepts for Advanced Computer Assisted Modelling*, ACM Computing Surveys(1979),

[Skeel 1980] Skeel, R.D., *Iterative Refinement Implies Numerical Stability for Gaussian Elimination*, Math. Comp. **35** (1980), 817–832.

[Higham 1996] Higham, N.J., *Accuracy and Stability of Numerical Algorithms*, SIAM Publications, Philadelphia, 1996.

[Trefethen, Bau 1997] Trefethen, L.N., Bau, III D., *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[Golub, Van Loan 1996] Golub, G.H., Van Loan, C.F., *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, 1996.

[Schwarz 1986] Schwarz, H.R., *Numerische Mathematik*, 2. Auflage, B.G. Teubner, Stuttgart, 1986.

[Heuser 1986/1] Heuser, H., *Lehrbuch der Analysis, Teil 1*, B.G. Teubner, Stuttgart, 1986.

[Heuser 1986/2] Heuser, H., *Lehrbuch der Analysis, Teil 2*, B.G. Teubner, Stuttgart, 1986.

[Bulirsch, Rutishauser 1968] Bulirsch, R., Rutishauser, H., *Interpolation und genäherte Quadratur*, in "Mathematische Hilfsmittel des Ingeneurs, Part III" (Sauer, R., Szabo, I., eds.), Springer Verlag, Berlin New York Heidelberg, 1968.

[Cooley, Tukey 1965] Cooley, J.W., Tukey, J.W., *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. **19** (1965), 297–301.

[Gentleman, Sande 1966] *Fast Fourier transforms — For fun and profit*, in "Proc. AFIPS 1966 Fall Joint Computer Conference",**29**, Spartan Books, Washington D.C., 1966.

[Good 1958] Good, I.J., *The interaction algorithm and practical Fourier series*, J. Roy. Statist. Soc. Ser. B **20** (1958), 361–372.; Addendum: **22**(1960), 372–375

[Risler 1992] Risler, J.–J., *Mathematical Methods for CAD*, Cambridge University Press, Cambridge, 1992.

[Rutishauser 1960] Rutishauser, H., *Bemerkungen zur glatten Interpolation*, ZaMP **11** (1960), 508–513.

[Böhmer 1974] Böhmer, K., *Spline–Funktionen*, B.G. Teubner, Stuttgart, 1974.

[de Boor 1978] Boor, C. de, *A Practical Guide to Splines*, Springer, Berlin New York Heidelberg, 1978.

[Schoenberg, Whitney 1953] Schoenberg, I.J., Whitney, A., *On Polya frequency functions, III: The positivity of translation determinants with an application to the interpolation problem by spline curves*, Trans. Amer. Math. Soc. **74** (1953), 246–259.

[Nielson 1974] Nielson, G. M., *Some Piecewise Polynomial Alternatives to Splines Under Tension*, in "Computer Aided Geometric Design" (Barnhill, R. E., Riesenfeld, R. F., eds.), Academic Press, New York San Francisco London, 1974.

[Kronrod 1965] Kronrod, A. S., *Nodes and Weights of Quadrature Formulas*, Consultants Bureau, New York(1965),

[Gander 1985] Gander, W., *Computermathematik*, Birkhäuser, Basel, 1985.

[Broyden et al. 1970] Broyden, C. G., Dennis, J. E., Moré, J. J., *On the local and superlinear convergence of quasi–Newton methods*, J. Inst. Math. Appl. **12** (1970), 223-245.

[Gill et al. 1974] Gill, P. E., Golub, G. H., Murray, W., Saunders, M. A., *Methods for modifying matrix factorizations*, Math. Comput. **28** (1974), 505–535.

[Oren, Luenberger 1974] Oren, S. S., Luenberger, D. G., *Self–scaling variable metric (SSVM) algorithms. I. Criteria and sufficient conditions for scaling a class of algorithms*, Manage. Sci. **20** 845–862.

[Knuth 1969] Knuth, D. E., *The Art of Computer Programming*, Addison–Wesley, Reading, Massachusetts, 1969.

[Hermeline 1982] Hermeline, F., *Triangulation automatique d'un polyèdre en dimension N*, RAIRO, Analyse numérique **16, 3** (1982), 211–242.

[Bronstein, Semendjajev 1989] Bronstein, I. N., Semendjajev, K. A., *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun Frankfurt/Main, 1989.

[Hart 1968] Hart, J. F., et al., *Computer Approximations*, John Wiley, New York, 1968.

[Metropolis, Ulam, 1949] Metropolis, N., Ulam, S. M., *The Monte Carlo method*, J. Amer. Statist. Assoc. **44** (1949), 335–341.

[Niederreiter 1992] Niederreiter, H., *Random Number Generation and Quasi–Monte Carlo Methods*, SIAM, Philadelphia, 1992.

[Kuipers, Niederreiter 1974] Kuipers, L., Niederreiter, H., *Uniform Distribution of Sequences*, John Wiley, New York, 1974.

[Koksma 1942/43] Koksma, J. F., *Een algemeene stelling uit de theorie der gelijkmatige verdeeling modulo 1*, Mathematica B (Zutphen) **11** (1942/43), 7–11.

[Hlawka 1961] Hlawka, E., *Funktionen von beschränkter Variation in der Theorie der Gleichverteilung*, Ann. Mat. Pura Appl. **54** (1961), 325–333.

[Schmidt 1972] Schmidt, W. M., *Irregularities of distribution. VII*, Acta Arith. **21** (1972), 45–50.

[IEEE 754–1985] *Standard for Binary Floating Point Arithmetic*, ANSI/IEEE Standard **754** (1985),

[Neumaier 2000] Neumaier, A., *Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, 2000.

[Neumaier 1990] Neumaier, A., *Interval Methods for Systems of Equations*, Encyclopedia of Mathematics and its Applications 37, Cambridge University Press, Cambridge, 1990.

[Neumaier 1974] *Rundungsfehleranalyse einiger Verfahren zur Summation endlicher Summen*, Z. Angew. Math. Mech. **54** (1974), 39–51.

[van der Sluis 1969] *Condition numbers and equilibration of matrices*, Numer. Math. **14** (1969), 14–23.

[Prager, Oettli 1964] *Compatibility of approximate solutions of linear equations with given error bounds for coefficients and right hand sides*, Numer. Math. **6** (1964), 405–409.

[Wilkinson 1968] *À Priori Error Analysis of Algebraic Processes*, Proc. International Congress Math.(1968), 629–639.

[Neumaier 1998] Neumaier, A., *Formstabile Interpolation*, Preprint(1998),

[Opitz 1958] *Gleichungsauflösung mittels einer speziellen Interpolation*, Z. Angew. Math. Mech. **38** (1958), 276–277.

Liste der Algorithmen