# Mathematische Methoden für Computer Vision, Robotik und Grafik

**Kursnotizen für CS 205A, Herbst 2013**

Justin Solomon
Abteilung für Computerwissenschaften
Universität in Stanford

# Inhalt

# Teil I

# Vorrunden

# Kapitel 0

# Mathematik-Rezension

In diesem Kapitel werden wir relevante Begriffe aus der linearen Algebra und der Multivariablenrechnung besprechen, die in unsere Diskussion über Rechentechniken einfließen. Es ist als Überblick über Hintergrundmaterial gedacht, wobei der Schwerpunkt auf Ideen und Interpretationen liegt, die in der Praxis häufig anzutreffen sind. Das Kapitel kann bedenkenlos übersprungen oder als Nachschlagewerk für Schüler mit ausgeprägteren Mathematikkenntnissen verwendet werden.

## 0.1 Vorbereitungen: Zahlen und Mengen

Anstatt algebraische (und manchmal philosophische) Diskussionen wie „Was ist eine Zahl?“ zu berücksichtigen, werden wir uns auf Intuition und mathematischen gesunden Menschenverstand verlassen, um einige Mengen zu

definieren: • Die natürlichen Zahlen N = {1, 2, **3 , . . .}**

• Die ganzen Zahlen **Z** = {. . . , ÿ2, ÿ1, 0, 1, 2, . . .}

• Die rationalen Zahlen **Q** = {a/b : a, b ÿ Z} • Die [1]

reellen Zahlen **R , die Q** sowie irrationale Zahlen wie $\sqrt{ÿ}$ und $\sqrt{ÿ 2}$ umfassen • Die komplexen Zahlen **C** = $\overline{\phantom{x}}$

{a + bi : a, b ÿ R }, wobei wir davon ausgehen, dass $\sqrt{i\, i} = \sqrt{ÿ\, ÿ1}$ erfüllt.

Es muss anerkannt werden, dass unsere Definition von **R** alles andere als streng ist. Die Konstruktion der reellen Zahlen kann ein wichtiges Thema für Praktiker von Kryptographietechniken sein, die alternative Zahlensysteme verwenden, aber diese Feinheiten sind für die vorliegende Diskussion irrelevant.

Wie alle anderen Mengen können N, Z, Q, R und **C** mithilfe generischer Operationen manipuliert werden, um neue Zahlenmengen zu erzeugen. Denken Sie insbesondere daran, dass wir das „euklidische Produkt“ zweier Mengen A und B definieren können als

$$A \times B = \{(a, b) : a ÿ A \text{ und } b ÿ B\}.$$

Wir können Potenzen von Mengen durch Schreiben ermitteln

$$A^N = \underbrace{A \times A \times \cdots \times A}_{n \text{ mal}}.$$

[1] Dies ist das erste von vielen Malen, dass wir die Notation {A : B} verwenden ; Die geschweiften Klammern sollten eine Menge bezeichnen und der Doppelpunkt kann als „so dass“ gelesen werden. Beispielsweise kann die Definition von **Q** als „die Menge der Brüche a/b, sodass a und b ganze Zahlen sind“ gelesen werden . Als zweites Beispiel könnten wir **N** = {n ÿ **Z :** n > 0} schreiben.

Diese Konstruktion ergibt das, was in den kommenden Kapiteln zu unserem Lieblingszahlensatz werden wird:

$$\mathbf{R}^N = \{(a1, a2, \ldots, an) : ai \ddot{y} \mathbf{R} \text{ für alle } i\}$$

# 0,2 Vektorräume

Einführungskurse in die lineare Algebra könnten problemlos den Titel „Einführung in endlichdimensionale Vektorräume“ tragen. Obwohl die Definition eines Vektorraums abstrakt erscheinen mag, werden wir viele konkrete Anwendungen finden, die alle die formalen Aspekte erfüllen und daher von der Maschinerie, die wir entwickeln werden, profitieren können.

## 0.2.1 Vektorräume definieren

Wir beginnen mit der Definition eines Vektorraums und geben eine Reihe von Beispielen:

**Definition 0.1** (Vektorraum). Ein Vektorraum ist eine Menge V , die durch Skalarmultiplikation und -addition abgeschlossen ist.

Für unsere Zwecke ist ein Skalar eine Zahl in R, und die Additions- und Multiplikationsoperationen erfüllen die üblichen Axiome (Kommutativität, Assoziativität usw.). Normalerweise ist es einfach, Vektorräume in freier Wildbahn zu erkennen, einschließlich der folgenden Beispiele:

**Beispiel 0.1 (Rn** als Vektorraum). Das häufigste Beispiel für einen Vektorraum ist Rn . Hier erfolgen Addition und Skalarmultiplikation Komponente für Komponente:

$$(1, 2) + (\ddot{y}3, 4) = (1 \ddot{y} 3, 2 + 4) = (\ddot{y}2, 6)$$

$$10 \cdot (\ddot{y}1, 1) = (10 \cdot \ddot{y}1, 10 \cdot 1) = (\ddot{y}10, 10)$$

**Beispiel 0.2** (Polynome). Ein zweites wichtiges Beispiel eines Vektorraums ist der „Ring“ von Polynomen mit reellen Zahleneingaben, bezeichnet mit R[x]. Ein Polynom p ÿ R[x] ist eine Funktion p : **R** ÿ **R** der Form2

$$p(x) = \ddot{y}_k \; k \, akx \cdot$$

Addition und Skalarmultiplikation werden auf die übliche Weise durchgeführt, z. B. wenn $p(x) = x^2 + 2x \ddot{y} 1$ und $q(x) = x + 3x^3$, dann ist $3p(x) + 5q(x) = 5x^3 + 6x \ddot{y} 3$, was ein weiteres Polynom ist. Abgesehen davon sind zukünftige Beispiele beachten, dass Funktionen wie $p(x) = (x \ddot{y} 1)(x + 1) + x^2$ gelten, obwohl sie nicht explizit in der $(X^3 \text{ für } \ddot{y} 5)$ immer noch gerade Polynome obigen Form geschrieben sind.

Elemente v ÿ V eines Vektorraums V werden Vektoren genannt, und eine gewichtete Summe der Form ÿi aivi , wobei ai ÿ **R** und vi ÿ V ist, wird als Linearkombination der vi bezeichnet . In unserem zweiten Beispiel sind die „Vektoren“ Funktionen, obwohl wir diese Sprache normalerweise nicht verwenden, um R[x] zu diskutieren. Eine Möglichkeit , Standpunkte mit der Sequenz zu verknüpfen, wäre die Identifizierung des Polynoms ÿk akx diese beiden (a0, a1, a2, · · ·); Denken Sie daran, dass Polynome endlich viele Terme haben, sodass die Folge letztendlich in einer Folge von Nullen endet.

---

2Die Notation f : A ÿ B bedeutet, dass f eine Funktion ist, die als Eingabe ein Element der Menge A nimmt und ein Element der Menge B ausgibt. Beispielsweise nimmt f : **R** ÿ **Z** als Eingabe eine reelle Zahl in **R** und gibt eine ganze Zahl Z aus , wie es bei f(x) = x der Fall sein könnte, der „Abrundungs“-Funktion.

## 0.2.2 Spanne, lineare Unabhängigkeit und Basen

Angenommen, wir beginnen mit den Vektoren v1, . . . ,vk ÿ V für den Vektorraum V. Nach Definition 0.1 haben wir zwei Möglichkeiten, mit diesen Vektoren zu beginnen und neue Elemente von V zu konstruieren: Addition und Skalarmultiplikation. Die Idee von span besteht darin, dass es alle Vektoren beschreibt, die Sie über diese beiden Operationen erreichen können:

**Definition 0,2** (Span). Die Spanne einer Menge S ÿ V von Vektoren ist die Menge

$$\text{span } S \ddot{y} \{a1v1 + \cdots + akvk : k \ddot{y} 0, vi \ddot{y} V \text{ für alle i und ai } \ddot{y} \mathbf{R} \text{ für alle i}\}.$$

Beachten Sie, dass die Spanne S ein Unterraum von V ist, also eine Teilmenge von V , die selbst ein Vektorraum ist. Wir können einige Beispiele nennen:

**Beispiel 0.3** (Mixologie). Der typische „Brunnen“ einer Cocktailbar enthält mindestens vier Zutaten, die dem Barkeeper zur Verfügung stehen: Wodka, Tequila, Orangensaft und Grenadine. Vorausgesetzt, wir haben diesen einfachen Brunnen, können wir Getränke als Punkte in R4 mit einem Platz für jede Zutat darstellen. Beispielsweise kann ein typischer „Tequila-Sonnenaufgang“ mit dem Punkt (0, 1,5, 6, 0,75) dargestellt werden, der die Mengen an Wodka, Tequila, Orangensaft und Grenadine (in Unzen) darstellt.

Das Getränkeset, das mit dem typischen Brunnen zubereitet werden kann, ist in enthalten

$$\text{span } \{(1, 0, 0, 0),(0, 1, 0, 0),(0, 0, 1, 0),(0, 0, 0, 1)\},$$

das heißt, alle Kombinationen der vier Grundzutaten. Ein Barkeeper, der Zeit sparen möchte, bemerkt jedoch möglicherweise, dass viele Getränke das gleiche Verhältnis von Orangensaft zu Grenadine haben, und mischt die Flaschen. Der neue, vereinfachte Brunnen lässt sich zwar leichter einschenken, kann aber wesentlich weniger Getränke zubereiten:

$$\text{Spanne } \{(1, 0, 0, 0),(0, 1, 0, 0),(0, 0, 6, 0{,}75)\}$$

**Beispiel 0.4** (Polynome). Definieren Sie $pk(x) \ddot{y} x^k$. Dann ist das leicht zu erkennen

$$R[x] = \text{span } \{pk : k \ddot{y} 0\}.$$

Stellen Sie sicher, dass Sie die Notation gut genug verstehen, um zu verstehen, warum dies der Fall ist.

Das Hinzufügen eines weiteren Elements zu einer Menge von Vektoren führt nicht immer zu einer Vergrößerung der Spanne. Für In R2 ist dies beispielsweise eindeutig der Fall

$$\text{Spanne } \{(1, 0),(0, 1)\} = \text{Spanne } \{(1, 0),(0, 1),(1, 1)\}.$$

In diesem Fall sagen wir, dass die Menge {(1, 0),(0, 1),(1, 1)} linear abhängig ist:

**Definition 0.3** (Lineare Abhängigkeit). Wir bieten drei äquivalente Definitionen. Eine Menge S ÿ V von Vektoren ist linear abhängig, wenn:

1. Eines der Elemente von S kann als lineare Kombination der anderen Elemente geschrieben werden, oder S enthält null.

2. Es existiert eine nichtleere Linearkombination von Elementen vk ÿ S , die ÿ ck = 0 für alle k ergibt. $\sum_{k=1}^{M} ckvk = 0$ wobei

3. Es gibt v ÿ S, so dass Spanne S = Spanne S\{v}. Das heißt, wir können einen Vektor ohne entfernen aus S Auswirkungen auf seine Spannweite.

Wenn S nicht linear abhängig ist, dann sagen wir, dass es linear unabhängig ist.

Für Schüler, die mit Notation und abstrakter Mathematik weniger vertraut sind, ist es eine lohnenswerte Übung, Beweise oder informelle Nachweise dafür zu erbringen, dass jede Definition ihren Entsprechungen entspricht (auf die Art und Weise, „wenn und nur wenn“).

Das Konzept der linearen Abhängigkeit führt zu einer Vorstellung von „Redundanz“ in einer Menge von Vektoren. In diesem Sinne stellt sich natürlich die Frage, wie groß die Menge sein kann, die wir wählen können, bevor das Hinzufügen eines weiteren Vektors die Spanne unmöglich vergrößern kann. Nehmen wir insbesondere an, wir haben eine linear unabhängige Menge S ÿ V und wählen nun einen zusätzlichen Vektor v ÿ V. Das Hinzufügen von v zu S führt zu einem von zwei möglichen Ergebnissen:

1. Die Spannweite von S ÿ {v} ist größer als die Spannweite von S.

2. Das Hinzufügen von v zu S hat keinen Einfluss auf die Spanne.

Die Dimension von V ist nichts anderes als die maximale Häufigkeit, mit der wir Ergebnis 1 erhalten, v zu S addieren und wiederholen können.

**Definition 0,4** (Dimension und Basis). Die Dimension von V ist die maximale Größe |S| einer linear unabhängigen Menge S ÿ V mit der Spanne S = V. Jede Menge S, die diese Eigenschaft erfüllt, heißt Basis für V.

**Beispiel 0,5 (Rn ).** Die Standardbasis für Rn ist die Menge der Vektoren der Form

$$e_k \ddot{y} (0, \ldots, \underbrace{0}_{k\ddot{y}1 \text{ Slots}}, 1, \underbrace{0, \ldots, 0}_{n\ddot{y}k \text{ Slots}}).$$

Das heißt, ek hat alle Nullen bis auf eine einzelne Eins im k-ten Slot. Es ist klar, dass diese Vektoren linear unabhängig sind und eine Basis bilden; Beispielsweise kann in R3 jeder Vektor (a, b, c) als ae1 + be2 + ce3 geschrieben werden.
Daher ist die Dimension von Rn erwartungsgemäß n.

**Beispiel 0.6** (Polynome). Es ist klar, dass die Menge {1, x, x , . . .} ist eine linear unabhängige Menge von Polynomen, die R[x] aufspannen. Beachten Sie, dass diese Menge unendlich groß ist und daher die Dimension von R[x] ÿ ist.

## 0.2.3 Unser Fokus: Rn

Von besonderer Bedeutung für unsere Zwecke ist der Vektorraum Rn, der klidische , das sogenannte n-dimensionale Eu Raum. Dies ist nichts anderes als der Satz von Koordinatenachsen, die man im Mathematikunterricht der Oberstufe antrifft:

- R1 ÿ **R** ist der Zahlenstrahl
- R2 ist die zweidimensionale Ebene mit den Koordinaten (x, y) • R3 stellt den dreidimensionalen Raum mit den Koordinaten (x, y, z) dar

Fast alle Methoden in diesem Kurs befassen sich mit Transformationen und Funktionen auf dem Rn .

Der Einfachheit halber schreiben wir Vektoren im Rn normalerweise wie folgt in „Spaltenform“.

$$(a_1, \ldots, a_n) \ddot{y} \begin{matrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{matrix} \ddot{y}$$

Diese Notation umfasst Vektoren als Sonderfälle von Matrizen, die unten besprochen werden.

Im Gegensatz zu einigen Vektorräumen hat Rn nicht nur eine Vektorraumstruktur, sondern auch eine zusätzliche Konstruktion, die den entscheidenden Unterschied macht: das Skalarprodukt.

**Definition 0,5** (Skalarprodukt). Das Skalarprodukt zweier Vektoren a = (a1, . . . , an) undb = (b1, . . . , bn) im Rn ist gegeben durch

$$a \cdot b = \sum_{k=1}^{N} a_k b_k .$$

**Beispiel 0,7 (R2 ).** Das Skalarprodukt von (1, 2) und (ÿ2, 6) ist 1 · ÿ2 + 2 · 6 = ÿ2 + 12 = 10.

Das Skalarprodukt ist ein Beispiel für eine Metrik, und seine Existenz gibt Rn eine Vorstellung von der Geometrie· Beispielsweise können wir den Satz des Pythagoras verwenden, um die Norm oder Länge eines Vektors als Quadratwurzel zu definieren

$$\sqrt{a_1^2 + a_2^2 + \cdots + a_N^2} = \sqrt{a \cdot a}.$$

Dann ist der Abstand zwischen zwei Punktena,b ÿ Rn einfach b ÿa2.

Punktprodukte liefern nicht nur Vorstellungen von Längen und Abständen, sondern auch von Winkeln. Erinnern Sie sich an die folgende Identität aus der Trigonometrie fora,b ÿ R3 :

$$a \cdot b = ab \cos \theta$$

wobei *ÿ* der Winkel zwischen a und b ist. Für n ÿ 4 ist der Begriff „Winkel“ für Rn jedoch viel schwieriger zu visualisieren . Wir könnten den Winkel *ÿ* zwischen a und b als den Wert *ÿ* definieren, der durch gegeben ist

$$\theta \ddot{y} \arccos \frac{a \cdot b}{ab}.$$

Wir müssen unsere Hausaufgaben machen, bevor wir eine solche Definition machen! Denken Sie insbesondere daran, dass die Kosinusausgabe Werte im Intervall [ÿ1, 1] ausgibt. Daher müssen wir überprüfen, ob die Eingabe für den Arcuskosinus (auch als cosÿ1 bezeichnet ) in diesem Intervall liegt. Glücklicherweise garantiert die bekannte Cauchy-Schwarz-Ungleichung a ·b ÿ ab genau diese Eigenschaft.

Wenn a = cb für ein c ÿ R, gilt *ÿ* = arccos 1 = 0, wie wir erwarten würden: Der Winkel zwischen parallelen Vektoren ist Null. Was bedeutet es, dass Vektoren senkrecht zueinander stehen? Ersetzen wir *ÿ* = 90ÿ . Dann haben wir

$$0 = \cos 90^\ddot{y} = \frac{a \cdot b}{ab}$$

Die Multiplikation beider Seiten mit ab motiviert die Definition:

**Definition 0,6** (Orthogonalität). Zwei Vektoren sind senkrecht oder orthogonal, wenna ·b = 0.

Diese Definition ist aus geometrischer Sicht etwas überraschend. Insbesondere ist es uns gelungen, zu definieren, was es bedeutet, senkrecht zu sein, ohne explizit Winkel zu verwenden. Diese Konstruktion wird es einfacher machen, bestimmte Probleme zu lösen, bei denen die Nichtlinearität von Sinus und Cosinus in einfacheren Situationen möglicherweise Kopfschmerzen bereitet hätte.

**Neben 0,1.** Hier gibt es viele theoretische Fragen, über die wir nachdenken müssen, einige davon werden wir in späteren Kapiteln behandeln, wenn sie motivierter sind:

- Lassen alle Vektorräume Skalarprodukte oder ähnliche Strukturen zu?
- Lassen alle endlichdimensionalen Vektorräume Skalarprodukte zu?
- Was könnte ein sinnvolles Skalarprodukt zwischen Elementen von R[x] sein?

Interessierte Studierende können Texte zur Real- und Funktionsanalyse zu Rate ziehen.

# 0,3 Linearität

Eine Funktion zwischen Vektorräumen, die die Struktur bewahrt, wird als lineare Funktion bezeichnet:

**Definition 0,7** (Linearität). Angenommen, V und V sind Vektorräume. Dann ist L : V ÿ V linear, wenn es die folgenden zwei Kriterien für alle v,v1,v2 ÿ V und c ÿ R erfüllt:

- L bewahrt Summen: L[v1 +v2] = L[v1] + L[v2]
- L bewahrt Skalarprodukte: L[cv] = cL[v]

Es ist einfach, lineare Abbildungen zwischen Vektorräumen zu erstellen, wie wir in den folgenden Beispielen sehen können:

**Beispiel 0.8** (Linearität in Rn ). Die folgende Abbildung f : R2 ÿ R3 ist linear:

$$f(x, y) = (3x, 2x + y, ÿy)$$

Wir können die Linearität wie folgt überprüfen:

- Summenerhaltung:

$$\begin{aligned} f(x1 + x2, y1 + y2) &= (3(x1 + x2), 2(x1 + x2) + (y1 + y2), ÿ(y1 + y2)) \\ &= (3x1, 2x1 + y1, ÿy1) + (3x2, 2x2 + y2, \\ &\quad ÿy2) = f(x1, y1) + f(x2, y2) \end{aligned}$$

- Skalare Produktkonservierung:

$$\begin{aligned} f(cx, cy) &= (3cx, 2cx + cy, ÿcy) = c(3x, \\ &\quad 2x + y, ÿy) = c\, f(x, y) \end{aligned}$$

Im Gegensatz dazu ist g(x, y) ÿ xy2 nicht linear. Zum Beispiel ist g(1, 1) = 1, aber g(2, 2) = 8 = 2 · g(1, 1), daher behält diese Form keine Skalarprodukte bei.

**Beispiel 0.9** (Integration). Das folgende „funktionale“ L von R[x] nach **R** ist linear:

$$L[p(x)] ÿ \int_0^1 p(x)\, dx.$$

Dieses etwas abstraktere Beispiel bildet Polynome p(x) auf reelle Zahlen L[p(x)] ab. Wir können zum Beispiel schreiben

$$L[3x^2 + x \ddot{y} 1] = \int_0^1 (3x^2 1 + x \ddot{y} 1)\, dx = 2.$$

Die Linearität ergibt sich aus den folgenden bekannten Fakten aus der Analysis:

$$\int_0^1 c \cdot f(x)\, dx = c \int_0^1 f(x)\, dx$$

$$\int_0^1 [f(x) + g(x)]\, dx = \int_0^1 f(x)\, dx + \int_0^1 g(x)\, dx$$

Wir können eine besonders schöne Form für lineare Abbildungen auf Rn schreiben . Denken Sie daran, dass der Vektor a = (a1, . . . , an) gleich der Summe ÿk akek ist , wobei ek der k-te Standardbasisvektor ist. Wenn L dann linear ist, wissen wir:

$$\begin{aligned} L[a] &= L \ddot{y}_k a_k e_k \text{ für die Standardbasis} \\ &= \ddot{y}_k L[a_k e_k] \text{ durch Summenerhaltung} \\ &= \ddot{y}_k a_k L[e_k] \text{ durch Skalarprodukterhaltung} \end{aligned}$$

Diese Ableitung zeigt die folgende wichtige Tatsache:

L **wird vollständig durch seine Wirkung auf die Standardbasis vectorsek bestimmt .**

Das heißt, für jeden Vektora ÿ , Wir können die obige Summe verwenden, um L[a] durch lineare Kombination zu bestimmen Rn L[e1], . . . ,L[en].

**Beispiel 0.10** (Erweitern einer linearen Karte). Erinnern Sie sich an die Karte in Beispiel 0.8, gegeben durch f(x, y) = (3x, 2x + y, ÿy). Es gilt f(e1) = f(1, 0) = (3, 2, 0) und f(e2) = f(0, 1) = (0, 1, ÿ1). Somit zeigt die obige Formel:

$$f(x, y) = x\, f(e_1) + y\, f(e_2) = x \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

## 0.3.1 Matrizen

Die obige Erweiterung linearer Karten deutet auf einen von vielen Kontexten hin, in denen es nützlich ist, mehrere Vektoren in derselben Struktur zu speichern. Allgemeiner gesagt, wir haben n Vektoren v1, . . . ,vn ÿ Rm. Wir können jeden als Spaltenvektor schreiben:

$$v_1 = \begin{pmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{m1} \end{pmatrix}, v_2 = \begin{pmatrix} v_{12} \\ v_{22} \\ \vdots \\ v_{m2} \end{pmatrix}, \cdots, v_n = \begin{pmatrix} v_{1n} \\ v_{2n} \\ \vdots \\ v_{mn} \end{pmatrix}$$

Diese getrennt mit sich herumzutragen kann in der Notation umständlich sein. Um die Sache zu vereinfachen, kombinieren wir sie einfach in einer einzigen m × n-Matrix:

$$\begin{pmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{pmatrix}$$

Wir nennen den Raum solcher Matrizen $\mathbb{R}^{m\times n}$ .

**Beispiel 0.11** (Identitätsmatrix). Wir können die Standardbasis für $\mathbb{R}^n$ in der $n \times n$ „Identitätsmatrix“ speichern. $I_{n\times n}$ gegeben durch:

$$I_{n\times n} = \begin{pmatrix} | & | & & | \\ e_1 & e_2 & \cdots & d_e \\ | & | & & | \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

Da wir Matrizen als bequeme Möglichkeit zum Speichern von Vektormengen konstruiert haben, können wir mithilfe der Multiplikation ausdrücken, wie sie linear kombiniert werden können. Insbesondere kann eine Matrix in $\mathbb{R}^{m\times n}$ mit einem Spaltenvektor in $\mathbb{R}^n$ wie folgt multipliziert werden:

$$\begin{pmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

Die Erweiterung dieser Summe ergibt die folgende explizite Formel für Matrix-Vektor-Produkte:

$$\begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} c_1 v_{11} + c_2 v_{12} + \cdots + c_n v_{1n} \\ c_1 v_{21} + c_2 v_{22} + \cdots + c_n v_{2n} \\ \vdots \\ c_1 v_{m1} + c_2 v_{m2} + \cdots + c_n v_{mn} \end{pmatrix}$$

**Beispiel 0.12** (Identitätsmatrixmultiplikation). Es ist eindeutig wahr, dass für jedes $x \in \mathbb{R}^n$ , wir können schreiben $x = I_{n\times n} x$ ist, wobei $I_{n\times n}$ die Identitätsmatrix aus Beispiel 0.11 ist.

**Beispiel 0.13** (Lineare Karte). Wir kehren noch einmal zum Ausdruck aus Beispiel 0.8 zurück, um eine weitere alternative Form zu zeigen:

$$f(x, y) = \begin{pmatrix} 3 & 0 & 2 \\ 1 & & \\ 0 & -1 & \end{pmatrix} \begin{pmatrix} x \\ \\ y \end{pmatrix}$$

Auf ähnliche Weise definieren wir ein Produkt zwischen einer Matrix in $M \in \mathbb{R}^{m\times n}$ und einer anderen Matrix in $\mathbb{R}^{n\times p}$ , indem wir einzelne Matrix-Vektor-Produkte verketten:

$$M \begin{pmatrix} | & | & & | \\ c_1 & c_2 & \cdots & c_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ Mc_1 & Mc_2 & \cdots & Mc_n \\ | & | & & | \end{pmatrix}$$

**Beispiel 0.14** (Mixologie). Nehmen wir in Fortsetzung von Beispiel 0.3 an, dass wir in unserem vereinfachten Brunnen einen Tequila Sunrise und eine zweite Mischung mit gleichen Teilen der beiden Liköre herstellen. Um herauszufinden, wie viele Grundzutaten in jeder Bestellung enthalten sind, könnten wir die Rezepte für jede Spalte spaltenweise kombinieren und eine Matrix verwenden Multiplikation:

| | Gut 1 | Gut 2 | Gut 3 |
|---|---|---|---|
| Wodka | 1 | 0 | 0 |
| Tequila | 0 | 1 | 0 |
| ABl. | 0 | 0 | 6 |
| Grenadine | 0 | 0 | 0,75 |

.

| | Trinken Sie 1, trinken Sie 2 |
|---|---|
| | 0 0,75 |
| | 1.5 0,75 |
| | 1 2 |

=

| | Trinken Sie 1, trinken Sie 2 | |
|---|---|---|
| Wodka | 0 | 0,75 |
| Tequila | 0,75 | 1.5 |
| ABl | 12 | |
| Grenadine | 1.5 | 0,75 |

Im Allgemeinen verwenden wir Großbuchstaben zur Darstellung von Matrizen, wie z. B. A ÿ Rm×n. Wir werden das verwenden Notation Aij ÿ **R ,** um das Element von A in Zeile i und Spalte j zu bezeichnen.

## 0.3.2 Skalare, Vektoren und Matrizen

Es ist keine Überraschung, dass wir einen Skalar als 1 × 1-Vektor c ÿ R1×1 schreiben können. Ähnlich, wie wir schon Wie in §0.2.3 vorgeschlagen, können Vektoren in Rn , wenn wir sie in Spaltenform schreiben, als n × 1-Matrizen betrachtet werden v ÿ Rn×1 . Beachten Sie, dass Matrix-Vektor-Produkte in diesem Zusammenhang leicht interpretiert werden können; Zum Beispiel, wenn A ÿ Rm×n, x ÿ Rn , undb ÿ Rm, dann können wir Ausdrücke schreiben wie

$$A\,x = b$$
$$m\times n \quad n\times 1 \quad m\times 1$$

Wir werden einen zusätzlichen Operator für Matrizen einführen, der in diesem Zusammenhang nützlich ist:

**Definition 0,8** (Transponieren). Die Transponierte einer Matrix A ÿ Rm×n ist eine Matrix A ÿ Rn×m mit Elementen $(A^\top)_{ij} = A_{ji}$.

**Beispiel 0,15** (Transposition). Die Transponierte der Matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

ist gegeben durch

$$A^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

Geometrisch gesehen können wir uns Transposition als Umkehrung einer Matrix auf ihrer Diagonale vorstellen.

Diese einheitliche Behandlung von Skalaren, Vektoren und Matrizen in Kombination mit Operationen wie Transposition und Multiplikation kann zu raffinierten Ableitungen bekannter Identitäten führen. Zum Beispiel,

Wir können die Skalarprodukte der Vektoren sa,b ÿ Rn berechnen , indem wir die folgende Reihe von Schritten ausführen:

$$a \cdot b = \sum_{k=1}^{N} a_k b_k$$

$$= a_1\ a_2 \cdots a_n \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{rd} \end{pmatrix}$$

$$= ab$$

Viele wichtige Identitäten aus der linearen Algebra lassen sich ableiten, indem man diese Operationen mit ein paar Regeln verkettet:

$$(A\ ) = A$$

$$(A + B) = A + B$$

$$(AB) = BA$$

**Beispiel 0,16** (Restnorm). Angenommen, wir haben eine Matrix A und zwei Vektoren x und b. Wenn wir wissen möchten, wie gut Ax b annähert, könnten wir einen Residuum r ÿ b ÿ Ax definieren; Dieses Residuum ist genau dann Null, wenn Ax = b. Andernfalls könnten wir die Norm r2 als Proxy für die Beziehung zwischen Ax und b verwenden.
Wir können die obigen Identitäten zur Vereinfachung verwenden:

$$R_2^2 = b \ddot{y} Ax_2^2$$

= (b ÿ Ax) · (b ÿ Ax) wie in §0.2.3 erklärt = (b ÿ Ax)

(b ÿ Ax) durch unseren Ausdruck für das Skalarprodukt oben = (b ÿx A )(b

ÿ Ax) durch Eigenschaften der Transposition =bb ÿb Ax ÿx A

b +x A Ax nach der Multiplikation

Alle vier Terme auf der rechten Seite sind Skalare oder entsprechend 1 × 1-Matrizen. Als Matrizen gedachte Skalare genießen trivialerweise eine zusätzliche schöne Eigenschaft c = c, da es nichts zu transponieren gibt! So können wir schreiben

$$x\ A\ b = (x\ A\ b) = b\ Ax$$

Dadurch können wir unseren Ausdruck noch weiter vereinfachen:

$$R_2^2 = bb \ddot{y} 2b\ Ax + x\ A\ Ax$$

$$= Axt_2^2 \ddot{y} 2b\ Ax + b_2^2$$

Wir hätten diesen Ausdruck mithilfe von Punktproduktidentitäten ableiten können, aber die obigen Zwischenschritte werden sich in unserer späteren Diskussion als nützlich erweisen.

**0.3.3 Modellproblem:** Ax =b

Im Einführungskurs in die Algebra verbringen die Schüler viel Zeit damit, lineare Systeme wie das folgende für Tripel (x, y, z) zu lösen:

$$\begin{aligned} 3x + 2y + 5z &= 0 \\ -4x + 9y - 3z &= -7\ 2x \\ -3y - 3z &= 1 \end{aligned}$$

Unsere Konstruktionen in §0.3.1 ermöglichen es uns, solche Systeme sauberer zu kodieren:

$$\begin{bmatrix} 3 & 2 & 5 \\ -4 & 9 & -3 \\ 0 & -3 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ -7 \\ 1 \end{bmatrix}$$

Allgemeiner gesagt können wir lineare Gleichungssysteme in der Form Ax = b schreiben, indem wir dem gleichen Muster wie oben folgen; hier ist der Vektor x unbekannt, während A und b bekannt sind. Es ist nicht immer garantiert, dass ein solches Gleichungssystem eine Lösung hat. Wenn A beispielsweise nur Nullen enthält, erfüllt eindeutig kein x Ax =b, wenn b =0. Wir werden eine allgemeine Betrachtung darüber, wann eine Lösung existiert, auf unsere Diskussion linearer Löser in zukünftigen Kapiteln verschieben.

Eine Schlüsselinterpretation des Systems Ax =b besteht darin, dass es folgende Aufgabe anspricht:

**Schreiben Sie b als lineare Kombination der Spalten von A.**

Warum? Erinnern Sie sich an §0.3.1, dass das Produkt Ax eine lineare Kombination der Spalten von A mit Gewichten kodiert, die in Elementen von x enthalten sind. Die Gleichung Ax =b verlangt also, dass die Linearkombination Ax gleich dem gegebenen Vektorb ist. Angesichts dieser Interpretation definieren wir den Spaltenraum von A als den Raum der rechten Seitenb, für den das System eine Lösung hat:

**Definition 0.9** (Spaltenraum). Der Spaltenraum einer Matrix A ∈ $\mathbb{R}^{m\times n}$ ist die Spanne der Spalten von A. Wir können schreiben als

$$\operatorname{col} A = \{Ax : x \in \mathbf{R}^n\}.$$

Ein wichtiger Fall ist etwas einfacher zu betrachten. Angenommen, A ist quadratisch, also können wir A ∈ $\mathbb{R}^{n\times n}$ schreiben . Nehmen wir außerdem an, dass das System Ax = b eine Lösung für alle Auswahlmöglichkeiten von b hat. Die einzige Bedingung für b ist, dass es ein Mitglied von Rn ist. Daraus , Das können wir durch unsere obige Interpretation von Ax =b lässt sich schließen, dass die Spalten von A Rn überspannen.

Da in diesem Fall das lineare System immer lösbar ist, setzen wir die Standardbasis e1 ein, . . . ,en, um die Vektoren x1, . . . ,xn, wobei Axk = ek für jedes k erfüllt ist. Dann können wir diese Ausdrücke „stapeln“, um Folgendes zu zeigen:

$$A \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ax_1 & Ax_2 & \cdots & Ax_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_n \\ | & | & & | \end{bmatrix} = I_{n\times n},$$

wobei In×n die Identitätsmatrix aus Beispiel 0.11 ist. Wir nennen die Matrix mit den Spalten xk die Umkehrung $A^{-1}$ , was befriedigt

$$AA^{-1} = A^{-1}A = I_{n\times n}.$$

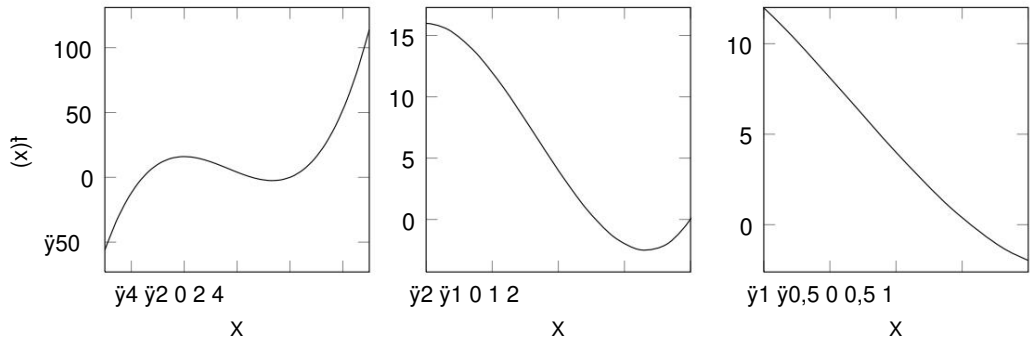



Abbildung 1: Je näher wir an f(x) = x heranzoomen $^{3\,2\,+\,x}$ ÿ 8x + 4, desto mehr sieht es wie eine Linie aus.

Es ist auch leicht zu überprüfen, dass $(\tilde{A}^{ÿ1})^{ÿ1} = \tilde{A}$. Wenn eine solche Umkehrung existiert, ist es einfach, das System Ax =b zu lösen. Insbesondere finden wir:

$$x = In \times nx = (A\ ÿ1A)x = A^{ÿ1}\ (Ax) = A\ ÿ1b$$

# 0,4 Nichtlinearität: Differentialrechnung

Während die Schönheit und Anwendbarkeit der linearen Algebra sie zu einem Hauptziel des Studiums macht, gibt es in der Natur viele Nichtlinearitäten und wir müssen oft Rechensysteme entwerfen, die mit dieser Tatsache des Lebens umgehen können. Denn auf der grundlegendsten Ebene macht das Quadrat in der berühmten Beziehung E = mc2 sie einer linearen Analyse nicht zugänglich.

### 0.4.1 Differenzierung

Während viele Funktionen global nichtlinear sind, zeigen sie lokal ein lineares Verhalten. Diese Idee der „lokalen Linearität“ ist einer der Hauptmotive der Differentialrechnung. Abbildung 1 zeigt beispielsweise, dass eine glatte Funktion, wenn man nah genug heranzoomt, letztendlich wie eine Linie aussieht. Die Ableitung f (x) einer Funktion f(x) : **R** ÿ **R** ist nichts anderes als die Steigung der Näherungsgeraden, berechnet durch Ermitteln der Steigung von Geraden durch immer näher an x liegende Punkte:

$$= \lim_{yÿx} \frac{f(y)\ ÿ\ f(x)\ f\ (x)}{y\ ÿ\ x}$$

Wir können die lokale Linearität ausdrücken, indem wir f(x + ÿx) = f(x) + ÿx · f (x) + O(ÿx schreiben$^{2}$).

Wenn die Funktion f mehrere Eingaben benötigt, kann sie wie folgt geschrieben werden : f(x) : Rn ÿ **R** für x ÿ Rn ; mit anderen Worten, jedem Punkt x = (x1, . . . , xn) im n-dimensionalen Raum weist f eine einzelne Zahl f(x1, . . . , xn) zu. Unsere Vorstellung von lokaler Linearität bricht hier etwas zusammen, da Linien eindimensionale Objekte sind. Die Festlegung aller Variablen bis auf eine reduziert sich jedoch auf den Fall der Einzelvariablenrechnung. Zum Beispiel könnten wir g(t) = f(t, x2, . . . , xn) schreiben, wobei wir einfach die Konstanten x2, . festlegen . . , xn. Dann ist g(t) eine differenzierbare Funktion einer einzelnen Variablen. Natürlich hätten wir t in jeden der Eingabefelder für f einfügen können, daher machen wir im Allgemeinen die folgende Definition der partiellen Ableitung von f:

f **Definition 0,10** (partielle Ableitung). Die k-te partielle Ableitung von f, notiert als $\ddot{y}xk$ , $\frac{\ddot{y}}{\text{die f in}}$ wird durch different gegeben seiner k-ten Eingabevariablen bindet:

$$\frac{\ddot{y}\, f}{dt}(x1, \ldots, xn) \ddot{y} \ddot{y}xk \overset{D}{\text{—}} f(x1, \ldots, xk\ddot{y}1, t, xk+1, \ldots, xn)|t=xk$$

Die Notation „|t=xk " sollte als „ausgewertet bei t = xk " gelesen werden.

**Beispiel 0,17** (Relativität). Die Beziehung E = mc2 kann man sich als Funktion von m und c zu E vorstellen. Somit könnten wir E(m, c) = mc2 schreiben und die Ableitungen ergeben

$$\frac{\ddot{y}E}{\ddot{y}m} \; 2 = c$$
$$\frac{\ddot{y}E}{\ddot{y}c} = 2mc$$

Unter Verwendung der Einvariablenrechnung können wir schreiben:

$$f(x + \ddot{y}x) = f(x1 + \ddot{y}x1, x2 + \ddot{y}x2, \ldots, xn + \ddot{y}xn) \; \ddot{y}\, f$$
$$f(x1, x2 + \ddot{y}x2, \ldots, xn + \ddot{y}xn) + \ddot{y}x1 \overset{=}{+} O(\ddot{y}x \; \ddot{y}x1 \quad {}^{2}_{1}) \text{ durch Einzelvariablenrechnung}$$
$$= f(x1, \ldots, xn) + \ddot{\ddot{y}}_{k=1}^{N} \frac{\ddot{y}\, f}{} \ddot{y}xk + O(\ddot{y}x \; \ddot{y}xk^{2}_{k}) \text{ , indem Sie dies n-mal wiederholen}$$
$$= f(x) + \ddot{y}\, f(x) \cdot \ddot{y}x + O(x \quad {}^{2})$$

wobei wir den Gradienten von f als definieren

$$\ddot{y}\, f\, \ddot{y} \quad \frac{\ddot{y}\, f}{\ddot{y}x1}, \frac{\ddot{y}\, f}{\ddot{y}x2}, \cdots, \frac{\ddot{y}\, f}{\ddot{y}xn} \quad \ddot{y}\, \mathbf{R}^{N}$$

Aus dieser Beziehung ist leicht zu erkennen, dass f in jede Richtung v differenziert werden kann; wir können diese Ableitung Dv f wie folgt auswerten:

$$Dv\, f(x) \ddot{y} f(\overset{D}{\ddot{y}} + tv)|t=0 \, dt =$$
$$f(x) \cdot v$$

**Beispiel 0,18 (R2 ).** Nehmen Sie f(x, y) = x 2y 3 . Dann,

$$\frac{\ddot{y}\, f}{\ddot{y}x} = 2xy3$$
$$\frac{\ddot{y}\, f}{\ddot{y}y} = 3x \quad {}^{2}_{j}{}^{2}$$

Somit können wir schreiben : ÿ f(x, y) = (2xy3 , 3x 2y 2 ). Die Ableitung von f bei (1, 2) in Richtung (ÿ1, 4) ist gegeben durch (ÿ1, 4) · ÿ f(1, 2) = (ÿ1, 4) · (16, 12) = 32.

**Beispiel 0.19** (Lineare Funktionen). Es ist offensichtlich, aber erwähnenswert, dass der Gradient von f(x) ÿa · x +c = (a1x1 + c1, . . . , anxn + cn) ist.

**Beispiel 0.20** (Quadratische Formen). Nehmen Sie eine beliebige Matrix A ÿ , und definiere f(x) ÿ x Ax. Erweitern Rn×n, die diese Funktion Element für Element zeigt

$$f(x) = \ddot{y} \underset{ij}{} Aijxixj ;$$

Es lohnt sich, f zu erweitern und diesen Zusammenhang explizit zu überprüfen. Nehmen Sie ein k ÿ {1, . . . , n}. Dann können wir alle Terme heraustrennen, die xk enthalten :

$$f(x) = Akkx \overset{2}{k} + xk \ddot{y} \underset{i=k}{} Aikxi + \ddot{y} \underset{j=k}{} Akjxj + \ddot{y} \underset{i,j=k}{} Aijxixj$$

Mit dieser Faktorisierung ist es leicht zu erkennen

$$\frac{\ddot{y}f}{\ddot{y}xk} = 2Akkxk + \ddot{y} \underset{i=k}{} Aikxi + \ddot{y} \underset{j=k}{} Akjxj$$
$$= \overset{N}{\ddot{y}} \underset{i=1}{} (Aik + Aki)xi$$

Diese Summe ist nichts anderes als die Definition der Matrix-Vektor-Multiplikation! So können wir schreiben

$$\ddot{y} f(x) = Ax + A x.$$

Wir haben von f : **R** ÿ **R** auf f : Rn ÿ R verallgemeinert. Um eine vollständige Allgemeingültigkeit zu erreichen, möchten wir f : Rn ÿ Rm betrachten. Mit anderen Worten: f nimmt n Zahlen auf und gibt m Zahlen aus. Glücklicherweise ist diese Erweiterung unkompliziert, da wir uns f als eine Sammlung einwertiger Funktionen f1, ... vorstellen können . . . , fm : Rn ÿ **R** zu einem einzigen Vektor zusammengeschlagen. Das heißt, wir schreiben:

$$f(x) = \begin{matrix} \ddot{y} & f1(x) & \ddot{y} \\ & f2(x) & \\ & \vdots & \\ \ddot{y}\ddot{y}\ddot{y}\ddot{y} & fm(x) & \ddot{y}\ddot{y}\ddot{y}\ddot{y} \end{matrix}$$

Jedes fk kann wie zuvor differenziert werden, sodass wir am Ende eine Matrix partieller Ableitungen erhalten, die Jacobian von f genannt wird:

**Definition 0.11** (Jakobianisch). Der Jacobi von f : Rn ÿ Rm ist die Matrix D f ÿ Rm×n mit Einträgen

$$(D f)ij \ddot{y} \ddot{y}xj \frac{\ddot{y}fi}{} .$$

**Beispiel 0.21** (Einfache Funktion). Angenommen, f(x, y) = (3x, ÿxy2 , x + y). Dann,

$$D f(x, y) = \begin{matrix} \ddot{y} & 3 & \ddot{y} \\ \ddot{y}y & \ddot{y}2 & 0 \\ \ddot{y} & \ddot{y}2xy & \ddot{y}1 \end{matrix} .$$

Stellen Sie sicher, dass Sie diese Berechnung manuell durchführen können.

**Beispiel 0,22** (Matrixmultiplikation). Es überrascht nicht, dass der Jacobi-Wert von f(x) = Ax für Matrix A durch D f(x) = A gegeben ist.

Hier stoßen wir auf einen gemeinsamen Punkt der Verwirrung. Angenommen, eine Funktion verfügt über eine Vektoreingabe und eine Skalarausgabe, das heißt f : Rn ÿ R. Wir haben den Gradienten von f als Spaltenvektor definiert. Um diese Definition mit der des Jacobi-Werts in Einklang zu bringen, müssen wir schreiben

$$D f = \ddot{y} f .$$

## 0.4.2 Optimierung

Erinnern wir uns an Minima und Maxima von f aus der Einzelvariablenrechnung: **R** ÿ **R** muss an Punkten x auftreten, die f (x) = 0 erfüllen. Natürlich ist diese Bedingung eher notwendig als ausreichend: Es kann Punkte x mit f (x) = 0 geben das sind keine Maxima oder Minima. Allerdings kann das Finden solcher kritischer Punkte von f ein Schritt eines Funktionsminimierungsalgorithmus sein, solange der nächste Schritt sicherstellt, dass das resultierende x tatsächlich ein Minimum/Maximum ist.

Wenn f : Rn ÿ **R** bei x minimiert oder maximiert ist, müssen wir sicherstellen, dass es keine einzige Richtung ÿx von x gibt, in der f abnimmt bzw. zunimmt. Nach der Diskussion in §0.4.1 bedeutet dies, dass wir Punkte finden müssen, für die ÿ f = 0 ist.

**Beispiel 0.23** (Einfache Funktion). Angenommen, f(x, y) = x 2x + 8y. $^{2}$ + 2xy + 4y $^{2}$ und ÿx ÿy $\underline{\ddot{y}\, f}\, \ddot{y} f$ . Dann ist = 2x + 2y ___ = Somit erfüllen kritische Punkte von f:

$$2x + 2y = 0\ 2x$$
$$+ 8y = 0$$

Offensichtlich wird dieses System bei (x, y) = (0, 0) gelöst. Tatsächlich ist dies das Minimum von f, wie man deutlicher erkennen kann, wenn man f(x, y) = (x + y) schreibt. $^{2}$ + 3J $^{2}$.

**Beispiel 0.24** (Quadratische Funktionen). Angenommen, f(x) = x Ax +bx + c. Dann können wir aus den Beispielen im vorherigen Abschnitt schreiben: ÿ f(x) = (A + A)x +b. Somit erfüllen kritische Punkte x von f (A + A)x +b = 0.

Im Gegensatz zur Einzelvariablenrechnung können wir bei der Berechnung von Rn Einschränkungen zu unserer Optimierung hinzufügen. Die allgemeinste Form eines solchen Problems sieht so aus:

$$\text{Minimiere } f(x) \text{ , so}$$
$$\text{dass } g(x) = 0$$

**Beispiel 0,25** (Rechteckflächen). Angenommen, ein Rechteck hat die Breite w und die Höhe h. Ein klassisches Geometrieproblem besteht darin, die Fläche mit einem festen Umfang zu maximieren 1:

$$\text{maximieren } wh$$
$$\text{so dass } 2w + 2h \ddot{y} 1 = 0$$

Wenn wir diese Einschränkung hinzufügen, können wir nicht mehr erwarten, dass kritische Punkte ÿ f(x) = 0 erfüllen, da diese Punkte möglicherweise nicht g(x) = 0 erfüllen.

Angenommen, g : Rn ÿ R. Betrachten Sie die Punktmenge S0 ÿ {x : g(x) = 0}. Offensichtlich erfüllen zwei beliebige x,y ÿ S0 die Beziehung g(y) ÿ g(x) = 0 ÿ 0 = 0. Angenommen, y = x + ÿx für kleine ÿx.
Dann ist g(y) ÿ g(x) = ÿg(x) · ÿx + O(ÿx ). Mit anderen Worten, wenn wir bei x beginnen und g(x) = 0 erfüllen, dann verschieben wir in der ÿx-Richtung ÿg(x) · ÿx ÿ 0, um diese Beziehung weiterhin zu erfüllen.
2

Erinnern Sie sich nun daran, dass die Ableitung von f in der Richtung v bei x durch ÿ f ·v gegeben ist. Wenn x ein Minimum des obigen eingeschränkten Optimierungsproblems ist, sollte jede kleine Verschiebung von x zu x + v einen Anstieg von f(x) zu f(x + v) bewirken. Da wir uns nur um Verschiebungen v kümmern, die die Einschränkung g(x + v) = c wahren, wollen wir aus unserem obigen Argument ÿ f · v = 0 für alle v, die ÿg(x) · v = 0 erfüllen. Mit anderen Worten, ÿ f und ÿg müssen parallel sein, eine Bedingung, die wir als ÿ f = ÿÿg für ein *ÿ* ÿ R schreiben können.

Definieren

$$ÿ(x, ÿ) = f(x) ÿ ÿg(x).$$

Dann erfüllen kritische Punkte von ÿ ohne Einschränkungen:

$$0 = \frac{ÿÿ}{ÿÿ} = ÿg(x)$$

$$0 = ÿxÿ = ÿ f(x) ÿ ÿÿg(x)$$

Mit anderen Worten, kritische Punkte von ÿ erfüllen g(x) = 0 und ÿ f(x) = ÿÿg(x), genau die Optimalitätsbedingungen, die wir abgeleitet haben!

Die Erweiterung auf multivariate Einschränkungen ergibt Folgendes:

**Satz 0.1** (Methode der Lagrange-Multiplikatoren). Kritische Punkte des obigen eingeschränkten Optimierungsproblems sind uneingeschränkte kritische Punkte der Lagrange-Multiplikatorfunktion

$$ÿ(x,ÿ) ÿ f(x) \ ÿÿ \cdot g(x),$$

sowohl bezüglich x als auch ÿ.

**Beispiel 0,26** (Bereich maximieren). In Fortsetzung von Beispiel 0.25 definieren wir die Lagrange-Multiplikatorfunktion ÿ(w, h, ÿ) = wh ÿ ÿ(2w + 2h ÿ 1). Differenzierend finden wir:

$$ÿw \frac{ÿÿ}{ÿÿ} = h ÿ 2ÿ \ 0 =$$

$$w ÿ 2ÿ \frac{= =}{ÿh}$$

$$\frac{ÿÿ}{ÿÿ} = 1 ÿ 2w ÿ 2h \ 0 =$$

Kritische Punkte des Systems sind also erfüllt

$$\begin{matrix} ÿ \\ ÿ \\ ÿ \end{matrix} \begin{matrix} 0 & 1 & ÿ2 \\ 1 & 0 & ÿ2 \\ 2 & 2 & 0 \end{matrix} \quad \begin{matrix} ÿ & ÿ & ÿ \\ ÿ & & h \end{matrix} \begin{matrix} w \\ ÿ \\ ÿ \end{matrix} = \begin{matrix} ÿ \\ ÿ \\ ÿ \end{matrix} \begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \begin{matrix} ÿ \\ ÿ \\ ÿ \end{matrix}$$

Die Lösung des Systems ergibt w = h = 1/4 und *ÿ* = 1/8. Mit anderen Worten: Bei einem festen Umfang ist das Rechteck mit der maximalen Fläche ein Quadrat.

**Beispiel 0.27** (Eigenprobleme). Angenommen, A ist eine symmetrische positiv definite Matrix, was bedeutet, dass A = A (Symmetrie) und x Ax > 0 für alle x ÿ Rn\{0} (positiv definit) ist. Oft möchten wir x Ax unter der Bedingung x = 1 für eine gegebene Matrix A ÿ Rn×n minimieren ; Beachten Sie, dass ohne die Einschränkung das Minimum trivialerweise bei x = 0 auftritt . Wir definieren die Lagrange-Multiplikatorfunktion

$$\ddot{y}(x, \ddot{y}) = x\,Ax \ddot{y} \ddot{y}(x \ddot{y} 1)$$
$$= x\,Ax \ddot{y} \ddot{y}(x\,x \ddot{y} 1).$$

Wenn wir nach x differenzieren, finden wir

$$0 = \ddot{y}x\ddot{y} = 2Ax \ddot{y} 2\ddot{y}x$$

Mit anderen Worten, x ist ein Eigenvektor der Matrix A:

$$Ax = \ddot{y}x.$$

## 0,5 Probleme

**Aufgabe 0.1.** Sei C1 **(R)** die Menge der Funktionen f : **R** ÿ **R ,** die eine erste Ableitung f (x) zulassen. Warum ist 1 C **(R)** ein Vektorraum? Beweisen Sie, dass C1 **(R)** die Dimension ÿ hat.

**Aufgabe 0.2.** Angenommen, die Zeilen von A ÿ Rm×n seien durch die Transponierten von r1, . gegeben . . ,rm ÿ Rn und die Spalten von A ÿ Rm×n sind gegeben durch c1, . . . ,cn ÿ Rm. Das ist,

$$A = \begin{bmatrix} \ddot{y} & r_1 & \ddot{y} \\ \ddot{y} & r_2 & \ddot{y} \\ & \vdots & \\ \ddot{y} & r_m & \ddot{y} \end{bmatrix} \quad \ddot{y}\ddot{y}\ddot{y}\ddot{y} = \begin{bmatrix} \ddot{y} & | & | & & | & \ddot{y} \\ \ddot{y} & c_1 & c_2 & \cdots & c_n & \ddot{y} \\ \ddot{y} & | & | & & | & \ddot{y} \end{bmatrix}.$$

Geben Sie Ausdrücke für die Elemente von AA und AA anhand dieser Vektoren an.

**Aufgabe 0.3.** Geben Sie ein lineares Gleichungssystem an, das durch Minima der Energie f(x) = Ax ÿb bezüglich x erfüllt wird, für x ÿ Rn und b ÿ Rm. Dieses System wird „Normalgleichungen“ genannt und wird an anderer Stelle in diesen Anmerkungen erscheinen; Dennoch lohnt es sich, die Ableitung durchzuarbeiten und vollständig zu verstehen.

**Aufgabe 0.4.** Angenommen A, B ÿ Rn×n . Formulieren Sie eine Bedingung dafür, dass Vektoren x ÿ Rn kritische Punkte des Subjekts für Bx = 1 sind. Geben Sie außerdem eine alternative Form für die optimalen Werte von Ax an.

**Aufgabe 0.5.** Legen Sie einen Vektora ÿ Rn\{0} fest und definieren Sie f(x) =a · x. Geben Sie einen Ausdruck für das Maximum von f(x) unter der Bedingung x = 1 an.

# Kapitel 1

# Numerik und Fehleranalyse

Beim Studium der numerischen Analyse gehen wir vom Umgang mit Ints und Longs zu Floats und Doubles über.
Dieser scheinbar harmlose Übergang bedeutet eine enorme Veränderung in der Art und Weise, wie wir über das Design und die Implementierung von Algorithmen nachdenken müssen. Im Gegensatz zu den Grundlagen diskreter Algorithmen können wir nicht mehr erwarten Unsere Algorithmen liefern in allen Fällen exakte Lösungen. „Big O“ und Vorgangszählung funktionieren nicht immer herrsche unumschränkt; Stattdessen sind wir gezwungen, selbst wenn wir die grundlegendsten Techniken verstehen, zu studieren der Kompromiss zwischen Timing, Näherungsfehler usw.

## 1.1 Speichern von Zahlen mit Bruchteilen

Denken Sie daran, dass Computer Daten im Allgemeinen im Binärformat speichern. Insbesondere ist jede Ziffer positiv Ganzzahl entspricht einer anderen Zweierpotenz. Beispielsweise könnten wir 463 in eine Binärzahl umwandeln anhand der folgenden Tabelle:

| 1 | 1 | 1 | 0 | 0 | 1 | | | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Mit anderen Worten: Diese Notation kodiert die Tatsache, dass 463 in Zweierpotenzen zerlegt werden kann eindeutig als:

$$463 = 2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$
$$= 256 + 128 + 64 + 8 + 4 + 2 + 1$$

Abgesehen von Überlaufproblemen können alle positiven ganzen Zahlen in dieser Form mit einer endlichen Anzahl von geschrieben werden Ziffern. Auch negative Zahlen können auf diese Weise dargestellt werden, entweder durch Einführung eines Vorzeichens Bit oder mit dem „Zweierkomplement“-Trick.

Eine solche Zerlegung inspiriert zu einer einfachen Erweiterung auf Zahlen, die Brüche enthalten: einfach negative Zweierpotenzen einschließen. Beispielsweise ist die Zerlegung von 463,25 so einfach wie die Addition von zwei Schlüssel:

| 1 | 1 | 1 | 0 | 0 | 1 | 3 | 2 | | 1 | 1 | 1. | 0 | 0 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | | | | | $2^2$ | $2^1$ | | | $2^{-1}$ | $2^{-2}$ | |

Genau wie im Dezimalsystem ist jedoch auch die Darstellung gebrochener Teile von Zahlen auf diese Weise möglich nicht annähernd so brav wie die Darstellung von ganzen Zahlen. Zum Beispiel den Bruch 1/3 binär schreiben ergibt den Ausdruck:

$$\frac{1}{3} = 0{,}0101010101\ldots$$

Solche Beispiele zeigen, dass es auf allen Skalen Zahlen gibt, die nicht mit a dargestellt werden können
endliche binäre Zeichenfolge. Tatsächlich sind Zahlen wie $\ddot{y}$ = 11,00100100001. . .2 haben unendlich lange Entwicklungen
unabhängig davon, welche (ganzzahlige) Basis Sie verwenden!

Aus diesem Grund müssen wir beim Entwerfen von Computersystemen, die mit **R** statt mit Z rechnen , Folgendes tun
sind gezwungen, Näherungen für nahezu jede einigermaßen effiziente numerische Darstellung vorzunehmen.
Dies kann beim Codieren zu vielen Verwirrungen führen. Betrachten Sie beispielsweise das folgende C++
Ausschnitt:

```
doppeltes x = 1,0;
doppeltes y = x / 3,0;
if ( x == y *3.0) cout << "Sie sind gleich! "                    ;
sonst cout <<           "Sie sind NICHT gleich.           "    ;
```

Entgegen der Intuition gibt dieses Programm „Sie sind NICHT gleich“ aus. Warum? Die Definition von y
macht eine Annäherung an 1/3 , da es nicht als abschließende Binärzeichenfolge geschrieben werden kann, Rundung
zu einer nahegelegenen Zahl, die es darstellen kann. Somit multipliziert y*3.0 nicht mehr 3 mit 1/3. Eine Möglichkeit, das Problem zu beheben
Dieses Problem ist unten:

```
doppeltes x = 1,0;
doppeltes y = x / 3,0;
if ( fabs (x - y *3.0) < numeric_limits < double >:: epsilon ) cout << "Sie sind gleich ! "          ;
                                                                 else cout << "Sie sind NICHT gleich. "
                                                                 ;
```

Hier überprüfen wir, ob x und y*3,0 innerhalb einer gewissen Toleranz zueinander liegen, anstatt dies zu überprüfen
exakte Gleichheit. Dies ist ein Beispiel für einen sehr wichtigen Punkt:

Der Operator == und seine Äquivalente **sollten selten, wenn überhaupt, für Bruchwerte verwendet werden.**
**Stattdessen sollte eine gewisse Toleranz verwendet werden, um zu prüfen, ob die Zahlen gleich sind.**

Natürlich gibt es hier einen Kompromiss: Die Größe der Toleranz definiert eine Grenze zwischen Gleichheit und
„Nähe, aber nicht gleich“, die für eine bestimmte Anwendung sorgfältig ausgewählt werden müssen.

Im Folgenden betrachten wir einige Möglichkeiten zur Darstellung von Zahlen auf einem Computer.

## 1.1.1 Fixpunktdarstellungen

Die einfachste Möglichkeit zum Speichern von Brüchen ist das Hinzufügen eines festen Dezimalpunkts. Das heißt, wie in
Im obigen Beispiel stellen wir Werte dar, indem wir 0/1-Koeffizienten vor Zweierpotenzen speichern
Bereich von 2ÿk bis 2 für ein k, ÿ Z. Zum Beispiel alle nichtnegativen Werte dazwischen darstellen
0 und 127,75 in Schritten von 1/4 ist so einfach wie k = 2 und = 7; In dieser Situation vertreten wir
Diese Werte werden mit 9 Binärziffern angegeben, von denen zwei nach dem Dezimalpunkt stehen.

Der Hauptvorteil dieser Darstellung besteht darin, dass nahezu alle arithmetischen Operationen möglich sind
erfolgt mit den gleichen Algorithmen wie bei ganzen Zahlen. Das ist zum Beispiel leicht zu erkennen

$$a + b = (a \cdot 2^{k} + b \cdot 2^{k}) \cdot 2^{\ddot{y}k} .$$

Die Multiplikation unserer festen Darstellung mit 2k garantiert, dass das Ergebnis ganzzahlig ist, also diese Beobachtung
zeigt im Wesentlichen, dass die Addition durch Ganzzahladdition im Wesentlichen durch „Ignorieren“ des Dezimalpunkts durchgeführt
werden kann. Anstatt also spezielle Hardware zu verwenden, wird die bereits vorhandene Ganzzahl verwendet
Die Arithmetik-Logik-Einheit (ALU) führt Festkomma-Mathematik schnell aus.

Festkomma-Arithmetik ist zwar schnell, kann jedoch unter schwerwiegenden Präzisionsproblemen leiden. Insbesondere,
Es kommt häufig vor, dass die Ausgabe einer binären Operation wie Multiplikation oder Division erforderlich sein kann
mehr Bits als die Operanden. Nehmen wir zum Beispiel an, wir berücksichtigen eine Dezimalstelle für die Genauigkeit und

möchte das Produkt 1/2 · 1/2 = 1/4 ausführen . Wir schreiben 0,12 × 0,12 = 0,012, was auf 0 gekürzt wird. In diesem System ist es ziemlich einfach, Festkommazahlen auf sinnvolle Weise zu kombinieren und ein unvernünftiges Ergebnis zu erhalten.

Aufgrund dieser Nachteile enthalten die meisten großen Programmiersprachen standardmäßig keinen Festkomma-Dezimaldatentyp. Die Geschwindigkeit und Regelmäßigkeit der Festpunktarithmetik kann jedoch ein erheblicher Vorteil für Systeme sein, die Timing gegenüber Genauigkeit bevorzugen. Tatsächlich implementieren einige Grafikprozessoren (GPUs) der unteren Preisklasse nur diese Vorgänge, da für viele Grafikanwendungen eine Genauigkeit von wenigen Dezimalstellen ausreicht.

## 1.1.2 Gleitkommadarstellungen

Eine der vielen numerischen Herausforderungen beim Schreiben wissenschaftlicher Anwendungen ist die Vielfalt der möglichen Skalen. Nur Chemiker beschäftigen sich mit Werten irgendwo zwischen 9,11 × 10ÿ31 und 6,022 × 1023. Eine so harmlose Operation wie eine Änderung der Einheiten kann einen plötzlichen Übergang zwischen diesen Regimen verursachen: Dieselbe Beobachtung, geschrieben in Kilogramm pro Lichtjahr, sieht in Megatonnen erheblich anders aus pro Sekunde. Als numerische Analysten besteht unsere Aufgabe darin, Software zu schreiben, die einen reibungslosen Übergang zwischen diesen Skalen ermöglicht, ohne dem Kunden unnatürliche Einschränkungen seiner Techniken aufzuerlegen.

Hierzu sind einige Vorstellungen und Beobachtungen aus der Kunst der wissenschaftlichen Messung relevant Diskussion. Erstens ist offensichtlich eine der folgenden Darstellungen kompakter als die andere:

$$6{,}022 \times 1023 = 602,200,000,000,000,000,000,000$$

Darüber hinaus ist der Unterschied zwischen 6,022 × 1023 und 6,022 × 1023 + 9,11 × 10ÿ31 mangels außergewöhnlicher wissenschaftlicher Ausrüstung vernachlässigbar. Eine Möglichkeit, zu dieser Schlussfolgerung zu gelangen, besteht darin, zu sagen, dass 6,022 × 1023 nur eine dreistellige Genauigkeit hat und wahrscheinlich einen Bereich möglicher Messungen darstellt [6,022 × 1023 ÿ ÿ, 6,011 × 1023 + ÿ] für einige $\ddot{y}$ ÿ 0,001 × 1023 .

Unsere erste Beobachtung war in der Lage, unsere Darstellung von 6,022 × 1023 zu verdichten , indem wir sie in wissenschaftlicher Notation schrieben. Dieses Zahlensystem trennt die „interessanten“ Ziffern einer Zahl von ihrer Größenordnung, indem es sie in der Form a × 10b für einige a ÿ 1 und b ÿ Z schreibt. Wir nennen dieses Format die Gleitkommaform einer Zahl, weil Im Gegensatz zur Festkomma-Einstellung in §1.1.1 „schwebt“ hier der Dezimalpunkt nach oben. Wir können Gleitkommasysteme mit wenigen Parametern beschreiben (CITE):

- Die Basis $\ddot{y}$ ÿ N; Für die oben erläuterte wissenschaftliche Schreibweise ist die Basis 10
- Die Genauigkeit p ÿ **N ,** die die Anzahl der Stellen in der Dezimalentwicklung darstellt
- Der Bereich der Exponenten [L, U] , der die möglichen Werte von b darstellt

Eine solche Erweiterung sieht folgendermaßen aus:

$$\underset{\text{Zeichen}}{\pm} \underbrace{(d0 + d1 \cdot \ddot{y}^{\ddot{y}1} + d2 \cdot \ddot{y}^{\ddot{y}2} + \cdots + dp\ddot{y}1 \cdot \ddot{y}^{1\ddot{y}p})}_{\text{Mantisse}} \times \underbrace{\ddot{y}^{B}}_{\text{Exponent}}$$

wobei jede Ziffer dk im Bereich [0, $\ddot{y}$ ÿ 1] liegt und b ÿ [L, U].

Gleitkommadarstellungen haben eine merkwürdige Eigenschaft, die sich auf unerwartete Weise auf Software auswirken kann: Ihre Abstände sind ungleichmäßig. Zum Beispiel die Anzahl der darstellbaren Werte zwischen $\ddot{y}$ und $\ddot{y}$ , obwohl das gleiche wie das zwischen $\ddot{y}$ und $\ddot{y}$ , ist $\ddot{y}$ mal größer. Um die mit einem gegebenen Zahlensystem möglichen Präzision zu verstehen, definieren wir die Maschinengenauigkeit ÿm als

kleinstes ÿm > 0, so dass 1 + ÿm darstellbar ist. Dann sind Zahlen wie *ÿ* + ÿm im Zahlensystem nicht darstellbar, weil ÿm zu klein ist!

Der mit Abstand gebräuchlichste Standard zum Speichern von Gleitkommazahlen ist der IEEE 754-Standard. Dieser Standard spezifiziert mehrere Klassen von Gleitkommazahlen. Beispielsweise wird eine Gleitkommazahl mit doppelter Genauigkeit in der Basis *ÿ* = 2 geschrieben (wie die meisten Zahlen auf dem Computer), mit einem einzelnen ± -Vorzeichenbit, 52 Ziffern für d und einem Exponentenbereich zwischen ÿ1022 und 1023. Der Standard legt auch fest, wie ±ÿ und Werte wie NaN oder „keine Zahl“ gespeichert werden , die für die Ergebnisse von Berechnungen wie 10/0 reserviert sind. Ein zusätzliches Maß an Präzision kann erreicht werden, indem man normalisierende Gleitkommawerte schreibt und davon ausgeht, dass die höchstwertige Ziffer d0 1 ist, und sie nicht schreibt.

Der IEEE-Standard enthält auch vereinbarte Optionen für den Umgang mit der endlichen Anzahl von Werten, die bei einer endlichen Anzahl von Bits dargestellt werden können. Beispielsweise besteht eine gängige unverzerrte Strategie für Rundungsberechnungen darin, auf den nächsten Wert zu runden und auf gerade Werte zu binden. Dabei werden äquidistante Bindungen aufgehoben, indem auf den nächsten Gleitkommawert mit einem geraden niedrigstwertigen Bit (ganz rechts) gerundet wird. Beachten Sie, dass es viele gleichermaßen legitime Strategien zum Runden gibt; Die Auswahl einer einzigen Option garantiert, dass wissenschaftliche Software auf allen Client-Rechnern, die denselben Standard implementieren, identisch funktioniert.

### 1.1.3 Weitere exotische Optionen

In Zukunft gehen wir davon aus, dass Dezimalwerte im Gleitkommaformat gespeichert werden, sofern nicht anders angegeben. Dies bedeutet jedoch nicht, dass es keine anderen numerischen Systeme gibt, und für bestimmte Anwendungen könnte eine alternative Wahl erforderlich sein. Wir erkennen hier einige dieser Situationen an.

Der Aufwand, Toleranzen hinzuzufügen, um Rundungsfehlern Rechnung zu tragen, könnte für einige Anwendungen unakzeptabel sein. Diese Situation tritt bei Anwendungen der Computergeometrie auf, z. B. wenn der Unterschied zwischen nahezu und vollständig parallelen Linien schwierig zu unterscheiden sein kann. Eine Lösung könnte darin bestehen, Arithmetik mit beliebiger Genauigkeit zu verwenden, das heißt, Arithmetik ohne Rundung oder Fehler jeglicher Art zu implementieren.

Arithmetik mit beliebiger Genauigkeit erfordert eine spezielle Implementierung und eine sorgfältige Überlegung, welche Arten von Werten dargestellt werden müssen. Beispielsweise könnte es sein, dass für eine gegebene Anwendung rationale Zahlen **Q** ausreichend sind, die als Verhältnisse a/b für a, b ÿ Z geschrieben werden können.
Grundrechenarten können in **Q** ohne Präzisionsverlust ausgeführt werden. Es ist zum Beispiel leicht zu erkennen

$$\frac{A}{B} \times \frac{C}{D} = \frac{ac}{bd} \qquad \frac{A}{B} \div \frac{C}{D} = \frac{\text{Anzeige}}{\text{v. Chr}}.$$

Die Arithmetik in den rationalen Zahlen schließt die Existenz eines Quadratwurzeloperators aus, da Werte wie $\sqrt{2}$ irrational sind. Auch diese Darstellung ist nicht eindeutig, da zB a/b = 5a/5b.

In anderen Fällen kann es nützlich sein, Fehler einzuklammern, indem Werte neben Fehlerschätzungen als Paar a,ÿ ÿ R dargestellt werden; Wir stellen uns das Paar (a,ÿ) als den Bereich a ± ÿ vor. Dann aktualisieren arithmetische Operationen nicht nur den Wert, sondern auch die Fehlerschätzung, wie in

$$(x \pm ÿ1) + (y \pm ÿ2) = (x + y) \pm (ÿ1 + ÿ2 + \text{Fehler}(x + y)),$$

wobei der letzte Term eine Schätzung des Fehlers darstellt, der durch die Addition von x und y verursacht wird.

# 1.2 Fehler verstehen

Mit Ausnahme der in §1.1.3 beschriebenen Systeme mit beliebiger Genauigkeit ist nahezu jede computergestützte Darstellung reeller Zahlen mit Bruchteilen gezwungen, Rundungs- und andere Näherungsschemata zu verwenden. Dieses Schema stellt eine von vielen Approximationsquellen dar, die typischerweise in numerischen Systemen anzutreffen sind:

- Der Kürzungsfehler ergibt sich aus der Tatsache, dass wir nur eine endliche Teilmenge aller möglichen Wertemengen in R darstellen können. Beispielsweise müssen wir lange oder unendliche Sequenzen über den Dezimalpunkt hinaus auf die Anzahl der Bits kürzen, die wir speichern möchten.

- Diskretisierungsfehler entstehen durch unsere computergestützten Anpassungen von Analysis, Physik und anderem Aspekte der kontinuierlichen Mathematik. Wir machen zum Beispiel eine Näherung

$$\frac{dy}{dx} \quad \ddot{y} \quad \frac{y(x + \ddot{y}) \ddot{y} y(x)}{\ddot{y}}.$$

  Wir werden lernen, dass diese Näherung legitim und nützlich ist, aber abhängig von der Wahl von $\ddot{y}$ möglicherweise nicht ganz korrekt ist.

- Modellierungsfehler entstehen durch unvollständige oder ungenaue Beschreibungen der Probleme, die wir lösen möchten. Beispielsweise könnte eine Simulation zur Vorhersage des Wetters in Deutschland den kollektiven Flügelschlag von Schmetterlingen in Malaysia vernachlässigen, obwohl die Luftverdrängung dieser Schmetterlinge ausreichen könnte, um die Wettermuster anderswo etwas zu stören.

- Empirische Konstantenfehler entstehen durch schlechte Darstellungen physikalischer oder mathematischer Konstanten. Zum Beispiel können wir $\ddot{y}$ mithilfe einer Taylor-Folge berechnen, die wir vorzeitig beenden, und selbst Wissenschaftler kennen die Lichtgeschwindigkeit möglicherweise nicht einmal auf mehr als einige Stellen genau.

- Eingabefehler können von benutzergenerierten Näherungen von Parametern eines bestimmten Systems (und von Tippfehlern!) herrühren. Simulations- und numerische Techniken können zur Beantwortung von „Was-wäre-wenn“-Fragen eingesetzt werden, bei denen explorative Auswahlmöglichkeiten für Eingabeeinstellungen getroffen werden, um eine Vorstellung davon zu bekommen, wie sich ein System verhält.

**Beispiel 1.1** (Computerphysik). Angenommen, wir entwerfen ein System zur Simulation von Planeten, die sich um die Erde drehen. Das System löst im Wesentlichen die Newtonsche Gleichung F = ma, indem es zeitlich vorwärts gerichtete Kräfte integriert. Beispiele für Fehlerquellen in diesem System könnten sein:

- Kürzungsfehler: Verwendung von IEEE-Gleitkomma zur Darstellung von Parametern und Ausgaben des Systems und Kürzung bei der Berechnung der Produktma

- Diskretisierungsfehler: Ersetzen der Beschleunigung a durch eine geteilte Differenz

- Modellierungsfehler: Es wurde versäumt, die Auswirkungen des Mondes auf die Bewegung der Erde innerhalb des Planeten zu simulieren System

- Empirischer Fehler: Die Masse des Jupiter wird nur vierstellig eingegeben

- Eingabefehler: Der Benutzer möchte möglicherweise die Kosten für die Verbringung von Müll in den Weltraum abschätzen, anstatt eine Wall-E-Ansammlung auf der Erde zu riskieren, kann aber nur die Menge an Müll schätzen, die die Regierung bereit ist, auf diese Weise abzuwerfen

### 1.2.1 Klassifizierungsfehler

Angesichts unserer vorherigen Diskussion könnte davon ausgegangen werden, dass die folgenden beiden Zahlen den gleichen potenziellen Fehler aufweisen:

$$1 \pm 0{,}01$$

$$105 \pm 0{,}01$$

Obwohl er die Größe des Bereichs [1 ÿ 0,01, 1 + 0,01] hat, scheint der Bereich [105 ÿ 0,01, 105 + 0,01] eine zuverlässigere Messung zu kodieren, da der Fehler 0,01 im Vergleich zu 105 viel kleiner ist als zu 1.

Die Unterscheidung zwischen diesen beiden Fehlerklassen wird durch die Unterscheidung zwischen ab beschrieben gelöster Fehler und relativer Fehler:

**Definition 1.1** (Absoluter Fehler). Der absolute Fehler einer Messung ergibt sich aus der Differenz zwischen dem Näherungswert und dem zugrunde liegenden wahren Wert.

**Definition 1.2** (Relativer Fehler). Der relative Fehler einer Messung ergibt sich aus dem absoluten Fehler dividiert durch den wahren Wert.

Eine Möglichkeit, zwischen diesen beiden Fehlerarten zu unterscheiden, ist die Verwendung von Einheiten gegenüber Prozentsätzen.

**Beispiel 1.2** (Absoluter und relativer Fehler). Hier sind zwei äquivalente Aussagen in gegensätzlicher Form:

Absolut: 2 Zoll ± 0,02 Zoll

Relativ: 2 Zoll ± 1 %

In den meisten Anwendungen ist der wahre Wert unbekannt; Wäre dies nicht der Fall, wäre die Verwendung einer Näherung anstelle des wahren Werts möglicherweise ein zweifelhaftes Unterfangen. Es gibt zwei beliebte Möglichkeiten, dieses Problem zu lösen. Die erste besteht einfach darin, bei der Durchführung von Berechnungen konservativ zu sein: Nehmen Sie bei jedem Schritt die größtmögliche Fehlerschätzung vor und übertragen Sie diese Schätzungen nach Bedarf weiter. Solche konservativen Schätzungen sind insofern aussagekräftig, als wir, wenn sie klein sind, sehr sicher sein können, dass unsere Lösung nützlich ist.

Eine alternative Auflösung hängt davon ab, was Sie messen können. Nehmen wir zum Beispiel an, wir möchten die Gleichung f(x) = 0 für x bei gegebener Funktion f : **R** ÿ R lösen. Wir wissen, dass es irgendwo eine Wurzel x0 gibt , die genau f(x0) = 0 erfüllt , aber wenn wir das wüssten Rooten unseres Algorithmus wäre gar nicht nötig. In der Praxis kann unser Rechensystem ein xest ergeben, das f(xest) = $\ddot{y}$ für ein $\ddot{y}$ mit /$\ddot{y}$/ erfüllt 1. Wir können die Differenz x0 ÿ xest möglicherweise nicht auswerten , da x0 unbekannt ist. Andererseits können wir durch einfaches Auswerten von f f(xest) ÿ f(x0) ÿ f(xest) berechnen , da wir per Definition wissen, dass f(x0) = 0 ist. Dieser Wert gibt einen Anhaltspunkt für den Fehler unserer Berechnung.

Dieses Beispiel veranschaulicht die Unterscheidung zwischen Vorwärts- und Rückwärtsfehler. Der durch eine Näherung verursachte Vorwärtsfehler definiert höchstwahrscheinlich unsere Intuition für die Fehleranalyse als die Differenz zwischen der angenäherten und der tatsächlichen Lösung, aber wie wir bereits besprochen haben, ist eine Berechnung nicht immer möglich. Der Rückwärtsfehler hat jedoch die Besonderheit, dass er berechenbar ist, aber nicht unser genaues Ziel bei der Lösung eines bestimmten Problems ist. Wir können unsere Definition und Interpretation des Rückwärtsfehlers anpassen, wenn wir uns verschiedenen Problemen nähern. Eine geeignete, wenn auch vage Definition lautet jedoch wie folgt:

**Definition 1.3** (Rückwärtsfehler). Der Rückwärtsfehler wird durch den Betrag angegeben, um den sich eine Problemstellung ändern müsste, um eine gegebene Annäherung an ihre Lösung zu erreichen.

Diese Definition ist etwas unklar, daher veranschaulichen wir ihre Verwendung anhand einiger Beispiele.

**Beispiel 1.3** (Lineare Systeme). Angenommen, wir möchten das n × n-lineare System Ax = b lösen . Nennen Sie die wahre Lösung x0 ÿ A ÿ1b. In Wirklichkeit liefert unser System aufgrund von Kürzungsfehlern und anderen Problemen eine nahezu lösungsorientierte Lösung . Der Vorwärtsfehler dieser Näherung wird offensichtlich anhand der Differenz xest ÿx0 gemessen; In der Praxis ist es unmöglich, diesen Wert zu berechnen, da wir x0 nicht kennen . In Wirklichkeit ist xest die exakte Lösung eines modifizierten Systems Ax = best for best ÿ Axest; Daher könnten wir den Rückwärtsfehler anhand der Differenz b ÿ best messen. Im Gegensatz zum Vorwärtsfehler lässt sich dieser Fehler leicht berechnen, ohne A umzukehren, und es ist leicht zu erkennen, dass xest genau dann eine Lösung für das Problem ist, wenn der Rückwärtsfehler (oder Vorwärtsfehler) Null ist.

**Beispiel 1.4** (Gleichungen lösen, CITE). Angenommen, wir schreiben eine Funktion zum Finden von Quadratwurzeln positiver Zahlen, die ÿ 2 ÿ 1,4 ausgibt. Der Vorwärtsfehler beträgt |1,4 ÿ 1,41421 · · · | ÿ 0,0142. Beachten Sie, dass 1,42 = 1,96, der Rückwärtsfehler also |1,96 ÿ 2| beträgt = 0,04.

Die beiden obigen Beispiele zeigen ein größeres Muster, dass der Rückwärtsfehler viel einfacher zu berechnen ist als der Vorwärtsfehler. Beispielsweise erforderte die Auswertung des Vorwärtsfehlers in Beispiel 1.3 die Invertierung einer Matrix A, während die Auswertung des Rückwärtsfehlers nur eine Multiplikation mit A erforderte. In ähnlicher Weise ersetzte der Übergang vom Vorwärtsfehler zum Rückwärtsfehler in Beispiel 1.4 die Quadratwurzelberechnung durch Multiplikation.

### 1.2.2 Konditionierung, Stabilität und Genauigkeit

In fast jedem numerischen Problem bedeutet ein Null-Rückwärtsfehler einen Null-Vorwärtsfehler und umgekehrt.
Daher kann eine Software, die zur Lösung eines solchen Problems entwickelt wurde, sicherlich abbrechen, wenn sie feststellt, dass eine Kandidatenlösung keinen Rückwärtsfehler aufweist. Was aber, wenn der Rückwärtsfehler ungleich Null, aber klein ist?
Bedeutet dies unbedingt einen kleinen Vorwärtsfehler? Solche Fragen motivieren die Analyse der meisten numerischen Techniken, deren Ziel die Minimierung von Vorwärtsfehlern ist, in der Praxis jedoch nur Rückwärtsfehler messen können.

Wir möchten Änderungen im Rückwärtsfehler relativ zum Vorwärtsfehler analysieren, damit unsere Algorithmen mit Sicherheit sagen können, dass sie nur unter Verwendung des Rückwärtsfehlers akzeptable Lösungen hervorgebracht haben.
Dieser Zusammenhang kann für jedes Problem, das wir lösen möchten, unterschiedlich sein, daher nehmen wir am Ende folgende grobe Klassifizierung vor:

- Ein Problem ist unempfindlich oder gut konditioniert, wenn kleine Mengen an Rückwärtsfehlern kleine Mengen an Vorwärtsfehlern implizieren. Mit anderen Worten: Eine kleine Störung der Aussage eines gut konditionierten Problems führt nur zu einer kleinen Störung der wahren Lösung.

- Ein Problem ist sensibel oder schlecht konditioniert, wenn dies nicht der Fall ist.

**Beispiel 1.5** (ax = b). Nehmen wir als Spielzeugbeispiel an, dass wir die Lösung x0 ÿ b/a der linearen Gleichung ax = b für a, x, b ÿ R finden wollen. Der Vorwärtsfehler einer möglichen Lösung x ist durch x ÿ x0 gegeben , der Rückwärtsfehler dagegen gegeben durch b ÿ ax = a(x ÿ x0). Wenn also |a| 1 ist das Problem gut konditioniert, da kleine Werte des Rückwärtsfehlers a(x ÿ x0) noch kleinere Werte von x ÿ x0 implizieren ; im Gegensatz dazu, wenn |a| 1 Das Problem ist schlecht konditioniert, denn selbst wenn a(x ÿ x0) klein ist, kann der Vorwärtsfehler x ÿ x0 ÿ 1/a · a(x ÿ x0) bei gegebenem 1/a- Faktor groß sein.

Wir definieren die Bedingungszahl als Maß für die Sensibilität eines Problems:

**Definition 1.4** (Konditionsnummer). Die Bedingungszahl eines Problems ist das Verhältnis zwischen der Änderung seiner Lösung und der Änderung seiner Aussage bei kleinen Störungen. Alternativ ist es das Verhältnis von Vorwärts- zu Rückwärtsfehler bei kleinen Änderungen in der Problemstellung.

**Beispiel 1.6** (ax = b, Teil zwei). Wenn wir Beispiel 1.5 fortsetzen, können wir die Bedingungszahl genau berechnen:

$$c = \frac{\text{Vorwärtsfehler,}}{\text{Rückwärtsfehler}} = \frac{x \ddot{y} x0}{a(x \ddot{y} x0)} \ddot{y} \frac{1}{A}$$

Im Allgemeinen ist die Berechnung von Zustandszahlen fast so schwierig wie die Berechnung des Vorwärtsfehlers, und daher ist ihre genaue Berechnung wahrscheinlich unmöglich. Dennoch ist es oft möglich, Grenzen oder Näherungen für Bedingungszahlen zu finden, um zu beurteilen, wie vertrauenswürdig eine Lösung ist.

**Beispiel 1.7** (Wurzelfindung). Angenommen, wir erhalten eine glatte Funktion f : **R** ÿ **R** und möchten Werte x mit f(x) = 0 finden. Beachten Sie, dass f(x + ÿ) ÿ f(x) + ÿf (x). Somit könnte eine Annäherung an die Bedingungszahl zum Finden von x sein

$$\frac{\text{Änderung des Vorwärtsfehlers}}{\text{Änderung des Rückwärtsfehlers}} = \frac{(x + \ddot{y}) \ddot{y} x f(x}{+ \ddot{y}) \ddot{y} f(x) \ddot{y}} \ddot{y} \frac{}{\ddot{y}f (x)} = \frac{1}{f (x)}$$

Beachten Sie, dass diese Näherung mit der in Beispiel 1.6 übereinstimmt. Wenn wir x nicht kennen, können wir f (x) natürlich nicht auswerten , aber wenn wir die Form von f und die Grenze | betrachten können f | in der Nähe von x haben wir eine Vorstellung von der Worst-Case-Situation.

Vorwärts- und Rückwärtsfehler sind Maße für die Genauigkeit einer Lösung. Aus Gründen der wissenschaftlichen Wiederholbarkeit möchten wir außerdem stabile Algorithmen ableiten, die selbstkonsistente Lösungen für eine Klasse von Problemen liefern. Beispielsweise lohnt es sich möglicherweise nicht, einen Algorithmus zu implementieren, der nur in einem Fünftel der Fälle sehr genaue Lösungen generiert, selbst wenn wir die oben genannten Techniken anwenden können, um zu überprüfen, ob die Kandidatenlösung gut ist.

## 1.3 Praktische Aspekte

Die Unendlichkeit und Dichte der reellen Zahlen **R** kann bei der Implementierung numerischer Algorithmen schädliche Fehler verursachen. Während uns die in §1.2 vorgestellte Theorie der Fehleranalyse letztendlich dabei helfen wird, Garantien für die Qualität numerischer Techniken zu geben, die in zukünftigen Kapiteln vorgestellt werden, ist es erwähnenswert, bevor wir fortfahren, eine Reihe häufiger Fehler und „Fallstricke“ zu beachten, die bei Implementierungen numerischer Methoden vorkommen.

Wir haben den größten Übeltäter absichtlich zu Beginn in §1.1 eingeführt und wiederholen ihn zur wohlverdienten Hervorhebung

in größerer Schrift: Der Operator == und seine Äquivalente **sollten selten, wenn überhaupt, für Bruchwerte verwendet werden.**

Die Suche nach einem geeigneten Ersatz für == und entsprechenden Bedingungen zum Beenden einer numerischen Methode hängt von der betrachteten Technik ab. Beispiel 1.3 zeigt, dass eine Methode zur Lösung von Ax = b enden kann, wenn das Residuum b ÿ Ax Null ist; Da wir nicht explizit prüfen wollen, ob A*x==b ist, prüfen Implementierungen in der Praxis norm(A*xb)<epsilon. Beachten Sie, dass dieses Beispiel zwei Techniken demonstriert:

- Die Verwendung von Rückwärtsfehlerb ÿ Ax anstelle von Vorwärtsfehler, um zu bestimmen, wann beendet werden soll, Und
- Überprüfen, ob der Rückwärtsfehler kleiner als Epsilon ist, um das verbotene ==0-Prädikat zu vermeiden.

Der Parameter Epsilon hängt davon ab, wie genau die gewünschte Lösung sein muss, sowie von der Auflösung des verwendeten Zahlensystems.

Ein Programmierer, der diese Datentypen und Operationen verwendet, muss wachsam sein, wenn es darum geht, fehlerhafte numerische Operationen zu erkennen und zu verhindern. Betrachten Sie beispielsweise den folgenden Codeausschnitt zur Berechnung der Norm x2 für einen Vektor x ÿ Rn , der als 1D-Array x[] dargestellt wird:

```
double normSquared = 0; for ( int i
= 0; i < n ; i ++) normSquared += x [ i ]* x
                [ i ];
return sqrt ( normSquared );
```

Es ist leicht zu erkennen, dass in der Theorie mini |xi | ÿ x2/ ÿ n ÿ¯maxi |xi |, das heißt, die Norm von x liegt in der Größenordnung der Werte der in x enthaltenen Elemente. In der Berechnung von x2 verbirgt sich jedoch der Ausdruck x[i]*x[i]. Wenn es ein i gibt, so dass x[i] in der Größenordnung von DOUBLE MAX liegt, läuft das Produkt x[i]*x[i] über, obwohl x2 immer noch im Bereich der Doubles liegt. Ein solcher Überlauf lässt sich leicht verhindern, indem man x durch seinen Maximalwert dividiert, die Norm berechnet und zurückmultipliziert:

```
double maxElement = epsilon ; // Ich möchte nicht durch Null dividieren! for ( int i = 0; i < n ; i ++)

                maxElement = max ( maxElement for ( int    ,    Fabs ( x [ i ]));
i = 0; i < n ; i ++) {
                doppelt skaliert = x [ i ] / maxElement ; normSquared
                += skaliert * skaliert ;
}
return sqrt (normSquared) * maxElement;
```

Der Skalierungsfaktor beseitigt das Überlaufproblem, indem er sicherstellt, dass die summierten Elemente nicht größer als 1 sind.

Dieses kleine Beispiel zeigt einen von vielen Umständen, in denen ein einzelnes Codezeichen zu einem nicht offensichtlichen numerischen Problem führen kann. Während unsere Intuition aus der kontinuierlichen Mathematik ausreicht, um viele numerische Methoden zu generieren, müssen wir immer noch einmal überprüfen, ob die von uns verwendeten Operationen aus diskreter Sicht gültig sind.

### 1.3.1 Beispiel im größeren Maßstab: Summierung

Wir liefern nun ein Beispiel für ein numerisches Problem, das durch Arithmetik mit endlicher Genauigkeit verursacht wird und mit einem weniger offensichtlichen algorithmischen Trick gelöst werden kann.

Angenommen, wir möchten eine Liste von Gleitkommawerten zusammenfassen, eine Aufgabe, die von Systemen in der Buchhaltung, beim maschinellen Lernen, in der Grafik und in fast allen anderen Bereichen problemlos benötigt wird. Ein Codeausschnitt zur Lösung dieser Aufgabe, der zweifellos in unzähligen Anwendungen vorkommt, sieht wie folgt aus:

```
Doppelsumme = 0; for
( int i = 0; i < n ; i ++) sum += x [ i ];
```

Bevor wir fortfahren, ist es erwähnenswert, dass dies für die überwiegende Mehrheit der Anwendungen eine vollkommen stabile und sicherlich mathematisch gültige Technik ist.

Aber was kann schiefgehen? Betrachten Sie den Fall, in dem n groß ist und die meisten Werte x[i] klein und positiv sind. Wenn i groß genug ist, ist in diesem Fall die Variablensumme im Verhältnis zu x[i] groß. Letztendlich könnte sum so groß sein, dass x[i] nur die niederwertigsten Bits von sum beeinflusst, und im Extremfall könnte sum so groß sein, dass das Hinzufügen von x[i] überhaupt keine Auswirkung hat. Während ein einzelner solcher Fehler vielleicht keine große Sache ist, könnte die Gesamtwirkung, wenn dieser Fehler wiederholt gemacht wird, die Summe übersteigen, der wir überhaupt vertrauen können.

Um diesen Effekt mathematisch zu verstehen, nehmen wir an, dass die Berechnung einer Summe a + b um bis zu *ÿ* > 0 abweichen kann. Dann kann die obige Methode eindeutig einen Fehler in der Größenordnung von *nÿ induzieren,* der linear mit n wächst. Wenn die meisten Elemente x[i] tatsächlich in der Größenordnung von ÿ liegen, kann man der Summe überhaupt nicht trauen! Das ist ein enttäuschendes Ergebnis: Der Fehler kann so groß sein wie die Summe selbst.

Glücklicherweise gibt es viele Möglichkeiten, es besser zu machen. Wenn Sie beispielsweise die kleinsten Werte zuerst addieren, könnte dies dazu beitragen, deren kumulierten Effekt zu erklären. Paarweise Methoden, die Wertepaare von x[] rekursiv addieren und eine Summe bilden, sind ebenfalls stabiler, es kann jedoch schwierig sein, sie so effizient zu implementieren wie die obige for-Schleife. Glücklicherweise bietet ein Algorithmus von Kahan (CITE) eine einfach zu implementierende Methode der „kompensierten Summierung“, die fast genauso schnell ist.

Die nützliche Beobachtung hier ist, dass wir tatsächlich eine Annäherung an den Fehler in der Summe während einer bestimmten Iteration verfolgen können. Betrachten Sie insbesondere den Ausdruck

$$((a + b) \ddot{y} a) \ddot{y} b.$$

Offensichtlich ist dieser Ausdruck algebraisch Null. Zahlenmäßig ist dies jedoch möglicherweise nicht der Fall. Insbesondere kann die Summe (a + b) das Ergebnis runden, um es im Bereich der Gleitkommawerte zu halten. Die Subtraktion von a und b nacheinander ergibt dann eine Näherung des durch diese Operation verursachten Fehlers; Beachten Sie, dass die Subtraktionsoperationen wahrscheinlich besser konditioniert sind, da der Übergang von großen zu kleinen Zahlen aufgrund der Aufhebung Ziffern an Genauigkeit hinzufügt.

Somit läuft die Kahan-Technik wie folgt ab:

```
Doppelsumme = 0;
Doppelkompensation = 0; // eine Annäherung an den Fehler

for ( int i = 0; i < n ; i ++) { // versuchen, sowohl
                zu x [ i ] als auch zum fehlenden Teil wieder hinzuzufügen double nextTerm = x [ i ] +
                Kompensation ;

                // Berechne das Summationsergebnis dieser Iteration double nextSum = sum +
                nextTerm ;

                // Berechnen Sie die Kompensation als Differenz zwischen dem Term, den Sie // hinzufügen wollten, und dem tatsächlichen
                Ergebnis. kompensation = nextTerm - (nextSum -
                sum);

                sum = nextSum ;
}
```

Anstatt einfach die Summe beizubehalten, verfolgen wir jetzt die Summe und führen eine Näherungskompensation der Differenz zwischen der Summe und dem gewünschten Wert durch. Bei jeder Iteration versuchen wir, etwas hinzuzufügen

Diese Kompensation zusätzlich zum aktuellen Element von x[], und dann berechnen wir die Kompensation neu, um den letzten Fehler zu berücksichtigen.

Die Analyse des Kahan-Algorithmus erfordert eine sorgfältigere Buchführung als die Analyse der einfacheren inkrementellen Technik. Am Ende dieses Kapitels werden Sie eine Ableitung eines Fehlerausdrucks durchgehen. Das endgültige mathematische Ergebnis wird sein, dass sich der Fehler von $n\ddot{y}$ auf $O(\ddot{y} + n\ddot{y}^2)$ verbessert, erhebliche Verbesserung, wenn $0 < \ddot{y} < 1$.

Die Implementierung der Kahan-Summierung ist unkompliziert, verdoppelt aber die Operationsanzahl des resultierenden Programms mehr als. Auf diese Weise gibt es einen impliziten Kompromiss zwischen Geschwindigkeit und Genauigkeit, den Softwareentwickler eingehen müssen, wenn sie entscheiden, welche Technik am besten geeignet ist.

Im weiteren Sinne ist Kahans Algorithmus eine von mehreren Methoden, die die Anhäufung numerischer Fehler im Verlauf einer Berechnung, die aus mehr als einer Operation besteht, umgeht. Weitere Beispiele sind Bresenhams Algorithmus zur Rasterung von Linien (CITE), der nur ganzzahlige Arithmetik zum Zeichnen von Linien verwendet, selbst wenn sie Pixelzeilen und -spalten an nicht ganzzahligen Stellen schneiden, und die schnelle Fourier-Transformation (CITE), die effektiv die binäre Partition nutzt oben beschriebenen Summationstrick.

## 1.4 Probleme

**Problem 1.1.** Hier ist ein Problem.

# Teil II

# Lineare Algebra

# Kapitel 2

# Lineare Systeme und die LU Zersetzung

In Kapitel 0 haben wir verschiedene Situationen besprochen, in denen lineare Gleichungssysteme Ax = b in der mathematischen Theorie und in der Praxis auftreten. In diesem Kapitel gehen wir das Grundproblem direkt an und untersuchen numerische Methoden zur Lösung solcher Systeme.

## 2.1 Lösbarkeit linearer Systeme

Wie in §0.3.3 eingeführt, sind lineare Gleichungssysteme wie

$$\begin{aligned} 3x + 2y &= 6 \\ -4x + y &= 7 \end{aligned}$$

kann in Matrixform geschrieben werden wie in

$$\begin{bmatrix} 3 & 2 \\ -4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \end{bmatrix}.$$

Allgemeiner können wir Systeme der Form Ax =b für $A \in \mathbb{R}^{m\times n}$ schreiben, $x \in \mathbb{R}^n$, und$b \in \mathbb{R}^m$.

Die Lösbarkeit des Systems muss in einen von drei Fällen fallen:

1. Das System lässt möglicherweise keine Lösungen zu, wie in:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

   Dieses System erfordert x = −1 und x = 1 gleichzeitig, offensichtlich zwei inkompatible Bedingungen.

2. Das System kann eine einzelne Lösung zulassen; Beispielsweise wird das System am Anfang dieses Abschnitts durch (x, y) = (−8/11, 45/11) gelöst.

3. Das System kann unendlich viele Lösungen zulassen, zB 0x =0. Beachten Sie, dass, wenn ein System Ax =b zwei Lösungen x0 und x1 zulässt, es automatisch unendlich viele Lösungen der Form cx0 + (1 ÿ c)x1 für c ÿ R hat, da

$$A(cx_0 + (1 - c)x_1) = cAx_0 + (1 - c)Ax_1 = cb + (1 - c)b = b.$$

Dieses lineare System würde als unterbestimmt bezeichnet.

Im Allgemeinen hängt die Lösbarkeit eines Systems sowohl von A als auch von b ab. Wenn wir zum Beispiel das obige unlösbare System so ändern, dass es ist

$$\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix} \quad \begin{matrix} x \\ j \end{matrix} \quad = \quad \begin{matrix} 1 \\ 1 \end{matrix} \quad ,$$

dann bewegt sich das System von null zu unendlich vielen Lösungen der Form (1, y). Tatsächlich lässt jede Matrix A eine rechte Seite b zu, so dass Ax = b lösbar ist, da Ax = 0 immer durch x ÿ 0 gelöst werden kann, unabhängig von A. Erinnern Sie sich an §0.3.1, dass die Matrix-Vektor-Multiplikation betrachtet werden kann als lineares Kombinieren der Spalten von A mit Gewichten von x. Daher können wir, wie in §0.3.3 erwähnt, erwarten, dass Ax =b genau dann lösbar ist, wenn b im Spaltenraum von A liegt.

Im Großen und Ganzen hat die „Form“ der Matrix A ÿ Rm×n erheblichen Einfluss auf die Lösbarkeit von Ax =b. Denken Sie daran, dass die Spalten von A m-dimensionale Vektoren sind. Betrachten Sie zunächst den Fall, dass A „breit“ ist, das heißt, wenn es mehr Spalten als Zeilen hat (n > m). Jede Spalte ist ein Vektor in Rm, daher kann der Spaltenraum höchstens die Dimension m haben. Da n > m ist, müssen die n Spalten von A dann linear abhängig sein; Dies impliziert, dass es ein x0 =0 gibt, so dass Ax0 =0 ist. Wenn wir dann Ax = b nach x auflösen können, dann ist A(x + ÿx0) = Ax + ÿAx0 = b +0 = b, was zeigt, dass es tatsächlich unendlich viele Lösungen x für Ax = b gibt. Mit anderen Worten: Wir haben gezeigt, dass kein breites Matrixsystem eine eindeutige Lösung zulässt.

Wenn A „groß“ ist, das heißt, wenn es mehr Zeilen als Spalten hat (m > n), können die n Spalten Rm nicht überspannen. Somit existiert ein Vektorb0 ÿ Rm\col A. Per Definition kann thisb0 Ax =b0 für kein x erfüllen. Mit anderen Worten: Jede hohe Matrix A lässt Systeme Ax =b0 zu , die nicht lösbar sind.

Beide oben genannten Situationen sind alles andere als günstig für den Entwurf numerischer Algorithmen. Wenn beispielsweise ein lineares System viele Lösungen zulässt, müssen wir zunächst definieren, welche Lösung vom Benutzer gewünscht wird: Schließlich ist die Lösung x + 1031x0 möglicherweise nicht so aussagekräftig wie x ÿ 0,1x0. Auf der anderen Seite ist im großen Fall jede kleine Störung Ax =b + ÿb0 nicht mehr lösbar , selbst wenn Ax = b für ein bestimmtes b lösbar ist; Diese Situation kann einfach deshalb auftreten, weil die im letzten Kapitel besprochenen Rundungsverfahren A und B überhaupt nur approximieren können.

Angesichts dieser Komplikationen werden wir in diesem Kapitel einige vereinfachende Annahmen treffen: •

Wir werden nur das Quadrat A ÿ Rn×n betrachten ·

• Wir gehen davon aus, dass A nichtsingulär ist, das heißt, dass Ax =b für jedes b lösbar ist.

Erinnern Sie sich an §0.3.3, dass die Nichtsingularitätsbedingung der Anforderung entspricht, dass die Spalten von A überspannt Rn und impliziert die Existenz einer Matrix $A^{ÿ1}$ erfüllt A ÿ1A = AAÿ1 = In×n.

Eine irreführende Beobachtung ist die Annahme, dass das Lösen von Ax =b gleichbedeutend mit der expliziten $A^{ÿ1}$ Berechnung der Matrix und der anschließenden Multiplikation ist, um x ÿ A ÿ1b zu finden. Obwohl diese Lösungsstrategie sicherlich gültig ist, kann sie einen erheblichen Overkill darstellen: Schließlich sind wir nur an den n 2 -Werten in x und nicht an nÿ1 Werte in A ÿ1 interessiert. Selbst wenn A sich gut benimmt, kann es außerdem sein, dass A geschrieben wird führt zu numerischen Schwierigkeiten, die umgangen werden können.

## 2.2 Ad-hoc-Lösungsstrategien

In der einführenden Algebra betrachten wir das Problem der Lösung eines linearen Gleichungssystems oft als eine Kunstform. Die Strategie besteht darin, Variablen zu „isolieren“ und iterativ alternative Formen des linearen Systems zu schreiben, bis jede Zeile die Form x = const hat.

Bei der Formulierung systematischer Algorithmen zur Lösung linearer Systeme ist es aufschlussreich, ein Beispiel dieses Lösungsprozesses durchzuführen. Betrachten Sie das folgende System:

$$\begin{aligned} y \ddot{y} z &= \ddot{y}1 \\ 3x \ddot{y} y + z &= 4 \\ x + y \ddot{y} 2z &= \ddot{y}3 \end{aligned}$$

Parallel dazu können wir eine Matrixversion dieses Systems pflegen. Anstatt Ax = b explizit auszugeben , können wir etwas Platz sparen, indem wir die folgende „erweiterte“ Matrix schreiben:

$$\left[\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right.\begin{array}{l} 0\ 1\ \ddot{y}1\ \ddot{y}1 \\ 3\ \ddot{y}1\ 1\ 4 \\ 1\ 1\ \ddot{y}2\ \ddot{y}3 \end{array}\left|\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right]$$

Wir können lineare Systeme immer auf diese Weise schreiben, solange wir uns darauf einigen, dass die Variablen auf der linken Seite der Gleichungen und die Konstanten auf der rechten Seite bleiben.

Vielleicht möchten wir uns zuerst mit der Variablen x befassen. Der Einfachheit halber können wir die Zeilen des Systems so vertauschen, dass die dritte Gleichung zuerst erscheint:

$$\begin{aligned} x + y \ddot{y} 2z &= \ddot{y}3 \\ y \ddot{y} z &= \ddot{y}1 \\ 3x \ddot{y} y + z &= 4 \end{aligned}$$

$$\left[\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right.\begin{array}{l} 1\ 1\ \ddot{y}2\ \ddot{y}3 \\ 0\ 1\ \ddot{y}1\ \ddot{y}1 \\ 3\ \ddot{y}1\ 1\ 4 \end{array}\left|\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right]$$

Wir können dann die erste Gleichung in die dritte einsetzen, um den 3x-Term zu eliminieren. Dies ist dasselbe, als würde man die Beziehung x + y ÿ 2z = ÿ3 um ÿ3 skalieren und das Ergebnis zur dritten Gleichung hinzufügen:

$$\begin{aligned} x + y \ddot{y} 2z &= \ddot{y}3 \\ y \ddot{y} z &= \ddot{y}1 \\ \ddot{y}4y + 7z &= 13 \end{aligned}$$

$$\left[\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right.\begin{array}{l} 1\ 1\ \ddot{y}2\ \ddot{y}3 \\ 0\ 1\ \ddot{y}1\ \ddot{y}1 \\ 0\ \ddot{y}4\ 7\ 13 \end{array}\left|\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right]$$

Um y aus der dritten Gleichung zu eliminieren, können wir auf ähnliche Weise die zweite Gleichung mit 4 multiplizieren und das Ergebnis zur dritten addieren:

$$\begin{aligned} x + y \ddot{y} 2z &= \ddot{y}3 \\ y \ddot{y} z &= \ddot{y}1 \\ 3z &= 9 \end{aligned}$$

$$\left[\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right.\begin{array}{l} 1\ 1\ \ddot{y}2\ \ddot{y}3 \\ 0\ 1\ \ddot{y}1\ \ddot{y}1 \\ 0\ 0\ 3\ 9 \end{array}\left|\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right]$$

Wir haben jetzt z isoliert! Daher können wir die dritte Zeile um 1/3 skalieren , um einen Ausdruck für z zu erhalten:

$$\begin{aligned} x + y \ddot{y} 2z &= \ddot{y}3 \\ y \ddot{y} z &= \ddot{y}1 \\ z &= 3 \end{aligned}$$

$$\left[\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right.\begin{array}{l} 1\ 1\ \ddot{y}2\ \ddot{y}3 \\ 0\ 1\ \ddot{y}1\ \ddot{y}1 \\ 0\ 0\ 1\ \ 3 \end{array}\left|\begin{array}{c} \\ \ddot{y} \\ \\ \ddot{y} \end{array}\right]$$

Jetzt können wir z = 3 in die anderen beiden Gleichungen einsetzen, um z aus allen bis auf die letzte Zeile zu entfernen:

$$\begin{array}{l} x + y = 3 \\ 2z = 3 \end{array} \qquad \left[\begin{array}{ccc|c} 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Schließlich führen wir eine ähnliche Substitution für y durch, um die Lösung abzuschließen:

$$\begin{array}{l} x = 1 \\ = 2 \\ z = 3 \end{array} \qquad \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Dieses Beispiel mag etwas pedantisch sein, aber wenn wir auf unsere Strategie zurückblicken, ergeben sich einige wichtige Beobachtungen dazu, wie wir bei der Lösung linearer Systeme vorgehen können:

- Wir haben aufeinanderfolgende Systeme $A_i x = b_i$ geschrieben , die als Vereinfachungen des Originals angesehen werden können $Ax_t = b$.
- Wir haben das System gelöst, ohne jemals A aufzuschreiben $A^{-1}$ .
- Wir haben wiederholt einige einfache Operationen verwendet: Skalieren, Hinzufügen und Permutieren von Zeilen der System.
- Die gleichen Operationen wurden auf A und b angewendet. Wenn wir die k-te Zeile von A skaliert haben, haben wir auch skaliert die k-te Reihe von$b$. Wenn wir die Zeilen k und von A hinzugefügt haben, haben wir die Zeilen k und of$b$ hinzugefügt.
- Weniger offensichtlich war, dass die Lösungsschritte nicht von b abhingen. Das heißt, dass alle unsere Lösungsentscheidungen dadurch motiviert waren, dass wir Werte ungleich Null in A eliminierten, anstatt die Werte in b;b zu untersuchen, die einfach mit von der Partie waren.
- Wir schlossen ab, als wir das System auf $I_{n \times n} x = b$ reduzierten.

Wir werden all diese allgemeinen Beobachtungen zur Lösung linearer Systeme zu unserem Vorteil nutzen.

# 2.3 Zeilenoperationen kodieren

Wenn wir uns das Beispiel in §2.2 noch einmal ansehen, sehen wir, dass die Lösung des linearen Systems eigentlich nur die Anwendung von drei Operationen erforderte: Permutation, Zeilenskalierung und Addition der Skalierung einer Zeile zur anderen. Tatsächlich können wir auf diese Weise jedes lineare System lösen, daher lohnt es sich, diese Operationen genauer zu untersuchen.

## 2.3.1 Permutation

Unser erster Schritt in §2.2 bestand darin, zwei Zeilen im Gleichungssystem zu vertauschen. Allgemeiner könnten wir die Zeilen einer Matrize mit den Zahlen 1, ... indizieren. . . , M. Dann kann eine Permutation dieser Zeilen als Funktion $\sigma$ geschrieben werden , sodass die Liste $\sigma(1), \ldots, \sigma(m)$ deckt den gleichen Satz von Indizes ab.

Wenn ek die k-te Standardbasisfunktion ist, ist leicht zu erkennen, dass das Produkt $e_k A$ ergibt die k-te Zeile der Matrix A. Daher können wir diese Zeilenvektoren vertikal „stapeln“ oder verketten, um eine Matrix zu erhalten, die die Zeilen gemäß ÿ permutiert :

$$P_ÿ = \begin{bmatrix} e_{ÿ(1)} \\ e_{ÿ(2)} \\ \vdots \\ e_{ÿ(m)} \end{bmatrix}$$

Das heißt, das Produkt *PÿA* ist genau die Matrix A mit gemäß ÿ permutierten Zeilen.

**Beispiel 2.1** (Permutationsmatrizen). Angenommen, wir möchten Zeilen einer Matrix in R3×3 mit ÿ(1) = 2, ÿ(2) = 3 und ÿ(3) = 1 permutieren. Nach unserer Formel hätten wir

$$P_ÿ = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Aus Beispiel 2.1 können wir sehen, dass *Pÿ* Einsen an Positionen der Form (k, ÿ(k)) und an anderen Stellen Nullen hat. Das Paar (k, ÿ(k)) stellt die Aussage dar: „Wir möchten, dass Zeile k der Ausgabematrix Zeile ÿ(k) aus der Eingabematrix ist.“ Basierend auf dieser Beschreibung einer Permutationsmatrix ist es leicht zu erkennen, dass die Umkehrung von *Pÿ* die Transponierte P ist, da dadurch einfach die Rollen der Zeilen und Spalten vertauscht werden – jetzt nehmen wir Zeile ÿ(k) der Eingabe und fügen sie ein Zeile k der Ausgabe. Mit anderen Worten, $P_ÿ P_ÿ = I_{m×m}$.

### 2.3.2 Zeilenskalierung

Angenommen, wir schreiben eine Liste von Konstanten a1, . . . , am und versuchen, die k-te Zeile einer Matrix A mit ak zu skalieren . Dies wird offensichtlich durch die Anwendung der Skalierungsmatrix Sa erreicht , die gegeben ist durch:

$$S_a = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_m \end{bmatrix}$$

Unter der Annahme, dass alle ak die Bedingung ak = 0 erfüllen, ist es einfach, Sa durch „Rückskalierung“ umzukehren :

$$S_A^{ÿ^{-1}} = S_{1/a_ÿ} = \begin{bmatrix} 1/a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1/a_m \end{bmatrix}$$

### 2.3.3 Eliminierung

Nehmen wir abschließend an, wir möchten Zeile k um eine Konstante c skalieren und das Ergebnis zu Zeile hinzufügen. Dieser Vorgang mag weniger natürlich erscheinen als die beiden vorherigen, ist aber tatsächlich recht praktisch: Es ist der einzige, den wir durchführen

müssen Gleichungen aus verschiedenen Zeilen des linearen Systems kombinieren! Wir werden diese Operation mithilfe einer „Eliminationsmatrix“ M realisieren, sodass das Produkt MA diese Operation auf Matrix A anwendet.

Denken Sie daran, dass das Produkt $e_k A$ wählt die k-te Zeile von A aus. Die Vormultiplikation mit e ergibt dann a matrizee k A, das Null ist, außer dass die -te Zeile gleich dem k-ten von A ist.

**Beispiel 2.2** (Konstruktion der Eliminierungsmatrix). Nehmen

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Angenommen, wir möchten die dritte Zeile von A ÿ R3×3 isolieren und in Zeile zwei verschieben. Wie oben erläutert, wird dieser Vorgang durch Schreiben erreicht:

$$e_{2e_3} A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & & \\ 7 & 8 & 9 & \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 7 & 8 & 9 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ 7 & 8 & 9 \\ 0 & 0 & 0 \end{pmatrix}$$

Natürlich haben wir oben von rechts nach links multipliziert, hätten das Produkt aber genauso gut als (e2e) gruppieren können$_3$ )A. Der Aufbau dieses Produkts ist leicht zu erkennen:

$$e_{2e_3} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad 0\,0\,1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & & \\ 0 & 0 & 0 & \end{pmatrix}$$

Es ist uns gelungen, Zeile k zu isolieren und in Zeile zu verschieben. Unsere ursprüngliche Eliminierungsoperation wollte c mal Zeile k zu Zeile A = k (In×n + cee) addieren, was wir nun als Summe A + cee erreichen können

$_k$ )A.

**Beispiel 2.3** (Ein System lösen). Wir können nun jede unserer Operationen aus Abschnitt 2.2 mithilfe der Matrizen kodieren, die wir oben erstellt haben:

1. Permutieren Sie die Zeilen, um die dritte Gleichung in die erste Zeile zu verschieben:

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

2. Skalieren Sie Zeile eins um -3 und addieren Sie das Ergebnis zu Zeile drei: E1 = I3×3 ÿ 3e3e 1

3. Skalieren Sie Zeile zwei mit 4 und addieren Sie das Ergebnis zu Zeile drei: E2 = I3×3 + 4e3e 2

4. Skalieren Sie Zeile drei um 1/3: S = diag(1, 1, 1/3)

5. Skalieren Sie Zeile drei um 2 und fügen Sie sie zur ersten Zeile hinzu: E3 = I3×3 + 2e1e3

6. Fügen Sie Zeile drei zu Zeile zwei hinzu: E4 = I3×3 +e2e 3

7. Skalieren Sie Zeile drei um -1 und addieren Sie das Ergebnis zu Zeile eins: E5 = I3×3 ÿe1e 3

Somit erfüllt die Umkehrung von A in Abschnitt 2.2

$$A^{ÿ1} = E5E4E3SE2E1P.$$

Stellen Sie sicher, dass Sie verstehen, warum diese Matrizen in umgekehrter Reihenfolge erscheinen!

## 2.4 Gaußsche Eliminierung

Die in Abschnitt 2.2 gewählte Schrittfolge war keineswegs eindeutig: Es gibt viele verschiedene Wege, die zur Lösung von Ax =b führen können. Unsere Schritte folgten jedoch der Strategie der Gaußschen Eliminierung, einem berühmten Algorithmus zur Lösung linearer Gleichungssysteme.

Nehmen wir allgemeiner an, dass unser System die folgende „Form“ hat:

$$A\,|\,b = \begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array} \left[\begin{array}{ccccc} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right] \left|\begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array}\right.$$

Der Algorithmus läuft in den unten beschriebenen Phasen ab.

### 2.4.1 Stürmerwechsel

Betrachten Sie das obere linke Element unserer Matrix:

$$A\,|\,b = \begin{array}{c} ÿ \\ \\ \\ ŷŷŷŷ \end{array} \left[\begin{array}{ccccc} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right] \left|\begin{array}{c} ÿ \\ \\ \\ ŷŷŷŷ \end{array}\right.$$

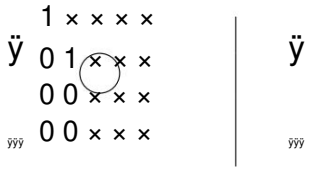
Wir nennen dieses Element unseren ersten Pivot und gehen davon aus, dass es ungleich Null ist; Wenn es Null ist, können wir Zeilen so permutieren, dass dies nicht der Fall ist. Wir wenden zunächst eine Skalierungsmatrix an, sodass der Pivot gleich eins ist:

$$\begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array} \left[\begin{array}{ccccc} 1 & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right] \left|\begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array}\right.$$

Jetzt verwenden wir die Zeile, die den Pivot enthält, um alle anderen Werte darunter in derselben Spalte zu entfernen:

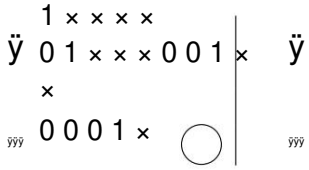
$$\begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array} \left[\begin{array}{ccccccccc} 1 & \times & \times & \times & \times & 0 & \times & \times & \times \\ \times & 0 & \times & \times & \times & \times & 0 & \times & \times \\ \times & \times & & & & & & & \end{array}\right] \left|\begin{array}{c} ÿ \\ \\ \\ ŷŷŷ \end{array}\right.$$

Wir verschieben nun unseren Pivot in die nächste Zeile und wiederholen eine ähnliche Reihe von Vorgängen:



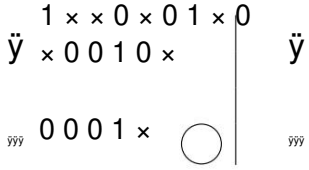
Beachten Sie, dass hier etwas Schönes passiert. Nachdem der erste Pivot aus allen anderen Zeilen entfernt wurde, ist die erste Spalte unter Zeile 1 eine Null. Das bedeutet, dass wir problemlos Vielfache von Zeile zwei zu den darunter liegenden Zeilen addieren können, ohne dass sich dies auf die Nullen in Spalte eins auswirkt.

Wir wiederholen diesen Vorgang, bis die Matrix die obere Dreiecksform annimmt:

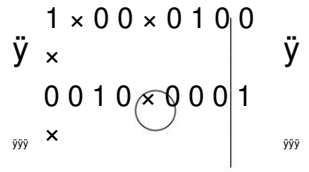


## 2.4.2 Rückenwechsel

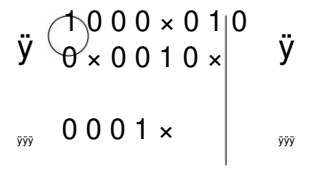
Das Entfernen der verbleibenden × aus dem System ist nun ein unkomplizierter Prozess. Nun gehen wir in umgekehrter Reihenreihenfolge vor und eliminieren rückwärts. Nach der ersten Reihe von Rückersetzungsschritten erhalten wir beispielsweise die folgende Form:



In ähnlicher Weise ergibt die zweite Iteration:



Nach unserem letzten Eliminierungsschritt bleibt uns die gewünschte Form übrig:



Die rechte Seite ist nun die Lösung des linearen Systems Ax =b.

## 2.4.3 Analyse der Gaußschen Eliminierung

Jede Zeilenoperation bei der Gaußschen Eliminierung – Skalierung, Eliminierung und Vertauschen zweier Zeilen – nimmt offensichtlich O(n) Zeit in Anspruch, da Sie über alle n Elemente einer Zeile iterieren müssen (bzw

zwei) von A. Sobald wir einen Pivot ausgewählt haben, müssen wir n Vorwärts- oder Rückwärtsersetzungen in den Zeilen unter bzw. über diesem Drehpunkt; Dies bedeutet, dass die Arbeit für einen einzelnen Pivot insgesamt O(n) beträgt $^2$ vornehmen . Insgesamt wählen wir einen Pivot pro Zeile und fügen einen Endfaktor von n hinzu. Daher ist es leicht, diese Gaußsche Funktion zu erkennen Eliminierungsläufe in O(n $^3$ ) Zeit.

Eine Entscheidung, die während der Gaußschen Eliminierung stattfindet und die wir nicht besprochen haben, ist die Wahl der Drehpunkte. Denken Sie daran, dass wir Zeilen des linearen Systems nach Belieben permutieren können, bevor wir eine Rück- oder Vorwärtssubstitution durchführen. Diese Operation ist notwendig, um mit allen möglichen Matrizen A umgehen zu können. Überlegen Sie beispielsweise, was passieren würde, wenn wir im Folgenden keine Pivotierung verwenden würden Matrix:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Beachten Sie, dass das eingekreiste Element genau Null ist, daher können wir nicht damit rechnen, Zeile eins durch eine beliebige Zahl zu dividieren, um diese 0 durch eine 1 zu ersetzen. Das bedeutet nicht, dass das System nicht lösbar ist, sondern nur, dass wir eine Pivotierung durchführen müssen, die durch Vertauschen erreicht wird erste und zweite Reihe, um einen Wert ungleich Null in diesen Steckplatz einzufügen.

Nehmen wir allgemeiner an, dass unsere Matrix wie folgt aussieht:

$$A = \begin{bmatrix} \ddot{y} & 1 \\ 1 & 0 \end{bmatrix},$$

wobei $0 < \ddot{y}$ 1. Wenn wir nicht schwenken, ergibt die erste Iteration der Gaußschen Eliminierung:

$$\tilde{A} = \begin{bmatrix} 1 & 1/\ddot{y} \\ \ddot{y}1/\ddot{y} & 0 \end{bmatrix},$$

Wir haben eine Matrix A, die fast wie eine Permutationsmatrix aussieht (tatsächlich eine sehr einfache $^{\ddot{y}1\ \ddot{y}\ A}$ , A Möglichkeit, das System zu lösen!) in ein System mit potenziell **großen** Werten *1/ÿ umgewandelt.*

Dieses Beispiel zeigt, dass es Fälle gibt, in denen wir möglicherweise einen Wechsel wünschen, auch wenn dies streng genommen nicht notwendig ist. Da wir mit dem Kehrwert des Pivotwerts skalieren, besteht die numerisch stabilste Option eindeutig darin, einen großen Pivotwert zu haben: Kleine Pivotwerte haben große Kehrwerte, wodurch Zahlen in Regimen, die wahrscheinlich an Präzision verlieren, auf große Werte skaliert werden. Es gibt zwei bekannte Pivot-Strategien:

1. Bei der teilweisen Pivotierung wird die aktuelle Spalte durchsucht und die Zeilen der Matrix so permutiert der größte absolute Wert erscheint auf der Diagonale.

2. Beim vollständigen Pivotieren wird die **gesamte** Matrix durchlaufen und sowohl Zeilen als auch Spalten vertauscht, um den größtmöglichen Wert auf der Diagonale zu erhalten. Beachten Sie, dass das Permutieren von Spalten einer Matrix eine gültige Operation ist: Sie entspricht dem Ändern der Beschriftung der Variablen im System oder dem Nachmultiplizieren von A mit einer Permutation.

**Beispiel 2.4** (Pivotieren). Angenommen, nach der ersten Iteration der Gaußschen Eliminierung erhalten wir die folgende Matrix:

$$\begin{bmatrix} 1 & 10 & \ddot{y}10 \\ \ddot{y} & 0 & 0{,}1 & 9 & \ddot{y} \\ \ddot{y}\ddot{y} & 0 & 4 & 6.2 & \ddot{y}\ddot{y} \end{bmatrix}$$

Wenn wir eine teilweise Pivotierung implementieren, schauen wir nur in der zweiten Spalte und vertauschen die zweite und dritte Zeile; Beachten Sie, dass wir die 10 in der ersten Zeile belassen, da diese bereits vom Algorithmus besucht wurde:

$$\begin{matrix} & 1\ 10\ ÿ10 & \\ ÿ & 0\ 4\ 6.2 & ÿ \\ ÿ & 0\ 0{,}1\ 9 & ÿ \end{matrix}$$

Wenn wir das vollständige Pivotieren implementieren, verschieben wir die 9:

$$\begin{matrix} & 1\ ÿ10\ 10 & \\ ÿ & 0\ 9\ 0{,}1 & ÿ \\ ÿ & 0\ 6{,}2\ 4 & ÿ \end{matrix}$$

Offensichtlich liefert die vollständige Pivotierung die bestmöglichen Zahlen, allerdings ist die Suche nach großen Elementen in der Matrix teurer.

## 2,5 LU-Faktorisierung

Es kommt oft vor, dass wir eine Folge von Problemen Ax1 =b1, Ax2 =b2, . lösen möchten . . . . Wie wir bereits besprochen haben, hängen die Schritte der Gaußschen Eliminierung zur Lösung von Ax =bk hauptsächlich von der Struktur von A und nicht von den Werten in einem bestimmten bk ab . Da A hier konstant gehalten wird, möchten wir uns vielleicht an die Schritte „erinnern“, die wir zur Lösung des Systems unternommen haben, damit wir nicht jedes Mal, wenn uns ein neues Problem präsentiert wird, bei Null anfangen müssen.

Dies bestätigt den Verdacht, dass wir einige der O(n) bewegen können$^3$) für die Gaußsche Eliminierung in der Vorberechnungszeit erinnern wir uns an das obere Dreieckssystem, das sich nach der Vorwärtssubstitutionsstufe ergibt:

$$\left[\begin{matrix} 1 \times \times \times \times & \\ ÿ\ 0\ 1 \times \times \times\ 0\ 0\ 1 \times & ÿ \\ \times & \\ ÿÿÿ\ 0\ 0\ 0\ 1 \times & ÿÿÿ \end{matrix}\right]$$

Tatsächlich benötigt die Lösung dieses Systems durch Rücksubstitution $^2$) Zeit! Warum? Zurück Substitution nur O(n, in diesem Fall ist dies dank der Struktur der Nullstellen im System viel einfacher. Beispielsweise erhalten wir in der ersten Reihe von Rücksubstitutionen die folgende Matrix:

$$\left[\begin{matrix} 1 \times \times\ 0 \times 0 \times 0 \times & 0 \\ ÿ\ 1\ 0 \times 1 & ÿ \\ 0 & \\ ÿÿÿ\ 0\ 0\ 0\ 1 \times & ÿÿÿ \end{matrix}\right]$$

Da wir wissen, dass die (eingekreisten) Werte links vom Pivot konstruktionsbedingt Null sind, müssen wir sie nicht explizit kopieren. Somit dauerte dieser Schritt nur O(n) Zeit und nicht O(n$^2$) für die Vorwärtssubstitution.

Nun führt unser nächster Pivot eine ähnliche Ersetzung durch:

$$\left[\begin{matrix} 1 \times 0\ 0 \times & \\ ÿ\ 0\quad 1\ 0\ 0 \times & ÿ \\ 0\quad 0\ 1\ 0 \times 0\ 0\ 1 \times & \\ ÿÿÿ\ 0 & ÿÿÿ \end{matrix}\right]$$

Auch hier müssen die Nullen auf beiden Seiten der 1 nicht explizit kopiert werden.

So haben wir herausgefunden:

**Überwachung.** Während die Gaußsche Eliminierung O(n) benötigt $^3$) Zeit, das Lösen von Dreieckssystemen dauert O(n$^2$) Zeit.

## 2.5.1 Konstruktion der Faktorisierung

Erinnern Sie sich an §2.3, dass alle Operationen der Gaußschen Eliminierung als Vormultiplikation von Ax = b mit verschiedenen Matrizen M betrachtet werden können, um ein einfacheres System (MA)x = Mb zu erhalten. Wie wir in Beispiel 2.3 gezeigt haben, stellt von diesem Standpunkt aus jeder Schritt der Gaußschen Eliminierung ein System (Mk · · · M2M1A)x = Mk · · · M2M1 b dar. Natürlich ist die explizite Speicherung dieser Matrizen Mk als n × n Objekte übertrieben, aber wenn man diese Interpretation aus theoretischer Sicht im Hinterkopf behält, werden viele unserer Berechnungen vereinfacht.

Nach der Vorwärtssubstitutionsphase der Gaußschen Eliminierung bleibt uns eine obere Dreiecksmatrix übrig, die wir U ÿ Rn×n nennen können . Aus der Perspektive der Matrixmultiplikation ist das möglich schreiben:

$$Mk \cdot \cdot \cdot M1A = U,$$

oder gleichwertig,

$$\begin{aligned} A &= (Mk \cdot \cdot \cdot M1)\ ÿ1U \\ &= (Mÿ1_1\ \ Mÿ1_2 \cdot \cdot \cdot Mÿ1_k)U \\ &ÿ LU, \text{ wenn wir die Definition } L ÿ Mÿ1 \text{ machen } {}_1\ Mÿ1_2 \cdot \cdot \cdot Mÿ1_k. \end{aligned}$$

Wir wissen noch nichts über die Struktur von L, aber wir wissen, dass Systeme der Form Uy = d einfacher zu lösen sind, da U eine obere Dreiecksform ist. Wenn L ebenso schön ist, könnten wir Ax =b in zwei Schritten lösen, indem wir (LU)x =b oder x = Uÿ1L ÿ1b schreiben: 1. Lösen Sie Ly =b nach y auf,

was y = L ÿ1b ergibt.

2. Nachdem wir nun y haben, lösen Sie Ux = y, was x = Uÿ1y = Uÿ1 (L ÿ1b) = (LU) ÿ1b = A ÿ1b ergibt.
Wir wissen bereits, dass dieser Schritt nur O(n) erfordert $^2$) Zeit.

Unsere verbleibende Aufgabe besteht darin, sicherzustellen, dass L eine schöne Struktur hat, die das Lösen von Ly = b einfacher macht als das Lösen von Ax =b. Glücklicherweise – und nicht überraschend – werden wir feststellen, dass L eine untere Dreiecksform ist und daher mit O(n) gelöst werden kann$^2$) Vorwärtssubstitution.

Um dies zu sehen, nehmen wir zunächst an, dass wir kein Pivotieren implementieren. Dann jede unserer Matrizen Mk ist entweder eine Skalierungsmatrix oder hat die Struktur

$$Mk = In\times n + cee\ _k,$$

wobei > k, da wir nur eine Vorwärtssubstitution durchgeführt haben. Denken Sie daran, dass diese Matrix einem bestimmten Zweck dient: Zeile k um c skalieren und das Ergebnis zur Zeile hinzufügen. Diese Operation lässt sich offensichtlich leicht rückgängig machen: Skalieren Sie Zeile k um c und subtrahieren Sie das Ergebnis von Zeile . Wir können dies formal überprüfen:

$$\begin{aligned} (In\times n + cee\ _k)(In\times n ÿ cee\ _k) &= In\times n + (ÿcee\ _k + cee\ k\ ) ÿ c\ ^2 ee\ ee\ kk \\ &= In\times n ÿ c\ ^2 e(e\ _k\ e)e\ _k \\ &= In\times n \text{ seite } e = ek \cdot e\ k\ , \text{ und } k = \end{aligned}$$

Die L-Matrix ist also das Produkt von Skalierungsmatrizen und Matrizen der Form Mÿ1 = In×n + cee k Sind unteres Dreieck, wenn > k. Skalierungsmatrizen sind diagonal und die Matrix M ist untere dreieckig. In Übung 2.1 werden Sie zeigen, dass das Produkt der unteren Dreiecksmatrizen die untere Dreiecksform hat, was wiederum zeigt, dass L bei Bedarf die untere Dreiecksform hat.

Wir haben gezeigt, dass Sie A = LU in das Produkt der unteren und oberen Dreiecksmatrizen faktorisieren können, wenn es möglich ist, die Gaußsche Eliminierung von A ohne Pivotierung durchzuführen . Vorwärts- und Rückwärtssubstitution benötigen jeweils $O(n^2)$ Zeit. Wenn diese Faktorisierung also im Voraus berechnet werden kann, kann die lineare Lösung schneller als die vollständige $O(n^3)$ -Lösung) Gaußsche Eliminierung. Du wirst vorbeischauen durchgeführt werden . Übung 2.2: Was passiert, wenn wir LU mit Pivotieren durchführen? Keine größeren Änderungen sind notwendig.

## 2.5.2 LU implementieren

Eine einfache Implementierung der Gaußschen Eliminierung zur Lösung von Ax = b ist recht einfach zu formulieren. Wie wir bereits besprochen haben, können wir insbesondere die erweiterte Matrix (A |b) bilden und Zeilenoperationen nacheinander auf diesen n × (n + 1) -Block anwenden, bis er wie (In×nA |b) aussieht. ÿ1 Dieser Prozess ist jedoch destruktiv, das heißt, am Ende kümmern wir uns nur um die letzte Spalte der erweiterten Matrix und haben keinen Beweis für unseren Lösungsweg aufbewahrt. Ein solches Verhalten ist für die LU-Faktorisierung eindeutig nicht akzeptabel.

Sehen wir uns an, was passiert, wenn wir zwei Eliminierungsmatrizen multiplizieren:

$$(\mathrm{In\times n} \ddot{y} \text{ cee}_{k})(\mathrm{In\times n} \ddot{y} \text{ cpepe k}) = \mathrm{In\times n} \ddot{y} \text{ cee}_{k} \ddot{y} \text{ cpepe k}$$

Wie bei unserer Konstruktion der Umkehrung einer Eliminierungsmatrix verschwindet das Produkt der beiden Ei -Terme, da die Standardbasis orthogonal ist. Diese Formel zeigt, dass nach der Skalierung des Pivots auf 1 das Produkt der Eliminierungsmatrizen, die zum Vorwärtsersatz für diesen Pivot verwendet werden, die Form hat:

$$M = \begin{matrix} 1\,0\,0\,0 \\ \ddot{y}\ 0\,1\,\bigcirc\ 0\,0\ \ddot{y} \\ 0\times 1\,0 \\ \ddot{y}\ddot{y}\ddot{y}\ 0\times 0\,1\ \ddot{y}\ddot{y}\ddot{y} \end{matrix},$$

wobei die Werte × die Werte sind, die zum Eliminieren des Rests der Spalte verwendet werden. Die Multiplikation von Matrizen dieser Form zeigt, dass die Elemente unterhalb der Diagonale von L lediglich von Koeffizienten stammen, die zur Durchführung der Substitution verwendet werden.

Wir können eine endgültige Entscheidung treffen, die Elemente entlang der Diagonale von L in der LU-Faktorisierung gleich 1 zu belassen. Diese Entscheidung ist legitim, da wir ein L immer mit einer Skalierungsmatrix S nachmultiplizieren können, um diese Elemente auf 1 zu bringen und schreiben Sie LU = (LS)(S ÿ1U) , ohne das Dreiecksmuster von L oder U zu beeinflussen. Mit dieser Entscheidung können wir unsere Speicherung von L und U in eine einzelne n × n-Matrix komprimieren, deren oberes Dreieck U ist und der gleich L unter der Diagonale ist; die fehlenden Diagonalelemente von L sind alle 1.

Wir sind nun bereit, Pseudocode für die einfachste LU-Faktorisierungsstrategie zu schreiben, bei der wir Zeilen oder Spalten nicht permutieren, um Pivots zu erhalten:

```
// Nimmt als Eingabe eine n-by-n-Matrix A [i, j]
// Bearbeitet A an Ort und Stelle, um die oben beschriebene kompakte LU-Faktorisierung zu erhalten

für Pivot von 1 nach n {
        PivotValue = A [Pivot, Pivot]; // Schlechte Annahme, dass dies ungleich Null ist!
```

```
        for EliminationRow from ( Pivot +1) to n { // Werte unterhalb des Pivots eliminieren
                // Wie stark die Pivot-Zeile skaliert werden soll, um den Wert in der aktuellen Zeile zu eliminieren
                // Reihe ; Beachten Sie, dass wir auch durch Pivot teilen, um Pivot auf 1 zu skalieren
                // Pivot-Wert
                Scale = A [EliminationRow, Pivot]/ PivotValue;

                // Da wir die Pivot-Zeile von der aktuellen Zeile multiplizieren / subtrahieren / skalieren
                // während der Gaußschen Eliminierung // , wir / addieren / es in der umgekehrten Operation
                in L gespeichert
                A [EliminationRow, Pivot] = Scale;

                // Führen Sie nun wie bei der Gaußschen Eliminierung die Zeilenoperation für den Rest durch
                // der Zeile: daraus wird U
                für EliminationCol von ( Pivot +1) bis n
                        Eine [ EliminationRow , EliminationCol] -= A [Pivot, EliminationCol]* Skala;
        }
}
```

## 2.6 Probleme

**Problem 2.1.** Das Produkt der unteren dreieckigen Dinge ist das untere Dreieck; Das Produkt der Pivotmatrizen sieht richtig aus

**Problem 2.2.** Implementieren Sie LU mit Pivotierung

**Problem 2.3.** Nicht quadratisches LU

# Kapitel 3

# Lineares Entwerfen und Analysieren Systeme

Da wir nun über einige Methoden zur Lösung linearer Gleichungssysteme verfügen, können wir diese zur Lösung einer Vielzahl von Problemen verwenden. In diesem Kapitel werden wir einige solcher Anwendungen und begleitende Analysetechniken untersuchen, um die Arten von Lösungen zu charakterisieren, die wir erwarten können.

## 3.1 Lösung quadratischer Systeme

Zu Beginn des vorherigen Kapitels haben wir mehrere Annahmen über die Arten linearer Systeme gemacht, die wir lösen wollten. Obwohl diese Einschränkung nicht trivial war, können tatsächlich viele, wenn nicht die meisten Anwendungen linearer Löser auf der Grundlage quadratischer, invertierbarer linearer Systeme dargestellt werden. Im Folgenden untersuchen wir einige gegensätzliche Anwendungen.

### 3.1.1 Regression

Wir beginnen mit einer einfachen Anwendung in der Datenanalyse, die als Regression bezeichnet wird. Angenommen, wir führen ein wissenschaftliches Experiment durch und möchten die Struktur unserer experimentellen Ergebnisse verstehen. Eine Möglichkeit, eine solche Beziehung zu modellieren, könnte darin bestehen, die unabhängigen Variablen des Experiments in einen Vektor x ÿ Rn zu schreiben und die abhängige Variable als Funktion f(x) : Rn ÿ R zu betrachten. Unser Ziel ist es, die Ausgabe von f vorherzusagen ohne das komplette Experiment durchzuführen.

**Beispiel 3.1** (Biologisches Experiment). Angenommen, wir möchten die Wirkung von Dünger, Sonnenlicht und Wasser auf das Pflanzenwachstum messen. Wir könnten eine Reihe von Experimenten durchführen, indem wir unterschiedliche Mengen an Dünger (in cm3 ), Sonnenlicht (in Watt) und Wasser (in ml) verwenden und nach ein paar Tagen die Höhe der Pflanze messen. Wir könnten unsere Beobachtungen als Funktion f : R3 ÿ **R** modellieren , indem wir die drei Parameter, die wir testen möchten, berücksichtigen und die Höhe der Pflanze ausgeben.

Bei der parametrischen Regression machen wir eine vereinfachende Annahme über die Struktur von f . Zum Beispiel, Nehmen wir an, dass f linear ist:

$$f(x) = a1x1 + a2x2 + \cdots + anxn.$$

Dann wird unser Ziel konkreter: die Koeffizienten ak zu schätzen .

Angenommen, wir führen eine Reihe von Experimenten durch, die die $x^{(k)}$ $y^{(k)}$- $y \approx f(x^{(k)})$. Durch den Anschluss an unsere Form für f zeigen, erhalten wir eine Reihe von Aussagen:

$$\begin{aligned} y^{(1)} &= f(x^{(1)}) = a_1 x_1^{(1)} + a_2 x_2^{(1)} + \cdots + a_N x_N^{(1)} \\ y^{(2)} &= f(x^{(2)}) = a_1 x_1^{(2)} + a_2 x_2^{(2)} + \cdots + a_N x_N^{(2)} \\ &\vdots \end{aligned}$$

Beachten Sie, dass im Gegensatz zu unserer Notation $Ax = b$ die Unbekannten hier die $a_k$ -Variablen und nicht die x-Variablen sind. Wenn wir n Beobachtungen machen, können wir schreiben:

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_N^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_N^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_N^{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Mit anderen Worten: Wenn wir n Versuche unseres Experiments durchführen und diese in die Spalten einer Matrix $X \in \mathbb{R}^{n \times n}$ schreiben und die abhängigen Variablen in einen Vektor $y \in \mathbb{R}^n$ schreiben, können die Koeffizienten a durch Lösen von $X a = y$ wiederhergestellt werden .

Tatsächlich können wir unseren Ansatz auf andere interessantere nichtlineare Formen für die Funktion f verallgemeinern. Wichtig ist hier, dass f eine lineare Kombination von Funktionen ist. Nehmen wir insbesondere an, dass f(x) die folgende Form annimmt:

$$f(x) = a_1 f_1(x) + a_2 f_2(x) + \cdots + a_m f_m(x),$$

wobei $f_k : \mathbb{R}^n \to \mathbf{R}$ und wir die Parameter $a_k$ schätzen wollen . Dann können wir durch eine parallele Ableitung gegebenen m Beobachtungen der Form $x^{(k)}$ $y^{(k)}$ bei die Parameter durch Lösen von: $y \approx$ finden

$$\begin{bmatrix} f_1(x^{(1)}) & f_2(x^{(1)}) & \cdots & f_m(x^{(1)}) \\ f_1(x^{(2)}) & f_2(x^{(2)}) & \cdots & f_m(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ f_1(x^{(m)}) & f_2(x^{(m)}) & \cdots & f_m(x^{(m)}) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \approx \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ J^{(M)} \end{bmatrix}$$

Das heißt, selbst wenn die f nichtlinear sind, können wir die Gewichte $a_k$ mit rein linearen Techniken lernen.

**Beispiel 3.2** (Lineare Regression). Das System $Xa = y$ kann aus der allgemeinen Formulierung wiederhergestellt werden, indem $f_k(x) = x_k$ angenommen wird .

**Beispiel 3.3** (Polynomielle Regression). Angenommen, wir beobachten eine Funktion einer einzelnen Variablen f(x) und möchten sie als Polynom n-ten Grades schreiben

$$f(x) \approx a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N.$$

Gegeben sind n Paare $x_k \approx y^{(k)}$ , Wir können die Parameter über das System ermitteln

$$\begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \cdots & (x^{(1)})^n \\ 1 & x^{(2)} & (x^{(2)})^2 & \cdots & (x^{(2)})^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & \cdots & (x^{(n)})^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \approx \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}.$$

Mit anderen Worten, wir nehmen $f_k(x) = x^k$ in unserer obigen allgemeinen Form. Übrigens ist die Matrix auf der linken Seite dieser Beziehung als Vandermonde-Matrix bekannt, die viele besondere Eigenschaften aufweist, die für ihre Struktur spezifisch sind.

**Beispiel 3.4** (Oszillation). Angenommen, wir möchten a und $\phi$ für eine Funktion $f(x) = a\cos(x + \phi)$ finden . Erinnern Sie sich an die Trigonometrie, dass wir $\cos(x + \phi) = \cos x \cos \phi \mp \sin x \sin \phi$ schreiben können . Wenn wir also zwei Beispielpunkte gegeben haben, können wir die obige Technik verwenden, um $f(x) = a_1 \cos x + a_2 \sin x$ zu finden, und unter Anwendung dieser Identität können wir schreiben

$$a = \sqrt{a_1^2 + a_2^2}$$

$$\phi = \arctan\frac{a_2}{a_1}$$

Diese Konstruktion kann erweitert werden, um $f(x) = \sum_k a_k \cos(x + \phi_k)$ zu finden, was eine Möglichkeit bietet, die diskrete Fourier-Transformation von f zu motivieren.

### 3.1.2 Kleinste Quadrate

Die Techniken in §3.1.1 bieten wertvolle Methoden zum Finden eines stetigen f, das genau zu einer Menge von Datenpaaren $x_k \approx y_k$ passt . Dieser Ansatz hat zwei damit verbundene Nachteile:

- Bei der Messung der Werte $x_k$ und $y_k$ kann es zu Fehlern kommen . In diesem Fall kann eine ungefähre Beziehung $f(x_k) \approx y_k$ akzeptabel oder sogar einer exakten Beziehung $f(x_k) = y_k$ vorzuziehen sein .
- Beachten Sie, dass wir genau m Beobachtungen$x_k \approx y_k$ benötigen, wenn es m Funktionen $f_k$ total gäbe . Zusätzliche Beobachtungen müssten verworfen werden, oder wir müssten die Form von f ändern.

Beide Probleme hängen mit dem größeren Problem der Überanpassung zusammen: Die Anpassung einer Funktion mit n Freiheitsgraden an n Datenpunkte lässt keinen Raum für Messfehler.

Nehmen wir allgemeiner an, wir möchten das lineare System $Ax = b$ nach x lösen . Wenn wir Zeile k von A mit $a_{sr\,k}$ bezeichnen , dann sieht unser System aus

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \approx \begin{bmatrix} a_{r\,1} \\ a_{r\,2} \\ \vdots \\ a_{r\,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \approx \begin{bmatrix} r_1 \cdot x \\ r_2 \cdot x \\ \vdots \\ r_m \cdot x \end{bmatrix}$$

nach Definition der Matrixmultiplikation.

Somit entspricht jede Zeile des Systems einer Beobachtung der Form$r_k \cdot x = b_k$ . Das heißt, eine weitere Möglichkeit, das lineare System $Ax = b$ zu interpretieren, ist als n Aussagen der Form „Das Skalarprodukt von x mit$r_k$ ist $b_k$ .“

Unter diesem Gesichtspunkt kodiert ein hohes System $Ax = b$ mit $A \in \mathbb{R}^{m\times n}$ und $m > n$ einfach mehr als n dieser Skalarproduktbeobachtungen. Wenn wir jedoch mehr als n Beobachtungen machen, können diese inkompatibel sein; Wie in Abschnitt 2.1 erläutert, werden große Systeme wahrscheinlich keine Lösung zulassen. In unserem oben erläuterten „experimentellen“ Aufbau könnte diese Situation auf Fehler bei der Messung der Paare $x_k \approx y_k$ zurückzuführen sein .

Wenn wir $Ax = b$ nicht exakt lösen können , könnten wir das Problem etwas lockern, um $Ax \approx b$ anzunähern. Insbesondere können wir verlangen, dass das Residuum $b - Ax$ so klein wie möglich ist, indem wir es minimieren

Norm b ÿ Ax. Beachten Sie, dass diese Norm bei Null minimiert wird, wenn es eine exakte Lösung für das lineare System gibt, da in diesem Fall b ÿ Ax = b ÿ b = 0 gilt. Die Minimierung von b ÿ Ax ist dasselbe wie die Minimierung von b ÿ Ax $^2$, was wir in Beispiel 0.16 erweitert haben zu:

$$b ÿ Ax \quad ^2 = x A Ax ÿ 2b Ax + b \quad ^2.1$$

Der Gradient dieses Ausdrucks in Bezug auf x muss im Minimum Null sein, was das folgende System ergibt:

$$0 = 2A Ax ÿ 2A b$$

Oder äquivalent: A Ax = A b.

Diese berühmte Beziehung verdient einen Satz:

**Satz 3.1** (Normalgleichungen). Minima des Residuums b ÿ Ax für A ÿ Rm×n (ohne Einschränkung auf m oder n) erfüllen AAx = A b.

Wenn mindestens n Zeilen von A linear unabhängig sind, dann ist die Matrix AA ÿ Rn×n invertierbar. In diesem Fall tritt das minimale Residuum (eindeutig) bei (A A) ÿ1A b auf, oder äquivalent dazu ist die Lösung des Problems der kleinsten Quadrate genauso einfach wie die Lösung des quadratischen linearen Systems A Ax = A b aus Satz 3.1. Daher haben wir unseren Satz an Lösungsstrategien auf A ÿ Rm×n mit m ÿ n erweitert, indem wir nur Techniken für quadratische Matrizen angewendet haben.

Der unterbestimmte Fall m < n ist wesentlich schwieriger zu handhaben. Insbesondere verlieren wir die Möglichkeit einer eindeutigen Lösung für Ax = b. In diesem Fall müssen wir eine zusätzliche Annahme für x treffen, um eine eindeutige Lösung zu erhalten, z. B. dass es eine kleine Norm hat oder viele Nullstellen enthält. Jede dieser Regularisierungsannahmen führt zu einer anderen Lösungsstrategie; Wir werden einige davon in den Übungen zu diesem Kapitel untersuchen.

## 3.1.3 Zusätzliche Beispiele

Eine wichtige Fähigkeit besteht darin, lineare Systeme „in freier Wildbahn“ identifizieren zu können. Hier zählen wir noch schnell ein paar weitere Beispiele auf.

#### Ausrichtung

Angenommen, wir machen zwei Fotos derselben Szene aus verschiedenen Positionen. Eine häufige Aufgabe bei Computer Vision und Grafik besteht darin, sie zusammenzufügen. Dazu kann der Benutzer (oder ein automatisches System) eine Anzahl von Punkten xk ,yk ÿ R2 markieren , sodass xk in Bild eins yk in Bild zwei entspricht . Natürlich wurden beim Abgleich dieser Punkte wahrscheinlich Fehler gemacht, daher möchten wir eine stabile Transformation zwischen den beiden Bildern finden, indem wir die Anzahl der erforderlichen Paare (x, y) überabtasten.

Vorausgesetzt, unsere Kamera verfügt über ein Standardobjektiv, sind die Kameraprojektionen linear, also ein vernünftiger Wert Die Annahme ist, dass es ein A ÿ R2×2 und einen Translationsvektorb ÿ R2 gibt , so dass

$$yk ÿ Axk + b.$$

[1] Es kann sinnvoll sein, an dieser Stelle noch einmal auf die Vorbemerkungen in Kapitel 0 zurückzukommen.

Unsere unbekannten Variablen sind hier A undb statt xk und yk .

In diesem Fall können wir die Transformation finden, indem wir Folgendes lösen:

$$\min_{A,b} \ddot{y}_{k=1}^{P} (Axk + b) \ddot{y} yk \quad ^2.$$

Dieser Ausdruck ist wiederum eine Summe quadrierter linearer Ausdrücke in unseren Unbekannten A und b und kann durch eine ähnliche Ableitung wie bei unserer Diskussion des Problems der kleinsten Quadrate linear gelöst werden.

**Dekonvolution**

Oftmals machen wir aus Versehen Fotos, die etwas unscharf sind. Während ein Foto, das völlig unscharf ist, ein verlorener Fehler sein kann, können wir bei örtlicher oder kleiner Unschärfe möglicherweise mithilfe von Computertechniken ein schärferes Bild wiederherstellen. Eine einfache Strategie ist die Entfaltung, die unten erläutert wird.

Wir können uns ein Foto als einen Punkt in Rp , wobei p die Anzahl der Pixel ist; natürlich, wenn die vorstellen. Wenn ein Foto in Farbe ist, benötigen wir möglicherweise drei Werte (RGB) pro Pixel, was zu einer ähnlichen Technik in R3p führt . Unabhängig davon sind viele einfache Bildunschärfen linear, z. B. Gaußsche Faltung oder Operationen zur Mittelung von Pixeln mit ihren Nachbarn auf dem Bild. In der Bildverarbeitung haben diese linearen Operationen oft andere spezielle Eigenschaften wie Verschiebungsinvarianz, aber für unsere Zwecke können wir uns Unschärfe als einen linearen Operator x ÿ G ÿ x vorstellen.

Angenommen, wir machen ein verschwommenes Foto x0 ÿ Rp . Dann könnten wir versuchen, das scharfe Bild wiederherzustellen x ÿ Rp durch Lösen des Problems der kleinsten Quadrate

$$\min_{\mathbf{x \ddot{y} Rp}} \quad x0 \ddot{y} G \ddot{y} x \quad ^2.$$

Das heißt, wir verlangen, dass Sie das beobachtete Foto x0 erhalten, wenn Sie x mit G verwischen. Natürlich können viele scharfe Bilder unter G das gleiche unscharfe Ergebnis liefern, daher fügen wir der obigen Minimierung oft zusätzliche Terme hinzu und verlangen, dass x0 nicht stark variiert.

## 3.2 Besondere Eigenschaften linearer Systeme

Unsere Diskussion der Gaußschen Eliminierung und der LU-Faktorisierung führte zu einer völlig generischen Methode zur Lösung linearer Gleichungssysteme. Obwohl diese Strategie immer funktioniert, können wir manchmal Geschwindigkeit oder numerische Vorteile erzielen, indem wir das spezielle System untersuchen, das wir lösen. Hier diskutieren wir einige gängige Beispiele, bei denen das Wissen über das lineare System Lösungsstrategien vereinfachen kann.

### 3.2.1 Positiv-definite Matrizen und die Cholesky-Faktorisierung

Wie in Satz 3.1 gezeigt, liefert die Lösung eines Problems der kleinsten Quadrate Ax ÿ b eine Lösung x, die das quadratische lineare System (A A)x = A b erfüllt. Unabhängig von A weist die Matrix AA einige besondere Eigenschaften auf, die dieses System zu etwas Besonderem machen.

Erstens ist es leicht zu erkennen, dass AA symmetrisch ist, da

$$(A A) = A (A ) = A A.$$

Hier haben wir einfach die Identitäten (AB) = BA und (A ) = A verwendet. Wir können diese Symmetrie indexweise ausdrücken, indem wir (A A)ij = (A A)ji für alle Indizes i, j schreiben. Diese Eigenschaft impliziert, dass es ausreicht, nur die Werte von AA auf oder über der Diagonale zu speichern, da die restlichen Elemente durch Symmetrie erhalten werden können.

Darüber hinaus ist AA eine positive semidefinite Matrix, wie unten definiert:

**Definition 3.1** (Positiv (Semi-)Definit). Eine Matrix B ÿ Rn×n ist positiv semidefinit, wenn für alle x ÿ Rn x Bx ÿ 0. B ، ist positiv definit, wenn x Bx > 0, wenn x =0.

Es ist leicht zu zeigen, dass AA positiv semidefinit ist, denn:

$$x\,A\,Ax = (Ax)\,(Ax) = (Ax)\cdot(Ax) = \|Ax\|_2^2 \ \text{ÿ}\ 0.$$

Wenn die Spalten von A tatsächlich linear unabhängig sind, dann ist AA positiv definit.

Nehmen wir allgemeiner an, wir möchten ein symmetrisches positiv definites System Cx = d lösen . Wie wir bereits untersucht haben, könnten wir die Matrix C LU-faktorisieren, aber tatsächlich können wir es etwas besser machen. Wir schreiben C ÿ Rn×n als Blockmatrix:

$$C = \begin{matrix} c11 & v \\ v & \tilde{C} \end{matrix}$$

wobei v ÿ Rnÿ1 und C˜ ÿ R(nÿ1)×(nÿ1) . Dank der besonderen Struktur von C können wir folgende Beobachtung machen:

$$Ce1 = \begin{matrix} 1 & 0 & \cdots & 0 \end{matrix} \begin{matrix} c11 & v \\ v & \tilde{C} \end{matrix} \begin{matrix} 1 \\ 0 \\ \vdots \\ 0 \end{matrix}$$

$$= \begin{matrix} 1 & 0 & \cdots & 0 \end{matrix} \begin{matrix} c11 \\ v \end{matrix}$$

$$= c11$$

$$> 0, \text{ da C positiv definit ist und } e1 \neq 0.$$

Dies zeigt, dass wir – abgesehen von numerischen Problemen – keine Pivotierung verwenden müssen, um sicherzustellen, dass c11 = 0 für den ersten Schritt der Gaußschen Eliminierung.

Wenn wir mit der Gaußschen Eliminierung fortfahren, können wir eine Vorwärtssubstitutionsmatrix E anwenden, die hat im Allgemeinen die Form

$$E = \begin{matrix} 1/\sqrt{c11} & 0 \\ r & I_{(n-1)\times(n-1)} \end{matrix}.$$

Hier enthält der Vektor r ÿ Rnÿ1 die Vielfachen von Zeile 1, um den Rest der ersten Spalte von C aufzuheben. Aus Gründen, die gleich klar werden, skalieren wir Zeile 1 auch um 1/ ÿ c11 !

Nach der Vorwärtssubstitution kennen wir das Produkt konstruktionsbedingt

$$EC = \begin{matrix} \sqrt{c11} & v/\sqrt{c11} \\ 0 & D \end{matrix}$$

für ein D ÿ R(nÿ1)×(nÿ1) .

Hier weichen wir von der Gaußschen Eliminierung ab: Anstatt mit der zweiten Zeile fortzufahren, können wir mit E nachmultiplizieren, um ein Produkt ECE zu erhalten:

$$\begin{aligned} ECE &= (EC)E \\ &= \begin{bmatrix} c11 & v / \\ 0 & D \end{bmatrix} \begin{bmatrix} 1/c11 & R \\ 0 & I_{(n-1)\times(n-1)} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & D^{\sim} \end{bmatrix} \end{aligned}$$

Das heißt, wir haben die erste Zeile und die erste Spalte von C eliminiert! Darüber hinaus lässt sich leicht überprüfen, dass die Matrix D˜ auch positiv definit ist.

Wir können diesen Vorgang wiederholen, um alle Zeilen und Spalten von C symmetrisch zu entfernen. Beachten dass wir sowohl Symmetrie als auch positive Bestimmtheit verwendet haben, um die Faktorisierung abzuleiten, da

- Die Symmetrie ermöglichte es uns, auf beiden Seiten dasselbe E anzuwenden, und
- Positive Bestimmtheit garantiert, dass $c11 > 0$, was impliziert, dass $\sqrt{c11}$ existiert.

Am Ende erhalten wir nun ähnlich wie bei der LU-Faktorisierung eine Faktorisierung C = LL für eine untere Dreiecksmatrix L. Dies ist als Cholesky-Faktorisierung von C bekannt. Wenn das Ziehen der Quadratwurzeln entlang der Diagonale zu numerischen Problemen führt, wird eine entsprechende LDL-Faktorisierung durchgeführt , wobei D eine Diagonalmatrix ist, vermeidet dieses Problem und lässt sich aus der obigen Diskussion einfach ableiten.

Die Cholesky-Faktorisierung ist aus mehreren Gründen wichtig. Besonders hervorzuheben ist, dass zum Speichern von L nur halb so viel Speicher benötigt wird wie für die LU-Faktorisierung von C oder sogar von C selbst, da die Elemente über der Diagonale Null sind und wie in LU das Lösen von Cx = d so einfach ist wie eine Vorwärts- und Rückwärtssubstitution. Weitere Eigenschaften der Faktorisierung erforschen Sie in den Übungen.

Letztendlich kann der Code für die Cholesky-Faktorisierung sehr prägnant sein. Um eine besonders com abzuleiten Nehmen wir an, wir wählen eine beliebige Zeile k aus und schreiben L in Blockform, um diese Zeile zu isolieren:

$$L = \begin{bmatrix} L_{11} & 0 & 0 \\ k & & \\ L_{31} & k & L_{33} \end{bmatrix}$$

Hier sind L11 und L33 beide untere dreieckige Quadratmatrizen. Dann ergibt die Durchführung eines Produkts:

$$\begin{aligned} LL &= \begin{bmatrix} L_{11} & 0 & 0 \\ k & & \\ L_{31} & k & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & 0 & 0 \\ k & L_{kk} & 0 \\ L_{31} & k & L_{33} \end{bmatrix} \\ &= \begin{bmatrix} \times & \times & \times \\ \times & L_{11}^2 + k \times & \times \\ \times & \times & \times \end{bmatrix} \end{aligned}$$

Werte des Produkts, die für unsere Ableitung nicht notwendig sind, lassen wir weg.

Am Ende wissen wir, dass wir C = LL schreiben können. Das mittlere Element des Produkts zeigt:

$$c_{kk} = L_{kk}^2 + k^2$$

wobei k ÿ Rkÿ1 die Elemente der k-ten Zeile von L links von der Diagonale enthält. Außerdem,
Das mittlere linke Element des Produkts wird angezeigt

$$L11k = ck$$

wobei ck die Elemente von C an derselben Position enthält ask . Da L11 die untere Dreiecksform hat, ist dies
System kann durch Vorwärtssubstitution gelöst werden!

Beachten Sie, dass unsere obige Diskussion einen Algorithmus zur Berechnung der Cholesky-Faktorisierung hervorbringt
von oben nach unten, da L11 bereits berechnet ist, wenn wir Zeile k erreichen. Wir bieten an
Pseudocode unten, angepasst von CITE:

```
// Nimmt als Eingabe eine n-by-n-Matrix A [i, j]
// Bearbeitet A an Ort und Stelle, um die Cholesky-Faktorisierung in seinem unteren Dreieck zu erhalten

für k von 1 bis n {
        // Zurück – ersetzen, um l_k zu finden
        für i von 1 bis k -1 { // Element i von l_k
                Summe = 0;
                für j von 1 bis i -1
                        sum += A [i , j ]* A [k , j ];
                Ein [k , i ] = ( A [k         , i ] - Summe )/ A [i   , ich ];
        }

        // Wende die Formel für l_kk an
        normSquared = 0
        für j von 1 bis i -1
                normSquared += A [k , j ]^2;
        Ein [k , k ] = sqrt ( A [k            , k ] - normSquared );
}
```

Wie bei der LU-Faktorisierung läuft dieser Algorithmus eindeutig in O(n). $^{3}$) Zeit.

## 3.2.2 Sparsität

Viele lineare Gleichungssysteme weisen von Natur aus Sparsity-Eigenschaften auf, was bedeutet, dass die meisten
Einträge von A im System Ax = b sind genau Null. Sparsity kann eine bestimmte Struktur in einem widerspiegeln
gegebenes Problem, einschließlich der folgenden Anwendungsfälle:

- In der Bildverarbeitung drücken viele Systeme zur Fotobearbeitung und zum Verständnis Beziehungen zwischen den Werten
  von Pixeln und denen ihrer Nachbarn im Bildraster aus. Ein Bild
  kann ein Punkt in Rp für p Pixel sein, aber wenn man Ax =b für ein neues Bild der Größe p auflöst, ist A ÿ Rp×p
  kann nur O(p) statt O(p) haben $^{2}$) ungleich Null, da jede Zeile nur ein einzelnes Pixel umfasst
  und seine Nachbarn oben/unten/links/rechts.

- Beim maschinellen Lernen verwendet ein grafisches Modell eine Graphenstruktur G ÿ (V, E) , um Wahrscheinlichkeitsverteilungen
  über mehrere Variablen auszudrücken. Jede Variable wird durch einen Knoten v ÿ V von dargestellt
  des Graphen und die Kante e ÿ E stellt eine probabilistische Abhängigkeit dar. Lineare Systeme entstehen in
  Dieser Kontext hat oft eine Zeile pro Scheitelpunkt v ÿ V mit Nicht-Nulls nur in beteiligten Spalten
  v und seine Nachbarn.

- In der Computergeometrie werden Formen häufig durch Ansammlungen verknüpfter Dreiecke ausgedrückt
  zu einem Netz zusammenfügen. Gleichungen für die Oberflächenglättung und andere Aufgaben verknüpfen wiederum
  Positionen und andere Werte an einem bestimmten Scheitelpunkt mit denen an seinen Nachbarn im Netz.

**Beispiel 3.5** (Harmonische Parametrisierung). Angenommen, wir möchten ein Bild verwenden, um ein Dreiecksnetz zu texturieren. Ein Netz kann als eine Ansammlung von Eckpunkten V ÿ R3 dargestellt werden, die durch Kanten E ÿ V × V miteinander verbunden sind, um Dreiecke zu bilden. Da sich die Eckpunkte der Geometrie im R3 befinden , müssen wir einen Weg finden, sie auf die Bildebene abzubilden, um die Textur als Bild zu speichern. Daher müssen wir jedem v ÿ V Texturkoordinaten t(v) ÿ R2 auf der Bildebene zuweisen . Eine Veranschaulichung finden Sie in Abbildung NUMMER.

Eine Strategie zur Erstellung dieser Karte beinhaltet eine einzige lineare Lösung. Angenommen, das Netz hat eine Scheibentopologie, das heißt, es kann auf das Innere eines Kreises in der Ebene abgebildet werden. Für jeden Scheitelpunkt vb an der Grenze des Netzes können wir die Position von vb angeben , indem wir ihn auf einem Kreis platzieren. Im Inneren können wir verlangen, dass die Position der Texturkarte der Durchschnitt ihrer benachbarten Positionen ist:

$$t(v) = \frac{1}{|n(v)|} \sum_{w \in n(v)} t(w)$$

Hier ist n(v) ÿ V die Menge der Nachbarn von v ÿ V auf dem Netz. Somit ist jedes v ÿ V mit einer linearen Gleichung verbunden, die es entweder auf den Rand fixiert oder verlangt, dass seine Position dem Durchschnitt seiner benachbarten Positionen entspricht. Dieses |V| × |V| Das Gleichungssystem führt zu einer stabilen Parametrisierungsstrategie, die als harmonische Parametrisierung bekannt ist. Die Matrix des Systems hat nur O(|V|) ungleich Nullen in Slots, die den Eckpunkten und ihren Nachbarn entsprechen.

Wenn A ÿ Rn×n so dünnbesetzt ist, dass es O(n) - statt O(n- Werte enthält, gibt es natürlich 2) ungleich Null keinen Grund, A als n × n-Matrix zu speichern. Stattdessen speichern Speichertechniken für spärliche Matrizen nur die O(n) ungleich Nullen in einer sinnvolleren Datenstruktur, z. B. einer Liste von Zeilen-/Spalten-/Werttripeln. Die Wahl einer Matrixdatenstruktur beinhaltet die Berücksichtigung der wahrscheinlichen Operationen, die auf der Matrix auftreten werden, möglicherweise einschließlich Multiplikation und Iteration über ungleich Nullen , oder Iteration über einzelne Zeilen oder Spalten.

Leider ist leicht zu erkennen, dass die LU-Faktorisierung eines dünn besetzten A möglicherweise nicht zu dünn besetzten L- und U-Matrizen führt. Dieser Strukturverlust schränkt die Anwendbarkeit dieser Methoden zur Lösung von Ax =b erheblich ein, wenn A groß, aber dünn besetzt ist. Glücklicherweise gibt es viele direkte Sparse-Löser, die LU an dünn besetzte Matrizen anpassen und eine LU-ähnliche Faktorisierung erzeugen können, ohne viel Füllung oder zusätzliche Nicht-Nullen zu induzieren; Die Diskussion dieser Techniken liegt außerhalb des Rahmens dieses Textes. Alternativ wurden iterative Techniken verwendet, um Näherungslösungen für lineare Systeme zu erhalten; Wir werden die Diskussion dieser Methoden auf zukünftige Kapitel verschieben.

Bestimmte Matrizen sind nicht nur dünn besetzt, sondern auch strukturiert. Zum Beispiel ein tridiagonales System von Lineare Gleichungen haben das folgende Muster von Werten ungleich Null:

$$\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

In den Übungen im Anschluss an dieses Kapitel werden Sie eine spezielle Version der Gaußschen Eliminierung für den Umgang mit dieser einfachen Bandstruktur herleiten.

In anderen Fällen sind Matrizen möglicherweise nicht dünn besetzt, lassen aber möglicherweise eine dünn besetzte Darstellung zu. Für die Prüfung Betrachten Sie beispielsweise die zyklische Matrix:

$$\begin{bmatrix} a & b & c & d & d \\ a & b & c & c & d \\ a & b & b & c & d \\ a & a & b & c & d \end{bmatrix}$$

Offensichtlich kann diese Matrix nur mit den Werten a, b, c, d gespeichert werden. Spezielle Techniken für diese und andere Matrizenklassen sind gut untersucht und oft effizienter als die generische Gaußsche Eliminierung.

## 3.3 Sensitivitätsanalyse

Wie wir gesehen haben, ist es wichtig, die Matrix eines linearen Systems zu untersuchen, um herauszufinden, ob sie spezielle Eigenschaften aufweist, die den Lösungsprozess vereinfachen können. Sparsität, positive Bestimmtheit, Symmetrie usw. können Hinweise auf den richtigen Löser für ein bestimmtes Problem geben.

Selbst wenn eine bestimmte Lösungsstrategie theoretisch funktionieren könnte, ist es jedoch ebenso wichtig zu verstehen, wie gut wir der Antwort eines bestimmten Lösers auf ein lineares System vertrauen können. Beispielsweise könnte es aufgrund von Rundungen und anderen diskreten Effekten der Fall sein, dass eine Implementierung der Gaußschen Eliminierung zur Lösung von Ax = b eine Lösung x0 ergibt , so dass 0 < Ax0 ÿb 1; mit anderen Worten, x0 löst das System nur näherungsweise.

Eine Möglichkeit, die Wahrscheinlichkeit dieser Näherungseffekte zu verstehen, ist die Sensitivitätsanalyse. In diesem Ansatz fragen wir, was mit x passieren könnte, wenn wir, anstatt Ax = b in Wirklichkeit zu lösen, ein gestörtes Gleichungssystem (A + ÿA)x = b + ÿb lösen. Es gibt zwei Möglichkeiten, die Schlussfolgerungen dieser Art von Analyse anzuzeigen:

1. Aufgrund von Rundungen und anderen Effekten machen wir wahrscheinlich Fehler bei der Darstellung von A und b. Diese Analyse zeigt dann die bestmögliche Genauigkeit, die wir für x angesichts der gemachten Fehler bei der Darstellung des Problems erwarten können.

2. Wenn unser Löser eine Näherung x0 für die Lösung von Ax =b generiert, ist es eine exakte Lösung für das System Ax0 = b0 , wenn wir b0 ÿ Ax0 definieren (stellen Sie sicher, dass Sie verstehen, warum dieser Satz keine Tautologie ist!). Wenn man versteht, wie sich Änderungen in x0 auf Änderungen inb0 auswirken, zeigt sich, wie empfindlich das System auf leicht falsche Antworten reagiert.

Beachten Sie, dass unsere Diskussion hier unseren Definitionen von Vorwärts- und Rückwärtsfehlern in den vorherigen Kapiteln ähnelt und tatsächlich durch diese motiviert ist.

### 3.3.1 Matrix- und Vektornormen

Bevor wir die Empfindlichkeit eines linearen Systems diskutieren können, müssen wir etwas sorgfältig definieren, was es bedeutet, dass eine Änderung ÿx „klein“ ist. Im Allgemeinen möchten wir die Länge oder Norm eines Vektors x messen. Wir sind bereits auf die Zweinorm eines Vektors gestoßen:

$$2 \; x2 \ddot{y} \; x \; 1 \qquad \sqrt{+ x \; \frac{2}{2} \; 2 + \cdots + x \; _N}$$

für x ÿ Rn . Diese Norm ist aufgrund ihrer Verbindung zur euklidischen Geometrie beliebt, aber sie ist keineswegs die einzige Norm auf dem Rn . Im Allgemeinen definieren wir eine Norm wie folgt:

**Definition 3.2** (Vektornorm). Eine Vektornorm ist eine Funktion · : Rn ÿ [0, ÿ), die Folgendes erfüllt Bedingungen:

• x = 0 genau dann, wenn x =0.

• cx = |c|x für alle Skalare c ÿ **R** und Vektoren x ÿ Rn .

• x +y ÿ x + y für alle x,y ÿ Rn .

Während wir die beiden Indizes · 2 verwenden , um die Zweinorm eines Vektors zu bezeichnen, verwenden wir, sofern wir nichts anderes vermerken, die Notation x, um die Zweinorm von x zu bezeichnen. Außer dieser Norm gibt es viele andere Beispiele: • Die p-

Norm xp für p ÿ 1, gegeben durch:

$$xp \ ÿ \ (|x1|\ p + |x2|\ p + \cdots + |xn|\ p\ )^{1/P}$$

Von besonderer Bedeutung ist die 1-Norm oder „Taxi“-Norm, gegeben durch

$$x1 \ ÿ \ ÿ_{k=1}^{N} |xk|.$$

Diese Norm erhält ihren Spitznamen, weil sie die Entfernung angibt, die ein Taxi zwischen zwei Punkten in einer Stadt zurücklegt, in der die Straßen nur in Nord-Süd- und Ost-West-Richtung verlaufen.

• Die ÿ-Norm xÿ gegeben durch:

$$xÿ \ ÿ \ \max(|x1|, |x2|, \cdots, |xn|).$$

In gewissem Sinne sind viele Normen für Rn gleich. Nehmen wir insbesondere an, wir sagen, zwei Normen seien äquivalent, wenn sie die folgende Eigenschaft erfüllen: **Definition 3.3**

(Äquivalente Normen). Zwei Normen · und · sind äquivalent, wenn es Konstanten clow und chigh gibt, so dass

$$clowx \ ÿ \ x \ ÿ \ chighx$$

für alle x ÿ Rn .

Diese Bedingung garantiert, dass bis auf einige konstante Faktoren alle Normen hinsichtlich der Vektoren übereinstimmen sind „klein“ und „groß“. Tatsächlich werden wir einen berühmten Satz aus der Analysis ohne Beweis angeben: **Satz 3.2**

(Äquivalenz von Normen auf Rn ). Alle Normen auf Rn sind äquivalent.

Dieses etwas überraschende Ergebnis impliziert, dass alle Vektornormen das gleiche grobe Verhalten aufweisen, die Wahl einer Norm zur Analyse oder Darstellung eines bestimmten Problems jedoch einen großen Unterschied machen kann. Beispielsweise geht die ÿ-Norm auf R3 davon aus, dass der Vektor (1000, 1000, 1000) dieselbe Norm wie (1000, 0, 0) hat , während die 2-Norm sicherlich von den zusätzlichen Werten ungleich Null beeinflusst wird.

Da wir nicht nur Vektoren, sondern auch Matrizen stören, müssen wir auch die Norm einer Matrix annehmen können. Natürlich ändert sich die grundlegende Definition einer Norm auf Rn×m nicht. Aus diesem Grund können wir jede Matrix in Rm×n zu einem Vektor in Rnm „abwickeln“, um jede Vektornorm in Matrizen zu übernehmen. Eine solche Norm ist die Frobenius-Norm, gegeben durch

$$AFro \ ÿ \ ÿ \ \sqrt{ÿ_{i,j} a_{ij}^2}$$

Solche Anpassungen von Vektornormen sind jedoch nicht immer sehr aussagekräftig. Insbesondere ist die Priorität für das Verständnis der Struktur einer Matrix A häufig ihre Wirkung auf Vektoren, d. h. die wahrscheinlichen Ergebnisse, wenn A mit einem beliebigen x multipliziert wird. Mit dieser Motivation können wir die durch eine Vektornorm induzierte Norm wie folgt definieren:

**Definition 3.4** (Induzierte Norm). Die durch eine Norm · auf Rn induzierte Norm auf Rm×n ist gegeben durch

$$A \ddot{y} \max\{Ax : x = 1\}.$$

Das heißt, die induzierte Norm ist die maximale Länge des Bildes eines Einheitsvektors multipliziert mit A.

Da Vektornormen cx = |c|x erfüllen, ist leicht zu erkennen, dass diese Definition dem Erfordernis entspricht

$$A \ddot{y} \max_{x \ddot{y} Rn\setminus\{0\}} \frac{Axt}{X}.$$

Unter diesem Gesichtspunkt ist die durch · induzierte Norm von A das größte erreichbare Verhältnis der Norm von Ax relativ zu der der Eingabe x.

Diese allgemeine Definition macht es etwas schwierig, die Norm A bei gegebener Matrix A und einer Auswahl von · zu berechnen . Glücklicherweise können die von vielen gängigen Vektornormen abgeleiteten Matrixnormen vereinfacht werden. Wir geben einige solcher Ausdrücke ohne Beweis an:

- Die induzierte Einnorm von A ist die maximale Summe einer beliebigen Spalte von A:

$$A1 = \max_{1 \ddot{y} j \ddot{y} n} \ddot{y}_{i=1}^{M} |aij|$$

- Die induzierte ÿ-Norm von A ist die maximale Summe einer beliebigen Zeile von A:

$$A\ddot{y} = \max_{1 \ddot{y} i \ddot{y} m} \ddot{y}_{j=1}^{N} |aij|$$

- Die induzierte Zweinorm oder Spektralnorm von A ÿ Rn×n ist die Quadratwurzel des größten Eigenwerts von A A. Das heißt,

$$A \ {}_{2}^{2} = \max\{\ddot{y} : \text{es existiert x } \ddot{y} \ \mathbf{R}^{N} \text{ mit A Ax} = \ddot{y}x\}$$

Zumindest die ersten beiden Normen sind relativ einfach zu berechnen; Auf die dritte werden wir bei der Diskussion von Eigenwertproblemen zurückkommen.

## 3.3.2 Konditionsnummern

Da wir nun über Werkzeuge zum Messen der Wirkung einer Matrix verfügen, können wir die Bedingungszahl eines linearen Systems definieren, indem wir unsere generische Definition von Bedingungszahlen aus Kapitel 1 anpassen. Wir folgen der in CITE dargestellten Entwicklung.

Angenommen, wir haben eine Störung ÿA einer Matrix A und eine entsprechende Störung ÿb. Für jede *ÿ* ÿ 0, unter Vernachlässigung technischer Invertierbarkeitsaspekte können wir einen Vektor *x(ÿ)* als Lösung schreiben

$$(A + \ddot{y} \cdot \ddot{y}A)x(\ddot{y}) = b + \ddot{y} \cdot \ddot{y}b.$$

Wenn wir beide Seiten nach *ÿ* differenzieren und die Produktregel anwenden, erhalten wir folgendes Ergebnis:

$$\ddot{y}A \cdot x + (A + \ddot{y} \cdot \ddot{y}A) = \ddot{y}b \ d\ddot{y} \frac{dx}{}$$

Insbesondere wenn $\ddot{y} = 0$ gilt

$$\ddot{y}A \cdot x(0) + A = \ddot{y}b - \frac{dx}{d\ddot{y}}\ddot{y}=0$$

oder gleichwertig,

$$\frac{dx}{d\ddot{y}}\ddot{y}=0 = A^{\ddot{y}1}(\ddot{y}b \ddot{y} \ddot{y}A \cdot x(0)).$$

Mit der Taylor-Entwicklung können wir schreiben

$$x(\ddot{y}) = x + \ddot{y}x(0) + O(\ddot{y}^2),$$

wobei wir x (0) = gestörtes $\frac{dx}{d\ddot{y}}\ddot{y}=0$ . Somit können wir den relativen Fehler erweitern, der durch die Lösung entsteht System definieren:

$$\frac{x(\ddot{y}) \ddot{y}x(0)}{x(0)} = \frac{\ddot{y}x(0) + O(\ddot{y}^2)}{x(0)} \text{ durch die Taylor-Entwicklung}$$

$$= \frac{\ddot{y}A^{\ddot{y}1}(\ddot{y}b \ddot{y} \ddot{y}A \cdot x(0)) + O(\ddot{y}\,x(0)^2)}{x(0)} \text{ durch die von uns berechnete Ableitung}$$

$$\ddot{y}\frac{|}{|\ddot{y}|}(A^{\ddot{y}1}\ddot{y}1\ddot{y}b + A \quad \ddot{y}A \cdot x(0))) + O(\ddot{y}^2)$$

x(0) durch die Dreiecksungleichung A + B ÿ A + B

$$\ddot{y}|\ddot{y}|A^{\ddot{y}1}\frac{\ddot{y}b}{x(0)} + \ddot{y}A + O(\ddot{y}^2) \text{ durch die Identität AB ÿ AB}$$

$$= |\ddot{y}|A^{\ddot{y}1}A\frac{\ddot{y}b}{Axt(0)} + \frac{\ddot{y}A}{A} + O(\ddot{y}^2)$$

$$\ddot{y}|\ddot{y}|A^{\ddot{y}1}A\frac{\ddot{y}b}{Axt(0)} + \frac{\ddot{y}A}{A} + O(\ddot{y}^2) \text{ da } Ax(0) \ddot{y} Ax(0)$$

$$= |\ddot{y}|A^{\ddot{y}1}A\frac{\ddot{y}b}{B} + \frac{\ddot{y}A}{A} + O(\ddot{y}^2) \text{ da } Ax(0) \ddot{y} Ax(0)$$

da per Definition Ax(0) =b ist

Hier haben wir einige Eigenschaften der Matrixnorm angewendet, die sich aus entsprechenden Eigenschaften für Vektoren ergeben. Beachten Sie, dass die Summe D ÿ ÿb/b + ÿA/A die relativen Störungen von A undb kodiert. Von diesem Standpunkt aus haben wir in erster Ordnung den relativen Fehler der Störung des Systems um $\ddot{y}$ unter Verwendung eines Faktors $\ddot{y}$ ÿ AA ÿ1 begrenzt:

$$\frac{x(\ddot{y}) \ddot{y}x(0)}{x(0)} \ddot{y} \ddot{y} \cdot D \cdot \ddot{y} + O(\ddot{y}^2)$$

Auf diese Weise begrenzt die Größe $\ddot{y}$ die Konditionierung linearer Systeme mit A. Aus diesem Grund treffen wir die folgende Definition:

**Definition 3.5** (Matrix-Bedingungsnummer). Die Bedingungszahl von A ÿ Rn×n für eine gegebene Matrixnorm · ist

$$\text{Bedingung } A \ddot{y} AA^{\ddot{y}1} .$$

Wenn A nicht invertierbar ist, nehmen wir die Bedingung A ÿ ÿ.

Es ist leicht zu erkennen, dass cond A ÿ 1 für alle A ist, dass die Skalierung von A keinen Einfluss auf seine Bedingungsnummer hat und dass die Bedingungsnummer der Identitätsmatrix 1 ist. Diese Eigenschaften stehen im Gegensatz zur Determinante, die nach oben und unten skaliert werden kann wenn Sie A erklimmen.

Wenn · durch eine Vektornorm induziert wird und A invertierbar ist, dann gilt

$$A^{\ddot{y}1} = \max_{x=0} \frac{A\ddot{y}1x}{X} \text{ per Definition}$$

$$= \max_{y=0} \frac{j}{Ja} \text{ durch Ersetzen von } y = A^{\ddot{y}1}x$$

$$= \min y=0 \frac{Ja}{j}^{\ddot{y}1} \text{ indem man den Kehrwert bildet}$$

In diesem Fall ist die Konditionsnummer von A gegeben durch:

$$\text{Leitung } A = \max_{x=0} \frac{Axt}{X} \quad \min_{y=0} \frac{Ja}{ja}^{\ddot{y}1}$$

Mit anderen Worten, cond A misst die maximale bis minimal mögliche Ausdehnung eines Vektors x unter A.

Allgemeiner gesagt besteht eine wünschenswerte Stabilitätseigenschaft eines Systems Ax =b darin, dass sich die Lösung x nicht wesentlich ändert, wenn eine Kugel gestört wird. Unsere Motivation für Bedingung A zeigt, dass bei einer kleinen Bedingungszahl die Änderung von x im Verhältnis zur Änderung von A-Kugel klein ist, wie in Abbildung NUMBER dargestellt. Andernfalls kann eine kleine Änderung der Parameter des linearen Systems zu großen Abweichungen in x führen; Diese Instabilität kann dazu führen, dass lineare Löser aufgrund von Rundungen und anderen Näherungen während des Lösungsprozesses große Fehler in x machen. kann genauso schwierig sein wie die Berechnung $A^{\ddot{y}1}$. Eine Möglichkeit zum Absenken der vollständigen inversen A-Grenze. Die Bedingung Nummer besteht darin, die Identität A ÿ1x ÿ A ÿ1x anzuwenden . können wir A schreiben $A^{\ddot{y}1}$ Somit gilt für jedes x = 0 ÿ A ÿ1x/x. Daher,

$$\text{Bedingung } A = AA^{\ddot{y}1} \ddot{y} \frac{AA\ddot{y}1x}{X} .$$

Wir können also die Bedingungszahl begrenzen, indem wir A ÿ1x für einige Vektoren x auflösen; Natürlich erzeugt die Notwendigkeit eines linearen Lösers, A ÿ1x zu finden , eine zirkuläre Abhängigkeit von der Bedingungszahl, um die Qualität der Schätzung zu bewerten! Wenn · durch die Zweinorm induziert wird, werden wir in zukünftigen Kapiteln zuverlässigere Schätzungen bereitstellen.

## 3.4 Probleme

Etwas wie:

• Kernel-Regression als Beispiel für §3.1.1

• Lösung der kleinsten Norm für Ax =b, die Matrix der kleinsten Quadrate ist ansonsten invertierbar

• Variationsversionen der Tikhonov-Regularisierung/„Ridge“-Regression (nicht der übliche Ansatz).
dazu, aber was auch immer); Vervollständigung der unbestimmten Geschichte auf diese Weise

1 • L Ansätze zur Regularisierung für den Kontrast – zeichnen Sie ein Bild davon, warum mit Sparsity zu rechnen ist, zeichnen Sie Einheitskreise, zeigen Sie, dass die p-Norm keine Norm für p < 1 ist, nehmen Sie den Grenzwert als p ÿ 0

• Mini-Riesz – Matrix für das innere Produkt ableiten, um zu zeigen, wie der Raum gedreht wird

• tridiagonale Lösung

• Eigenschaften der Konditionsnummer

# Kapitel 4

# Spaltenräume und QR

Eine Möglichkeit, das lineare Problem Ax = b für x zu interpretieren, besteht darin, dass wir b als lineare Kombination der Spalten von A mit den in x angegebenen Gewichten schreiben möchten. Diese Perspektive ändert sich nicht, wenn wir zulassen, dass A ÿ Rm×n nicht quadratisch ist, aber die Lösung existiert möglicherweise nicht oder ist abhängig von der Struktur des Spaltenraums nicht eindeutig. Aus diesen Gründen streben einige Techniken zum Faktorisieren von Matrizen und zum Analysieren linearer Systeme nach einfacheren Darstellungen des Spaltenraums, um die Lösbarkeit eindeutiger zu machen und eine explizitere Spannweite zu erreichen als zeilenbasierte Faktorisierungen wie LU.

## 4.1 Die Struktur der Normalgleichungen

Wie wir gezeigt haben, ist eine notwendige und hinreichende Bedingung dafür, dass x eine Lösung des Problems der kleinsten Quadrate Ax ÿ b ist, dass x die Normalgleichungen (A A)x = A b erfüllt. Dieser Satz legt nahe, dass die Lösung der kleinsten Quadrate eine recht einfache Erweiterung linearer Techniken ist. Methoden wie die Cholesky-Faktorisierung zeigen auch, dass die spezielle Struktur von Problemen der kleinsten Quadrate zum Vorteil des Lösers genutzt werden kann.

Es gibt jedoch ein großes Problem, das die Verwendung dieses Ansatzes einschränkt. Nehmen wir zunächst einmal an, dass A quadratisch ist; dann können wir schreiben:

$$\begin{aligned}\text{cond } AA &= \|A\|\,\|(A)^{-1}\| \\ &\leq \|A\|\,\|A^{-1}\|\,\|(A)^{-1}\| \\ &= \|A\|^2\,\|A^{-1}\|^2 \\ &= (\text{Bedingung } A)^2\end{aligned}$$

abhängig von der Wahl des ·

Das heißt, die Konditionszahl von AA ist ungefähr das **Quadrat** der Konditionszahl von A! Während also generische lineare Strategien bei AA funktionieren könnten, wenn das Problem der kleinsten Quadrate „einfach“ ist, erzeugen diese Strategien wahrscheinlich erhebliche Fehler, wenn die Spalten von A nahezu linear abhängig sind, da sie sich nicht direkt mit A befassen.

Intuitiv liegt ein Hauptgrund dafür, dass cond A groß sein kann, darin, dass die Spalten von A möglicherweise „ähnlich“ aussehen. Stellen Sie sich jede Spalte von A als einen Vektor in Rm vor. Wenn zwei Spalten ai und aj ai ÿ erfüllen , würde die aj -Restlänge der kleinsten Quadrate b ÿ Ax wahrscheinlich nicht viel darunter leiden, wenn wir Vielfache von ai durch Vielfache von aj ersetzen oder umgekehrt. Dieses breite Spektrum an nahezu – aber nicht vollständig – äquivalenten Lösungen führt zu einer schlechten Konditionierung. Der resultierende Vektor x ist zwar instabil, das Produkt jedoch

Axe bleibt aufgrund unserer Substitution nahezu unverändert. Wenn wir daher Ax ÿb einfach lösen wollen, um b in den Spaltenraum von A zu schreiben, würde jede Lösung ausreichen.

Um solche schlecht konditionierten Probleme zu lösen, werden wir eine alternative Strategie anwenden, die stärker auf den Spaltenraum von A achtet, anstatt Zeilenoperationen wie bei der Gaußschen Eliminierung einzusetzen. Auf diese Weise können wir solche Beinahe-Abhängigkeiten explizit identifizieren und numerisch stabil damit umgehen.

## 4.2 Orthogonalität

Wir haben festgestellt, wann das Problem der kleinsten Quadrate schwierig ist, können uns aber auch fragen, wann es am einfachsten ist. Wenn wir ein System auf den einfachen Fall reduzieren können, ohne dabei Konditionierungsprobleme hervorzurufen, haben wir einen stabileren Weg gefunden, die in §4.1 erläuterten Probleme zu umgehen.

Offensichtlich ist das am einfachsten zu lösende lineare System In×nx = b: Die Lösung ist einfach x ÿ b! Es ist unwahrscheinlich, dass wir dieses spezielle lineare System explizit in unseren Löser eingeben, es kann jedoch sein, dass wir dies versehentlich beim Lösen der Methode der kleinsten Quadrate tun. Insbesondere selbst wenn A = In×n – A muss tatsächlich keine quadratische Matrix sein – können wir unter besonders glücklichen Umständen feststellen, dass die normale Matrix AA AA = In×n erfüllt. Um Verwechslungen mit dem allgemeinen Fall zu vermeiden, verwenden wir zur Darstellung einer solchen Matrix den Buchstaben Q.

Das einfache Beten, dass QQ = In×n unwahrscheinlich ist, wird zu einer wünschenswerten Lösungsstrategie führen, aber wir können diesen Fall untersuchen, um zu sehen, wie er so günstig wird. Schreiben Sie die Spalten von Q als Vektoren q1, · · · ,qn ÿ Rm. Dann lässt sich leicht überprüfen, ob das Produkt QQ die folgende Struktur hat:

$$QQ = \begin{pmatrix} \ddot{y} & q_1 & \ddot{y} \\ \ddot{y} & q_2 & \ddot{y} \\ & \vdots & \\ \ddot{y} & q_n & \ddot{y} \end{pmatrix} \begin{pmatrix} \ddot{y} & | & | & & | & \ddot{y} \\ \ddot{y} & q_1 & q_2 & \cdots & q_n & \ddot{y} \\ \ddot{y} & | & | & & | & \ddot{y} \end{pmatrix} = \begin{pmatrix} \ddot{y} & q_1 \cdot q_1 & q_1 \cdot q_2 & \cdots & q_1 \cdot q_n & q_2 \cdot q_1 & \ddot{y} \\ & q_2 \cdot q_2 & \cdots & q_2 \cdot q_n & & & \\ & \vdots & & \vdots & \cdots & \vdots & \\ \ddot{y} & q_n \cdot q_1 & q_n \cdot q_2 & \cdots & q_n \cdot q_n & & \ddot{y} \end{pmatrix}$$

Setzt man den Ausdruck auf der rechten Seite gleich In×n, erhält man die folgende Beziehung:

$$q_i \cdot q_j = \begin{cases} 1 \text{ wenn } i = j \\ 0 \text{ wenn } i = j \end{cases}$$

Mit anderen Worten, die Spalten von Q haben eine Einheitslänge und sind orthogonal zueinander. Wir sagen, dass sie eine Orthonormalbasis für den Spaltenraum von Q bilden:

**Definition 4.1** (Orthonormal; orthogonale Matrix). Eine Menge von Vektoren {v1, · · · ,vk} ist orthonormal, wenn vi = 1 für alle i und vi ·vj = 0 für alle i = j. Eine quadratische Matrix, deren Spalten orthonormal sind, wird orthogonale Matrix genannt.

Wir haben unsere Diskussion mit der Frage motiviert, wann wir mit QQ = In×n rechnen können . Nun ist leicht zu erkennen, dass dies geschieht, wenn die Spalten von Q orthonormal sind. Wenn Q außerdem quadratisch und invertierbar ist mit QQ = In×n, dann erhalten wir durch einfaches Multiplizieren beider Seiten dieses Ausdrucks mit Qÿ1 Qÿ1 = Q. Somit ist die Lösung von Qx = b in diesem Fall so einfach wie das Multiplizieren beide Seiten durch das transponierte Q.

Orthonormalität hat auch eine starke geometrische Interpretation. Erinnern Sie sich an Kapitel 0, dass wir zwei orthogonale Vektoren a und b als senkrecht betrachten können. Also ein orthonormaler Satz von Vektoren

Ist einfach eine Menge senkrechter Vektoren mit Einheitslänge in Rn , die · Wenn Q orthogonal ist, dann ist seine Wirkung orthogonal die Länge der Vektoren nicht beeinflussen:

$$\|Qx\|^2 = x\,Q\,Qx = x\,I_{n\times n}x = x\cdot x = \|x\|^2$$

Ebenso kann Q den Winkel zwischen zwei Vektoren nicht beeinflussen, da:

$$(Qx)\cdot(Qy) = x\,Q\,Qy = x\,I_{n\times n}y = x\cdot y \quad \text{Wenn } Q$$

orthogonal ist, stellt Q eine Isometrie von Rn dar , das heißt, es behält Längen und Winkel‚bei. Mit anderen Worten: Es kann Vektoren drehen oder spiegeln, sie jedoch nicht skalieren oder scheren.
Von einem hohen Niveau aus ist die lineare Algebra orthogonaler Matrizen einfacher, da ihre Wirkung die Geometrie des zugrunde liegenden Raums in keiner nichttrivialen Weise beeinflusst.

## 4.2.1 Strategie für nicht-orthogonale Matrizen

Außer unter besonderen Umständen sind die meisten unserer Matrizen A bei der Lösung von Ax =b oder dem entsprechenden Problem der kleinsten Quadrate nicht orthogonal, sodass die Maschinerie von §4.2 nicht direkt anwendbar ist. Aus diesem Grund müssen wir einige zusätzliche Berechnungen durchführen, um den allgemeinen Fall mit dem orthogonalen zu verbinden.

Nehmen Sie eine Matrix $A \ddot{y} R^{m\times n}$, und bezeichnen seinen Spaltenraum als col A; Denken Sie daran, dass Spalte A die Spanne der Spalten von A darstellt. Nehmen wir nun an, dass eine Matrix $B \ddot{y} R^{n\times n}$ invertierbar ist. Wir können eine einfache Beobachtung über den Spaltenraum von AB relativ zu dem

von A machen: **Lemma 4.1** (Spaltenrauminvarianz). Für jedes $A \ddot{y} R^{m\times n}$ und das invertierbare $B \ddot{y}‚R^{n\times n}$

$$\text{Spalte } A = \text{Spalte } AB.$$

Nachweisen. Angenommen, b ÿ col A. Dann gibt es nach der Definition der Multiplikation mit A x mit Ax = b. Dann ist (AB) · (B ÿ1x) = Ax =b, sob ÿ col AB. Umgekehrt sei c ÿ col AB, also existiert y mit (AB)y = c. Dann ist A · (By) = c, was zeigt, dass c in Spalte A steht. □

Erinnern Sie sich an die „Eliminationsmatrix“-Beschreibung der Gaußschen Eliminierung: Wir begannen mit einer Matrix A und wendeten Zeilenoperationsmatrizen Ei an , so dass die Sequenz A, E1A, E2E1A, . . . dargestellt sequentiell einfachere lineare Systeme. Das obige Lemma schlägt eine alternative Strategie für Situationen vor, in denen uns der Spaltenraum wichtig ist: Wenden Sie Spaltenoperationen durch Nachmultiplikation auf A an, bis die Spalten orthonormal sind. Das heißt, wir erhalten ein Produkt Q = AE1E2 · · · Ek , so dass Q orthonormal ist. Solange die Ei invertierbar sind, zeigt das Lemma, dass col Q = col A. Das Invertieren dieser Operationen ergibt eine Faktorisierung A = QR für R = E. Wenn wir wie bei der LU- $\ddot{y}1\ \ddot{y}1\ddot{y}1\ E_k \cdots E_{k\ddot{y}1}\ddot{y}1$.

Faktorisierung R sorgfältig entwerfen, ist die Lösung der kleinsten Quadratprobleme Ax ÿb können vereinfacht werden. Insbesondere wenn A = QR ist, können wir die Lösung von A Ax = A b wie folgt schreiben:

$$\begin{aligned} x &= (A\,A)\ddot{y}1A\,b \\ &= (RQ\,QR)\ddot{y}1R\,Q\,b, \text{ da } A = QR \\ &= (R\,R)\ddot{y}1R\,Q\,b, \text{ da } Q \text{ orthogonal ist} \\ &= R^{\ddot{y}1}(R^{\ddot{y}1})\ddot{y}1R\,Q\,b \text{ da } (AB)^{\ddot{y}1} = B\ddot{y}1A\ddot{y}1 \\ &= R\ddot{y}1Q\,b \end{aligned}$$

Oder äquivalent: Rx = Q b

Wenn wir also R als Dreiecksmatrix entwerfen, ist die Lösung des linearen Systems Rx = Qb so einfach wie eine Rücksubstitution.

Unsere Aufgabe für den Rest des Kapitels besteht darin, Strategien für eine solche Faktorisierung zu entwerfen.

# 4.3 Gram-Schmidt-Orthogonalisierung

Unser erster Ansatz zum Finden von QR-Faktorisierungen ist am einfachsten zu beschreiben und zu implementieren, kann jedoch mit numerischen Problemen behaftet sein. Wir verwenden es hier als erste Strategie und werden es dann durch bessere Abläufe verbessern.

## 4.3.1 Projektionen

Angenommen, wir haben zwei Vektoren a undb. Dann könnten wir leicht fragen: „Welches Vielfache von a kommt am nächsten zu b?“ Mathematisch entspricht diese Aufgabe der Minimierung von ca ÿb über alle möglichen c $\overset{2}{\ddot{y}}$ R. Wenn wir uns a und b als n × 1-Matrizen und c als 1 × 1-Matrix vorstellen , dann ist dies nichts weiter als ein unkonventionelles Problem der kleinsten Quadrate · c ÿb. In diesem Fall lauten die Normalgleichungena · c = ab, oder

$$c = \frac{a \cdot b}{\text{ein} \cdot \text{ein}} = \frac{a \cdot b}{A^{2}}.$$

Wir bezeichnen diese Projektion vonb aufa als:

$$\text{proj}_{A} = ca = \frac{a \cdot bb}{a \cdot a} a = \frac{a \cdot b}{A^{\text{eine 2}}}$$

Offensichtlich proj $_{A}$ b ist parallel zu a. Was ist mit dem Rest b ÿ proj $_{A}$ B? Wir können ein einfaches machen Berechnung um herauszufinden:

$$\begin{aligned} a \cdot (b \ddot{y} \text{proj}_{A} b) &= a \cdot b \ddot{y} a \cdot \frac{a \cdot b}{A^{\text{eine 2}}} \\ &= a \cdot b \ddot{y} \frac{a \cdot b}{A^{2}} (a \cdot a) \\ &= a \cdot b \ddot{y} a \cdot b \\ &= 0 \end{aligned}$$

Somit haben wir b in eine Komponente parallel zu a und eine weitere Komponente orthogonal zerlegt zu einem.

Nehmen wir nun an, dass aˆ1, aˆ2, · · · , aˆk orthonormal sind; Wir werden Vektoren mit Einheitslänge mit Hüten überziehen. Dann können wir für jedes einzelne i Folgendes sehen:

$$\text{proj}_{\hat{a}i} b = (\hat{a}i \cdot b)\hat{a}i$$

Der Normterm kommt nicht vor, da per Definition aˆi = 1 ist. Wir könnten b auf die Spanne {aˆ1, · · · , aˆk} projizieren , indem wir die folgende Energie über c1, minimieren . . . , ck ÿ R:

$$\|c_1\hat{a}_1 + c_2\hat{a}_2 + \cdots + c_k\hat{a}_k - b\|^2 = \sum_{i=1}^{k}\sum_{j=1}^{k} c_i c_j(\hat{a}_i \cdot \hat{a}_j) - 2b \cdot \sum_{i=1}^{k} c_i\hat{a}_i + b \cdot b$$

durch Anwendung von $\|v\|^2 = v \cdot v$

$$= \sum_{i=1}^{k} c_i^2 - 2c_i\, b \cdot \hat{a}_i + \|b\|^2 \quad \text{da die } \hat{a}_i \text{ orthonormal sind}$$

Beachten Sie, dass der zweite Schritt hier nur aufgrund der Orthonormalität gültig ist. Zumindest ist die Ableitung nach ci für jedes ci null , was Folgendes ergibt:

$$c_i = \hat{a}_i \cdot b$$

Wir haben also gezeigt, dass, wenn aˆ1, · · · , aˆk orthonormal sind, die folgende Beziehung gilt:

$$\text{projspan}\{\hat{a}_1,\cdots,\hat{a}_k\} b = (\hat{a}_1 \cdot b)\hat{a}_1 + \cdots + (\hat{a}_k \cdot b)\hat{a}_k$$

Dies ist lediglich eine Erweiterung unserer Projektionsformel, und anhand eines ähnlichen Beweises ist dies leicht zu erkennen

$$\hat{a}_i \cdot (b - \text{projspan}\{\hat{a}_1,\cdots,\hat{a}_k\} b) = 0.$$

Das heißt, wir haben b in eine Komponente parallel zur Spanne der aˆi und ein dazu senkrechtes Residuum aufgeteilt.

## 4.3.2 Gram-Schmidt-Orthogonalisierung

Unsere obigen Beobachtungen führen zu einem einfachen Algorithmus zur Orthogonalisierung oder zum Finden einer orthogonalen Basis {aˆ1, · · · , aˆk} , deren Spanne dieselbe ist wie die einer Menge linear unabhängiger Eingabevektoren {v1, · · · ,vk}:

1. Einstellen

$$\hat{a}_1 - \frac{v_1}{\|v_1\|}.$$

Das heißt, wir nehmen an, dass aˆ1 ein Einheitsvektor parallel zu v1 ist.

2. Für i von 2 bis k,

(a) Berechnen Sie die Projektion

$$p_i - \text{projspan}\{\hat{a}_1,\cdots,\hat{a}_{i-1}\} v_i.$$

Per Definition sind aˆ1, · · · , aˆiÿ1 orthonormal, daher gilt unsere obige Formel.

(b) Definieren

$$\hat{a}_i - \frac{v_i - p_i}{\|v_i - p_i\|}.$$

Diese als „Gram-Schmidt-Orthogonalisierung“ bekannte Technik ist eine einfache Anwendung unserer obigen Diskussion. Der Schlüssel zum Beweis dieser Technik besteht darin, zu beachten, dass span {v1, · · · ,vi} = span {aˆ1, · · · , aˆi} für jedes i ÿ {1, · · · , k}. Schritt 1 macht dies eindeutig für i = 1 der Fall , und für i > 1 entfernt die Definition von aˆi in Schritt 2b einfach die Projektion auf die Vektoren, die wir bereits gesehen haben.

Wenn wir mit einer Matrix A beginnen, deren Spalten v1, · · · ,vk sind, können wir Gram Schmidt als eine Reihe von Spaltenoperationen für A implementieren. Die Division der Spalte i von A durch ihre Norm entspricht der Nachmultiplikation von A mit ak × k Diagonalmatrix. Ebenso ist das Subtrahieren der Projektion einer Spalte auf die orthonormalen Spalten links davon wie in Schritt 2 gleichbedeutend mit einer Nachmultiplikation mit einer oberen Dreiecksmatrix: Stellen Sie sicher, dass Sie verstehen, warum dies der Fall ist! Somit gilt unsere Diskussion in §4.2.1 und wir können Gram-Schmidt verwenden, um eine Faktorisierung A = QR zu erhalten.

Leider kann der Gram-Schmidt-Algorithmus aufgrund des Subtraktionsschritts zu schwerwiegenden numerischen Instabilitäten führen. Nehmen wir zum Beispiel an, wir stellen die Vektoren v1 = (1, 1) und v2 = (1 + ÿ, 1) als Eingabe für Gram-Schmidt für 0 < ÿ 1 bereit. Beachten Sie, dass eine offensichtliche Basis für span {v1,v2} { (1, 0),(0, 1)}. Wenn wir jedoch Gram-Schmidt anwenden, erhalten wir:

$$a\hat{}1 = \frac{v1}{\|v1\|} = \frac{1}{\sqrt{2}} \begin{matrix}1\\1\end{matrix}$$

$$p2 = \frac{2+\ddot{y}}{2} \begin{matrix}1\\1\end{matrix}$$

$$v2 - p2 = \begin{matrix}1+\ddot{y}\\1\end{matrix} - \frac{2+\ddot{y}}{2} \begin{matrix}1\\1\end{matrix}$$

$$= \frac{1}{2} \begin{matrix}\ddot{y}\\-\ddot{y}\end{matrix}$$

Beachten Sie, dass v2 ÿ p2 = ( ÿ 2/2) · ÿ, sodass die Berechnung von aˆ2 eine Division durch einen Skalar in der Größenordnung von ÿ erfordert . Die Division durch kleine Zahlen ist eine instabile numerische Operation, die wir vermeiden sollten.

## 4.4 Haushaltstransformationen

In §4.2.1 haben wir die Konstruktion der QR-Faktorisierung durch Postmultiplikation und Spaltenoperationen motiviert. Diese Konstruktion ist im Kontext der Analyse von Spaltenräumen sinnvoll, aber wie wir bei unserer Ableitung des Gram-Schmidt-Algorithmus gesehen haben, können die resultierenden numerischen Techniken instabil sein.

Anstatt mit A zu beginnen und nachträglich mit Spaltenoperationen zu multiplizieren, um Q = AE1 · · · Ek zu erhalten , Wir können jedoch unsere High-Level-Strategie vor der Gaußschen Eliminierung bewahren. Das heißt, wir können mit A beginnen und mit orthogonalen Matrizen Qi vormultiplizieren, um Qk · · · Q1A = R zu erhalten ; Diese Qs wirken wie Zeilenoperationen und eliminieren Elemente von A, bis das resultierende Produkt R die obere Dreiecksform hat. Dank der Orthogonalität der Qs können wir dann A = QR schreiben und so den QR-Faktor erhalten
isierung, da das Produkt orthogonaler Matrizen orthogonal ist.

Die Zeilenoperationsmatrizen, die wir bei der Gaußschen Eliminierung und LU verwendet haben, reichen für die QR-Faktorisierung nicht aus, da sie nicht orthogonal sind. Es wurde eine Reihe von Alternativen vorgeschlagen; Wir werden eine gemeinsame Strategie vorstellen, die 1958 von Alston Scott Householder eingeführt wurde.

Der Raum orthogonaler n × n-Matrizen ist sehr groß, daher müssen wir einen kleineren Raum von Qi finden , mit dem wir einfacher arbeiten können. Aus unseren geometrischen Diskussionen in §4.2 wissen wir, dass orthogonale Matrizen Winkel und Längen beibehalten müssen, sodass sie intuitiv nur Vektoren drehen und reflektieren können.

Glücklicherweise lassen sich die Überlegungen leicht in Form von Projektionen beschreiben, wie in Abbildung NUMMER dargestellt. Angenommen, wir haben einen Vektor b, den wir über einen Vektor v spiegeln möchten. Wir haben gezeigt, dass das Residuum $r_v b$ steht senkrecht auf v. Wie in Abbildung NUMMER die Differenz $\ddot{y} b \ddot{y}_v b \ddot{y} b$ reflektiertb über v.

proj 2proj ist. Wir können unsere Reflexionsformel wie folgt erweitern:

$$2\text{proj}_v \ddot{y} b = 2 \frac{v \cdot bb}{v \cdot v} v \ddot{y} b \text{ nach Definition der Projektion}$$

$$= 2v \cdot \frac{vb}{Vv} \ddot{y} b \text{ unter Verwendung der Matrixschreibweise}$$

$$= \frac{2vv}{Vv} \ddot{y} \text{ In×n } b$$

ÿ ÿHv b, wobei das Negativ eingeführt wird, um es an andere Behandlungen anzupassen

Daher können wir uns die Spiegelung von b über v so vorstellen, als würden wir einen linearen Operator ÿHv auf b anwenden! Natürlich ist Hv ohne das Negativ immer noch orthogonal, daher werden wir es von nun an verwenden.

Angenommen, wir machen den ersten Schritt der Vorwärtssubstitution während der Gaußschen Eliminierung. Dann möchten wir A mit einer Matrix vormultiplizieren, die die erste Spalte von A, die wir mit a bezeichnen werden, zu einem Vielfachen des ersten Identitätsvektors e1 führt. Mit anderen Worten, wir wollen für ein c ÿ R:

$$ce1 = Hva$$

$$= \text{In×n } \ddot{y} \frac{2vv}{Vv} A$$

$$= a \ddot{y} 2v \frac{va}{Vv}$$

Begriffe in Shows verschieben

$$v = (a \ddot{y} ce1) \cdot \frac{Vv}{2va}$$

Mit anderen Worten, v muss parallel zur Differenz a ÿ ce1 sein. Tatsächlich hat die Skalierung von v keinen Einfluss auf die Formel für Hv , sodass wir v =a ÿ ce1 wählen können. Dann müssen wir es haben, damit unsere Beziehung hält

$$1 = \frac{Vv}{2va}$$

$$= \frac{A^2 \ddot{y} 2ce1 \cdot a + c^2}{2(a \cdot a \ddot{y} ce1 \cdot a)}$$

$$\text{Oder } 0 = a^{2\,2\ddot{y}\,c} = \ddot{y} c = \pm a$$

Mit dieser Wahl von c haben wir gezeigt:

$$HvA = \begin{matrix} \ddot{y} & c \times \times \times & \\ & 0 \times \times \times & \\ & \vdots \quad \vdots \quad \vdots \quad \vdots & \\ \text{¯¯¯} & 0 \times \times \times & \text{¯¯¯} \end{matrix} \ddot{y}$$

Wir haben gerade einen Schritt erreicht, der der Vorwärtseliminierung nur mit orthogonalen Matrizen ähnelt!

Weitergehend haben wir in der Notation von CITE während des k-ten Schritts der Triangularisierung einen Vektor a, das wir in zwei Komponenten aufteilen können:

$$a = \begin{matrix} a1 \\ a2 \end{matrix}$$

Hier gilt a1 ÿ Rk und a2 ÿ Rmÿk . Wir möchten v so finden

$$Hva = \begin{matrix} ÿ \\ \\ \\ \\ \end{matrix} \begin{matrix} a1 \\ 0 \\ \vdots \\ 0 \end{matrix} \begin{matrix} ÿ \\ \\ \\ \\ \end{matrix}$$

Folgt man einer parallelen Ableitung zur obigen, lässt sich das leicht zeigen

$$v = \begin{matrix} 0 \\ \\ a2 \end{matrix} \quad ÿ \text{ cek}$$

führt genau diese Transformation durch, wenn c = ±a2; Normalerweise wählen wir das Vorzeichen von c, um eine Aufhebung zu vermeiden, indem wir dafür sorgen, dass es das Vorzeichen hat, das dem des k-ten Werts ina entgegengesetzt ist.

Der Algorithmus für Householder QR ist daher ziemlich einfach. Für jede Spalte von A berechnen wir v, indem wir die unteren Elemente der Spalte vernichten, und wenden Hv auf A an. Das Endergebnis ist eine obere Dreiecksmatrix R = Hvn · · · Hv1 A. Die orthogonale Matrix Q ist durch das Produkt H which gegeben kann implizit als Liste von Vektoren v gespeichert werden, die in das untere Dreieck wie oben gezeigt v1 vn passt .

## 4.5 Reduzierte QR-Faktorisierung

Wir schließen unsere Diskussion ab, indem wir zum allgemeinsten Fall Ax ÿ b zurückkehren, wenn A ÿ Rm×n nicht quadratisch ist. Beachten Sie, dass beide in diesem Kapitel besprochenen Algorithmen nichtquadratische Matrizen A in Produkte QR faktorisieren können, die Ausgabe jedoch etwas anders ist:

- Bei der Anwendung von Gram-Schmidt führen wir Spaltenoperationen an A durch, um Q durch Orthogonalisierung zu erhalten. Aus diesem Grund ist die Dimension von A die von Q, was Q ÿ Rm×n und R·ÿ Rn×n ergibt
- Bei Verwendung von Householder-Reflexionen erhalten wir Q als Produkt einer Zahl von m × m Reflexionsmatrizen, sodass R ÿ Rm×n bleibt·

Angenommen, wir befinden uns im typischen Fall der kleinsten Quadrate, für den m n gilt. Aufgrund der numerischen Stabilität bevorzugen wir immer noch die Householder-Methode, aber jetzt ist die m × m-Matrix Q möglicherweise zu groß zum Speichern! Zum Glück wissen wir, dass R die obere Dreiecksform hat. Betrachten Sie beispielsweise die Struktur einer 5 × 3-Matrix R:

$$R = \begin{matrix} ÿ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \begin{matrix} ÿ \\ \\ \\ \\ \\ \end{matrix}$$

Es ist leicht zu erkennen, dass alles unterhalb des oberen n × n-Quadrats von R Null sein muss, was eine Vereinfachung ergibt Kation:

$$A = QR = Q1\ Q2 \begin{matrix} R1 \\ 0 \end{matrix} = Q1R1$$

Hier enthält Q1 ÿ Rm×n und R1 ÿ Rn×n noch das obere Dreieck von R. Dies wird als „reduziertes“ Dreieck bezeichnet. QR-Faktorisierung von A, da die Spalten von Q1 eine Basis für den Spaltenraum von A und nicht für den gesamten Rm enthalten; es nimmt viel weniger Platz ein. Beachten Sie, dass die Diskussion in §4.2.1 weiterhin gilt, sodass die reduzierte QR-Faktorisierung auf ähnliche Weise für kleinste Quadrate verwendet werden kann.

## 4.6 Probleme

- Tridiagonalisierung mit Householder

- Gegebenheiten

- Unterbestimmter QR

# Kapitel 5

# Eigenvektoren

Wir wenden uns nun einem nichtlinearen Problem über Matrizen zu: der Ermittlung ihrer Eigenwerte und Eigenvektoren. Eigenvektoren x und ihre entsprechenden Eigenwerte *ÿ* einer quadratischen Matrix A werden durch die Gleichung Ax = ÿx bestimmt. Es gibt viele Möglichkeiten zu erkennen, dass dieses Problem nichtlinear ist.
Zum Beispiel gibt es ein Produkt der Unbekannten *ÿ* und x, und um die triviale Lösung x = 0 zu vermeiden, beschränken wir x = 1; Diese Einschränkung ist eher kreisförmig als linear. Dank dieser Struktur werden sich unsere Methoden zum Finden von Eigenräumen erheblich von Techniken zum Lösen und Analysieren linearer Gleichungssysteme unterscheiden.

## 5.1 Motivation

Trotz der willkürlich wirkenden Form der Gleichung Ax = ÿx entsteht in vielen Fällen natürlicherweise das Problem, Eigenvektoren und Eigenwerte zu finden. Wir motivieren unsere Diskussion mit einigen Beispielen unten.

### 5.1.1 Statistik

Angenommen, wir verfügen über Maschinen zum Sammeln mehrerer statistischer Beobachtungen über eine Sammlung von Gegenständen. Beispielsweise können wir in einer medizinischen Studie Alter, Gewicht, Blutdruck und Herzfrequenz von 100 Patienten erfassen. Dann kann jeder Patient i durch einen Punkt xi in R4 dargestellt werden , der diese vier Werte speichert.

Natürlich können solche Statistiken eine starke Korrelation aufweisen. Beispielsweise kann es bei Patienten mit höherem Blutdruck wahrscheinlich zu einem höheren Gewicht oder einer höheren Herzfrequenz kommen. Obwohl wir unsere Daten in R4 gesammelt haben, können sie aus diesem Grund in Wirklichkeit – bis zu einem gewissen Grad – in einem niedrigerdimensionalen Raum leben und die Beziehungen zwischen den verschiedenen Variablen besser erfassen.

Nehmen wir zunächst einmal an, dass es tatsächlich einen eindimensionalen Raum gibt, der unserem Datensatz nahekommt. Dann erwarten wir, dass alle Datenpunkte nahezu parallel zu einem Vektor v sind, sodass jeder als xi ÿ civ für verschiedene ci ÿ R geschrieben werden kann. Von zuvor wissen wir, dass die beste Näherung für xi parallel zu v proj ist $_v$ xi :

$$\text{proj}_v = \frac{x_i \cdot v}{v \cdot v} v \text{ per Definition}$$

$$= (x_i \cdot \hat{v})\hat{v} \text{ , da } v \cdot v = v^2$$

Hier definieren wir vˆ ÿ v/v. Natürlich spielt die Größe von v für das vorliegende Problem keine Rolle, daher ist es sinnvoll, den Raum der Einheitsvektoren vˆ zu durchsuchen.

Nach dem Muster der kleinsten Quadrate haben wir ein neues Optimierungsproblem:

$$\text{minimiere } \ddot{y}\, x_i \ddot{y}\, \text{proj}_{\hat{v}} x_i \quad {}^2$$

$$\text{so dass } \hat{v} = 1$$

Wir können unser Optimierungsziel etwas vereinfachen:

$$\ddot{y}\, x_i \ddot{y}\, \text{proj}_{\hat{v}} x_i \quad {}^2 = \ddot{y}\, x_i \ddot{y}\, (x_i \cdot \hat{v})\hat{v} \quad {}^2 \text{ per Definition von Projektion}$$

$$= \ddot{y}\, x_i \quad {}^2 \ddot{y}\, (x_i \cdot \hat{v}) \quad {}^2 \quad \text{da } \hat{v} = 1 \text{ und } w \quad {}^2 = w \cdot w$$

$$= \text{konst.} \ddot{y}\, \ddot{y}\, (x_i \cdot \hat{v}) \quad {}^2$$

Diese Ableitung zeigt, dass wir ein äquivalentes Optimierungsproblem lösen können:

$$\text{maximiere } X\hat{v} \quad {}^2$$

$$\text{so dass } \hat{v} \quad {}^2 = 1,$$

wobei die Spalten von X die Vektoren xi sind . Beachten Sie, dass $X\hat{v} = {}^2\hat{v}\, XX\hat{v}$, sodass nach Beispiel 0.27 der Vektor vˆ dem Eigenvektor von XX mit dem höchsten Eigenwert entspricht. Der Vektor vˆ ist als erste Hauptkomponente des Datensatzes bekannt.

## 5.1.2 Differentialgleichungen

Viele physikalische Kräfte können als Funktionen der Position geschrieben werden. Beispielsweise kann die von einer Feder ausgeübte Kraft zwischen zwei Teilchen an den Positionen x und y in R3 nach dem Hookeschen Gesetz als k(x ÿ y) geschrieben werden ; Solche Federkräfte werden in vielen Simulationssystemen zur Annäherung an die Kräfte verwendet, die Stoffe zusammenhalten. Obwohl diese Kräfte nicht unbedingt eine lineare Position haben, nähern wir sie uns oft linear an. Insbesondere in einem physikalischen System mit n Teilchen kodieren Sie die Positionen aller Teilchen gleichzeitig in einem Vektor X ÿ R3n . Wenn wir dann eine solche Näherung annehmen, können wir schreiben, dass die Kräfte im System für eine Matrix A ungefähr F ÿ AX sind.

Erinnern Sie sich an Newtons zweites Bewegungsgesetz F = ma, oder Kraft ist gleich Masse mal Beschleunigung. In unserem Kontext können wir eine diagonale Massenmatrix M ÿ R3n×3n schreiben , die die Masse jedes Teilchens im System enthält. Dann wissen wir F = MX, wobei eine Primzahl die zeitliche Differenzierung bezeichnet. Natürlich ist X = (X ) , also haben wir am Ende ein Gleichungssystem erster Ordnung:

$$\frac{D}{dt}\begin{bmatrix} X \\ V \end{bmatrix} = \begin{bmatrix} 0 & I_{3n\times 3n} \\ M\ddot{y}1A\ 0 & \end{bmatrix}\begin{bmatrix} X \\ V \end{bmatrix}$$

Hier berechnen wir gleichzeitig beide Positionen in X ÿ R3n und Geschwindigkeiten V ÿ R3n aller n Teilchen als Funktionen der Zeit.

Allgemeiner gesehen treten Differentialgleichungen der Form x = Ax in vielen Zusammenhängen auf, einschließlich der Simulation von Stoffen, Federn, Hitze, Wellen und anderen Phänomenen. Angenommen, wir kennen Eigenvektoren

x1, . . . ,xk von A, so dass Axi = ÿixi . Wenn wir die Anfangsbedingung der Differentialgleichung anhand der Eigenvektoren schreiben, als

$$x(0) = c1x1 + \cdots + ckxk ,$$

dann kann die Lösung der Gleichung in geschlossener Form geschrieben werden:

$$x(t) = c1e \quad ÿ1t\, ÿk\, t\, x1 + \cdots + cke\, xk ,$$

Diese Lösung ist leicht von Hand zu überprüfen. Das heißt, wenn wir die Anfangsbedingungen dieser Differentialgleichung anhand der Eigenvektoren von A schreiben, dann kennen wir ihre Lösung für alle Zeiten t ÿ 0 kostenlos.
Natürlich ist diese Formel nicht das Ende der Geschichte der Simulation: Das Finden des vollständigen Satzes von Eigenvektoren von A ist teuer, und A kann sich im Laufe der Zeit ändern.

## 5.2 Spektrale Einbettung

Angenommen, wir haben eine Sammlung von n Elementen in einem Datensatz und ein Maß wij ÿ 0 dafür, wie ähnlich jedes Elementpaar i und j ist; Wir gehen davon aus, dass wij = wji ist. Vielleicht erhalten wir zum Beispiel eine Sammlung von Fotos und verwenden wij , um die Ähnlichkeit ihrer Farbverteilungen zu vergleichen. Möglicherweise möchten wir die Fotos nach ihrer Ähnlichkeit sortieren, um die Betrachtung und Erkundung der Sammlung zu vereinfachen.

Ein Modell zum Ordnen der Sammlung könnte darin bestehen , jedem Element i eine Nummer xi zuzuweisen und zu verlangen, dass ähnlichen Objekten ähnliche Nummern zugewiesen werden. Mithilfe der Energie können wir messen, wie gut eine Aufgabe ähnliche Objekte gruppiert

$$E(x) = ÿ\ wij(xi ÿ xj)^2 .$$
$$ij$$

Das heißt, E(x) verlangt, dass die Elemente i und j mit hohen Ähnlichkeitswerten auf nahegelegene Werte abgebildet werden .

Natürlich ergibt die Minimierung von E(x) ohne Einschränkungen ein offensichtliches Minimum: xi = const. für alle ich. Durch das Hinzufügen einer Einschränkung x = 1 wird diese konstante Lösung nicht entfernt! Insbesondere ergibt die Annahme von xi = 1/ ÿ n für alle i auf uninteressante Weise x = 1 und E(x) = 0. Daher müssen wir auch diesen Fall entfernen:

$$\text{E(x) minimieren}$$
$$\text{so dass } x^2 = 1$$
$$1 \cdot x = 0$$

Beachten Sie, dass unsere zweite Einschränkung verlangt, dass die Summe von x Null ist.

Noch einmal können wir die Energie vereinfachen:

$$E(x) = ÿ\ wij(xi ÿ xj)^2$$
$$ij$$
$$= ÿ\ wij(x^2 ÿ 2xixj + x^2 j)$$
$$ij$$
$$= ÿ\ aix^2 i ÿ 2ÿ\ wijxixj + ÿ\ bjx^2 j$$
$$ij \qquad J$$

für ai ÿ ÿ wij und bj ÿ ÿ wij
J

= x (A ÿ 2W + B)x wobei diag(A) =a und diag(B) =b = x (2A ÿ 2W)x aufgrund der Symmetrie von W

Es lässt sich leicht überprüfen, dass 1 ein Eigenvektor von 2A ÿ 2W mit dem Eigenwert 0 ist. Interessanter ist, dass der Eigenvektor, der dem zweitkleinsten Eigenwert entspricht, der Lösung unseres obigen Minimierungsziels entspricht! (TODO: KKT-Beweis aus Vorlesung hinzufügen)

## 5.3 Eigenschaften von Eigenvektoren

Wir haben eine Vielzahl von Anwendungen etabliert, die eine Eigenraumberechnung benötigen. Bevor wir jedoch Algorithmen für diesen Zweck untersuchen können, werden wir die Struktur des Eigenwertproblems genauer untersuchen.

Wir können mit einigen Definitionen beginnen, die an dieser Stelle wahrscheinlich offensichtlich sind:

**Definition 5.1** (Eigenwert und Eigenvektor). Ein Eigenvektor x = 0 einer Matrix A ÿ Rn×n ist jeder Vektor, der Ax = ÿx für ein *ÿ* ÿ R erfüllt; das entsprechende *ÿ* wird als Eigenwert bezeichnet. Komplexe Eigenwerte und Eigenvektoren erfüllen die gleichen Beziehungen mit *ÿ* ÿ **C** und x ÿ Cn .

**Definition 5.2** (Spektrum und Spektralradius). Das Spektrum von A ist die Menge der Eigenwerte von A. Der Spektralradius ÿ(A) ist der Eigenwert *ÿ ,* der |ÿ| maximiert .

Der Maßstab eines Eigenvektors ist nicht wichtig. Insbesondere ergibt die Skalierung eines Eigenvektors x mit c A(cx) = cAx = *cÿx* = ÿ(cx), sodass cx ein Eigenvektor mit demselben Eigenwert ist. Wir schränken unsere Suche oft ein, indem wir eine Einschränkung x = 1 hinzufügen. Selbst diese Einschränkung beseitigt die Mehrdeutigkeit nicht vollständig, da nun ±x beide Eigenvektoren mit demselben Eigenwert sind.

Die algebraischen Eigenschaften von Eigenvektoren und Eigenwerten könnten leicht ein Buch füllen. Wir werden unsere Diskussion auf einige wichtige Theoreme beschränken, die den Entwurf numerischer Algorithmen beeinflussen; Wir werden die Entwicklung von CITE AXLER verfolgen. Damit unsere Suche nicht umsonst ist, sollten wir zunächst prüfen, ob jede Matrix mindestens einen Eigenvektor hat. Unsere übliche Strategie besteht darin, zu beachten, dass, wenn *ÿ* ein Eigenwert mit Ax = ÿx ist, (A ÿ ÿIn×n)x =0; somit ist *ÿ* genau dann ein Eigenwert, wenn die Matrix A ÿ ÿIn×n nicht vollrangig ist.

**Lemma 5.1** (CITE-Theorem 2.1). Jede Matrix A ÿ Rn×n hat mindestens einen (komplexen) Eigenvektor.

Nachweisen. Nehmen Sie einen beliebigen Vektor x ÿ Rn\{0}. Die Menge {x, Ax, A 2x, · · · A nx} muss linear abhängig sein, da sie n + 1 Vektoren in n Dimensionen enthält. Es gibt also Konstanten c0, . . . , cn ÿ **R** mit cn = 0, so dass

$$0 = c0x + c1Ax + \cdots + cnA^{N}x.$$

Wir können ein Polynom aufschreiben

$$f(z) = c0 + c1z + \cdots + cnz^{N}.$$

Nach dem Fundamentalsatz der Algebra gibt es n Wurzeln zi ÿ **C** mit f(z) = cn(z ÿ z1)

$$(z \ddot{y} z2)\cdot\cdot\cdot(z \ddot{y} zn).$$

Dann haben wir:

$$\begin{aligned} 0 &= c0x + c1Ax + \cdots + cnA^{N}x \\ &= (c0\ In\times n + c1A + \cdots + cnA^{n})x \\ &= cn(A \ddot{y} z1\ In\times n)\cdot\cdot\cdot(A \ddot{y} zn\ In\times n)x \text{ durch unsere Faktorisierung} \end{aligned}$$

Somit hat mindestens ein A ÿ zi In×n einen Nullraum, was zeigt, dass es je nach Bedarf v mit Av = ziv gibt. □

Es gibt noch eine weitere Tatsache, die es wert ist, überprüft zu werden, um unsere Diskussion über Eigenvektorberechnungen zu motivieren
Stationierung:

**Lemma 5.2** (CITE Proposition 2.2). Eigenvektoren, die unterschiedlichen Eigenwerten entsprechen, müssen linear unabhängig sein.

Nachweisen. Angenommen, dies ist nicht der Fall. Dann gibt es Eigenvektoren x1, · · · ,xk mit unterschiedlichen Eigenwerten ÿ1, · · · , ÿk , die linear abhängig sind. Dies impliziert, dass es Koeffizienten c1, . gibt . . , ck nicht alle Null mit 0 = c1x1 + · · · + ckxk .
Wenn wir mit der Matrix (A ÿ ÿ2 In×n)· · ·(A ÿ ÿk In×n) vormultiplizieren ,
finden wir:

$$0 = (A - \ddot{y}_2 I_{n\times n})\cdots(A - \ddot{y}_k I_{n\times n})(c_1x_1 + \cdots + c_kx_k) = c_1(\ddot{y}_1 - \ddot{y}_2)\cdots(\ddot{y}_1 - \ddot{y}_k)x_1 \text{ da } Ax_i = \ddot{y}_i x_i$$

Da alle ÿi unterschiedlich sind, ergibt sich c1 = 0. Ein ähnlicher Beweis zeigt, dass die restlichen ci Null sein müssen, was der linearen Abhängigkeit widerspricht. □

Dieses Lemma zeigt, dass eine n × n-Matrix höchstens n verschiedene Eigenwerte haben kann, da eine Menge von n Eigenwerten n linear unabhängige Vektoren ergibt. Die maximale Anzahl linear unabhängiger Eigenvektoren, die einem einzelnen Eigenwert *ÿ* entsprechen , wird als geometrische Vielfachheit von ÿ bezeichnet.

Es stimmt jedoch nicht, dass eine Matrix genau n linear unabhängige Eigenvektoren haben muss.
Dies ist bei vielen Matrizen der Fall, die wir als nicht fehlerhaft bezeichnen:

**Definition 5.3** (Nicht defekt). Eine Matrix A ÿ Rn×n ist nichtdefekt oder diagonalisierbar, wenn ihre Eigenvektoren Rn aufspannen .

Wir nennen eine solche Matrix aus folgendem Grund diagonalisierbar: Wenn eine Matrix diagonalisierbar ist, dann hat sie n Eigenvektorenx1, . . . ,xn ÿ Rn mit entsprechenden (ggf. nicht eindeutigen) Eigenwerten ÿ1, . . . , ÿn.
Nehmen Sie die Spalten von X als die Vektoren xi und definieren Sie D als die Diagonalmatrix mit den Eigenwerten ÿ1, . . . , ÿn entlang der Diagonale. Dann gilt per Definition der Eigenwerte AX = XD; Dies ist einfach eine „gestapelte“ Version von Axi = ÿixi .
Mit anderen Worten,

$$D = X^{-1}AX,$$

bedeutet, dass A durch eine Ähnlichkeitstransformation A ÿ X ÿ1AX diagonalisiert wird: **Definition**

**5.4** (Ähnliche Matrizen). Zwei Matrizen A und B sind ähnlich, wenn es T mit B = T ÿ1AT gibt .

Ähnliche Matrizen haben die gleichen Eigenwerte, denn wenn Bx = ÿx, dann ist T ÿ1ATx = ÿx. Äquivalent ist A(Tx) = ÿ(Tx), was zeigt, dass Tx ein Eigenvektor mit dem Eigenwert ÿ ist.

## 5.3.1 Symmetrische und positiv definite Matrizen

Angesichts unserer besonderen Berücksichtigung normaler Matrizen AA überrascht es nicht, dass symmetrische und/oder positiv definite Matrizen eine besondere Eigenvektorstruktur aufweisen. Wenn wir eine dieser Eigenschaften überprüfen können, können spezielle Algorithmen verwendet werden, um ihre Eigenvektoren schneller zu extrahieren.

Erstens können wir eine Eigenschaft symmetrischer Matrizen beweisen, die die Notwendigkeit einer Komplexität überflüssig macht Arithmetik. Wir beginnen mit einer Verallgemeinerung symmetrischer Matrizen auf Matrizen in Cn×n :

**Definition 5.5** (Komplexes Konjugat). Die komplexe Konjugierte einer Zahl z ÿ a + bi ÿ **C** ist $\bar{z}$ ÿ a ÿ bi.

**Definition 5.6** (Konjugierte Transponierung). Die konjugierte Transponierte von A ÿ Cm×n ist AH ÿ A¯.

**Definition 5.7** (Hermitesche Matrix). Eine Matrix A ÿ Cn×n ist hermitesch, wenn A = A H.

Beachten Sie, dass eine symmetrische Matrix A ÿ Rn×n automatisch hermitesch ist, da sie keinen komplexen Teil hat. Mit dieser leichten Verallgemeinerung können wir eine Symmetrieeigenschaft für Eigenwerte beweisen.
Unser Beweis wird das Skalarprodukt der Vektoren in Cn verwenden , gegeben durch

$$x,y = \ddot{y}\ xi\bar{y}\bar{i}\ ,$$

wobei x,y ÿ Cn . Beachten Sie, dass diese Definition wiederum mit x ·y übereinstimmt, wenn x,y ÿ Rn . Die Eigenschaften dieses inneren Produkts stimmen größtenteils mit denen des Skalarprodukts auf Rn überein , mit der Ausnahme, eine bemerkenswerte dass v, w = w ,v.

**Lemma 5.3.** Alle Eigenwerte hermitescher Matrizen sind reell.

Nachweisen. Angenommen, A ÿ Cn×n ist hermitesch mit Ax = ÿx. Durch Skalierung können wir x 1 $^2$ = x,x = annehmen. Dann gilt:

$\ddot{y}$ = ÿx,x, da x die Norm 1 hat
= ÿx,x durch Linearität von ·, ·
= Ax,x, da Ax = ÿx
= (Ax) x¯ nach Definition von = x ·, ·
(A¯x) durch Entwicklung des Produkts und Verwendung von ab = a¯ ¯b =
x, A Hx nach Definition von A $^H$ und ·, ·
= x, Ax, da A = A = ÿ¯x,x, da $^H$
Ax = ÿx = ÿ¯, da x die Norm 1
hat

Somit ist $\ddot{y}$ = ÿ¯, was je nach Bedarf nur dann passieren kann, wenn $\ddot{y}$ ÿ R. □

Symmetrische und hermitesche Matrizen verfügen außerdem über eine besondere Orthogonalitätseigenschaft für ihre Eigenvec Toren:

**Lemma 5.4.** Eigenvektoren, die unterschiedlichen Eigenwerten hermitescher Matrizen entsprechen, müssen orthogonal sein.

Nachweisen. Angenommen, A ÿ Cn×n sei hermitesch und sei $\ddot{y}$ = $\mu$ mit Ax = ÿx und Ay = µy. Durch das vorherige Lemma wissen wir ÿ, $\mu$ ÿ R. Dann ist Ax,y = ÿx,y. Da A aber hermitesch ist, können wir auch Ax,y = x, A Hy = x, Ay = µx,y schreiben. Somit ist ÿx,y = µx,y. Da $\ddot{y}$ = µ, müssen wir x,y = 0 haben.

□

Schließlich können wir ohne Beweis ein krönendes Ergebnis der linearen Algebra angeben, den Spektralsatz. Dieser Satz besagt, dass keine symmetrische oder hermitesche Matrix fehlerhaft sein kann, was bedeutet, dass eine n × n-Matrix, die diese Eigenschaft erfüllt, genau n orthogonale Eigenvektoren hat.

**Satz 5.1** (Spektralsatz). Angenommen, A ÿ Cn×n ist hermitesch (wenn A ÿ Rn×n , nehmen wir an, dass es symmetrisch ist). Dann hat A genau n orthonormale Eigenvektoren x1, · · · ,xn mit (ggf. wiederholten) Eigenwerten ÿ1, . . . , ÿn . Mit anderen Worten: Es gibt eine Orthonormalmatrix X von Eigenvektoren und eine Diagonalmatrix D von Eigenwerten, sodass D = X AX ist.

Dieser Satz impliziert, dass jeder Vektor y ÿ Rn in eine lineare Kombination der Eigenvektoren einer hermiteschen Matrix A zerlegt werden kann. Viele Berechnungen sind auf dieser Grundlage einfacher, wie unten gezeigt:

**Beispiel 5.1** (Berechnung mit Eigenvektoren). Nimm x1, . . . ,xn ÿ Rn sind die Einheitslängen-Eigenvektoren der symmetrischen Matrix A ÿ Rn×n . Angenommen, wir möchten Ay =b lösen . Wir können schreiben

$$b = c_1x_1 + \cdots + c_nx_n,$$

wobei ci =b · xi durch Orthonormalität. Es ist leicht, die folgende Lösung zu erraten:

$$\underline{\underline{c_1 \; c_n}}\; x_1 + \cdots + y = \underline{\underline{x_n.}}\; \ddot{y}_1 \; \ddot{y}_n$$

Insbesondere finden wir:

$$Ay = A \quad \underline{\underline{c_1 \; c_n}}\; x_1 + \cdots + x_n \underline{\underline{\ddot{y}_1 \; \ddot{y}_n \; c_1 \; c_n}}$$

$$= \underline{\underline{\ddot{y}_1 \; \ddot{y}_n}}\; Ax_1 + \cdots + \underline{\underline{\quad}}\; Ax_n$$

$$= c_1x_1 + \cdots + c_nx_n$$

$$= b, \text{ wie gewünscht.}$$

Die obige Berechnung ergibt sowohl ein positives als auch ein negatives Ergebnis. Es zeigt, dass Operationen wie die Inversion angesichts der Eigenvektoren von symmetrischem A unkompliziert sind. Auf der anderen Seite bedeutet dies, dass das Finden des vollständigen Satzes von Eigenvektoren einer symmetrischen Matrix A „mindestens“ so schwierig ist wie das Lösen von Ax =b.

Nach unserem Streifzug durch die komplexen Zahlen kehren wir zu den reellen Zahlen zurück, um eine letzte nützliche, wenn auch einfache Tatsache über positiv definite Matrizen zu beweisen:

**Lemma 5.5.** Alle Eigenwerte positiv definiter Matrizen sind nichtnegativ.

Nachweisen. Nehmen Sie A ÿ Rn×n positiv definit und nehmen Sie Ax = ÿx mit x = 1 an. Durch positive Definitheit wissen wir, dass x Ax ÿ 0 ist . Aber je nach Bedarf gilt x Ax = x (ÿx) = ÿx = ÿ. $^2$ □

## 5.3.2 Spezialisierte Eigenschaften1

#### Charakteristisches Polynom

Denken Sie daran, dass die Determinante einer Matrix det A genau dann die Beziehung erfüllt, dass det A = 0 ist, wenn A invertierbar ist. Daher besteht eine Möglichkeit, Eigenwerte einer Matrix zu finden, darin, Wurzeln des charakteristischen Polynoms zu finden

$$pA(\ddot{y}) = \det(A \ddot{y} \ddot{y}In\times n).$$

Wir werden in unserer Diskussion hier keine Determinanten definieren, aber die Vereinfachung von pA zeigt, dass es sich um ein Polynom n-ten Grades in ÿ handelt. Dies liefert einen alternativen Grund, warum es höchstens n verschiedene Eigenwerte gibt, da es höchstens n Wurzeln dieser Funktion gibt.

Aus dieser Konstruktion können wir die algebraische Multiplizität eines Eigenwerts als seine Multiplizität als Wurzel von pA definieren. Es ist leicht zu erkennen, dass die algebraische Multiplizität mindestens so groß ist wie die geometrische

1Dieser Abschnitt kann übersprungen werden, wenn den Lesern nicht genügend Hintergrundwissen vorliegt, er wird jedoch der Vollständigkeit halber eingefügt.

Vielzahl. Wenn die algebraische Multiplizität 1 ist, heißt die Wurzel einfach, weil sie einem einzelnen Eigenvektor entspricht, der von allen anderen linear abhängig ist. Eigenwerte, bei denen die algebraische und geometrische Multiplizität nicht gleich sind, werden als defekt bezeichnet.

In der numerischen Analyse vermeiden wir die Diskussion der Determinante einer Matrix. Obwohl es sich um eine praktische theoretische Konstruktion handelt, ist ihr praktischer Nutzen begrenzt. Determinanten sind schwer zu berechnen. Tatsächlich versuchen Eigenwertalgorithmen nicht, Wurzeln von pA zu finden , da dies die Auswertung einer Determinante erfordern würde. Darüber hinaus hat die Determinante det A nichts mit der Konditionierung von A zu tun, sodass eine Determinante von det(A ÿ ÿIn×n) nahe Null möglicherweise nicht zeigt, dass *ÿ* nahezu ein Eigenwert von A ist.

### Jordan-Normalform

Wir können eine Matrix nur dann diagonalisieren, wenn sie einen vollständigen Eigenraum hat. Alle Matrizen ähneln jedoch einer Matrix in Jordan-Normalform, die folgende Form hat: •

Werte ungleich Null befinden sich auf den Diagonaleinträgen aii und auf der „Superdiagonalen“ ai(i+1) .

- Diagonalwerte sind Eigenwerte, die so oft wie ihre Multiplizität wiederholt werden; die Matrix ist Blockdiagonale um diese Cluster.
- Außerdiagonale Werte sind 1 oder 0.

Somit sieht die Form in etwa wie folgt aus

$$\begin{bmatrix} \ddots & & & & \\ & \ddots & 1 & & \\ & & \ddots & & \\ & & & \ddots & 1 \\ & & & & \ddots \end{bmatrix}$$

Die Jordan-Normalform ist theoretisch attraktiv, weil sie immer existiert, aber die 1/0-Struktur ist diskret und unter numerischen Störungen instabil.

## 5.4 Berechnung von Eigenwerten

Die Berechnung und Schätzung der Eigenwerte einer Matrix ist ein gut untersuchtes Problem mit vielen möglichen Lösungen. Jede Lösung ist auf eine andere Situation abgestimmt und das Erreichen einer maximalen Konditionierung oder Geschwindigkeit erfordert das Experimentieren mit mehreren Techniken. Hier behandeln wir einige der beliebtesten und einfachsten Lösungen für das Eigenwertproblem, die in der Praxis häufig vorkommen.

### 5.4.1 Power-Iteration

Nehmen wir zunächst an, dass A ÿ Rn×n symmetrisch ist. Dann können wir nach dem Spektralsatz die Eigenvektoren x1, ... schreiben . . . ,xn ÿ Rn ; Wir sortieren sie so, dass ihre entsprechenden Eigenwerte | ÿ1| erfüllen ÿ |ÿ2| ÿ · · · ÿ |ÿn|.

Angenommen, wir nehmen einen beliebigen Vektor v. Da die Eigenvektoren von A Rn aufspannen, wir können schreiben:

$$v = c_1x_1 + \cdots + c_nx_n.$$

Dann,

$$\begin{aligned} Av &= c_1Ax_1 + \cdots + c_nAx_n \\ &= c_1\ddot{y}_1x_1 + \cdots + c_n\ddot{y}_nx_n \text{ , da } Ax_i = \ddot{y}_ix_i \\ \ddot{y}_2 \ddot{y}_n &= \ddot{y}_1 \underline{c_1x_1 + \underline{c_2x_2 + \cdots + c_nx_n} \ddot{y}_1 \ddot{y}_1} \end{aligned}$$

$$A^2v = \ddot{y}_1^2 \left[ c_1x_1 + \left(\frac{\ddot{y}_2}{\ddot{y}_1}\right)^2 c_2x_2 + \cdots + \left(\frac{\ddot{y}_n}{\ddot{y}_1}\right)^2 c_nx_n \right]$$

$$\vdots$$

$$A^kv = \ddot{y}_1^k \left[ c_1x_1 + \left(\frac{\ddot{y}_2}{\ddot{y}_1}\right)^k c_2x_2 + \cdots + \left(\frac{\ddot{y}_n}{\ddot{y}_1}\right)^k c_nx_n \right]$$

Beachten Sie, dass für k ÿ ÿ das Verhältnis $(\ddot{y}_i/\ddot{y}_1)^k \ddot{y} 0$ ist, es sei denn, $\ddot{y}_i = \ddot{y}_1$, da $\ddot{y}_1$ per Definition den größten Betrag aller Eigenwerte hat. Wenn also x die Projektion von v auf den Raum der Eigenvektoren mit den Eigenwerten $\ddot{y}_1$ ist, dann gilt für k ÿ ÿ die folgende Näherung immer genauer:

$$\ddot{y}_1^k A^k v \ddot{y} x$$

Diese Beobachtung führt zu einem äußerst einfachen Algorithmus zur Berechnung eines Eigenvektors x von A entspricht dem größten Eigenwert $\ddot{y}_1$:

1. Nehmen Sie $v_1 \ddot{y} R^n$ als einen beliebigen Vektor ungleich Null.
2. Iterieren Sie bis zur Konvergenz zur Erhöhung von k:

$$v_k = Av_k\ddot{y}_1$$

Dieser als Power-Iteration bekannte Algorithmus erzeugt Vektoren $v_k$, die immer paralleler zum gewünschten $x_1$ sind. Es ist garantiert, dass sie konvergiert, selbst wenn A asymmetrisch ist, obwohl der Beweis dieser Tatsache aufwändiger ist als die obige Herleitung. Das einzige Mal, dass diese Technik fehlschlagen kann, ist, wenn wir versehentlich $v_1$ so wählen, dass $c_1 = 0$, aber die Wahrscheinlichkeit, dass dies geschieht, ist gering bis gar nicht vorhanden.

Natürlich, wenn $|\ddot{y}_1| > 1$, dann $v_k \ddot{y}$ ÿ als k ÿ ÿ, eine unerwünschte Eigenschaft für Gleitkommaarithmetik. Denken Sie daran, dass uns nur die Richtung des Eigenvektors und nicht seine Größe wichtig ist, sodass die Skalierung keinen Einfluss auf die Qualität unserer Lösung hat. Um diese Divergenzsituation zu vermeiden, können wir einfach bei jedem Schritt normalisieren und so den normalisierten Potenziterationsalgorithmus erstellen:

1. Nehmen Sie $v_1 \ddot{y} R^n$ als einen beliebigen Vektor ungleich Null.
2. Iterieren Sie bis zur Konvergenz zur Erhöhung von k:

$$w_k = Av_k\ddot{y}_1$$

$$v_k = \frac{w_k}{\|w_k\|}$$

Beachten Sie, dass wir die Norm · nicht mit einem bestimmten Index versehen haben. Mathematisch gesehen reicht jede Norm aus, um das Divergenzproblem zu verhindern, da wir gezeigt haben, dass alle Normen auf Rn äquivalent sind. In der Praxis verwenden wir oft die Unendlichkeitsnorm · ÿ; In diesem Fall lässt sich leicht überprüfen, dass w k ÿ |ÿ1|.

## 5.4.2 Inverse Iteration

Wir haben jetzt eine Strategie, um den Eigenwert ÿ1 mit dem größten Betrag zu finden . Angenommen, A ist invertierbar, sodass wir y = A ÿ1v auswerten können , indem wir Ay = v mithilfe der in den vorherigen Kapiteln behandelten Techniken lösen.

Wenn Ax = ÿx, dann ist x = ÿA ÿ1x oder gleichwertig

$$A^{\ddot{y}1}x = \frac{1}{\ddot{y}}x.$$

Damit haben wir gezeigt, dass 1/ÿ ein Eigenwert von A und dann |b| $^{\ddot{y}1}$ mit Eigenvektor x. Beachten Sie, dass wenn |a| ÿ |b| ist ÿ |a| für jedes a, b ÿ $^{\ddot{y}1}$, also ist der kleinste Eigenwert von A der größte. Diese Beobachtung ergibt eine Strategie zum Finden Betragseigenvektor von A, sogenannte$^{\ddot{y}1}$ von ÿn anstelle von ÿ1
inverse Potenziteration:

1. Nehmen Sie v1 ÿ Rn als einen beliebigen Vektor ungleich Null.
2. Iterieren Sie bis zur Konvergenz zur Erhöhung von k:
   (a) Lösen Sie nach w k auf : Aw k = vkÿ1
   (b) Normalisieren: vk = $\frac{w\,k}{w\,k}$

Wir lösen wiederholt Gleichungssysteme mit derselben Matrix A, was eine perfekte Anwendung der Faktorisierungstechniken aus früheren Kapiteln ist. Wenn wir beispielsweise A = LU schreiben, könnten wir eine äquivalente, aber wesentlich effizientere Version der Umkehrpotenziteration formulieren:

1. Faktor A = LU
2. Nehmen Sie v1 ÿ Rn als einen beliebigen Vektor ungleich Null.
3. Iterieren Sie bis zur Konvergenz zur Erhöhung von k:
   (a) Lösen Sie nach yk durch Vorwärtssubstitution auf: Lyk = vkÿ1 (b)
   Lösen Sie nach w k durch Rückwärtssubstitution auf: Uw k = yk
   (c) Normalisieren Sie: vk = $\frac{w\,k}{w\,k}$

## 5.4.3 Schalten

Angenommen, ÿ2 ist der Eigenwert mit der zweitgrößten Größe von A. Angesichts unserer ursprünglichen Ableitung der Potenziteration ist leicht zu erkennen, dass die Potenziteration am schnellsten konvergiert, wenn |ÿ2/ÿ1| ist klein, da in diesem Fall die Leistung (ÿ2/ÿ1) schnell abfällt. Wenn dieses$^{k}$ Verhältnis hingegen nahezu 1 beträgt, kann es viele Iterationen der Potenziteration erfordern, bis ein einzelner Eigenvektor isoliert wird.

Wenn die Eigenwerte von A ÿ1 sind, . . . , ÿn, dann ist es leicht zu erkennen, dass die Eigenwerte von A ÿ ÿIn×n ÿ1 ÿ ÿ, . . . , ÿn ÿ ÿ. Dann besteht eine Strategie, um die Potenziteration schnell konvergieren zu lassen, darin, *ÿ* so zu wählen, dass:

$$\frac{\ddot{y}2 \ddot{y} \ddot{y}}{\ddot{y}1 \ddot{y} \ddot{y}} < \frac{\ddot{y}2}{\ddot{y}1} .$$

Natürlich kann es eine Kunst sein, ein solches *ÿ* zu erraten, da die Eigenwerte von A offensichtlich zunächst nicht bekannt sind. Wenn wir ebenfalls annehmen, dass *ÿ* nahe einem Eigenwert von A liegt, dann hat A ÿ ÿIn×n einen Eigenwert nahe 0, den wir durch inverse Iteration ermitteln können.

Eine Strategie, die sich diese Beobachtung zunutze macht, ist als Rayleigh-Quotienten-Iteration bekannt. Wenn wir eine feste Schätzung für einen Eigenvektor x von A haben, dann ist durch NUMBER die Kleinste-Quadrate-Approximation des entsprechenden Eigenwerts *ÿ* gegeben

$$\bar{y}\,\bar{y} \quad \frac{x\ Axt}{X\ \begin{matrix}2\\2\end{matrix}} .$$

Dieser Bruch wird als Rayleigh-Quotient bezeichnet. Daher können wir versuchen, die Konvergenz zu erhöhen, indem wir wie folgt iterieren:

1. Nehmen Sie an , dass v1 ÿ Rn ein beliebiger Vektor ungleich Null oder eine anfängliche Schätzung eines Eigenvektors ist.

2. Iterieren Sie bis zur Konvergenz zur Erhöhung von k:

   (a) Schreiben Sie die aktuelle Schätzung des Eigenwerts

$$\ddot{y}k = \frac{v\ k\ddot{y}1Avk\ddot{y}1}{vk\ddot{y}1\ \begin{matrix}2\\2\end{matrix}}$$

   (b) Lösen Sie nach w k auf : (A ÿ ÿk In×n)w k (c) Normalisieren $= vk\ddot{y}1$

   Sie: vk = $\frac{w\,k}{w\,k}$

Bei einer guten Ausgangsschätzung konvergiert diese Strategie viel schneller, aber die Matrix A ÿ ÿk In×n ist bei jeder Iteration unterschiedlich und kann nicht mit LU oder den meisten anderen Strategien vorfaktorisiert werden. Somit sind weniger Iterationen notwendig, aber jede Iteration nimmt mehr Zeit in Anspruch!

## 5.4.4 Finden mehrerer Eigenwerte

Bisher haben wir Techniken zum Finden eines einzelnen Eigenwert/Eigenvektor-Paares beschrieben: Potenzielle Iteration, um den größten Eigenwert zu finden, inverse Iteration, um den kleinsten zu finden, und Verschiebung zu Zielwerten dazwischen. Für viele Anwendungen reicht ein einzelner Eigenwert natürlich nicht aus. Zum Glück können wir unsere Strategien erweitern, um auch diesen Fall zu bearbeiten.

#### Deflation

Erinnern Sie sich an unsere Potenziterationsstrategie: Wählen Sie ein beliebiges v1 und multiplizieren Sie es iterativ mit A, bis nur noch der größte Eigenwert ÿ1 übrig bleibt. Nehmen Sie x1 als den entsprechenden Eigenvektor.

Wir haben jedoch schnell einen unwahrscheinlichen Fehlermodus dieses Algorithmus verworfen, wenn v1 ·x1 = 0. In diesem Fall werden Sie niemals einen Vektor wiederherstellen, egal wie oft Sie mit A vormultiplizieren

parallel zu x1, da man eine Nullkomponente nicht verstärken kann. Die Wahrscheinlichkeit, eine solche v1 zu wählen , ist genau null, sodass die Power-Iteration in allen Fällen, außer in den schädlichsten Fällen, sicher bleibt.

Wir können diesen Nachteil auf den Kopf stellen und eine Strategie formulieren, um mehr als einen Eigenwert zu finden, wenn A symmetrisch ist. Angenommen, wir finden x1 und ÿ1 wie zuvor durch Potenziteration. Jetzt starten wir die Power-Iteration neu, jedoch bevor wir mit Projekt x1 aus v1 beginnen. Da die Eigenvektoren von A dann orthogonal sind, wird durch die Potenziteration der zweitgrößte Eigenwert wiederhergestellt!

Aufgrund numerischer Probleme kann es vorkommen, dass die Anwendung von A auf einen Vektor eine kleine Komponente parallel zu x1 einführt. In der Praxis können wir diesen Effekt vermeiden, indem wir in jeder Iteration projizieren. Am Ende ergibt diese Strategie den folgenden Algorithmus zur Berechnung der Eigenwerte in der Reihenfolge absteigender Größe:

- Für jeden gewünschten Eigenwert = 1, 2, . . .
  1. Nehmen Sie v1 ÿ Rn als einen beliebigen Vektor ungleich Null.
  2. Iterieren Sie bis zur Konvergenz zur Erhöhung von

     k: (a) Projizieren Sie die Eigenvektoren, die wir bereits berechnet haben:

     $$uk = vkÿ1 ÿ projspan\{x1,...,xÿ1\} vkÿ1$$

     (b) Multiplizieren Sie Auk = w

     k (c) Normalisieren Sie: vk = $\frac{wk}{wk}$
  3. Fügen Sie das Ergebnis der Iteration zur Menge von xi hinzu

Die innere Schleife entspricht einer Potenziteration auf der Matrix AP, wobei P x1, . herausprojiziert . . ,xÿ1 .
Es ist leicht zu erkennen, dass AP die gleichen Eigenvektoren wie A hat; seine Eigenwerte sind *ÿ , . . . ,* ÿn , wobei die restlichen Eigenwerte auf Null gesetzt werden.

Allgemeiner gesagt besteht die Deflationsstrategie darin, die Matrix A so zu modifizieren, dass die Potenziteration einen Eigenvektor offenbart, den Sie noch nicht berechnet haben. Beispielsweise ist AP eine Modifikation von A, sodass die großen Eigenwerte, die wir bereits berechnet haben, auf Null gesetzt werden.

Unsere Projektionsstrategie schlägt fehl, wenn A asymmetrisch ist, da seine Eigenvektoren in diesem Fall möglicherweise nicht orthogonal sind. Andere, weniger offensichtliche Deflationsstrategien können in diesem Fall funktionieren. Angenommen, Ax1 = ÿ1x1 mit x1 = 1. Nehmen Sie H als die Householder-Matrix an, sodass Hx1 = e1, der erste Standardbasisvektor. Ähnlichkeitstransformationen wirken sich wiederum nicht auf die Menge der Eigenvektoren aus, daher könnten wir versuchen, mit H zu konjugieren. Überlegen Sie, was passiert, wenn wir HAH mit e1 multiplizieren:

HAHe1 = HAHe1 , da H symmetrisch ist

= HAx1, da Hx1 =e1 und H = ÿ1Hx1 , $^{2}$ = In×n

da Ax1 = ÿ1x1 = ÿ1e1 nach Definition von H

Somit ist die erste Spalte von HAH ÿ1e1, was zeigt, dass HAH die folgende Struktur hat (CITE HEIDE):

$$HAH = \begin{matrix} ÿ1 & b \\ 0 & B \end{matrix} .$$

Die Matrix B ÿ R(nÿ1)×(nÿ1) hat Eigenwerte ÿ2, . . . , ÿn. Daher besteht eine andere Strategie zur Deflation darin, immer kleinere B-Matrizen zu konstruieren, wobei jeder Eigenwert mithilfe einer Potenziteration berechnet wird.

**QR-Iteration**

Die Deflation hat den Nachteil, dass wir jeden Eigenvektor separat berechnen müssen, was langsam sein kann und zu Fehlern führen kann, wenn einzelne Eigenwerte nicht genau sind. Unsere verbleibenden Strategien versuchen, mehr als einen Eigenvektor gleichzeitig zu finden.

Denken Sie daran, dass ähnliche Matrizen A und B = T ÿ1AT dieselben Eigenwerte haben müssen. Somit kann ein Algorithmus, der versucht, die Eigenwerte von A zu finden, Ähnlichkeitstransformationen frei auf A anwenden. Natürlich kann die Anwendung von T im Allgemeinen schwierig sein, da sie effektiv eine Invertierung von T erfordern würde. Daher suchen wir nach T-Matrizen, deren Umkehrungen leicht zu ermitteln sind anwenden.

Einer unserer Beweggründe für die Ableitung der QR-Faktorisierung war, dass die Matrix Q orthogonal ist und Qÿ1 = Q erfüllt. Daher sind Q und Qÿ1 gleichermaßen einfach anzuwenden, was orthogonale Matrizen zu einer guten Wahl für Ähnlichkeitstransformationen macht.

Aber welche orthogonale Matrix Q sollten wir wählen? Idealerweise sollte Q die Struktur von A beinhalten und gleichzeitig einfach zu berechnen sein. Es ist unklar, wie man einfache Transformationen wie Householder-Matrizen strategisch anwendet, um mehrere Eigenwerte aufzudecken,2 aber wir wissen, wie man ein solches Q einfach durch Faktorisieren von A = QR erzeugt. Dann könnten wir A mit Q konjugieren und finden: Q ÿ1AQ = Q AQ = Q (QR)Q = (Q Q)RQ = RQ

Erstaunlicherweise ist die Konjugation von A = QR durch die orthogonale Matrix Q identisch mit dem Schreiben des Produkts RQ!

Auf der Grundlage dieser Überlegungen stellten in den 1950er Jahren mehrere Gruppen europäischer Mathematiker Hypothesen auf bemessen den gleichen eleganten iterativen Algorithmus zum Finden der Eigenwerte einer Matrix A:

1. Nehmen Sie A1 = A.
2. Für k = 1, 2, . . .
   (a) Faktor Ak = QkRk . (b)
   Schreiben Sie Ak+1 = RkQk .

Nach unserer obigen Ableitung haben die Matrizen Ak alle die gleichen Eigenwerte wie A. Nehmen wir außerdem an, dass die Ak -Werte gegen ein Aÿ konvergieren. Dann können wir Aÿ = QÿRÿ faktorisieren und durch Konvergenz wissen wir Aÿ = QÿRÿ = RÿQÿ. Nach ZAHL sind die Eigenwerte von Rÿ einfach die Werte entlang der Diagonale von Rÿ, und nach ZAHL muss das Produkt RÿQÿ = Aÿ wiederum die gleichen Eigenwerte haben. Schließlich hat Aÿ konstruktionsbedingt die gleichen Eigenwerte wie A. Wir haben also gezeigt, dass die QR-Iteration bei Konvergenz die Eigenwerte von A auf einfache Weise offenbart.

Natürlich geht die obige Ableitung davon aus, dass es Aÿ mit Ak ÿ Aÿ als k ÿ ÿ gibt. Tatsächlich ist die QR-Iteration eine stabile Methode, die in vielen wichtigen Situationen garantiert konvergiert, und die Konvergenz kann sogar durch eine Änderung der Strategie verbessert werden. Wir werden hier keine genauen Bedingungen herleiten, sondern können stattdessen eine Vorstellung davon geben, warum diese scheinbar willkürliche Strategie konvergieren sollte. Im Folgenden geben wir einige Hinweise für den symmetrischen Fall A = A, der dank der Orthogonalität der Eigenvektoren in diesem Fall einfacher zu analysieren ist.

Angenommen, die Spalten von A sind durch a1, . gegeben . . ,an, und betrachten Sie die Matrix $A^k$ für großes k. Wir kann schreiben:

$$A^k = A^{k\ddot{y}1} \cdot A = \begin{bmatrix} | & | & & | \\ A\ddot{y}1a1 & A\ddot{y}1a2 & \cdots & A\ddot{y}1an \\ | & | & & | \end{bmatrix}$$

2Fortgeschrittenere Techniken bewirken jedoch genau dies!

Nach unserer Ableitung der Potenziteration ist die erste Spalte von A im Allgemeinen parallel zum Eigenvektor x1 mit dem größten Betrag |ÿ1| da wir einen Vektor a1 genommen und ihn viele Male mit A multipliziert haben.
Wenn wir unsere Intuition aus der Deflation anwenden, nehmen wir an, dass wir a1 aus der zweiten Spalte von A projizieren. Ein Vektor muss nahezu parallel zu x2 sein, da es sich um den zweitdominantesten Eigenwert handelt! Weiter Induktiv würde die Faktorisierung von A = QR einen Satz von Eigenvektoren als Spalten von Q in der Reihenfolge abnehmender Eigenwertgröße mit den entsprechenden Eigenwerten entlang der Diagonale von R ergeben.

Natürlich nimmt die Berechnung von A für große k die Bedingungszahl von A zur k-ten Potenz, sodass QR auf der resultierenden Matrix wahrscheinlich fehlschlägt; Dies ist deutlich zu erkennen, da alle Spalten von A für große k wie x1 aussehen sollten. Wir können jedoch folgende Beobachtung machen:

$$
\begin{aligned}
A &= Q_1R_1 \\
A^2 &= (Q_1R_1)(Q_1R_1) \\
&= Q_1(R_1Q_1)R_1 \\
&= Q_1Q_2R_2R_1 \text{ unter Verwendung der obigen Notation der QR-Iteration, da } A_2 = R_1Q_1 \\
&\vdots \\
A^k &= Q_1Q_2 \cdots Q_kR_kR_k\ddot{y}_1 \cdots R_1
\end{aligned}
$$

Die separate Gruppierung der Qi -Variablen und der Ri- Variablen liefert eine QR-Faktorisierung von A. Wir erwarten, dass die $A^k$. Daher, Spalten von $Q_1 \cdots Q_k$ zu den Eigenvektoren von A konvergieren.

Durch ein ähnliches Argument können wir finden

$$
\begin{aligned}
A &= Q_1R_1 \\
A_2 &= R_1Q_1 \text{ durch unsere Konstruktion der QR-Iteration} \\
&= Q_2R_2 \text{ nach Definition der Faktorisierung} \\
A_3 &= R_2Q_2 \text{ aus QR-Iteration} \\
&= Q_2^{\cdot}A_2Q_2, \text{ da } A_2 = Q_2R_2 \\
&= Q_2^{\cdot}R_1Q_1Q_2, \text{ da } A_2 = R_1Q_1 \\
&= Q_2^{\cdot}Q_1^{\cdot}AQ_1Q_2, \text{ da } A = Q_1R_1 \\
&\vdots \\
A_{k+1} &= Q_k^{\cdot} \cdots Q_1^{\cdot}AQ_1 \cdots Q_k \text{ induktiv} \\
&= (Q_1 \cdots Q_k)^{\cdot}A(Q_1 \cdots Q_k)
\end{aligned}
$$

wobei Ak die k-te Matrix aus der QR-Iteration ist. Somit ist Ak+1 einfach die durch das Produkt $Q_k^{\cdot}$ ÿ $Q_1 \cdots Q_k$ konjugierte Matrix A. Wir haben zuvor argumentiert, dass die Spalten von $Q_k^{\cdot}$ gegen die Eigenvektoren von A konvergieren. Da die Konjugation durch die Matrix der Eigenvektoren eine diagonale Matrix von Eigenwerten ergibt, wissen wir also $A_{k+1} = Q_k^{\cdot}AQ_k^{\cdot}$ wird entlang seiner Diagonale ungefähre Eigenwerte von A haben, wenn k ÿ ÿ.

**Krylov-Subraummethoden**

Unsere Begründung der QR-Iteration umfasste die Analyse der Spalten von A der Power- $^{k}$ als $k \ddot{y} \ddot{y}$ als Erweiterung Iteration. Allgemeiner ausgedrückt gilt für einen Vektorb $\ddot{y}$ Rn , Wir können die sogenannte Krylov-Matrix untersuchen

$$Kk = \begin{matrix} \ddot{y} & | & | & | & | \\ \ddot{y} & | & & & \ddot{y} \end{matrix} \begin{matrix} b & Ab & A2b & \cdot \\ & & \cdot \cdot & A\,k\ddot{y}1b & | & | & | \end{matrix}$$

Methoden zur Analyse von Kk zur Ermittlung von Eigenvektoren und Eigenwerten werden im Allgemeinen als Krylov-Unterraummethoden bezeichnet. Beispielsweise verwendet der Arnoldi-Iterationsalgorithmus die Gram-Schmidt-Orthogonalisierung, um eine orthogonale Basis {q1, . . . ,qk} für den Spaltenraum von Kk :

1. Beginnen Sie damit, dass q1 ein beliebiger Einheitsnormvektor ist
2. Für k = 2, 3, . . .

   (a) Takeak = Aqkÿ1 (b)

   Projizieren Sie die qs, die Sie bereits berechnet haben:

   $$bk = ak \ddot{y} projspan \{q1,...,qk\ddot{y}1\} ak$$

   (c) Renormieren Sie, um das nächste qk = bk/bk zu finden.

Die Matrix Qk , deren Spalten die oben gefundenen Vektoren sind, ist eine orthogonale Matrix mit demselben Spaltenraum wie Kk , und Eigenwertschätzungen können aus der Struktur von Q AQk wiederhergestellt werden . $^{k}$
Die Verwendung von Gram-Schmidt macht diese Technik instabil und das Timing wird mit zunehmendem k immer schlechter. Es sind jedoch viele Erweiterungen erforderlich, um sie durchführbar zu machen. Eine Strategie besteht beispielsweise darin, einige Arnoldi-Iterationen auszuführen, die Ausgabe zu verwenden, um eine bessere Schätzung für das anfängliche q1 zu generieren, und dann neu zu starten. Methoden dieser Klasse eignen sich für Probleme, die mehrere Eigenvektoren an einem Ende des Spektrums erfordern, ohne den vollständigen Satz zu berechnen.

## 5.5 Sensibilität und Konditionierung

Wie gewarnt, haben wir nur einige Eigenwerttechniken aus einer umfangreichen und langjährigen Literatur skizziert. Mit nahezu jeder algorithmischen Technik wurde zum Auffinden von Spektren experimentiert, von iterativen Methoden über die Wurzelfindung auf dem charakteristischen Polynom bis hin zu Methoden, die Matrizen zur parallelen Verarbeitung in Blöcke unterteilen.

Genau wie bei linearen Lösern können wir die Konditionierung eines Eigenwertproblems unabhängig von der Lösungstechnik bewerten. Diese Analyse kann helfen zu verstehen, ob ein vereinfachtes iteratives Schema zum Finden der Eigenvektoren einer gegebenen Matrix erfolgreich ist oder ob komplexere Methoden erforderlich sind. Es ist wichtig zu beachten, dass die Konditionierung eines Eigenwertproblems nicht mit der Bedingungszahl der Matrix zur Lösung von Systemen identisch ist, da es sich um separate Probleme handelt.

Angenommen, eine Matrix A hat einen Eigenvektor x mit dem Eigenwert ÿ. Die Analyse der Konditionierung des Eigenwertproblems beinhaltet die Analyse der Stabilität von x und *ÿ* gegenüber Störungen in A. Zu diesem Zweck könnten wir A durch eine kleine Matrix ÿA stören und so den Satz der Eigenvektoren ändern. Insbesondere können wir Eigenvektoren von A + ÿA als Störungen von Eigenvektoren von A schreiben, indem wir das Problem lösen

$$(A + \ddot{y}A)(x + \ddot{y}x) = (\ddot{y} + \ddot{y}\ddot{y})(x + \ddot{y}x).$$

Die Erweiterung beider Seiten ergibt:

$$Ax + A\ddot{y}x + \ddot{y}A \cdot x + \ddot{y}A \cdot \ddot{y}x = \ddot{y}x + \ddot{y}\ddot{y}x + \ddot{y}\ddot{y} \cdot x + \ddot{y}\ddot{y} \cdot \ddot{y}x$$

Unter der Annahme, dass ÿA klein ist, gehen wir davon aus3 , dass ÿx und *ÿÿ* ebenfalls klein sind. Produkte zwischen diesen Variablen sind dann vernachlässigbar, was die folgende Näherung ergibt:

$$Ax + A\ddot{y}x + \ddot{y}A \cdot x \ddot{y} \ddot{y}x + \ddot{y}\ddot{y}x + \ddot{y}\ddot{y} \cdot x$$

Da Ax = ÿx, können wir diesen Wert von beiden Seiten subtrahieren, um Folgendes zu finden:

$$A\ddot{y}x + \ddot{y}A \cdot x \ddot{y} \ddot{y}\ddot{y}x + \ddot{y}\ddot{y} \cdot x$$

Wir wenden nun einen analytischen Trick an, um unsere Ableitung zu vervollständigen. Da Ax = ÿx, wissen wir (A ÿ ÿIn×n)x =0, also ist A ÿ ÿIn×n nicht vollrangig. Die Transponierte einer Matrix ist nur dann vollrangig, wenn die Matrix vollrangig ist. Wir wissen also, dass (A ÿ ÿIn×n) = A ÿ ÿIn×n auch einen Nullraumvektor y hat. Also A y = ÿy; wir können y den linken Eigenvektor nennen, der x entspricht. Wir können unsere obige Störungsschätzung mit y linksmultiplizieren:

$$y \ (A\ddot{y}x + \ddot{y}A \cdot x) \ddot{y} y \ (\ddot{y}\ddot{y}x + \ddot{y}\ddot{y} \cdot x)$$

Da A y = ÿy ist, können wir Folgendes vereinfachen:

$$y \ \ddot{y}A \cdot x \ddot{y} \ddot{y}\ddot{y}y \ x$$

Erträge neu ordnen:

$$\ddot{y} \quad \frac{y \ (\ddot{y}A)x \ \ddot{y}\ddot{y}}{yx}$$

Angenommen, x = 1 und y = 1. Wenn wir dann Normen auf beiden Seiten nehmen, finden wir:

$$|\ddot{y}\ddot{y}| \quad \frac{\ddot{y}A2 \ |}{y \cdot x|}$$

Im Allgemeinen hängt die Konditionierung des Eigenwertproblems also – wie erwartet – von der Größe der Störung ÿA und dem Winkel zwischen den linken und rechten Eigenvektoren x und y ab. Wir können 1/x·y als ungefähre Bedingungszahl verwenden. Beachten Sie, dass x = y, wenn A symmetrisch ist, was eine Bedingungszahl von 1 ergibt; Dies spiegelt die Tatsache wider, dass die Eigenvektoren symmetrischer Matrizen orthogonal und daher maximal getrennt sind.

## 5.6 Probleme

3Diese Annahme sollte in einer strengeren Behandlung überprüft werden!

# Kapitel 6

# Einzelwertzerlegung

In Kapitel 5 haben wir eine Reihe von Algorithmen zur Berechnung der Eigenwerte und Eigenvektoren von Matrizen A ÿ Rn×n abgeleitet . Nachdem wir diese Maschinerie entwickelt haben, schließen wir unsere anfängliche Diskussion der numerischen linearen Algebra ab, indem wir eine letzte Matrixfaktorisierung ableiten und verwenden, die für jede Matrix A ÿ Rm×n existiert : die Singularwertzerlegung (SVD).

## 6.1 Ableitung der SVD

Für A ÿ Rm×n , Wir können uns die Funktion x ÿ Ax als eine Karte vorstellen , die Punkte in Rn zu Punkten in Rm führt. Aus dieser Perspektive könnten wir fragen, was dabei mit der Geometrie von Rn passiert und insbesondere welche Auswirkung A auf die Längen und Winkel zwischen Vektoren hat.

Wenn wir unseren üblichen Ausgangspunkt für Eigenwertprobleme anwenden, können wir nach der Auswirkung fragen, die A auf hat die Längen von Vektoren durch Untersuchung kritischer Punkte des Verhältnisses

$$R(x) = \frac{Axt}{X}$$

über verschiedene Werte von x. Die Skalierung von x spielt keine Rolle, da

$$R(ÿx) = \frac{A \cdot ÿx}{ÿx} = \frac{|ÿ|}{|ÿ|} \cdot \frac{Axt}{X} = \frac{Axt}{X} = R(x).$$

Somit können wir unsere Suche auf x mit x = 1 beschränken. Da R(x) ÿ 0 ist, können wir außerdem stattdessen [R(x)]2 = Ax 2 = x A Ax betrachten . Wie wir jedoch in den vorherigen Kapiteln gezeigt haben, sind kritische Punkte von x A Ax unter der Bedingung x = 1 genau die Eigenvektoren xi , die A Axi = ÿixi erfüllen ; Beachten Sie ÿi ÿ 0 und xi · xj = 0, wenn i = j, da AA symmetrisch und positiv semidefinit ist.

Basierend auf unserer Verwendung der Funktion R ist die {xi} -Basis eine sinnvolle Basis für die Untersuchung der geometrischen Effekte von A. Zurück zu diesem ursprünglichen Ziel: Definieren Sie yi ÿ Axi . Wir können eine zusätzliche Beobachtung zu yi machen , die eine noch stärkere Eigenwertstruktur offenbart:

$$\begin{aligned} ÿiyi &= ÿi \cdot Axi \text{ nach Definition von } yi \\ &= A(ÿixi) \\ &= A(A\,Axi) \text{ , da } xi \text{ ein Eigenvektor von } AA \text{ ist} \\ &= (AA)(Axi) \text{ durch Assoziativität} \\ &= (AA)yi \end{aligned}$$

Wir haben also zwei Fälle:

1. Wenn ÿi = 0, dann ist yi = 0. In diesem Fall ist xi ein Eigenvektor von AA und yi = Axi ist AAxi = ein entsprechender Eigenvektor von AA mit yi = Axi = Axi ÿ ÿixi.

2. Wenn ÿi = 0, ist yi =0.

Ein identischer Beweis zeigt, dass, wenn y ein Eigenvektor von AA ist, x ÿ A y entweder Null oder ein Eigenvektor von AA mit demselben Eigenwert ist.

Nehmen Sie k als die Anzahl der oben diskutierten streng positiven Eigenwerte ÿi > 0 an. Mit unserer obigen Konstruktion können wir x1, . . . ,xk ÿ Rn als Eigenvektoren von AA und entsprechende Eigenvektoren y1, . . . ,yk ÿ Rm von AA so dass

$$A\,Axi = \ddot{y}ixi$$

$$AAyi = \ddot{y}iyi$$

für Eigenwerte ÿi > 0; hier normalisieren wir so, dass xi = yi = 1 für alle i. Der traditionellen Notation folgend, können wir Matrizen V¯ ÿ Rn×k und U¯ ÿ Rm×k definieren, deren Spalten xi bzw. yi sind .

Wir können die Auswirkung dieser neuen Basismatrizen auf A untersuchen. Nehmen Sie ei als den i-ten Standardbasisvektor. Dann,

$$\begin{aligned}
U^{-}AV^{-}ei &= U^{-}Axi \text{ nach Definition von } V^{-} \\
&= \frac{1}{\ddot{y}i}U^{-}A(\ddot{y}ixi) \text{ , da wir } \ddot{y}i > 0 \text{ } \ddot{y}i \text{ } 1 \text{ angenommen} \\
&= \frac{1}{\ddot{y}i}U^{-}A(A\,Axi) \text{ , da xi ein Eigenvektor von AA } \ddot{y}i \text{ } 1 \text{ ist haben} \\
&= \frac{1}{\ddot{y}i}U^{-}(AA)Axi \text{ durch Assoziativität } \ddot{y}i \text{ } 1 \\
&= \frac{1}{\ddot{y}i}U^{-}(AA)yi \text{ , da wir neu skaliert haben, sodass } yi = 1 \text{ } \ddot{y} \\
&\ddot{y}iU^{-}yi \text{ , da } AAyi = \ddot{y}iyi \\
&= \ddot{y}iei
\end{aligned}$$

Nehmen Sie $\ddot{y}^{-} = \text{diag}(\ddot{y}\ddot{y}1, \ldots, \ddot{y}\ddot{y}k)$. Dann zeigt die obige Ableitung, dass $U^{-}AV^{-} = \ddot{y}^{-}$ .

Vervollständigen Sie die Spalten von U¯ und V¯ zu U ÿ Rm×m und V ÿ Rn×n , indem Sie Orthonormalvektoren xi undyi mit A Axi =0 bzw. AAyi =0 hinzufügen. In diesem Fall ist es einfach, UAVei =0 und/oder UAV =0 $e_i$ anzuzeigen . Also, wenn wir nehmen

$$\ddot{y}ij \ddot{y} \begin{cases} \ddot{y}\ddot{y}i & i = j \text{ und } i \ddot{y} k \\ 0 & \text{sonst} \end{cases}$$

Dann können wir unsere vorherige Beziehung erweitern, um UAV = ÿ oder eine gleichwertige Gleichung zu zeigen

$$A = U\ddot{y}V .$$

Diese Faktorisierung ist genau die Singularwertzerlegung (SVD) von A. Die Spalten von U überspannen den Spaltenraum von A und werden dessen linke Singulärvektoren genannt; Die Spalten von V überspannen seinen Zeilenraum und sind die rechten singulären Vektoren. Die Diagonalelemente ÿi von ÿ sind die Singulärwerte von A; Normalerweise sind sie so sortiert, dass ÿ1 ÿ ÿ2 ÿ · · · ÿ 0. Sowohl U als auch V sind orthogonale Matrizen.

Die SVD liefert eine vollständige geometrische Charakterisierung der Wirkung von A. Da U und V orthogonal sind, können sie als Rotationsmatrizen betrachtet werden; Als Diagonalmatrix skaliert ÿ einfach einzelne Koordinaten. Somit sind alle Matrizen A ÿ Rm×n eine Zusammensetzung aus einer Rotation, einer Skala und einer zweiten Rotation.

### 6.1.1 Berechnung der SVD

Denken Sie daran, dass die Spalten von V einfach die Eigenvektoren von AA sind und daher mit den im vorherigen Kapitel beschriebenen Techniken berechnet werden können. Da A = UÿV ist, wissen wir AV = Uÿ. Somit sind die Spalten von U, die singulären Werten ungleich Null in ÿ entsprechen, einfach normalisierte Spalten von AV; Die verbleibenden Spalten erfüllen AAui = 0, was mithilfe der LU-Faktorisierung gelöst werden kann.

Diese Strategie ist keineswegs der effizienteste oder stabilste Ansatz zur Berechnung der SVD, funktioniert aber für viele Anwendungen einigermaßen gut. Wir werden speziellere Ansätze zur Ermittlung der SVD weglassen, beachten jedoch, dass es sich bei vielen um einfache Erweiterungen der Potenziteration und anderer Strategien handelt, die wir bereits behandelt haben und die ohne explizite Bildung von AA oder AA funktionieren.

## 6.2 Anwendungen des SVD

Den Rest dieses Kapitels widmen wir der Vorstellung vieler Anwendungen der SVD. Die SVD taucht sowohl in der Theorie als auch in der Praxis der numerischen linearen linearen Algebra unzählige Male auf, und ihre Bedeutung kann kaum überbewertet werden.

### 6.2.1 Lösung linearer Systeme und der Pseudoinversen

Im Spezialfall, in dem A ÿ Rn×n quadratisch und invertierbar ist, ist es wichtig zu beachten, dass die SVD zur Lösung des linearen Problems Ax =b verwendet werden kann. Insbesondere gilt UÿV x =b, oder

$$x = V\ddot{y}\ \ddot{y}1U\ b.$$

In diesem Fall ist ÿ eine quadratische Diagonalmatrix, also ÿ $^{\ddot{y}1}$ ist einfach die Matrix, deren diagonale Einträge sind *1/ÿi* .

Die Berechnung der SVD ist weitaus aufwendiger als die meisten linearen Lösungstechniken, die wir in Kapitel 2 vorgestellt haben, daher ist diese erste Beobachtung hauptsächlich von theoretischem Interesse. Nehmen wir allgemeiner an, wir möchten eine Lösung der kleinsten Quadrate für Ax ÿ b finden, wobei A ÿ Rm×n nicht unbedingt quadratisch ist. Aus unserer Diskussion der Normalgleichungen wissen wir, dass x A Ax = A b erfüllen muss. Bisher haben wir den Fall, dass A „kurz“ oder „unterbestimmt“ ist, also wenn A mehr Spalten als Zeilen hat, größtenteils außer Acht gelassen. In diesem Fall ist die Lösung der Normalgleichungen nicht eindeutig.

Um alle drei Fälle abzudecken, können wir ein Optimierungsproblem der folgenden Form lösen:

$$\begin{aligned} &x \text{ minimieren} && {}^{2} \\ &\text{so dass } A\ Ax = A\ b \end{aligned}$$

In Worten ausgedrückt verlangt diese Optimierung, dass x die Normalgleichungen mit der kleinstmöglichen Norm erfüllt.
Schreiben wir nun A = UÿV . Dann,

$$\begin{aligned} AA &= (U\ddot{y}V)\,(U\ddot{y}V) \\ &= V\ddot{y}\,U\,U\ddot{y}V \text{ da } (AB) = BA \\ &= V\ddot{y}\,\ddot{y}V, \text{ da U orthogonal ist} \end{aligned}$$

Daher ist die Frage, dass A Ax = A b ist, dasselbe wie die Frage

$$V\ddot{y}\,\ddot{y}V\,x = V\ddot{y}U\,b$$

$$\text{Oder äquivalent: } \ddot{y}y = d$$

wenn wir d ÿ Ub andy ÿ V x annehmen . Beachten Sie, dass y = x, da U orthogonal ist, sodass unsere Optimierung zu Folgendem führt:

$$\begin{aligned} &\text{Minimiere y so, } {}^{2} \\ &\text{dass } \ddot{y}y = d \end{aligned}$$

Da ÿ jedoch diagonal ist, besagt die Bedingung ÿy = d einfach ÿiyi = di ; Wenn also ÿi = 0, müssen wir yi = di/ ÿi haben . Wenn ÿi = 0, gibt es keine Einschränkung für yi . Da wir also y minimieren, können wir genauso gut yi = 0 annehmen. Mit anderen Worten, die Lösung für diese Optimierung ist y = ÿ +d, wobei ÿ + ÿ Rn×m hat die folgende Form:

$$^{+}_{\ddot{y}\,ij^{\ddot{y}}} \qquad \begin{matrix} 1/\ddot{y}i \text{ i = j, ÿi = 0 und sonst i ÿ k} \\ 0 \end{matrix}$$

Diese Form wiederum ergibt x = Vy = Vÿ +d = Vÿ +Ub.
Aus dieser Motivation heraus treffen wir folgende Definition:

**Definition 6.1** (Pseudoinvers). Die Pseudoinverse von A = UÿV ÿ Rm×n ist A+ ÿ Vÿ +U ÿ Rn×m.

Unsere obige Ableitung zeigt, dass die Pseudoinverse von A die folgenden Eigenschaften aufweist:

- Wenn A quadratisch und invertierbar ist, ist $A^{+} = A^{\ddot{y}1}$ .
- Wenn A überbestimmt ist, ergibt A +b die Lösung der kleinsten Quadrate für Ax ÿb.
- Wenn A unterbestimmt ist, ergibt A +b die Lösung der kleinsten Quadrate für Ax ÿ b mit minimalem (Euklidische) Norm.

Auf diese Weise können wir endlich die unterbestimmten, vollständig bestimmten und überbestimmten Fälle von Ax ÿb vereinheitlichen.

### 6.2.2 Zerlegung in äußere Produkte und Approximationen mit niedrigem Rang

Entwickeln wir das Produkt A = UÿV , Es ist leicht zu zeigen, dass diese Beziehung Folgendes impliziert:

$$A = \ddot{y}_{i=1} \ddot{y}iuiv\,i\,,$$

wobei ÿ min{m, n} und ui und vi die i-ten Spalten von U bzw. V sind. Unsere Summe geht nur zu min{m, n}, da wir wissen, dass die verbleibenden Spalten von U oder V durch ÿ auf Null gesetzt werden.

Dieser Ausdruck zeigt, dass jede Matrix als Summe der äußeren Produkte von zerlegt werden kann Vektoren:

**Definition 6.2** (Äußeres Produkt). Das äußere Produkt von u ÿ Rm und v ÿ Rn ist die Matrix u ÿ v ÿ uv ÿ Rm×n .

Angenommen, wir möchten das Produkt Ax schreiben. Dann könnten wir stattdessen schreiben:

$$Ax = \ddot{y} \sum_{i=1} \ddot{y}iuiv\, i\, x$$

$$= \ddot{y} \sum_{i=1} \ddot{y}iui(v \quad X)$$

$$= \ddot{y} \sum_{i=1} \ddot{y}i(vi \cdot x)ui \text{ , da } x \cdot y = xy$$

Das Anwenden von A auf x ist also dasselbe wie das lineare Kombinieren der ui- Vektoren mit Gewichten ÿi(vi · x). Diese Strategie zur Berechnung von Ax kann zu erheblichen Einsparungen führen, wenn die Anzahl der ÿi -Werte ungleich Null relativ klein ist. Darüber hinaus können wir kleine Werte von ÿi ignorieren und diese Summe effektiv kürzen, um Ax mit weniger Aufwand anzunähern.

In ähnlicher Weise können wir aus §6.2.1 die Pseudoinverse von A wie folgt schreiben:

$$A_{+} = \ddot{y} \sum_{\ddot{y}i=0} \frac{viu\, i}{\ddot{y}i}.$$

Offensichtlich können wir den gleichen Trick anwenden, um A +x auszuwerten , und tatsächlich können wir A +x annähern, indem wir nur diejenigen Terme in der Summe auswerten, für die ÿi relativ klein ist. In der Praxis berechnen wir die Singulärwerte ÿi als Quadratwurzeln der Eigenwerte von AA oder AA, und Methoden wie Potenziteration können verwendet werden, um einen Teilsatz anstelle eines vollständigen Satzes von Eigenwerten aufzudecken. Wenn wir also eine Reihe von Problemen der kleinsten Quadrate Axi ÿ bi für verschiedene bi lösen müssen und mit einer Näherung für xi zufrieden sind , kann es sinnvoll sein, zunächst die kleinsten ÿi- Werte zu berechnen und die obige Näherung zu verwenden. Diese Strategie vermeidet außerdem, jemals die vollständige A + -Matrix berechnen oder speichern zu müssen, und kann genau sein, wenn A einen großen Bereich singulärer Werte aufweist.

Wenn wir zu unserer ursprünglichen Notation A = UÿV, zurückkehren, zeigt unser obiges Argument effektiv, dass eine potenziell nützliche Näherung für A A˜ ÿ Uÿ˜ V ist, wobei ÿ˜ kleine Werte von ÿ auf Null rundet. Es lässt sich leicht überprüfen, dass der Spaltenraum von A˜ eine Dimension hat, die der Anzahl der Werte ungleich Null auf der Diagonale von ÿ˜ entspricht. Tatsächlich handelt es sich bei dieser Näherung nicht um eine Ad-hoc-Schätzung, sondern um die Lösung eines schwierigen Optimierungsproblems, das von dem folgenden berühmten Autor veröffentlicht wurde Satz (ohne Beweis angegeben):

**Satz 6.1** (Eckart-Young, 1936). Angenommen, A wird aus A = UÿV erhalten, indem alle außer A ÿ A˜Fro und A ÿ A˜2 größten Singulärwerten ÿi von A auf Null. Dann minimiert A beide gekürzt werden , abhängig von den k Einschränkungen, dass der Spaltenraum von A höchstens die Dimension k hat.

### 6.2.3 Matrixnormen

Die Konstruktion der SVD ermöglicht es uns auch, zu unserer Diskussion der Matrixnormen aus §3.3.1 zurückzukehren. Erinnern Sie sich beispielsweise daran, dass wir die Frobenius-Norm von A als definiert haben

$$\|A\|_{Her}^2 \ddot{y} \ddot{y} \|a_{ij}\|_2^2.$$

Wenn wir A = UÿV schreiben , Wir können diesen Ausdruck vereinfachen:

$$\begin{aligned}
\|A\|_{Her}^2 &= \ddot{y}_J \|Ae_j\|_2^2 \text{ da dieses Produkt die j-te Spalte von A ist} \\
&= \ddot{y}_J \|U\ddot{y}V e_j\|_2^2, \text{ ersetzt die SVD} \\
&= \ddot{y}_J \|\ddot{y}Ve_j\|_2^2 \text{ seit } \|x\|_2^2 = xx \text{ und U ist orthogonal} \\
&= \|\ddot{y}V\|_{Hin}^2 \text{ und her nach der gleichen Logik} \\
&= \|V\ddot{y}\|_{Her}^2 \text{ da eine Matrix und ihre Transponierte dieselbe Frobenius-Norm haben} \\
&= \ddot{y}_J \|V\ddot{y}e_j\|_2^2 = \ddot{y}_J \ddot{y}_j^2 \|Ve_j\|_2^2 \text{ durch Diagonalität von } \ddot{y} \\
&= \ddot{y}_J \ddot{y}_j^2 \text{ da V orthogonal ist}
\end{aligned}$$

Somit ist die Frobenius-Norm von A ÿ Rm×n die Summe der Quadrate seiner Singulärwerte.

Dieses Ergebnis ist von theoretischem Interesse, aber praktisch gesehen ist die grundlegende Definition der Frobe-nius-Norm bereits einfach zu bewerten. Noch interessanter ist, dass die induzierte Zwei-Norm von A gegeben ist durch

$$\|A\|_2^2 = \max\{\ddot{y} : \text{es existiert } x \ddot{y} \mathbf{R}^N \text{ mit } A^A x = \ddot{y}x\}.$$

Nachdem wir nun Eigenwertprobleme untersucht haben, erkennen wir, dass dieser Wert die Quadratwurzel des größten Eigenwerts von AA oder ein Äquivalent ist

$$\|A\|_2 = \max\{\ddot{y}_i\}.$$

Mit anderen Worten: Wir können die Zweinorm von A direkt aus seinen Eigenwerten ablesen.

Denken Sie auch daran, dass die Bedingungszahl von A durch cond A = A2A ÿ12 gegeben ist. Bei uns müssen Ableitung von A +, die Singulärwerte von A $\ddot{y}_1$ die Kehrwerte der singulären Werte von A sein. Kombinieren Sie dies mit unserer Vereinfachung der A2- Erträge:

$$\text{cond } A = \frac{\ddot{y}_{max}}{\ddot{y}_{min}}.$$

Dieser Ausdruck liefert eine Strategie zur Bewertung der Konditionierung von A. Natürlich erfordert die Berechnung von ÿmin die Lösung von Systemen Ax = b, ein Prozess, der an sich unter einer schlechten Konditionierung von A leiden kann; Wenn dies ein Problem darstellt, kann die Konditionierung mithilfe verschiedener Näherungen der Singulärwerte von A begrenzt und angenähert werden.

### 6.2.4 Das Procrustes-Problem und die Ausrichtung

Viele Techniken in der Computer Vision beinhalten die Ausrichtung dreidimensionaler Formen. Angenommen, wir haben einen dreidimensionalen Scanner, der zwei Punktwolken desselben starren Objekts aus verschiedenen Ansichten sammelt. Eine typische Aufgabe könnte darin bestehen, diese beiden Punktwolken in einem einzigen Koordinatenrahmen auszurichten.

Da das Objekt starr ist, erwarten wir, dass es eine Rotationsmatrix R und eine Translation t ÿ R3 gibt , so dass durch Drehen der ersten Punktwolke um R und anschließendes Verschieben um t die beiden Datensätze ausgerichtet werden. Unsere Aufgabe ist es, diese beiden Objekte abzuschätzen.

Wenn sich die beiden Scans überschneiden, kann der Benutzer oder ein automatisiertes System n entsprechende Punkte markieren, die zwischen den beiden Scans übereinstimmen; wir können diese in zwei Matrizen X1, X2 ÿ R3×n speichern . Dann erwarten wir für jede Spalte x1i von X1 und x2i von X2 Rx1i +t = x2i . Wir können eine Energiefunktion schreiben, die misst, inwieweit diese Beziehung zutrifft:

$$E \ddot{y} \ddot{y} \quad Rx1i + t \ddot{y} x2i \quad ^2.$$

Wenn wir R fixieren und bezüglich tot minimieren, wird die Optimierung von E offensichtlich zu einem Problem der kleinsten Quadrate. Nehmen wir nun an, wir optimieren für R mit festem Wert. Dies ist dasselbe wie das Minimieren $\overset{T}{2}_{Her,}$ von RX1 ÿ X, wobei die $\overset{T}{2}$ sind die von X2 übersetzten Bytes, vorausgesetzt, dass R eine 3 × 3-Rotationsmatrix ist, Spalten von X, also RR = I3×3, sind. Dies ist als orthogonales Procrustes-Problem bekannt.

Um dieses Problem zu lösen, führen wir die Spur einer quadratischen Matrix wie folgt ein:

**Definition 6.3** (Trace). Die Spur von A ÿ Rn×n ist die Summe ihrer Diagonalen:

$$tr(A) \ddot{y} \ddot{y} \quad aii.$$

Es ist einfach zu überprüfen, ob $A \quad \overset{2}{Her} = tr(A\ A)$. Somit können wir E wie folgt vereinfachen:

$$\begin{aligned} RX1 \ddot{y} X \quad \overset{T}{2} \quad \overset{2}{Her} &= tr((RX1 \ddot{y} X \quad \overset{\cdot}{2}) (RX1 \ddot{y} X \overset{\cdot}{2}^T)) \\ &= tr(X\ 1 \quad X1 \ddot{y} X \quad {}_1 RX \quad \overset{T}{2} \overset{\cdot}{2} \ddot{y} X^T \quad RX1 + X \quad \overset{T}{2} X2) \\ &= konst. \ddot{y} 2tr(X\ 2 \dot{R} X1) \\ &\quad da\ tr(A + B) = tr\ A + tr\ B\ und\ tr(A\ ) = tr(A) \end{aligned}$$

Daher möchten wir tr(X RX1) mit RR = $I3\overset{T}{\ddot{y}}3$ maximieren . In den Übungen beweisen Sie, dass tr(AB) = tr(BA). Somit lässt sich unser Ziel leicht zu tr(RC) mit C ÿ X1X 2 vereinfachen . Wenn wir die SVD anwenden und C = UÿV $\overset{T}{z}$erlegen, können wir noch mehr vereinfachen:

$$\begin{aligned} tr(RC) = tr(RU\ddot{y}V\ )\ &per\ Definition = tr((V \\ &RU)\ddot{y})\ da\ tr(AB) = tr(BA) = tr(R\tilde{}\ddot{y})\ wenn\ wir\ R\tilde{} = \\ &V\ RU\ definieren\ ,\ was\ ebenfalls\ orthogonal\ ist = \ddot{y}\ \ddot{y}ir\tilde{}ii\ ,\ da\ \ddot{y}\ diagonal\ ist \end{aligned}$$

Da R˜ orthogonal ist, haben alle seine Spalten eine Einheitslänge. Dies impliziert, dass r˜ii ÿ 1, da sonst die Norm der Spalte i zu groß wäre. Da ÿi ÿ 0 für alle i ist, zeigt dieses Argument, dass wir tr(RC) maximieren können , indem wir R˜ = I3×3 annehmen. Wenn wir unsere Ersetzungen rückgängig machen, ergibt sich R = VRU˜ = VU.

Allgemeiner gesagt haben wir Folgendes gezeigt:

**Satz 6.2** (Orthogonaler Prokrustes). Die orthogonale Matrix R minimiert RX ÿ Y VU, wobei SVD auf $^2$ ist gegeben durch den Faktor XY = UÿV angewendet wird.

Zurück zum Ausrichtungsproblem: Eine typische Strategie ist ein alternierender Ansatz:

1. Fixiere R und minimiere E bezüglich tot.

2. Fixieren Sie das Resultierende und minimieren Sie E in Bezug auf R unter der Voraussetzung RR = I3×3.

3. Kehren Sie zu Schritt 1 zurück.

Die Energie E nimmt mit jedem Schritt ab und konvergiert somit zu einem lokalen Minimum. Da wir nie gleichzeitig optimieren und R, können wir nicht garantieren, dass das Ergebnis der kleinstmögliche Wert von E ist, aber in der Praxis funktioniert diese Methode gut.

## 6.2.5 Hauptkomponentenanalyse (PCA)

Erinnern Sie sich an den Aufbau aus §5.1.1: Wir möchten eine niedrigdimensionale Näherung für eine Menge von Datenpunkten finden, die wir in einer Matrix X ÿ Rn×k für k Beobachtungen in n Dimensionen speichern können. Zuvor haben wir gezeigt, dass, wenn uns nur eine einzige Dimension erlaubt ist, die bestmögliche Richtung durch den dominanten Eigenvektor von XX gegeben ist.

Angenommen, wir dürfen stattdessen auf die Spanne von d Vektoren mit d ÿ min{k, n} projizieren und möchten diese Vektoren optimal auswählen. Wir könnten sie in einer n × d-Matrix C schreiben; Da wir Gram-Schmidt auf jede Menge von Vektoren anwenden können, können wir annehmen, dass die Spalten von C orthonormal sind, was CC = Id×d zeigt . Da C orthonormale Spalten hat, ist die Projektion von X auf den Spaltenraum von C durch die Normalengleichungen durch CCX gegeben.

In diesem Aufbau möchten wir X ÿ CCXFro unter der Bedingung CC = Id×d minimieren . Wir können vereinfachen Unser Problem einigermaßen:

$$\begin{aligned} \|X - CCX\|_{Her}^2 &= \operatorname{tr}((X - CCX)(X - CCX)) \text{ da } A = \operatorname{tr}(X \quad \|A\|_{Her}^2 = \operatorname{tr}(A A) \\ &\quad X - 2X CCX + X CCCCX) = \text{const.} - \\ &\quad \operatorname{tr}(X CCX) \text{ da } CC = Id\times d \\ &= -\|CX\|_{Her}^2 + \text{Konst.} \end{aligned}$$

Wir können also gleichbedeutend mit CX den $^2$Her; Für Statistiker zeigt dies, wann die Zeilen von X vorhanden sind Mittelwert Null maximieren, sodass wir die Varianz der Projektion C X maximieren möchten.

Nehmen wir nun an, wir faktorisieren X = UÿV . Dann möchten wir C UÿV Fro ÿ˜ CFro durch $= \|C\tilde{}\|_{Fro} =$ Orthogonalität von V maximieren, wenn wir C˜ = CU annehmen. Wenn die Elemente von C˜ c˜ij sind, dann ergibt die Erweiterung dieser Norm

$$\|C\tilde{}\|_{Her}^2 = \sum_{ij} \sigma_i^2 \tilde{c}_{ij}^2.$$

Aufgrund der Orthogonalität der Spalten von C˜ wissen wir, dass $\sum_i \sigma_i^2 \tilde{c}_{ij}^2 = 1$ für alle j und da C˜ weniger als n Spalten haben kann, ist $\sum_i \sigma_i^2 \tilde{c}_{ij}^2 \le 1$. Somit ist der Koeffizient neben $\sigma_1^2$ in der obigen Summe höchstens 1, wenn wir also klar sind Das Maximum wird erreicht, indem unsere Koordinatenänderung , man die Spalten von C˜ so sortiert, dass $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n$ bee1, . . . ,ed . Wenn wir rückgängig machen, sehen wir, dass unsere Wahl von C die ersten d Spalten von U sein sollte.

Wir haben gezeigt, dass die SVD von X zur Lösung eines solchen Hauptkomponentenanalyseproblems (PCA) verwendet werden kann. In der Praxis werden die Zeilen von X normalerweise so verschoben, dass sie vor der SVD den Mittelwert Null haben; Wie in Abbildung NUMBER gezeigt, zentriert dies den Datensatz um den Ursprung und liefert aussagekräftigere PCA-Vektoren ui .

## 6.3 Probleme

# Teil III

# Nichtlineare Techniken

# Kapitel 7

# Nichtlineare Systeme

So sehr wir uns auch bemühen, es ist einfach nicht möglich, alle Gleichungssysteme in dem linearen Rahmen auszudrücken, den wir in den letzten Kapiteln entwickelt haben. Es ist kaum notwendig, die Verwendung von Logarithmen, Exponentialfunktionen, trigonometrischen Funktionen, Absolutwerten, Polynomen usw. in praktischen Problemen zu motivieren, aber außer in einigen Sonderfällen ist keine dieser Funktionen linear. Wenn diese Funktionen auftreten, müssen wir einen allgemeineren, wenn auch weniger effizienten Maschinensatz einsetzen.

## 7.1 Probleme mit einer Variablen

Wir beginnen unsere Diskussion mit der Betrachtung von Problemen einer einzelnen Skalarvariablen. Insbesondere möchten wir für eine gegebene Funktion f(x) : **R** ÿ R Strategien entwickeln, um Punkte x ÿ **R** zu finden , so dass f(x ÿ ) = 0; wir nennen xa Wurzel von f . Probleme mit einer Variablen in der linearen Algebra sind nicht besonders = b/ interessant; Schließlich können wir die Gleichung ax ÿ b = 0 in geschlossener Form als x a. Die Lösung nichtlineare Gleichung lösen, $2 + e$ eines cos y ÿ 3 = 0 ist jedoch weitaus weniger offensichtlich (übrigens die Lösung). z. B. y ist y = ±1,30246 . . .).

### 7.1.1 Charakterisierung von Problemen

Wir können nicht mehr davon ausgehen, dass f linear ist, aber ohne Annahmen über seine Struktur werden wir bei der Lösung von Systemen mit einer Variablen wahrscheinlich keine Fortschritte machen. Beispielsweise kann ein Löser garantiert nicht die Nullstellen von f(x) finden, die durch gegeben sind

$$f(x) = \begin{cases} \ddot{y}1 \; x \ddot{y} \; 0 \\ 1 \; x > 0 \end{cases}$$

Oder schlimmer:

$$f(x) = \begin{cases} \ddot{y}1 \; x \ddot{y} \; \mathbf{Q} \\ 1 \; \text{sonst} \end{cases}$$

Diese Beispiele sind in dem Sinne trivial, dass ein rationaler Kunde einer Root-Findungssoftware in diesem Fall wahrscheinlich nicht erwarten würde, dass dies gelingt, aber weit weniger offensichtliche Fälle sind nicht viel schwieriger konstruieren.

Aus diesem Grund müssen wir einige „regulierende“ Annahmen darüber hinzufügen, dass f einen Halt in die Möglichkeit bietet, Techniken zur Wurzelfindung zu entwerfen. Typische Annahmen dieser Art sind unten aufgeführt, in aufsteigender Reihenfolge ihrer Stärke:

- Stetigkeit: Eine Funktion f ist stetig, wenn sie gezeichnet werden kann, ohne einen Stift anzuheben; mehr formal ist f stetig, wenn die Differenz f(x) ÿ f(y) für x ÿ y verschwindet.
- Lipschitz: Eine Funktion f ist Lipschitz-stetig, wenn es eine Konstante C mit | gibt f(x) ÿ f(y)| ÿ C|x ÿ y|; Lipschitz-Funktionen müssen nicht differenzierbar sein, sind aber in ihren Änderungsraten begrenzt.
- Differenzierbarkeit: Eine Funktion f ist differenzierbar, wenn ihre Ableitung f für alle x existiert.
- dass k, wenn es k Ableitungen gibt und f differenzierbar ist und jede dieser k Ableitungen stetig ist; • C: Eine Funktion ist C ÿ und zeigt an, $C^{ÿ}$ f existieren und stetig sind.

Indem wir immer stärkere Annahmen über f hinzufügen, können wir effektivere Algorithmen entwerfen, um f(x ÿ ) = 0 zu lösen. Wir werden diesen Effekt durch die Betrachtung einiger nachfolgender Algorithmen veranschaulichen.

## 7.1.2 Kontinuität und Halbierung

Nehmen wir an, alles, was wir über f wissen, ist, dass es stetig ist. In diesem Fall können wir einen intuitiven Satz aus der Standardrechnung mit einzelnen Variablen formulieren:

**Satz 7.1** (Zwischenwertsatz). Angenommen, f : [a, b] ÿ **R** ist stetig. Angenommen, f(x) < u < f(y). Dann existiert z zwischen x und y, so dass f(z) = u.

Mit anderen Worten: Die Funktion f muss jeden Wert zwischen f(x) und f(y) erreichen.

Angenommen, wir erhalten als Eingabe die Funktion f sowie zwei Werte und r, so dass f() · f(r) < 0; Beachten Sie, dass dies bedeutet, dass f() und f(r) entgegengesetzte Vorzeichen haben. Dann wissen wir durch den Zwischenwertsatz, dass es irgendwo zwischen und r eine Wurzel von f gibt! Dies bietet eine offensichtliche Halbierungsstrategie zum Finden von x $^{ÿ}$:

1. Berechnen Sie c = +r/2.
2. Wenn f(c) = 0, gib x zurück $^{ÿ}$ = c.
3. Wenn f() · f(c) < 0, nehme r ÿ c. Ansonsten nimm ÿ c.
4. Wenn |r ÿ | < ÿ, gib x zurück $^{ÿ}$ ÿ c.
5. Gehen Sie zurück zu Schritt 1

Diese Strategie teilt einfach das Intervall [,r] iterativ in zwei Hälften, wobei jedes Mal die Seite beibehalten wird, auf der bekanntermaßen eine Wurzel existiert. Nach dem Zwischenwertsatz konvergiert es eindeutig bedingungslos, in dem Sinne, dass, solange f()· f(r) < 0, schließlich beide und r garantiert zu einer gültigen Wurzel x konvergieren $^{ÿ}$.

## 7.1.3 Analyse der Wurzelfindung

Die Bisektion ist die einfachste, aber nicht unbedingt die effektivste Technik zur Wurzelfindung. Wie bei den meisten Eigenwertmethoden ist die Halbierung von Natur aus iterativ und liefert möglicherweise nie eine exakte Lösung. Wir können $X^{ÿ}$ jedoch fragen, wie nahe der Wert ck von c in der k-ten Iteration an der Wurzel x ÿ liegt , die wir berechnen möchten. Diese Analyse bietet eine Grundlage für den Vergleich mit anderen Methoden.

Nehmen wir im Allgemeinen an, dass wir eine Fehlergrenze Ek festlegen können , so dass die Schätzung xk der
X Wurzel ÿ während der k-ten Iteration einer Wurzelfindungsmethode |xk ÿ x ÿ | erfüllt < Ek . Offensichtlich stellt jeder Algorithmus mit Ek ÿ 0 ein konvergentes Schema dar; Die Konvergenzgeschwindigkeit kann jedoch durch die Geschwindigkeit charakterisiert werden, mit der sich Ek 0 nähert.

Beispielsweise in der Halbierung, da sowohl der ck- als ÿ Sie liegen im Intervall [k ,rk ], einer oberen Grenze von. auch der x-Fehler durch Ek ÿ | gegeben sind rk ÿ k |. Da wir das Intervall bei jeder Iteration halbieren, wissen wir Ek+1 = 1/2Ek . Da Ek+1 in Ek linear ist , sagen wir, dass die Halbierung eine lineare Konvergenz aufweist.

### 7.1.4 Fixpunktiteration

Es ist garantiert, dass die Halbierung für jede stetige Funktion f gegen eine Wurzel konvergiert. Wenn wir jedoch mehr über f wissen, können wir Algorithmen formulieren, die schneller konvergieren können.

Nehmen wir als Beispiel an, wir möchten x finden ÿ mit g(x ÿ ) = x ÿ ; Natürlich ist dieser Aufbau äquivalent zum Wurzelfindungsproblem, da das Lösen von f(x) = 0 dasselbe ist wie das Lösen von f(x) + x = x.
Als zusätzliche Information könnten wir jedoch auch wissen, dass g Lipschitz mit der Konstante C < 1 ist.

Das System g(x) = x legt eine mögliche Strategie nahe, die wir annehmen könnten:

1. Nehmen Sie x0 als eine anfängliche Schätzung einer Wurzel.
2. Iteriere xk = g(xkÿ1).

Wenn diese Strategie konvergiert, ist das Ergebnis eindeutig ein Fixpunkt von g, der die oben genannten Kriterien erfüllt.

Glücklicherweise stellt die Lipschitz-Eigenschaft sicher, dass diese Strategie zu einer Wurzel konvergiert, sofern eine solche existiert. Wenn wir Ek = |xk ÿ x ÿ | annehmen , dann haben wir die folgende Eigenschaft:

$$
\begin{aligned}
Ek &= |xk \ddot{y} x \ddot{y} \\
&\quad | = |g(xk\ddot{y}1) \ddot{y} g(x \ddot{y} )| \text{ durch Design des iterativen Schemas und Definition } \ddot{y} \\
&\quad \text{von } x \ddot{y} C|xk\ddot{y}1 \ddot{y} x \ddot{y} | \text{ da g Lipschitz ist} \\
&\quad = CEk\ddot{y}1
\end{aligned}
$$

Die induktive Anwendung dieser Aussage zeigt, dass Ek ÿ C k |E0| ÿ 0 als k ÿ ÿ. Also Festkomma-Iteration zum gewünschten x konvergiert ÿ !

Wenn g tatsächlich Lipschitz mit der Konstante C < 1 in einer Umgebung [x ÿ ÿ ÿ, x ÿ + ÿ], dann konvergiert die Fixpunktiteration, solange x0 in diesem Intervall gewählt wird. Dies ist wahr, da unser obiger Ausdruck für Ek zeigt, dass er bei jeder Iteration kleiner wird. und |g (x ÿ )| < 1. Aufgrund

Ein wichtiger Fall *tritt* ein , wenn g C ist und man 1 der Stetigkeit von g in diesem Fall wir + ÿ] mit |g (x)| < 1 ÿ *ÿ* für weiß, dass es eine Umgebung N = [ x ÿ ÿ ÿ, ÿ jedes x ÿ N,

$$
\begin{aligned}
|g(x) \ddot{y} g(y)| &= |g(\ddot{y})| \cdot |x \ddot{y} y| \text{ nach dem Mittelwertsatz der Grundrechnung für ein } \ddot{y} \ddot{y} [x, y] \\
&< (1 \ddot{y} \ddot{y})|x \ddot{y} y|
\end{aligned}
$$

Dies zeigt, dass g Lipschitz mit der Konstante 1 ÿ *ÿ* < 1 in N ist. Wenn g also stetig differenzierbar ist und g (x ÿ ) < 1, konvergiert die Festpunktiteration gegen x ÿ , wenn die anfängliche Schätzung x0 nahe beieinander liegt.

---

1Diese Aussage ist schwer zu verstehen: Stellen Sie sicher, dass Sie sie verstehen!

Bisher haben wir kaum einen Grund, die Fixpunktiteration zu verwenden: Wir haben gezeigt, dass die Konvergenz nur dann garantiert ist, wenn g Lipschitz ist, und unser Argument über die Ek zeigt eine lineare Konvergenz wie eine Halbierung. Es gibt jedoch einen Fall, in dem die Festkomma-Iteration einen Vorteil bietet.

Angenommen, g ist differenzierbar mit g (x ÿ ) = 0. Dann verschwindet der Term erster Ordnung in der Taylor-Reihe für g und hinterlässt:

$$1\ g(xk) = g(x\ \ddot{y}\ ) + g\ (\bar{x}\ \ddot{y}\ )(xk\ \ddot{y}\ x\ \ddot{y}\ )\ 2 \qquad ^{2} + O\ (xk\ \ddot{y}\ x\ \ddot{y}\ ) \qquad ^{3}\ .$$

Somit haben wir in diesem Fall:

$$Ek = |xk\ \ddot{y}\ x\ \ddot{y}\ | = |$$

$$g(xk\ddot{y}1)\ \ddot{y}\ g(x\ \ddot{y}\ )| \text{ wie vorher}$$

$$= \quad \bar{1}\ |g\ (x\ \ddot{y}\ )|(xk\ddot{y}1\ \ddot{y}\ x\ \ddot{y}\ )\ 2 \quad ^{2} + O((xk\ddot{y}1\ \ddot{y}\ x\ \ddot{y}\ ) \qquad ^{3}) \text{ aus dem Taylor-Argument}$$

$$1\ \ddot{y}\bar{} \ (|g\ (x\ \ddot{y}\ )| + \ddot{y})|(xk\ddot{y}1\ \ddot{y}\ x\ \ddot{y}\ ) \qquad ^{2} \text{ für einige } \ddot{y}, \text{ solange } xk\ddot{y}1 \text{ nahe bei } x \text{ liegt} \qquad ^{\ddot{y}}$$

$$= \quad \frac{2}{2}\ 1\ (|g\ (x\ \ddot{y}\ )| + \ddot{y})E \quad ^{2}_{k\ddot{y}1}$$

In diesem Fall ist Ek also quadratisch in Ekÿ1 , wir sagen also, dass eine Festpunktiteration quadratische Konvergenz haben kann; Beachten Sie, dass dieser Beweis der quadratischen Konvergenz nur gilt, weil wir Ek ÿ 0 bereits aus unserem allgemeineren Konvergenzbeweis kennen. Dies impliziert, dass Ek ÿ 0 viel schneller ist, sodass wir weniger Iterationen benötigen, um eine vernünftige Nullstelle zu erreichen.

**Beispiel 7.1** (Konvergenz der Festkomma-Iteration).

## 7.1.5 Newtons Methode

Wir verschärfen unsere Funktionsklasse noch einmal, um eine Methode abzuleiten, die eine konsistentere quadratische Konvergenz aufweist. Nehmen wir nun erneut an, wir möchten f(x ÿ ) = 0 lösen, gehen aber nun davon aus, dass $^{1}, ^{A}$ f eine etwas engere Bedingung als Lipschitz ist.

An einem Punkt xk ÿ R können wir ihn, da f jetzt differenzierbar ist, mit einer Tangente approximieren: f(x) ÿ

$$f(xk) + f\ (xk)(x\ \ddot{y}\ xk)$$

Das Auflösen dieser Näherung nach f(x) ÿ 0 ergibt eine Wurzel

$$xk+1 = xk\ \ddot{y} \qquad \frac{f(xk)\ f}{(xk)}\ .$$

Die Iteration dieser Formel ist als Newtons Methode zur Wurzelfindung bekannt und läuft darauf hinaus, eine lineare Näherung des nichtlinearen Problems iterativ zu lösen.

Beachten Sie das, wenn wir definieren

$$g(x) = x\ \ddot{y}\ f\ (x)\ \frac{f(x)}{\quad},$$

dann läuft Newtons Methode auf eine Festpunktiteration auf g hinaus. Durch Differenzieren finden wir:

$$g\ (x) = 1\ \ddot{y} \qquad \frac{f\ (x) \quad ^{2}\ \ddot{y}\ f(x)f\ (x) \text{ nach}}{f(x)f \quad ^{2}} \text{der Quotientenregel } f\ (x)$$

$$= \frac{(x)}{f\ (x) \quad ^{2}}$$

Fixpunktiteration ist eine einfache Wurzel, was f (x ÿ ) = 0 bedeutet. Dann ist g (x ÿ ) = 0, und durch unsere Ableitung der obigen „Angenommen x“ wissen wir, dass Newtons Methode für einen hinreichend nahen Punkt quadratisch gegen x ÿ konvergiert erste Vermutung. Wenn also f mit einer einfachen Wurzel differenzierbar ist, liefert das Newton-Verfahren eine Festkomma-Iterationsformel, die garantiert quadratisch konvergiert; Wenn x jedoch nicht einfach ist, kann die Konvergenz linear oder schlechter sein. Ist

Die Ableitung der Newton-Methode legt andere Methoden nahe, die durch die Verwendung weiterer Terme in der Taylor-Reihe abgeleitet werden. Beispielsweise fügt die „Halley-Methode“ den Iterationen Terme hinzu, an denen f beteiligt ist, und eine Klasse von „Haushaltsmethoden“ akzeptiert eine beliebige Anzahl von Ableitungen. Diese Techniken bieten eine noch höhere Konvergenz, allerdings auf Kosten der Auswertung komplexerer Iterationen und der Möglichkeit exotischerer Fehlermodi. Andere Methoden ersetzen Taylor-Reihen durch andere Grundformen; Beispielsweise verwendet die lineare Bruchinterpolation rationale Funktionen, um Funktionen mit Asymptotenstruktur besser anzunähern.

## 7.1.6 Sekantenmethode

Ein Effizienzproblem, auf das wir noch nicht eingegangen sind, sind die Kosten für die Bewertung von f und seinen Derivaten. Wenn f eine sehr komplizierte Funktion ist, möchten wir möglicherweise die Häufigkeit, mit der wir f berechnen müssen, oder noch schlimmer, f minimieren. Höhere Konvergenzordnungen helfen bei diesem Problem, aber wir können auch numerische Methoden entwerfen, die die Auswertung kostspieliger Ableitungen vermeiden.

**Beispiel 7.2** (Design). Angenommen, wir entwerfen eine Rakete und möchten wissen, wie viel Treibstoff wir dem Motor hinzufügen müssen. Für eine gegebene Gallonenzahl x können wir eine Funktion f(x) schreiben , die die maximale Höhe der Rakete angibt; Unsere Ingenieure haben angegeben, dass die Rakete eine Höhe h erreichen soll, also müssen wir f(x) = h lösen. Die Bewertung von f(x) erfordert die Simulation einer Rakete beim Start und die Überwachung ihres Treibstoffverbrauchs, was eine teure Angelegenheit ist, und obwohl wir vermuten könnten, dass f differenzierbar ist, können wir f möglicherweise nicht innerhalb einer praktikablen Zeitspanne bewerten.

Eine Strategie zum Entwerfen von Methoden mit geringeren Auswirkungen besteht darin, Daten so weit wie möglich wiederzuverwenden. Für Beispielsweise könnten wir leicht Folgendes annähern:

$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Das heißt, da wir f(xkÿ1) in der vorherigen Iteration berechnen mussten , verwenden wir einfach die Steigung zu f(xk) , um die Ableitung anzunähern. Sicherlich funktioniert diese Näherung gut, insbesondere wenn xk nahe der Konvergenz liegt.

Wenn wir unsere Näherung in die Newton-Methode integrieren, ergibt sich ein neues iteratives Schema:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Beachten Sie, dass der Benutzer zwei anfängliche Schätzungen x0 und xÿ1 angeben muss, um dieses Schema zu starten, oder eine einzelne Iteration von Newton ausführen kann, um es zu starten.

Die Analyse der Sekantenmethode ist etwas komplizierter als die der anderen von uns betrachteten Methoden, da sie sowohl f(xk) als auch f(xkÿ1) verwendet; Der Beweis seiner Konvergenz liegt außerhalb des Rahmens unserer Diskussion. Interessanterweise zeigt die Fehleranalyse, dass der Fehler zwischen linear und quadratisch mit einer Rate von $1+\sqrt{5}/2$ (dem „Goldenen Schnitt“) abnimmt; Da die Konvergenz der des Newton-Verfahrens nahe kommt, ohne dass f ausgewertet werden muss, kann das Sekantenverfahren eine starke Alternative darstellen.

### 7.1.7 Hybridtechniken

Es kann zusätzliches Engineering durchgeführt werden, um zu versuchen, die Vorteile verschiedener Wurzelfindungsalgorithmen zu kombinieren. Beispielsweise könnten wir die folgenden Beobachtungen zu zwei Methoden machen, die wir besprochen haben:

- Die Halbierung konvergiert garantiert bedingungslos, jedoch nur mit einer linearen Rate.
- Die Sekantenmethode konvergiert schneller, wenn sie eine Wurzel erreicht, in manchen Fällen jedoch nicht konvergieren.

Angenommen, wir haben eine Wurzel von f(x) in einem Intervall [k ,rk ] wie bei einer Halbierung eingeklammert. Wir können das sagen, aktuellen Schätzung von x xk und ist gegeben durch xk = k wenn | f(k)| < | f(rk)| andernfalls xk = rk . Wenn wir bei unserer xkÿ1 im Auge behalten , könnten wir xk+1 als die nächste durch die Sekantenmethode gegebene Schätzung der Wurzel annehmen. Wenn xk jedoch außerhalb des Intervalls [k ,rk ] liegt, können wir es durch k+rk/2 ersetzen. Diese Korrektur garantiert, dass xk+1 ÿ [k ,rk ], und unabhängig von der Wahl können wir durch Untersuchung des Vorzeichens von f(xk+1) eine gültige Klammer [k+1 ,rk+1 ] wie in der Halbierung erhalten. Dieser Algorithmus ist als „Dekker-Methode“ bekannt.

Die obige Strategie versucht, die unbedingte Konvergenz der Halbierung mit den stärkeren Wurzelschätzungen der Sekantenmethode zu kombinieren. In vielen Fällen ist es erfolgreich, aber seine Konvergenzrate ist etwas schwierig zu analysieren; Spezielle Fehlermodi können diese Methode auf lineare Konvergenz oder Schlimmeres reduzieren – tatsächlich kann Halbierung in manchen Fällen überraschenderweise schneller konvergieren! Andere Techniken, z. B. die „Brent-Methode“, führen häufiger Halbierungsschritte durch, um diesen Fall zu vermeiden, und können auf Kosten einer etwas komplexeren Implementierung ein garantiertes Verhalten zeigen.

### 7.1.8 Einzelvariablenfall: Zusammenfassung

Wir haben nun eine Reihe von Methoden zur Lösung von f(x ÿ ) = 0 im Fall einer einzelnen Variablen vorgestellt und analysiert. An diesem Punkt ist es wahrscheinlich offensichtlich, dass wir nur an der Oberfläche solcher Techniken gekratzt haben; Es gibt viele iterative Schemata zur Wurzelfindung, alle mit unterschiedlichen Garantien, Konvergenzraten und Einschränkungen. Unabhängig davon können wir aufgrund unserer Erfahrungen eine Reihe von Beobachtungen machen:

- Aufgrund der möglichen generischen Form von f ist es unwahrscheinlich, dass wir Wurzeln x ÿ exakt finden können und uns stattdessen mit iterativen Verfahren zufrieden geben.
- Wir wünschen uns, dass die Folge xk der Wurzelschätzungen x erreicht schnellstens. Wenn Ek eine Fehlergrenze ist, können wir eine Reihe von Konvergenzsituationen charakterisieren, indem wir Ek ÿ 0 als k ÿ ÿ annehmen. Eine vollständige Liste der Bedingungen, die gelten müssen, wenn k groß genug ist, finden Sie unten:
  1. Lineare Konvergenz: Ek+1 ÿ CEk für einige C < 1 2. Superlineare
  Konvergenz: Ek+1 ÿ CEr für r > 1 (jetzt benötigen wir kein C < 1, da sich die r-Potenz aufheben kann, wenn Ek klein genug ist die Auswirkungen von C)
  3. Quadratische Konvergenz: Ek+1 ÿ CE2 4. Kubische
  Konvergenz: Ek+1 ÿ CE3 (und so weiter)
- Eine Methode konvergiert möglicherweise schneller, erfordert aber bei jeder einzelnen Iteration zusätzliche Berechnungen. Aus diesem Grund kann es vorzuziehen sein, mehr Iterationen einer einfacheren Methode durchzuführen als weniger Iterationen einer komplexeren Methode.

## 7.2 Multivariable Probleme

Einige Anwendungen erfordern möglicherweise die Lösung eines allgemeineren Problems f(x) =0 für eine Funktion f : Rn ÿ Rm. Wir haben bereits einen Fall dieses Problems bei der Lösung von Ax =b gesehen, was dem Finden von Wurzeln von f(x) ÿ Ax ÿb entspricht, aber der allgemeine Fall ist wesentlich schwieriger. Insbesondere Strategien wie die Halbierung lassen sich nur schwer erweitern, da wir nun weitgehend garantieren, dass m verschiedene Werte gleichzeitig alle Null sind.

### 7.2.1 Newtons Methode

Glücklicherweise lässt sich eine unserer Strategien auf unkomplizierte Weise erweitern. Denken Sie daran, dass wir für f : Rn ÿ Rm die Jacobi-Matrix schreiben können, die die Ableitung jeder Komponente von f in jeder Koordinatenrichtung angibt:

$$(D\,f)_{ij} \ddot{y}\, \frac{d\,f_i}{dx_j}$$

Wir können die Jacobi-Funktion von f verwenden, um unsere Ableitung der Newton-Methode auf mehrere Dimensionen zu erweitern. Insbesondere ist die Näherung erster Ordnung von f gegeben durch:

$$f(x) \ddot{y}\, f(x_k) + D\,f(x_k) \cdot (x \ddot{y} x_k).$$

Das Einsetzen des gewünschten f(x) =0 ergibt das folgende lineare System für die nächste Iteration xk+1 :

$$D\,f(x_k) \cdot (x_{k+1} \ddot{y} x_k) = \ddot{y} f(x_k)$$

Diese Gleichung kann mit der Pseudoinversen gelöst werden, wenn m < n; Wenn m > n, kann man die Methode der kleinsten Quadrate ausprobieren, aber sowohl die Existenz einer Wurzel als auch die Konvergenz dieser Technik sind unwahrscheinlich. Wenn D f jedoch quadratisch ist, entspricht die , wir erhalten die typische Iteration für Newton Methode f : Rn ÿ Rn :

$$x_{k+1} = x_k \ddot{y} [D\,f(x_k)]\ddot{y}1\, f(x_k), \text{ wobei}$$

wir wie immer die Matrix [D f(xk)]ÿ1 nicht explizit berechnen , sondern sie vielmehr verwenden, um die Lösung eines linearen Systems zu signalisieren.

Die Konvergenz von Festkommamethoden wie der Newton-Methode, die xk+1 = g(xk) iteriert , erfordert, dass der Eigenwert der maximalen Größe des Jacobi-Dg kleiner als 1 ist. Nach Überprüfung dieser Annahme zeigt sich ein Argument, das dem eindimensionalen Fall ähnelt dass Newtons Methode ÿ kann , für die D f(x ÿ ) nichtsingulär ist. haben Wurzeln x quadratische Konvergenz in der Nähe von

### 7.2.2 Newton schneller machen: Quasi-Newton und Broyen

Wenn m und n zunehmen, wird die Newton-Methode sehr teuer. Für jede Iteration muss eine andere Matrix D f(xk) invertiert werden; Da es sich so oft ändert, hilft die Vorfaktorisierung von D f(xk) = LkUk nicht.

Einige Quasi-Newton-Strategien versuchen, unterschiedliche Approximationsstrategien anzuwenden, um einzelne Iterationen zu vereinfachen. Ein einfacher Ansatz könnte beispielsweise D f aus früheren Iterationen wiederverwenden und gleichzeitig f(xk) unter der Annahme neu berechnen, dass sich die Ableitung nicht sehr schnell ändert. Wir werden auf diese Strategien zurückkommen, wenn wir die Anwendung der Newton-Methode auf die Optimierung diskutieren.

Eine andere Möglichkeit besteht darin, zu versuchen, eine Parallele zu unserer Ableitung der Sekantenmethode herzustellen. So wie die Sekantenmethode immer noch eine Division enthält, machen solche Näherungen nicht unbedingt die Notwendigkeit, eine Matrix zu invertieren, zu verringern, sie ermöglichen jedoch die Durchführung einer Optimierung ohne explizite Berechnung des Jacobian D f . Solche Erweiterungen sind nicht ganz offensichtlich, da geteilte Differenzen keine vollständige approximative Jacobi-Matrix ergeben.

Bedenken Sie jedoch, dass die Richtungsableitung von f in Richtung v durch Dv f = D f · v gegeben ist. Wie bei der Sekantenmethode können wir diese Beobachtung zu unserem Vorteil nutzen, indem wir verlangen, dass unsere Näherung J einer Jacobi-Methode erfüllt

$$J \cdot (x_k - x_{k-1}) = f(x_k) - f(x_{k-1}).$$

Broydens Methode ist eine solche Erweiterung der Sekantenmethode, die nicht nur eine Schätzung xk von x verfolgt, sondern auch eine Matrix Jk, die den Jacobi-Wert schätzt; Es müssen beide Anfangsschätzungen J0 und x0 angegeben werden. Angenommen, wir haben eine vorherige Schätzung Jk−1 des Jacobi-Werts aus der vorherigen Iteration. Wir haben jetzt einen neuen Datenpunkt xk , an dem wir f(xk) ausgewertet haben , daher möchten wir Jk−1 unter Berücksichtigung dieser neuen Beobachtung auf einen neuen Jacobi- Jk aktualisieren. Ein vernünftiges Modell besteht darin, zu verlangen, dass die neue Näherung der alten Näherung bis auf die Richtung xk − xk−1 möglichst ähnlich ist :

$$\begin{aligned} &\text{minimieren} J_k \quad \|J_k - J_{k-1}\|_{\text{Her}}^2, \\ &\text{so dass } J_k \cdot (x_k - x_{k-1}) = f(x_k) - f(x_{k-1}) \end{aligned}$$

Um dieses Problem zu lösen, definieren Sie $\Delta J = J_k - J_{k-1}$ und $\Delta x = x_k - x_{k-1}$ . Die Substitution $\Delta y = f(x_k) - f(x_{k-1}) - J_{k-1} \cdot \Delta x$. dieser Substitutionen ergibt die folgende Form:

$$\begin{aligned} &\text{minimiere } \|\Delta J\|_{\text{Her}}^2 \\ &\text{so, dass } \Delta J \cdot \Delta x = d \end{aligned}$$

Wenn wir *y* als Lagrange-Multiplikator betrachten, ist diese Minimierung gleichbedeutend mit der Suche nach kritischen Punkten der Lagrange-Funktion :

$$\Lambda = \|\Delta J\|_{\text{Her}}^2 + y^T (\Delta J \cdot \Delta x - d)$$

Differenzieren nach $(\Delta J)_{ij}$ zeigt:

$$0 = \frac{\partial \Lambda}{\partial (\Delta J)_{ij}} = 2(\Delta J)_{ij} + y_i(\Delta x)_j = [y \Delta x]_{ij} = [y (y^T \Delta J)]_{ij} = [(y y^T) \Delta J]_{ij}$$

Das Einsetzen in $\Delta J \cdot \Delta x = d$ ergibt $(y y^T)(\Delta x) = 2d$, oder äquivalent $y = 2d/\Delta x^2$ . Schließlich können wir Folgendes ersetzen:

$$\Delta J = \frac{1}{2} (y y^T)(\Delta x) = \frac{d}{2} \frac{(y y^T)}{\Delta x}$$

Die Erweiterung unserer Vertretung zeigt:

$$\begin{aligned} J_k &= J_{k-1} + \Delta J \\ &= J_{k-1} + \frac{d}{2} \frac{(y y^T)}{\Delta x} \\ &= J_{k-1} + \frac{(f(x_{k-1}) - J_{k-1} \cdot \Delta x) (x_k - x_{k-1})}{\|x_k - x_{k-1}\|^2} \end{aligned}$$

Daher wechselt Broydens Methode einfach zwischen dieser Aktualisierung und dem entsprechenden Newton f(xk). In Schritt xk+1 = xk ÿ J $^{ÿ1}_{k}$ manchen Fällen lässt sich zusätzliche Effizienz erzielen, indem man den Überblick behält die Matrix J $^{ÿ1}_{k}$ explizit anstelle der Matrix Jk , die mit einer ähnlichen Formel aktualisiert werden kann.

## 7.3 Konditionierung

Wir haben bereits in Beispiel 1.7 gezeigt, dass die Bedingungszahl der Wurzelfindung in einer einzelnen Variablen ist:

$$\text{condx ÿ f} = \left| \frac{1}{f\,(x\ ÿ)} \right|$$

Wie in Abbildung NUMBER dargestellt, zeigt diese Bedingungsnummer, dass die bestmögliche Situation für die Wurzelfindung dann auftritt, wenn sich f in der Nähe von x schnell ändert, da in diesem Fall eine Störung von x ÿ dazu führt, dass f Werte weit von 0 annimmt.

Die Anwendung eines identischen Arguments, wenn f mehrdimensional ist, ergibt eine Bedingungszahl von D f(x ÿ ) $^{ÿ1}$ . Beachten Sie, dass die Bedingungszahl unendlich ist, wenn D f nicht invertierbar ist. Diese Kuriosität tritt auf, weil Störungen erster Ordnung x zu $^{ÿ}$ bewahrt f(x) = 0, und eine solche Bedingung kann dies tatsächlich anspruchsvollen Wurzelfindungsfällen führen, wie sie in Abbildung NUMMER dargestellt sind.

## 7.4 Probleme

Viele Möglichkeiten, darunter:

- Viele mögliche Festkomma-Iterationsschemata für ein gegebenes Wurzelfindungsproblem, grafische Version der Festkomma-Iteration
- Mittlere Felditeration in ML
- Müllers Methode – komplexe Wurzeln
- Iterative Methoden höherer Ordnung – Householder-Methoden
- Interpretation von Eigensachen als Wurzelfindung
- Konvergenz der Sekantenmethode
- Wurzeln von Polynomen
- Newton-Fourier-Methode (!)
- „Modifizierte Newton-Methode bei nichtquadratischer Konvergenz“
- Konvergenz –¿ Spektralradius für mehrdimensionales Newton; quadratische Konvergenz
- Sherman-Morrison-Update für Broyden

# Kapitel 8

# Uneingeschränkte Optimierung

In den vorherigen Kapiteln haben wir uns für einen weitgehend variierenden Ansatz zur Ableitung von Standardalgorithmen für die rechnerische lineare Algebra entschieden. Das heißt, wir definieren eine Zielfunktion, möglicherweise mit Einschränkungen, und stellen unsere Algorithmen als Minimierungs- oder Maximierungsproblem dar. Eine Auswahl aus unserer vorherigen Diskussion ist unten aufgeführt:

| Problem | Zielsetzung | Einschränkungen |
|---|---|---|
| Kleinsten Quadrate | $E(x) = Ax \ddot{y} b$ $^2$ | Keiner |
| Projektb aufa | $E(c) = ca \ddot{y} b$ | Keiner |
| Eigenvektoren der symmetrischen Matrix | $E(x) = x Ax$ | $x = 1$ |
| Pseudoinvers | $E(x) = x$ $^2$ | $A Ax = A b$ |
| Hauptkomponentenanalyse | $E(C) = X \ddot{y} CCXFro$ | $CC = Id_{\times d}$ |
| Broyden-Schritt | $E(Jk) = Jk \ddot{y} Jk\ddot{y}1$ $^2_{Her}$ | $Jk \cdot (xk \ddot{y} xk\ddot{y}1) = f(xk) \ddot{y} f(xk\ddot{y}1)$ |

Offensichtlich ist die Formulierung von Problemen auf diese Weise ein wirkungsvoller und allgemeiner Ansatz. Aus diesem Grund ist es wertvoll, Algorithmen zu entwerfen, die ohne eine spezielle Form für die Energie E funktionieren, genauso wie wir Strategien entwickelt haben, um Wurzeln von f zu finden, ohne die Form von vornherein zu kennen.

## 8.1 Uneingeschränkte Optimierung: Motivation

In diesem Kapitel werden wir unbeschränkte Probleme betrachten, das heißt Probleme, die als Minimierung oder Maximierung einer Funktion $f : Rn \ddot{y} \mathbf{R}$ ohne Anforderungen an die Eingabe gestellt werden können. Es ist nicht schwer, in der Praxis auf solche Probleme zu stoßen; Nachfolgend listen wir einige Beispiele auf.

**Beispiel 8.1** (Nichtlineare kleinste Quadrate). Angenommen, wir erhalten eine Anzahl von Paaren $(xi , yi)$, so dass $f(xi) \ddot{y} yi$ , und wir möchten den besten Näherungswert für f innerhalb einer bestimmten Klasse finden. Beispielsweise können wir erwarten, dass f exponentiell ist. In diesem Fall sollten wir $f(x) = ceax$ für einige c und einige a schreiben können; Unsere Aufgabe ist es, diese Parameter zu finden. Eine einfache Strategie könnte darin bestehen, zu versuchen, die folgende Energie zu minimieren:

$$E(a, c) = \ddot{y} (yi \ddot{y} ceaxi)^2.$$

Diese Form für E ist in a nicht quadratisch, daher sind unsere linearen Methoden der kleinsten Quadrate nicht anwendbar.

**Beispiel 8.2** (Maximum-Likelihood-Schätzung). Beim maschinellen Lernen besteht das Problem der Parameterschätzung darin, die Ergebnisse eines randomisierten Experiments zu untersuchen und zu versuchen, sie mithilfe einer Wahrscheinlichkeitsverteilung einer bestimmten Form zusammenzufassen. Beispielsweise könnten wir die Körpergröße jedes Schülers in einer Klasse messen und so eine Liste der Körpergrößen hi für jeden Schüler i erhalten. Wenn wir viele Schüler haben, können wir die Verteilung der Schülergrößen mithilfe einer Normalverteilung modellieren:

$$g(h; \mu, \ddot{y}) = \frac{1}{\ddot{y}\,\ddot{y}\,2\ddot{y}} e^{\ddot{y}(h\ddot{y}\mu)\,2/2\ddot{y}\,2},$$

Dabei ist $\mu$ der Mittelwert der Verteilung und $\ddot{y}$ die Standardabweichung.

Unter dieser Normalverteilung ist die Wahrscheinlichkeit, dass wir die Körpergröße hi für Schüler i beobachten, durch g(hi ; µ, ÿ) gegeben , und unter der (vernünftigen) Annahme, dass die Körpergröße von Schüler i wahrscheinlich unabhängig von der von Schüler j ist, Die Wahrscheinlichkeit, den gesamten Satz beobachteter Höhen zu beobachten, ist durch das Produkt gegeben

$$P(\{h1, \ldots, hn\}; \mu, \ddot{y}) = \ddot{y}\, g(hi ; \mu, \ddot{y}).$$

Eine übliche Methode zum Schätzen der Parameter $\mu$ und $\ddot{y}$ von g besteht darin, P, betrachtet als Funktion von $\mu$ und $\ddot{y}$, mit festem {hi} zu maximieren; Dies wird als Maximum-Likelihood-Schätzung von $\mu$ und ÿ bezeichnet. In der Praxis optimieren wir normalerweise die logarithmische Wahrscheinlichkeit *(µ,* ÿ) ÿ log P({h1, . . . hn}; µ, ÿ); Diese Funktion hat die gleichen Maxima, verfügt aber über bessere numerische und mathematische Eigenschaften.

**Beispiel 8.3** (Geometrische Probleme). Viele Geometrieprobleme, die in Grafik und Vision auftreten, lassen sich nicht auf Energien der kleinsten Quadrate reduzieren. Angenommen, wir haben eine Anzahl von Punkten x1, . . . ,xk ÿ R3 . Wenn wir diese Punkte gruppieren möchten, möchten wir sie möglicherweise mit einem einzigen x zusammenfassen, indem wir Folgendes minimieren:

$$E(x) \ddot{y} \ddot{y} \quad x \ddot{y} xi2.$$

Das x ÿ R3 , das E minimiert, ist als geometrischer Median von {x1, . . . ,xk}. Beachten Sie, dass die Norm der Differenz x ÿ xi in E nicht quadratisch ist, sodass die Energie in den Komponenten von x nicht mehr quadratisch ist.

**Beispiel 8.4** (Physikalische Gleichgewichte, angepasst von CITE). Angenommen, wir befestigen einen Gegenstand an einem Satz Federn. Jede Feder ist am Punkt xi ÿ R3 verankert und hat die natürliche Länge Li und die Konstante ki . In Abwesenheit der Schwerkraft, wenn sich unser Objekt an der Position p ÿ R3 befindet, hat das Quellennetz verfügt über potentielle Energie

$$E(p) = 2 - \frac{1}{\ddot{y}\, ki\, (p\, \ddot{y} xi2\, \ddot{y}\, Li)}^{2}$$

Gleichgewichte dieses Systems sind durch Minima von E gegeben und spiegeln Punkte p wider, an denen die Federkräfte alle ausgeglichen sind. Solche Gleichungssysteme werden verwendet, um Graphen G = (V, E) zu visualisieren, indem für jedes Paar in E Eckpunkte in V mit Federn verbunden werden.

## 8.2 Optimalität

Bevor wir diskutieren, wie eine Funktion minimiert oder maximiert werden kann, sollten wir uns darüber im Klaren sein, wonach wir suchen. Beachten Sie, dass das Maximieren von f dasselbe ist wie das Minimieren von ÿf, sodass das Minimierungsproblem für unsere Betrachtung ausreichend ist. Für ein bestimmtes f : Rn ÿ **R** und x ÿ Rn müssen wir f(x ÿ ) hat. Optimalitätsbedingungen, die ableiten, dass ÿ den kleinstmöglichen Wert

verifizieren, dass x Natürlich möchten wir im Idealfall globale Optima von f finden:

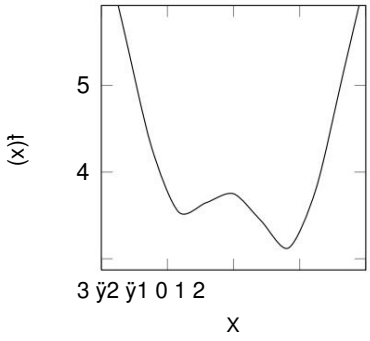



Abbildung 8.1: Eine Funktion f(x) mit mehreren Optima.

**Definition 8.1** (Globales Minimum). Der Punkt x f(x ÿ ) ÿ ÿ ÿ Rn ist ein globales Minimum von f : Rn ÿ R wenn f(x) für alle x ÿ Rn .

Das Finden eines globalen Minimums von f ohne Informationen über die Struktur von f erfordert effektiv eine Suche im Dunkeln. Angenommen, ein Optimierungsalgorithmus identifiziert das lokale Minimum in der Nähe von x = ÿ1 in der Funktion in Abbildung 8.1. Es ist fast unmöglich, einfach durch Erraten der x-Werte zu erkennen, dass es ein zweites, niedrigeres Minimum in der Nähe von x = 1 gibt – soweit wir wissen, könnte es bei x = 1000 ein drittes, noch niedrigeres Minimum von f geben!

Daher begnügen wir uns in vielen Fällen damit, ein lokales Minimum zu finden:

**Definition 8.2** (Lokales Minimum). Der Punkt x f(x) für ÿ ÿ Rn ist ein lokales Minimum von f : Rn ÿ R, wenn f(x ÿ ) ÿ < $\ddot{y}$ alle x ÿ Rn , die x ÿx erfüllen ÿ für ein $\ddot{y}$ > 0.

Definition erfordert, dass der x-Radius ÿ erreicht den kleinsten Wert in einer Umgebung, die durch definiert ist. Diese ÿ ist. Beachten Sie, dass lokale Optimierungsalgorithmen eine schwerwiegende Einschränkung haben, da sie nicht garantieren können, dass sie den niedrigstmöglichen Wert von f liefern, wie in Abbildung 8.1, wenn das linke lokale Minimum erreicht wird; Viele heuristische und andere Strategien werden angewendet, um die Landschaft möglicher x-Werte zu erkunden und so die Gewissheit zu gewinnen, dass ein lokales Minimum den bestmöglichen Wert hat.

## 8.2.1 Differenzielle Optimalität

Eine bekannte Geschichte aus der Ein- und Mehrvariablenrechnung besagt, dass das Finden potenzieller Minima und Maxima einer Funktion f : Rn ÿ **R** einfacher ist, wenn f differenzierbar ist. Denken Sie daran, dass der Gradientenvektor ÿ f = *(ÿ* f/ÿx1, . . . , *ÿ* f/ÿxn) in die Richtungzeigt, in der f am stärksten zunimmt; der Vektor ÿÿ f zeigt in Richtung der größten Abnahme. Eine Möglichkeit, dies zu erkennen, besteht darin, sich daran zu erinnern, dass f in der Nähe von a aussieht wie der lineare Funktionspunkt x0 ÿ Rn

$$f(x) \ddot{y} f(x0) + \ddot{y} f(x0) \cdot (x \ddot{y}x0).$$

Wenn wir x ÿx0 = ÿÿ f(x0) nehmen, dann finden wir:

$$f(x0 + \ddot{y}\ddot{y} f(x0)) \ddot{y} f(x0) + \ddot{y}\ddot{y} f(x0) \quad ^2$$

Wenn ÿ f(x0) > 0, bestimmt das Vorzeichen von *ÿ ,* ob f zunimmt oder abnimmt.

Es ist nicht schwer, das obige Argument zu formalisieren, um zu zeigen, dass, wenn x0 ein lokales Minimum ist, ÿ f(x0) =0 gelten muss . Beachten Sie, dass diese Bedingung notwendig, aber nicht ausreichend ist: Maxima und Sattel

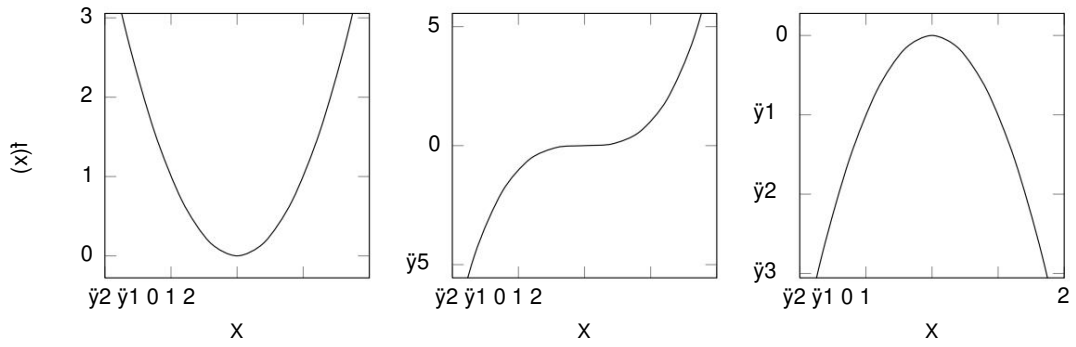



Abbildung 8.2: Kritische Punkte können viele Formen annehmen; Hier zeigen wir ein lokales Minimum, einen Sattelpunkt und ein lokales Maximum.

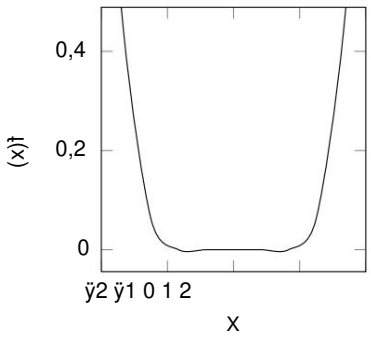



Abbildung 8.3: Eine Funktion mit vielen stationären Punkten.

Punkte haben auch ÿ f(x0) = 0, wie in Abbildung 8.2 dargestellt. Dennoch liefert diese Beobachtung über Minima differenzierbarer Funktionen eine gemeinsame Strategie zur Wurzelfindung:

1. Finden Sie Punkte xi , die ÿ f(xi) =0 erfüllen .

2. Prüfen Sie, welcher dieser Punkte ein lokales Minimum und kein Maximum oder Sattelpunkt ist.

Aufgrund ihrer wichtigen Rolle in dieser Strategie geben wir den von uns gesuchten Punkten einen besonderen Namen:

**Definition 8.3** (Stationärer Punkt). Ein stationärer Punkt von f : Rn ÿ **R** ist ein Punkt x ÿ Rn , der ÿ f(x) =0 erfüllt .

Das heißt, unsere Minimierungsstrategie kann darin bestehen, stationäre Punkte von f zu finden und dann diejenigen zu eliminieren, die keine Minima sind.

Es ist wichtig, im Auge zu behalten, wann wir mit dem Erfolg unserer Minimierungsstrategien rechnen können. In den meisten Fällen, wie beispielsweise in den in Abbildung 8.2 gezeigten, sind die stationären Punkte von f isoliert, was bedeutet, dass wir sie in eine diskrete Liste {x0,x1, . schreiben können . . .}. Ein degenerierter Fall ist jedoch in Abbildung 8.3 dargestellt; Hier besteht das gesamte Intervall [ÿ1/2, 1/2] aus stationären Punkten, so dass es unmöglich ist, sie einzeln zu betrachten. Wir werden Probleme wie degenerierte Fälle größtenteils ignorieren, werden aber auf sie zurückkommen, wenn wir die Konditionierung des Minimierungsproblems betrachten.

Angenommen, wir identifizieren einen Punkt x ÿ **R** als stationären Punkt von f und möchten nun prüfen, ob es sich um ein lokales Minimum handelt. Wenn f zweimal differenzierbar ist, besteht eine Strategie, die wir anwenden können, darin, seine Hessesche Funktion zu schreiben

Matrix:

$$Hf(x) = \begin{bmatrix} \frac{\ddot{y}^2}{\text{f } \ddot{y}x_1} & \frac{\ddot{y}^2}{\text{f } \ddot{y}x_1\ddot{y}x_2} & \cdots & \frac{\ddot{y}^2}{\text{f } \ddot{y}x_1\ddot{y}x_n} \\ \frac{\ddot{y}^2}{\text{f } \ddot{y}x_2\ddot{y}x_1} & \frac{\ddot{y}^2}{\text{f } \ddot{y}^2 x_2} & \cdots & \frac{\ddot{y}^2}{\text{f } \ddot{y}x_2\ddot{y}x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\ddot{y}^2}{\text{f } \ddot{y}x_n\ddot{y}x_1} & \frac{\ddot{y}^2}{\text{f } \ddot{y}x_n\ddot{y}x_2} & \cdots & \frac{\ddot{y}^2}{\text{f } \ddot{y}^2 x_n} \end{bmatrix}$$

Wir können unserer Taylor-Entwicklung von f einen weiteren Term hinzufügen, um die Rolle von Hf zu sehen :

$$f(x) \ddot{y} f(x_0) + \ddot{y} f(x_0) \cdot (x \ddot{y} x_0) + \frac{1}{2} (x \ddot{y} x_0) Hf(x \ddot{y} x_0)$$

Wenn wir einen stationären Punkt x ersetzen $^{\ddot{y}}$, dann wissen wir per Definition:

$$f(x) \ddot{y} f(x^{\ddot{y}}) + \frac{1}{2} (x \ddot{y} x^{\ddot{y}}) Hf(x \ddot{y} x^{\ddot{y}})$$

Wenn Hf positiv definit ist, dann zeigt dieser Ausdruck f(x) ÿ f(x $^{\ddot{y}}$) und somit x $^{\ddot{y}}$ ist ein lokales Minimum.
Im Allgemeinen kann eine der folgenden Situationen auftreten:

- Wenn Hf positiv definit ist, dann x $^{\ddot{y}}$ ist ein lokales Minimum von f .
- Wenn Hf negativ definit ist, dann x $^{\ddot{y}}$ ist ein lokales Maximum von f .
- Wenn Hf unbestimmt ist, dann x $^{\ddot{y}}$ ist ein Sattelpunkt von f .
- Wenn Hf nicht invertierbar ist, können Kuriositäten wie die Funktion in Abbildung 8.3 auftreten.

Um zu überprüfen, ob eine Matrix positiv definit ist, kann man prüfen, ob ihre Cholesky-Faktorisierung existiert, oder – langsamer – indem man prüft, ob alle ihre Eigenwerte positiv sind. Wenn also die Hesse-Funktion von f bekannt ist, können wir mithilfe der obigen Liste stationäre Punkte auf Optimalität prüfen. Viele Optimierungsalgorithmen, einschließlich derjenigen, die wir besprechen werden, ignorieren einfach den letzten Fall und benachrichtigen den Benutzer, da dies relativ unwahrscheinlich ist.

### 8.2.2 Optimalität über Funktionseigenschaften

Gelegentlich können wir, wenn wir mehr Informationen über f : Rn ÿ **R** kennen , Optimalitätsbedingungen angeben, die stärker oder einfacher zu überprüfen sind als die oben genannten.

Eine Eigenschaft von f, die starke Auswirkungen auf die Optimierung hat, ist die Konvexität, dargestellt in Abbildung NUMMER:

**Definition 8.4** (Konvex). Eine Funktion f : Rn ÿ **R** ist konvex, wenn für alle x,y ÿ Rn und $\ddot{y}$ ÿ (0, 1) die folgende Beziehung gilt:

$$f((1 \ddot{y} \ddot{y})x + \ddot{y}y) \ddot{y} (1 \ddot{y} \ddot{y})f(x) + \ddot{y} f(y).$$

Wenn die Ungleichung streng ist, ist die Funktion streng konvex.

Konvexität bedeutet, dass, wenn man im Rn zwei Punkte mit einer Linie verbindet, die Werte von f entlang der Linie kleiner oder gleich denen sind, die man durch lineare Interpolation erhalten würde.

Konvexe Funktionen verfügen über viele starke Eigenschaften, von denen die grundlegendste die folgende ist:

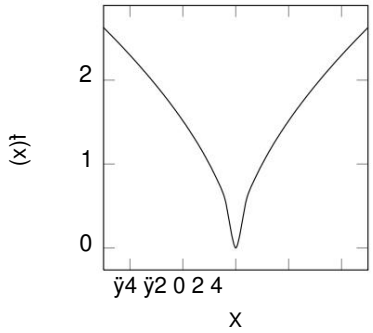



Abbildung 8.4: Eine quasikonvexe Funktion.

**Vorschlag 8.1.** Ein lokales Minimum einer konvexen Funktion f : Rn ÿ **R** ist notwendigerweise ein globales Minimum.

Nachweisen. Nehmen Sie x als ein solches lokales Minimum und nehmen Sie an, dass es $x^{ÿ} = x$ mit $f(x ÿ) < f(x)$. gibt. Dann ist für $ÿ$ ÿ (0, 1)

$$f(x + ÿ(x ÿ ÿx)) ÿ (1 ÿ ÿ)f(x) + ÿ f(x ÿ) \text{ durch Konvexität}$$
$$< f(x) \text{ da } f(x ÿ) < f(x)$$

Aber wenn man $ÿ$ ÿ 0 annimmt, zeigt sich, dass x unmöglich ein lokales Minimum sein kann. □

Dieser Satz und damit verbundene Beobachtungen zeigen, dass es möglich ist, zu überprüfen, ob Sie ein globales Minimum einer konvexen Funktion erreicht haben, indem Sie einfach die Optimalität erster Ordnung anwenden. Daher ist es wertvoll, von Hand zu prüfen, ob eine zu optimierende Funktion zufällig konvex ist, eine Situation, die im wissenschaftlichen Rechnen überraschend häufig vorkommt; Eine hinreichende Bedingung, die einfacher zu überprüfen ist, wenn f zweimal differenzierbar ist, besteht darin, dass Hf überall positiv definit ist.

Andere Optimierungstechniken haben Garantien unter anderen Annahmen über f . Für die Prüfung Beispielsweise ist eine schwächere Version der Konvexität die Quasikonvexität, die erreicht wird, wenn

$$f((1 ÿ ÿ)x + ÿy) ÿ \max(f(x), f(y)).$$

Ein Beispiel für eine quasikonvexe Funktion ist in Abbildung 8.4 dargestellt; Obwohl sie nicht die charakteristische „Schalenform“ einer konvexen Funktion aufweist, verfügt sie über ein einzigartiges Optimum.

# 8.3 Eindimensionale Strategien

Wie im letzten Kapitel beginnen wir mit der eindimensionalen Optimierung von f : **R** ÿ **R** und erweitern dann unsere Strategien auf allgemeinere Funktionen f : Rn ÿ R.

## 8.3.1 Newtons Methode

Unsere Hauptstrategie zur Minimierung differenzierbarer Funktionen f : Rn ÿ **R** besteht darin, sta ÿ zu finden, das ÿ f(x Maxima, Minima ÿ ) = 0 erfüllt . Angenommen, wir können überprüfen, ob stationäre Punkte Tionspunkte sind x oder Sattelpunkte als Post- In diesem Verarbeitungsschritt konzentrieren wir uns auf das Problem, die stationären Punkte x zu finden $^{ÿ}$ .

Nehmen wir dazu an, dass f : **R** ÿ **R** differenzierbar ist. Dann wie in unserer Ableitung von Newton
Methode zur Wurzelfindung können wir Folgendes annähern:

$$1\ f(x) \ddot{y} f(xk) + f\,(xk)(x \ddot{y} xk) + f\,(xk)(x \ddot{y} xk)\ 2^{2}.$$

Die Näherung auf der rechten Seite ist eine Parabel, deren Scheitelpunkt bei xk ÿ f (xk )/f (xk ) liegt .
Natürlich ist f in Wirklichkeit nicht unbedingt eine Parabel, daher iteriert Newtons Methode einfach die Formel

$$xk+1 = xk \ddot{y} \frac{f\,(xk)}{f\,(xk)}.$$

Diese Technik lässt sich angesichts der Arbeit, die wir bereits im vorherigen Kapitel in das Verständnis der New-Ton-Methode zur Wurzelfindung gesteckt haben, leicht analysieren. Eine alternative Möglichkeit, die obige Formel abzuleiten, besteht insbesondere darin, die Wurzel auf f (x) zu finden, da stationäre Punkte f (x) = 0 erfüllen.
Daher weist Newtons Optimierungsverfahren in den meisten Fällen quadratische Konvergenz auf, vorausgesetzt, dass die anfängliche Schätzung x0 hinreichend nahe an x liegt $^{\ddot{y}}$.

Eine natürliche Frage ist, ob die Sekantenmethode auf analoge Weise angewendet werden kann.
Unsere obige Ableitung der Newtonschen Methode findet Wurzeln von f , sodass die Sekantenmethode verwendet werden könnte, um die Auswertung von f, aber nicht von f zu eliminieren; Situationen, in denen wir f, aber nicht f kennen, sind relativ selten. Eine geeignetere Parallele besteht darin, die zur Annäherung von f in der Sekantenmethode verwendeten Liniensegmente durch Parabeln zu ersetzen. Diese Strategie, bekannt als sukzessive parabolische Interpolation, minimiert auch eine quadratische Näherung von f bei jeder Iteration, aber anstatt f(xk), f (xk) und f (xk) zum Konstruieren der Näherung zu verwenden, verwendet sie f(xk), f(xkÿ1) und f(xkÿ2). Die Ableitung dieser Technik ist relativ einfach und sie konvergiert superlinear.

### 8.3.2 Suche im Goldenen Schnitt

Wir haben die Halbierung in unserer Parallele zu Techniken zur Wurzelfindung mit einer Variablen übersprungen. Für dieses Versäumnis gibt es viele Gründe. Unsere Motivation für die Halbierung war, dass sie nur die schwächste Annahme über f verwendete, die zum Finden von Wurzeln erforderlich war: Kontinuität. Der Zwischenwertsatz lässt sich jedoch nicht intuitiv auf Minima anwenden, sodass es den Anschein hat, dass es einen derart einfachen Ansatz nicht gibt.

Es ist jedoch wertvoll, mindestens eine Minimierungsstrategie zur Verfügung zu haben, die keine Differenzierbarkeit von f als zugrunde liegende Annahme erfordert; Schließlich gibt es nicht differenzierbare Funktionen, die eindeutige Minima haben, wie f(x) ÿ |x| bei x = 0. Zu diesem Zweck könnte eine alternative Annahme sein, dass f unimodular ist:

**Definition 8.5** (Unimodular). Eine Funktion f : [a, b] ÿ **R** ist unimodular, wenn es xÿ ÿ [a, b] gibt , sodass f für x ÿ [a, x ÿ ] abnehmend und für x ÿ [x ÿ , b] steigend ist.

Mit anderen Worten: Eine unimodulare Funktion nimmt für einige Zeit ab und beginnt dann zuzunehmen; Es sind keine lokalisierten Minima zulässig. Beachten Sie, dass dies wie |x| funktioniert sind nicht differenzierbar, aber dennoch unimodular.

Angenommen, wir haben zwei Werte x0 und x1, so dass a < x0 < x1 < b. Wir können zwei Beobachtungen machen tionen, die uns bei der Formulierung einer Optimierungstechnik helfen werden:

- Wenn f(x0) ÿ f(x1), dann wissen wir, dass f(x) ÿ f(x1) für alle x ÿ [a, x0]. Somit ist das Intervall [a, x0] kann bei unserer Suche nach einem Minimum von f verworfen werden.

• Wenn f(x1) ÿ f(x0), dann wissen wir, dass f(x) ÿ f(x0) für alle x ÿ [x1, b], und können daher verwerfen [x1, b].

Diese Struktur legt eine mögliche Strategie zur Minimierung nahe, die mit dem Intervall [a, b] beginnt und Teile gemäß den oben genannten Regeln iterativ entfernt.

Ein wichtiges Detail bleibt jedoch bestehen. Unsere Konvergenzgarantie für den Halbierungsalgorithmus beruhte auf der Tatsache, dass wir in jeder Iteration die Hälfte des betreffenden Intervalls entfernen konnten.
Wir könnten auf ähnliche Weise vorgehen und jedes Mal ein Drittel des Intervalls entfernen; Dies erfordert zwei Auswertungen von f während jeder Iteration an neuen x0- und x1 -Positionen. Wenn die Auswertung von f jedoch teuer ist, möchten wir möglicherweise Informationen aus früheren Iterationen wiederverwenden, um mindestens eine dieser beiden Auswertungen zu vermeiden.

Vorerst gilt a = 0 und b = 1; Die Strategien, die wir unten ableiten, funktionieren allgemeiner durch Verschiebung und Skalierung. In Ermangelung weiterer Informationen über f könnten wir genauso gut eine symmetrische Wahl x0 = *ÿ* und x1 = 1 ÿ *ÿ* für ein *ÿ* ÿ (0, 1/2) treffen. Angenommen, unsere Iteration entfernt das Intervall ganz rechts [x1, b]. Dann wird das Suchintervall zu [0, 1 ÿ ÿ] und wir kennen *f(ÿ)* aus der vorherigen Iteration. Die nächste Iteration wird [0, 1 ÿ ÿ] so dividieren, dass x0 = ÿ(1 ÿ ÿ) und x1 = (1 ÿ ÿ) $^2$. Wenn wir Wenn Sie *f(ÿ)* aus der vorherigen Iteration wiederverwenden möchten , können Sie (1 ÿ ÿ) festlegen. $^2$ = ÿ, ergibt:

$$\ddot{y} = \frac{1}{2}(3 \ddot{y} \ddot{y} \overline{5})$$

$$1 \ddot{y} \ddot{y} = 2 \frac{1}{-}(\ddot{y} \overline{5} \ddot{y} 1)$$

Der Wert von 1 ÿ *ÿ* ÿ *ÿ* oben ist der Goldene Schnitt! Es ermöglicht die Wiederverwendung einer der Funktionsauswertungen aus den vorherigen Iterationen; Ein symmetrisches Argument zeigt, dass die gleiche Wahl von *ÿ* funktioniert, wenn wir das linke Intervall anstelle des rechten entfernt hätten.

Der Golden-Section-Suchalgorithmus nutzt diese Konstruktion (CITE):

1. Nehmen Sie *ÿ* ÿ $\frac{1}{2}$(ÿ $\overline{5}$ ÿ 1). und initialisieren Sie a und b so, dass f unimodular auf [a, b] ist.

2. Machen Sie eine anfängliche Unterteilung x0 = a + (1 ÿ ÿ)(b ÿ a) und x1 = a + ÿ(b ÿ a).

3. Initialisieren Sie f0 = f(x0) und f1 = f(x1).

4. Iterieren Sie, bis b ÿ a ausreichend klein ist:

   (a) Wenn f0 ÿ f1, dann entfernen Sie das Intervall [a, x0] wie folgt:

   • Nach links verschieben: a

   ÿ x0 • Vorherige Iteration wiederverwenden: x0 ÿ

   x1, f0 ÿ f1 • Neue Stichprobe generieren: x1 ÿ a + ÿ(b ÿ a), f1 ÿ f(x1)

   (b) Wenn f1 > f0, dann entfernen Sie das Intervall [x1, b] wie folgt:

   • Nach rechts verschieben: b

   ÿ x1 • Vorherige Iteration wiederverwenden: x1 ÿ

   x0, f1 ÿ f0 • Neue Stichprobe generieren: x0 ÿ a + (1 ÿ ÿ)(b ÿ a), f0 ÿ f(x0)

Dieser Algorithmus konvergiert offensichtlich bedingungslos und linear. Wenn f nicht global unimodal ist, kann es schwierig sein , [a, b] zu finden, sodass f in diesem Intervall unimodal ist, was die Anwendungen dieser Technik etwas einschränkt; Im Allgemeinen wird [a, b] geschätzt, indem versucht wird, ein lokales Minimum von f einzuklammern.

## 8.4 Multivariable Strategien

Wir setzen unsere Parallele zu unserer Diskussion über die Wurzelfindung fort, indem wir unsere Diskussion auf multivariable Probleme erweitern. Wie bei der Wurzelfindung sind multivariable Probleme erheblich schwieriger als Probleme mit einer einzelnen Variablen, kommen aber in der Praxis so oft vor, dass sie eine sorgfältige Betrachtung wert sind.

Wir betrachten hier nur den Fall, dass f : Rn ÿ **R** differenzierbar ist. Optimierungsmethoden, die eher der Suche nach nicht differenzierbaren Funktionen im Goldenen Schnitt ähneln, haben begrenzte Anwendungsmöglichkeiten und sind schwer zu formulieren.

### 8.4.1 Gradientenabstieg

Erinnern Sie sich an unsere vorherige Diskussion, dass ÿ f(x) in die Richtung des „steilsten Anstiegs“ von f bei x zeigt; In ähnlicher Weise ist der Vektor ÿÿ f(x) die Richtung des „steilsten Abstiegs“. Wenn nichts anderes garantiert, garantiert diese Definition, dass wir für kleine *ÿ* > 0 haben müssen, wenn ÿ f(x) =0

$$f(x \ddot{y} \ddot{y}\ddot{y} f(x)) \ddot{y} f(x).$$

Angenommen, unsere aktuelle Schätzung des Ortes des Minimums von f ist xk . Dann möchten wir vielleicht xk+1 wählen, sodass f(xk+1) < f(xk) für eine iterative Minimierungsstrategie ist. Eine Möglichkeit, die Suche nach xk+1 zu vereinfachen , wäre die Verwendung eines unserer eindimensionalen Algorithmen aus §8.3 auf ein einfacheres Problem. Betrachten Sie insbesondere die Funktion gk(t) ÿ f(xk ÿ tÿ f(xk)), die f auf die Gerade durch xk parallel zu ÿ f(xk) beschränkt. Dank unserer Diskussion des Gradienten wissen wir, dass ein kleines t zu einer Verringerung von f führt.

Der Gradientenabstiegsalgorithmus löst diese eindimensionalen Probleme iterativ, um unsere Schätzung von xk zu verbessern :

1. Wählen Sie eine erste Schätzung x0
2. Iteriere bis zur Konvergenz von xk :

   (a) Nehmen Sie gk(t) ÿ f(xk ÿ tÿ f(xk)) (b)

   Verwenden Sie einen eindimensionalen Algorithmus, um t zu ÿ Minimierung von gk über alle t ÿ 0 („Zeilensuche“)

   finden (c) Nehmen Sie xk+1 ÿ xk ÿ t ÿÿ f(xk)

Jede Iteration des Gradientenabstiegs verringert f(xk), sodass die Zielwerte konvergieren. Der Algorithmus endet nur, wenn ÿ f(xk) ÿ 0, was zeigt, dass der Gradientenabfall mindestens ein lokales Minimum erreichen muss; Die Konvergenz ist jedoch für die meisten Funktionen f langsam. Der Liniensuchprozess kann durch eine Methode ersetzt werden, die das Ziel einfach um einen nicht vernachlässigbaren, wenn auch suboptimalen Betrag verringert, obwohl es in diesem Fall schwieriger ist, Konvergenz zu gewährleisten.

### 8.4.2 Newtons Methode

Parallel zu unserer Ableitung des Einzelvariablenfalls können wir eine Taylor-Reihennäherung von f : Rn ÿ **R** unter Verwendung seines hessischen Hf schreiben :

$$f(x) \ddot{y} f(xk) + \ddot{y} f(xk) \cdot (x \ddot{y} xk) + \frac{1}{2} (x \ddot{y} xk) \cdot Hf(xk) \cdot (x \ddot{y} xk)$$

Wenn man nach x differenziert und das Ergebnis gleich Null setzt, erhält man das folgende iterative Schema:

$$x_{k+1} = x_k - [Hf(x_k)]^{-1} \nabla f(x_k)$$

Es lässt sich leicht überprüfen, dass dieser Ausdruck eine Verallgemeinerung des Ausdrucks in §8.3.1 ist, und er konvergiert erneut quadratisch, wenn x0 nahe einem Minimum liegt.

Abhängig vom Optimierungsziel f kann die Newton-Methode effizienter sein als der Gradientenabstieg. Denken Sie daran, dass jede Iteration des Gradientenabstiegs möglicherweise viele Auswertungen von f während der Liniensuchprozedur erfordert. Andererseits müssen wir den Hf -Wert bei jeder Iteration der Newton-Methode auswerten und invertieren. Beachten Sie, dass diese Faktoren keinen Einfluss auf die Anzahl der Iterationen, wohl aber auf die Laufzeit haben: Dies ist ein Kompromiss, der bei herkömmlichen Analysen möglicherweise nicht offensichtlich ist.

Es ist intuitiv, warum die Newton-Methode schnell konvergiert, wenn sie sich einem Optimum nähert. Insbesondere hat der Gradientenabstieg keine Kenntnis von Hf ; Es verhält sich analog zum Bergabgehen, indem man nur auf die Füße schaut. Durch die Verwendung von Hf erhält Newtons Methode ein umfassenderes Bild der Form von f in der Nähe.

Wenn Hf jedoch nicht positiv definit ist, sieht das Ziel lokal eher wie ein Sattel oder eine Spitze als wie eine Schüssel aus. In diesem Fall ist es möglicherweise nicht sinnvoll, zu einem ungefähren stationären Punkt zu springen. Daher könnten adaptive Techniken prüfen, ob Hf positiv definit ist, bevor ein Newton-Schritt angewendet wird. Wenn es nicht positiv definit ist, können die Methoden auf den Gradientenabstieg zurückgreifen, um eine bessere Annäherung an das Minimum zu finden. Alternativ können sie Hf modifizieren , indem sie beispielsweise auf die nächste positiv definite Matrix projizieren.

## 8.4.3 Optimierung ohne Derivate: BFGS

Es kann schwierig sein, die Newton-Methode auf komplizierte Funktionen f : Rn ÿ R anzuwenden. Die zweite Ableitung von f ist möglicherweise wesentlich komplizierter als die Form von f , und Hf ändert sich mit jeder Iteration, was es schwierig macht, Arbeiten aus früheren Iterationen wiederzuverwenden. Darüber hinaus hat Hf die Größe n × n, sodass die Speicherung von Hf O(n) erfordert $^2$) Platz, der inakzeptabel sein kann.

Wie in unserer Erörterung der Wurzelfindung werden Minimierungstechniken, die die Newton-Methode imitieren, aber Näherungsableitungen verwenden, als Quasi-Newton-Methoden bezeichnet. Oftmals können sie ähnlich starke Konvergenzeigenschaften aufweisen, ohne dass bei jeder Iteration eine explizite Neubewertung und sogar Faktorisierung des Hesse-Werts erforderlich ist. In unserer Diskussion werden wir die Entwicklung von (CITE NO CEDAL AND WRIGHT) verfolgen.

Angenommen, wir möchten f : Rn ÿ **R** mithilfe eines iterativen Schemas minimieren. Nahe der aktuellen Schätzung xk der Wurzel, könnten wir f mit einem quadratischen Modell schätzen:

$$f(x_k + \delta x) \approx f(x_k) + \nabla f(x_k) \cdot \delta x + \frac{1}{2} (\delta x)^T B_k(\delta x).$$

Beachten Sie, dass wir verlangt haben, dass unsere Näherung mit f erster Ordnung bei xk übereinstimmt ; Wie bei Broydens Methode zur Wurzelfindung lassen wir jedoch zu, dass unsere Schätzung des hessischen Bk variiert.

Dieses quadratische Modell wird minimiert, indem $\delta x = -B_k^{-1} \nabla f(x_k)$. Falls $\delta x_2$ groß ist und wir keinen so großen Schritt machen möchten, erlauben wir uns, diesen Unterschied um eine Schrittweite $\delta k$ zu skalieren , was ergibt

$$x_{k+1} = x_k - \delta k B_k^{-1} \nabla f(x_k).$$

Unser Ziel ist es, durch Aktualisierung von Bk eine vernünftige Schätzung Bk+1 zu finden , damit wir diesen Vorgang wiederholen können.

Die Hesse-Funktion von f ist nichts anderes als die Ableitung von ÿ f , daher können wir eine Bedingung im Sekantenstil für Bk+1 schreiben :

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k).$$

Wir ersetzensk ÿ xk+1 ÿxk und yk ÿ ÿ f(xk+1) ÿ ÿ f(xk), was eine äquivalente Bedingung Bk+1sk = yk ergibt .

Angesichts der vorliegenden Optimierung wünschen wir uns, dass Bk zwei Eigenschaften hat:

- Bk sollte eine symmetrische Matrix sein, wie die hessische Hf .
- Bk sollte positiv (semi-)definit sein, sodass wir eher nach Minima als nach Maxima suchen Sattelpunkte.

Die Symmetriebedingung reicht aus, um die Möglichkeit der Verwendung der Broyden-Schätzung auszuschließen, die wir im vorherigen Kapitel entwickelt haben.

Die positiv definite Einschränkung stellt implizit eine Bedingung für die Beziehung zwischen sk und dar. Insbesondere wird die Beziehung Bk+1sk = yk mit s Bk+1sk = s k yk vormultipliziert . Für dich . $_k$ zeigt s k Damit Bk+1 positiv definit ist, müssen wir dannsk · yk > 0 haben. Diese Beobachtung kann unsere Wahl von ÿk leiten ; es ist leicht zu erkennen, dass dies für hinreichend kleine ÿk > 0 gilt.

Nehmen Sie an, dass sk und yk unsere Kompatibilitätsbedingung erfüllen. Wenn das erledigt ist, können wir schreiben eine Optimierung im Broyden-Stil, die zu einer möglichen Näherung Bk+1 führt :

$$\begin{aligned} &\text{minimieren}_{B_{k+1}} \ \|B_{k+1} - B_k\|, \text{ so} \\ &\text{dass } B_{k+1} \\ &\qquad B_{k+1}s_k = y_k \end{aligned} = B_{k+1}$$

Zur geeigneten Auswahl von Normen · Iteratives , Diese Optimierung ergibt das bekannte DFP (Davidon Schema von Fletcher-Powell.

Anstatt die Details des DFP-Schemas auszuarbeiten, gehen wir zu einer populäreren Methode über, die als BFGS-Formel (Broyden-Fletcher-Goldfarb-Shanno) bekannt ist und in vielen modernen Systemen vorkommt. Beachten Sie, dass – wenn wir vorerst unsere Wahl von ÿk ignorieren – unsere Näherung zweiter Ordnung minimiert wurde, indem ÿx = ÿB angenommen wurde $\nabla B^{-1}_k \nabla f(x_k)$. Somit ist am Ende das Verhalten unseres iterativen Schemas wird durch die inverse Matrix B k vorgegeben $B^{-1}_k$. Die Annahme, dass Bk+1 ÿ Bk klein ist, kann dennoch relativ große Unterschiede zwischen der Wirkung von B k implizieren $B^{-1}_k$ und das von B ! $B^{-1}_{k+1}$

Unter Berücksichtigung dieser Beobachtung nimmt das BFGS-Schema eine kleine Änderung an der obigen Ableitung vor. Anstatt Bk bei jeder Iteration zu berechnen, können wir seinen Kehrwert Hk ÿ B direkt berechnen . $B^{-1}_k$ Jetzt wird unsere Bedingung Bk+1sk = yk umgekehrt zusk = Hk+1yk ; Die Bedingung, dass Bk symmetrisch ist, ist dasselbe wie die Forderung, dass Hk symmetrisch ist. Wir lösen eine Optimierung

$$\begin{aligned} &\text{minimieren}_{H_{k+1}} \ \|H_{k+1} - H_k\|, \text{ so} \\ &\text{dass } H_{k+1} \\ &\qquad s_k = H_{k+1}y_k \end{aligned} = H_{k+1}$$

Diese Konstruktion hat den schönen Nebenvorteil, dass zur Berechnung von ÿx = ÿHkÿ f(xk) keine Matrixinversion erforderlich ist .

Um eine Formel für Hk+1 abzuleiten , müssen wir uns für eine Matrixnorm entscheiden · . Wie bei unserer vorherigen Diskussion kommt die Frobenius-Norm der Optimierung der kleinsten Quadrate am nächsten, was es wahrscheinlich macht, dass wir einen Ausdruck in geschlossener Form für Hk+1 generieren können, anstatt die obige Minimierung als Unterprogramm der BFGS-Optimierung lösen zu müssen.

Die Frobenius-Norm hat jedoch einen gravierenden Nachteil für hessische Matrizen. Denken Sie daran, dass die Hesse-Matrix Einträge (Hf)ij = *ÿ fi/ÿxj* hat . Oftmals können die Größen xi für verschiedene i unterschiedliche Einheiten haben; Erwägen Sie beispielsweise die Maximierung des Gewinns (in Dollar), der durch den Verkauf eines Cheeseburgers mit dem Radius r (in Zoll) und dem Preis p (in Dollar) erzielt wird, was zu f : (Zoll, Dollar) ÿ Dollar führt. Es macht keinen Sinn, diese unterschiedlichen Größen zu quadrieren und zu addieren.

Angenommen, wir finden eine symmetrische positiv definite Matrix W, so dass Wsk = yk ; Wir werden in den Übungen prüfen, ob eine solche Matrix existiert. Eine solche Matrix nimmt die Einheiten von sk = xk+1 ÿ xk zu denen von yk = ÿ f(xk+1) ÿ ÿ f(xk) auf. Inspiriert von unserem Ausdruck $A = \mathrm{Tr}(A^{\mathrm{her}}A)$ können wir eine gewichtete Frobenius-Norm einer Matrix A definieren als

$$\|A\|_W^2 \ ÿ \ \mathrm{Tr}(A^{\mathrm{her}}WAW)$$

Es ist einfach zu überprüfen, ob dieser Ausdruck konsistente Einheiten hat, wenn er auf unsere Optimierung für Hk+1 angewendet wird . Wenn sowohl W als auch A symmetrisch zu den Spalten w i bzw. ai sind , ergibt die Erweiterung des obigen Ausdrucks:

$$\|A\|_W^2 = ÿ_{ij} (w_i \cdot a_j)(w_j \cdot a_i).$$

Diese Wahl der Norm in Kombination mit der Wahl von W ergibt eine besonders saubere Formel für Hk+1 und yk : gegeben Hk ,sk ,

$$H_{k+1} = (I_{n\times n} ÿ \frac{y_k s_k^{\mathrm{her}}}{y_k^{\mathrm{her}} s_k})H_k(I_{n\times n} ÿ \frac{s_k y_k^{\mathrm{her}}}{y_k^{\mathrm{her}} s_k}) + \frac{y_k y_k^{\mathrm{her}}}{y_k^{\mathrm{her}} s_k},$$

wobei ÿk ÿ 1/y·s. Wir zeigen im Anhang zu diesem Kapitel, wie man diese Formel herleitet.

Der BFGS-Algorithmus vermeidet die Notwendigkeit, eine Hesse-Matrix für f zu berechnen und zu invertieren,erfordert aber dennoch $O(n^2)$ Speicherzum HVariante namens L-BFGS („Limited-Memory BFGS“) vermeidet dieses Problem, indem sie eine begrenzte Historie der Vektoren yk führt andsk und Anwenden von Hk durch rekursive Erweiterung seiner Formel. Dieser Ansatz kann trotz seiner kompakten Raumnutzung tatsächlich bessere numerische Eigenschaften haben; insbesondere sind alte Vektoren yk undsk möglicherweise nicht mehr relevant und sollten ignoriert werden.

## 8.5 Probleme

Ideenliste:

- Leiten Sie Gauss-Newton ab
- Stochastische Methoden, AdaGrad
- VSCG-Algorithmus
- Wolfe-Bedingungen für den Gradientenabstieg; An BFGS anschließen
- Sherman-Morrison-Woodbury-Formel für Bk für BFGS
- Beweisen Sie, dass BFGS konvergiert; zeigen Existenz einer Matrix W
- (Verallgemeinerter) reduzierter Gradientenalgorithmus
- Konditionsnummer zur Optimierung

# Anhang: Ableitung von BFGS Update1

Unsere Optimierung für Hk+1 hat den folgenden Lagrange-Multiplikatorausdruck (zur Vereinfachung der Notation nehmen wir Hk+1 ÿ H und Hk = Hÿ ):

$$\ddot{y}\,\ddot{y}\,\ddot{y}\,(w_i \cdot (hj\,\ddot{y}h_{\ddot{y}_J}))(w_j \cdot (hi\,\ddot{y}h_{\ddot{y}_{\dots}})) \ddot{y}\,\ddot{y}\,\ddot{y}ij(Hij\,\ddot{y}\,Hji)\,\ddot{y}\ddot{y}\,(Hyk\,\ddot{y}sk)$$

$$= \ddot{y}\,(w_i \cdot (hj\,\ddot{y}h_{\ddot{y}_J}))(w_j \cdot (hi\,\ddot{y}h_{\ddot{y}_{\dots}})) \ddot{y}\,\ddot{y}\,\ddot{y}ijHij\,\ddot{y}\ddot{y}\,(Hyk\,\ddot{y}sk) \text{ , wenn wir } \ddot{y}ij = \ddot{y}\ddot{y}ji \text{ annehmen}$$

Die Bildung von Ableitungen zum Finden kritischer Punkte zeigt (für y ÿ yk ,s ÿsk ):

$$0 = \frac{\ddot{y}\ddot{y}}{2\ddot{y}} = \ddot{y}\,2wi(w_j \cdot (h\,\ddot{y}h\,\ddot{y})) \ddot{y}\,\ddot{y}ij\,\ddot{y}\,\ddot{y}iyj\,\ddot{y}Hij =$$

$$wi(W(H\,\ddot{y}\,H\,\ddot{y}))j\,\ddot{y}\,\ddot{y}ij\,\ddot{y}\,\ddot{y}iyj$$

$$= 2\ddot{y}\,(W(H\,\ddot{y}\,H\,\ddot{y}))jwi\ddot{y}\,\ddot{y}ij\,\ddot{y}\,\ddot{y}iyj \text{ aufgrund der Symmetrie von W}$$

$$= 2(W(H\,\ddot{y}\,H\,\ddot{y})W)ji\,\ddot{y}\,\ddot{y}ij\,\ddot{y}\,\ddot{y}iyj$$

$$= 2(W(H\,\ddot{y}\,H\,\ddot{y})W)ij\,\ddot{y}\,\ddot{y}ij\,\ddot{y}\,\ddot{y}iyj \text{ aufgrund der Symmetrie von W und H}$$

In Matrixform haben wir also die folgende Liste von Fakten:

$$0 = 2W(H\,\ddot{y}\,H\,\ddot{y})W\,\ddot{y}\,A\,\ddot{y}\ddot{y}y \text{ , wobei } A_{ij} = \ddot{y}ij$$

$$A = \ddot{y}A,\ W = W,\ H = H,(H\,\ddot{y}_{\ddot{y}}) = H$$

$$Hy = s,\ Ws = y$$

Durch Transposition in Kombination mit der Symmetrie von H und W und der Asymmetrie von A können wir ein Beziehungspaar erreichen:

$$0 = 2W(H\,\ddot{y}\,H\,\ddot{y})W\,\ddot{y}\,A\,\ddot{y}\ddot{y}y$$

$$0 = 2W(H\,\ddot{y}\,H\,\ddot{y})W + A\,\ddot{y}y\ddot{y} = \ddot{y}\ 0 =$$

$$4W(H\,\ddot{y}\,H\,\ddot{y})W\,\ddot{y}\ddot{y}y\,\ddot{y}y\ddot{y}$$

Die nachträgliche Multiplikation dieser Beziehung mit s zeigt:

$$0 = 4(y\,\ddot{y}\,WH\ddot{y}\,y)\,\ddot{y}\ddot{y}(y \cdot s)\,\ddot{y}y(\ddot{y} \cdot s)$$

Nehmen Sie nun das Skalarprodukt mit:

$$0 = 4(y \cdot s)\,\ddot{y}\,4(y\,H\,\ddot{y}\,y)\,\ddot{y}\,2(y \cdot s)(\ddot{y} \cdot s)$$

Das zeigt:

$$\ddot{y} \cdot s = 2\ddot{y}y\,(s\,\ddot{y}\,H\,\ddot{y}\,y), \text{ für } \ddot{y}\,\ddot{y}\,1/y \cdot s$$

1Besonderer Dank geht an Tao Du für das Debuggen mehrerer Teile dieser Ableitung.

Nun setzen wir dies in unsere Vektorgleichung ein:

$$0 = 4(y \ddot{y} WH\ddot{y} y)\ddot{y}\ddot{y}(y \cdot s)\ddot{y}y(\ddot{y} \cdot s) \text{ von vorher} =$$
$$4(y \ddot{y} WH\ddot{y} y)\ddot{y}\ddot{y}(y \cdot s)\ddot{y}y[2\ddot{y}y (s \ddot{y} H \ddot{y} y)] \text{ aus unserer Vereinfachung}$$
$$=\ddot{y}\ \ddot{y} = 4\ddot{y}(y \ddot{y} WH\ddot{y} y)\ddot{y}\ 2\ddot{y}\ \ {}^{2}y (s \ddot{y} H \ddot{y} y)y$$

Die nachträgliche Multiplikation mit y ergibt:

$$\ddot{y}y = 4\ddot{y}(y \ddot{y} WH\ddot{y} y)y \ddot{y}\ 2\ddot{y}\ \ {}^{2}y (s \ddot{y} H \ddot{y} y)yy$$

Nehmen Sie die Transponierung,

$$y\ddot{y} = 4\ddot{y}y(y \ddot{y}y H \ddot{y}W) \ddot{y}\ 2\ddot{y}\ \ {}^{2}y (s \ddot{y} H \ddot{y} y)yy$$

Kombinieren Sie diese Ergebnisse und dividieren Sie sie durch vier Shows:

$$\frac{1}{4}(\ddot{y}y + y\ddot{y}) = \ddot{y}(2yy \ddot{y} WH\ddot{y}\ yy \ddot{y}yy H \ddot{y}W) \ddot{y}\ \ddot{y}\ \ {}^{2}y (s \ddot{y} H \ddot{y} y)yy$$

Jetzt führen wir eine Vor- und Nachmultiplikation mit Wÿ1 durch . Da Ws = y, können wir äquivalent schreiben : = Wÿ1y; außerdem wissen wir dann aufgrund der Symmetrie von W, dass y Wÿ1 = s ist. Die Anwendung dieser Identitäten auf den obigen Ausdruck zeigt:

$$\frac{1}{4}W\ddot{y}1\ (\ddot{y}y + y\ddot{y})W\ddot{y}1 = 2\ddot{y}ss \ddot{y} \ddot{y}H\ {}^{\ddot{y}}ys \ddot{y} \ddot{y}sy H\ {}^{\ddot{y}}\ddot{y}\ \ddot{y}\ {}^{2}(y s)ss + \ddot{y}\ {}^{2}(y H \ddot{y} y)ss$$
$$= 2\ddot{y}ss \ddot{y} \ddot{y}H\ {}^{\ddot{y}}ys \ddot{y} \ddot{y}sy H\ {}^{\ddot{y}}\ddot{y}\ddot{y}ss + s\ddot{y}\ {}^{2}(y H \ddot{y} y)s \text{ nach Definition von } \ddot{y}$$
$$= \ddot{y}ss \ddot{y} \ddot{y}H\ {}^{\ddot{y}}ys \ddot{y} \ddot{y}sy H\ {}^{\ddot{y}} + s\ddot{y}\ {}^{2}(y H \ddot{y} y)s$$

Abschließend können wir unsere Ableitung des BFGS-Schritts wie folgt abschließen:

$$0 = 4W(H \ddot{y} H \ddot{y})W\ \ddot{y}\ddot{y}y\ \ddot{y}y\ddot{y} \text{ von vorher}$$
$$=\ddot{y}\ H = 4\ {}_{=}\frac{1}{-}W\ddot{y}1\ (\ddot{y}y + y\ddot{y})W\ddot{y}1 + H\ {}^{\ddot{y}}$$
$$\ddot{y}ss \ddot{y} \ddot{y}H\ {}^{\ddot{y}}ys \ddot{y} \ddot{y}sy H\ {}^{\ddot{y}} + s\ddot{y}\ {}^{2}(y H \ddot{y} y)s + H\ {}^{\ddot{y}} \text{ aus dem letzten Absatz}$$
$$= H\ \ddot{y}\ (I \ddot{y} \ddot{y}ys) + \ddot{y}ss \ddot{y} \ddot{y}sy H \ddot{y}\ (I \ddot{y}\ {}^{\ddot{y}} + (\ddot{y}sy)H \ddot{y}\ (\ddot{y}ys)$$
$$= H\ \ddot{y}ys) + \ddot{y}ss \ddot{y} \ddot{y}sy H \ddot{y}\ (I \ddot{y} \ddot{y}ys) = \ddot{y}ss + (I \ddot{y}$$
$$\ddot{y}sy)H \ddot{y}\ (I \ddot{y} \ddot{y}ys)$$

Dieser letzte Ausdruck ist genau der im Kapitel vorgestellte BFGS-Schritt.

# Kapitel 9

# Optimierungsprobleme

Wir setzen unsere Betrachtung von Optimierungsproblemen fort, indem wir den eingeschränkten Fall untersuchen. Diese Probleme haben die folgende allgemeine Form:

$$\text{Minimiere } f(x) \text{ so,}$$
$$\text{dass } g(x) = 0 \; h(x) \ddot{y} 0$$

Hier gilt f : Rn ÿ R, g : Rn ÿ Rm und h : Rn ÿ Rp . Offensichtlich ist diese Form äußerst generisch, sodass es nicht schwer vorherzusagen ist, dass Algorithmen zur Lösung solcher Probleme ohne zusätzliche Annahmen zu f, g oder h schwer zu formulieren sein können und Entartungen wie lokalen Minima und dem Fehlen von solchen unterliegen Konvergenz. Tatsächlich kodiert diese Optimierung andere Probleme, die wir bereits berücksichtigt haben; Wenn wir f(x) ÿ 0 annehmen, wird diese eingeschränkte Optimierung zur Wurzelfindung auf g, während sie sich, wenn wir g(x) = h(x) ÿ0 annehmen, auf eine uneingeschränkte Optimierung auf f reduziert.

Trotz dieser etwas düsteren Aussichten können Optimierungen für allgemeine eingeschränkte Fälle wertvoll sein, wenn f, g und h keine nützliche Struktur haben oder zu spezialisiert sind, um eine spezielle Behandlung zu verdienen. Wenn f ohnehin heuristisch ist, ist es darüber hinaus wertvoll , einfach ein zulässiges x zu finden, für das f(x) < f(x0) für eine anfängliche Schätzung x0 gilt. Eine einfache Anwendung in diesem Bereich wäre ein Wirtschaftssystem, in dem f die Kosten misst; Natürlich möchten wir die Kosten minimieren, aber wenn x0 die aktuelle Konfiguration darstellt, ist jedes x, das f verringert, eine wertvolle Ausgabe.

## 9.1 Motivation

In der Praxis ist es nicht schwierig, auf eingeschränkte Optimierungsprobleme zu stoßen. Tatsächlich haben wir viele Anwendungen dieser Probleme bereits aufgelistet, als wir Eigenvektoren und Eigenwerte besprochen haben, da dieses Problem so gestellt werden kann, dass kritische Punkte von x Ax unter der Bedingung x2 = 1 gefunden werden; Natürlich gibt es für den besonderen Fall der Eigenwertberechnung spezielle Algorithmen, die das Problem einfacher machen.

Hier listen wir andere Optimierungen auf, die nicht die Struktur von Eigenwertproblemen aufweisen:

**Beispiel 9.1** (Geometrische Projektion). Viele Flächen S im R3 können für ein g implizit in der Form g(x) = 0 geschrieben werden . Beispielsweise ergibt sich die Einheitskugel aus der Annahme g(x) ÿ x ÿ 1, während dies $\frac{2}{2}$bei einem Würfel der Fall ist

kann durch die Annahme von g(x) = x1 ÿ 1 konstruiert werden . Tatsächlich erlauben es einige 3D-Modellierungsumgebungen Benutzern, „blobby“-Objekte, wie in Abbildung NUMBER, als Summen anzugeben

$$g(x) \ddot{y} c + \ddot{y} \quad \ddot{y}bixÿxi aie \quad {}^{2}_{2}.$$

Angenommen, wir erhalten einen Punkt y ÿ R3 und möchten den Punkt auf S finden, der y am nächsten liegt. Dieses Problem wird durch die folgende eingeschränkte Minimierung gelöst:

$$\text{minimierex } x \ddot{y}y2 \text{ , so}$$
$$\text{dass } g(x) = 0$$

**Beispiel 9.2** (Fertigung). Angenommen, Sie haben m verschiedene Materialien; Sie haben Si- Einheiten von jedem Material auf Lager. Sie können k verschiedene Produkte herstellen; Produkt j bringt Ihnen Gewinn pj und verwendet cij des Materials i für die Herstellung. Um den Gewinn zu maximieren, können Sie die folgende Optimierung für die Gesamtmenge xj durchführen , die Sie von jedem Artikel j herstellen sollten:

$$\text{maximierenx } \sum_{j=1}^{k} pj\ddot{y}xj$$
$$\text{so dass } xj \ddot{y} 0 \ddot{y}j \ddot{y} \{1, \ldots, k\}$$
$$\sum_{j=1}^{k} \ddot{y} cijxj \ddot{y} si \ddot{y}i \ddot{y} \{1, \ldots, m\}$$

Die erste Einschränkung stellt sicher, dass Sie bei keinem Produkt negative Mengen herstellen, und die zweite stellt sicher, dass Sie von jedem Material nicht mehr als Ihren Lagerbestand verbrauchen.

**Beispiel 9.3** (Nichtnegative kleinste Quadrate). Wir haben bereits zahlreiche Beispiele für Kleinste-Quadrate-Probleme gesehen, aber manchmal sind negative Werte im Lösungsvektor möglicherweise nicht sinnvoll. In der Computergrafik könnte ein animiertes Modell beispielsweise als sich verformende Knochenstruktur plus vernetzter „Haut“ ausgedrückt werden; Für jeden Punkt auf der Haut kann eine Liste von Gewichten berechnet werden, um den Einfluss der Positionen der Knochengelenke auf die Position der Hautscheitelpunkte zu approximieren (CITE). Solche Gewichte sollten auf nichtnegative Werte beschränkt werden, um ein degeneriertes Verhalten bei der Verformung der Oberfläche zu vermeiden. In einem solchen Fall können wir das Problem der „nichtnegativen kleinsten Quadrate“ lösen:

$$\text{minimierenx } Ax \ddot{y}b2 \text{ , so}$$
$$\text{dass } xi \ddot{y} 0 \ddot{y}i$$

Neuere Forschungen umfassen die Charakterisierung der Sparsität von nichtnegativen Lösungen der kleinsten Quadrate, die oft mehrere Werte xi haben, die xi = 0 genau erfüllen (CITE).

**Beispiel 9.4** (Bündelanpassung). Nehmen wir beim Computer Vision an, dass wir ein Objekt aus mehreren Winkeln fotografieren. Eine natürliche Aufgabe besteht darin, die dreidimensionale Form des Objekts zu rekonstruieren. Dazu könnten wir auf jedem Bild eine entsprechende Menge von Punkten markieren; Insbesondere können wir xij ÿ R2 als die Position des Merkmalspunkts j auf dem Bild i annehmen. In Wirklichkeit hat jeder Merkmalspunkt eine Position yj ÿ R3 im Raum, die wir berechnen möchten. Darüber hinaus müssen wir die Positionen der Kameras selbst finden, die wir als darstellen können

unbekannte Projektionsmatrizen Pi . Dieses als Bündelanpassung bekannte Problem kann mit einer Optimierungsstrategie angegangen werden:

$$\text{minimieren}_{yj,Pi} \sum_{ij} \left\| Piyj - xij \right\|_2^2$$

so dass Pi orthogonal ist

Die Orthogonalitätsbeschränkung stellt sicher, dass die Kameratransformationen sinnvoll sind.

## 9.2 Theorie der eingeschränkten Optimierung

In unserer Diskussion gehen wir davon aus, dass f, g und h differenzierbar sind. Es gibt einige Methoden, die nur schwache Kontinuitäts- oder Lipschitz-Annahmen zugrunde legen, aber diese Techniken sind ziemlich spezialisiert und erfordern fortgeschrittene analytische Überlegungen.

Obwohl wir noch keine Algorithmen für die allgemeine Optimierung unter Nebenbedingungen entwickelt haben, haben wir bei der Betrachtung von Eigenwertmethoden implizit die Theorie solcher Probleme genutzt. Erinnern Sie sich insbesondere an die Methode der Lagrange-Multiplikatoren, die in Satz 0.1 eingeführt wurde. Bei dieser Technik werden kritische Punkte f(x) , die g(x) unterliegen, als kritische Punkte der unbeschränkten La-Grange-Multiplikatorfunktion ÿ(x,ÿ) ÿ f(x) *ÿÿ* · g(x) in Bezug auf beide ÿ charakterisiert und x gleichzeitig.
Dieser Satz ermöglichte es uns, Variationsinterpretationen von Eigenwertproblemen bereitzustellen; Allgemeiner gesagt gibt es ein alternatives (notwendiges, aber nicht ausreichendes) Kriterium dafür, dass x ein kritischer Punkt einer gleichheitsbeschränkten Optimierung ist.

Es kann jedoch eine erhebliche Herausforderung sein, einfach ein x zu finden, das die Einschränkungen erfüllt. Wir können diese Probleme trennen, indem wir einige Definitionen vornehmen:

**Definition 9.1** (Zulässiger Punkt und zulässige Menge). Ein zulässiger Punkt eines eingeschränkten Optimierungsproblems ist jeder Punkt x, der g(x) =0 und h(x) ÿ0 erfüllt. Die zulässige Menge ist die Menge aller Punkte x, die diese Einschränkungen erfüllen.

**Definition 9.2** (Kritischer Punkt der eingeschränkten Optimierung). Ein kritischer Punkt einer eingeschränkten Optimierung ist einer, der die Einschränkungen erfüllt und gleichzeitig ein lokales Maximum, Minimum oder Sattelpunkt von f innerhalb der zulässigen Menge ist.

Beschränkte Optimierungen sind schwierig, weil sie gleichzeitig Wurzelfindungsprobleme (die g(x) =0-Einschränkung), Erfüllbarkeitsprobleme (die h(x) ÿ0-Einschränkung) und Minimierung (die Funktion f) lösen. Abgesehen davon müssen wir, um unsere Differentialtechniken vollständig allgemeingültig zu machen, einen Weg finden, Ungleichheitsbeschränkungen zum Lagrange-Multiplikatorsystem hinzuzufügen. Angenommen, wir haben das Minimum der Optimierung gefunden, das mit x bezeichnet wird. Für jede Ungleichheitsbedingung hi(x ÿ ) ÿ 0 haben wir zwei Möglichkeiten: ÿ

- hi(x ÿ ) = 0: Eine solche Einschränkung ist aktiv, was wahrscheinlich darauf hindeutet, dass sich das Optimum ändern könnte, wenn die Einschränkung entfernt würde.
- hi(x ÿ ) > 0: Eine solche Einschränkung ist inaktiv, was bedeutet, dass wir in einer Umgebung von x ÿ auch dann das gleiche Minimum erreicht hätten, wenn wir diese Einschränkung entfernt hätten.

Natürlich wissen wir erst nach der Berechnung, welche Einschränkungen bei x ÿ aktiv oder inaktiv sein werden.

Wenn alle unsere Einschränkungen aktiv wären, könnten wir unsere Einschränkung h(x) ÿ0 in eine Gleichheit ändern, ohne das Minimum zu beeinflussen. Dies könnte dazu motivieren, das folgende Lagrange-Multiplikatorsystem zu studieren:

$$ÿ(x,ÿ,\mu) ÿ f(x) ÿÿ \cdot g(x) ÿ\mu \cdot h(x)$$

Wir können jedoch nicht mehr sagen, dass x ein kritischer Punkt von ÿ ist, da inaktive Einschränkungen die oben genannten Terme entfernen würden. Wenn wir diese (wichtige!) Frage vorerst ignorieren, könnten wir blind vorgehen und nach kritischen Punkten dieses neuen ÿ in Bezug auf x fragen, die Folgendes erfüllen:

$$0 = ÿ f(x) ÿ ÿ ÿiÿgi(x) ÿ ÿ µjÿhj(x)$$

Hier haben wir die einzelnen Komponenten von g und h herausgetrennt und als Skalarfunktionen behandelt, um eine komplexe Notation zu vermeiden.

Ein cleverer Trick kann diese Optimalitätsbedingung auf ungleichheitsbeschränkte Systeme erweitern. Beachten Sie, dass, wenn wir µj = 0 angenommen hätten, wann immer hj inaktiv ist, dies die irrelevanten Terme aus den Optimalitätsbedingungen entfernt. Mit anderen Worten, wir können den Lagrange-Multiplikatoren eine Einschränkung hinzufügen:

$$µjhj(x) = 0.$$

Mit dieser Einschränkung wissen wir, dass mindestens eines von µj und hj(x) Null sein muss, und daher gilt unsere Optimalitätsbedingung erster Ordnung immer noch!

Bisher hat unsere Konstruktion nicht zwischen der Einschränkung hj(x) ÿ 0 und der Einschränkung hj(x) ÿ 0 unterschieden. Wenn die Einschränkung inaktiv wäre, hätte sie weggelassen werden können, ohne das Ergebnis der Optimierung lokal zu beeinflussen, so wir Betrachten Sie den Fall, dass die Einschränkung aktiv ist. Intuitiv[1] gehen wir in diesem Fall davon aus, dass es eine Möglichkeit gibt, f durch Verletzung der Einschränkung zu verringern. Lokal gesehen ist die Richtung, in der f abnimmt, ÿÿ f(x ÿ ) und die Richtung, in der hj abnimmt, ist ÿÿhj(x ÿ ). Beginnend bei x können wir f noch weiter verringern, indem wir die Einschränkung hj(x) ÿ 0 verletzen, wenn ÿ f(x ÿ ) · ÿhj(x ÿ ) ÿ 0.

Natürlich ist es schwierig, mit Produkten der Gradienten von f und hj zu arbeiten. Bedenken Sie jedoch, dass unsere Optimalitätsbedingung erster Ordnung Folgendes besagt:

$$ÿ f(x ÿ ) = ÿ \quad ÿ_{i\ ÿgi(x ÿ ) + ÿ} \quad \mu_{j \text{ aktiv}}\ _{j\ ÿhj(x ÿ )}$$

Die inaktiven µj- Werte sind Null und können entfernt werden. Tatsächlich können wir die Einschränkungen für g(x) = 0 entfernen, indem wir die Ungleichheitsbeschränkungen g(x) ÿ 0 und g(x) ÿ 0 zu h hinzufügen; Dies ist eher eine mathematische Erleichterung beim Verfassen eines Beweises als ein numerisches Manöver. Dann ergibt die Bildung von Skalarprodukten mit ÿhk für jedes feste k:

$$ÿ_{\mu j \text{ aktiv}}\ _{j} ÿhj(x ÿ ) · ÿhk(x ÿ ) = ÿ f(x ÿ ) · ÿhk(x ÿ ) ÿ 0$$

Die Vektorisierung dieses Ausdrucks zeigt Dh(x ÿ )Dh(x ÿ ) $\mu$ inite, ÿ0. Da Dh(x ÿ )Dh(x ÿ ) positiv semidef ist dies impliziert $\mu$ die ÿ 0. Somit manifestiert sich die Beobachtung ÿ f(x ÿ ) · ÿhj(x ÿ ) ÿ 0 einfach durch Tatsache, dass µj ÿ 0.

Unsere Beobachtungen können formalisiert werden, um eine Optimalitätsbedingung erster Ordnung für ungleichheitsbeschränkte Optimierungen zu beweisen:

---

[1] Sie sollten unsere Diskussion nicht als formalen Beweis betrachten, da wir nicht viele Grenzfälle betrachten.

**Satz 9.1** (Karush-Kuhn-Tucker (KKT)-Bedingungen). Der Vektor x ÿ Rn ist ein kritischer Punkt ÿ für die Minimierung von f, sofern g(x) =0 und h(x) ÿ0 gilt, wenn ÿ ÿ Rm und $\mu$ ÿ Rp existieren, so dass:

- 0 = ÿ f(x ÿ ) ÿ ÿi ÿiÿgi(x ÿ ) ÿ ÿj µjÿhj(x ÿ ) („Stationarität“)
- g(x ÿ ) =0 und h(x ÿ ) ÿ0 („primale Machbarkeit“) • µjhj(x ÿ ) = 0 für alle j („komplementäre Slackness“) • µj ÿ 0 für alle j („duale Machbarkeit “) ")

Beachten Sie, dass sich dieser Satz auf das Lagrange-Multiplikatorkriterium reduziert, wenn h entfernt wird.

**Beispiel 9.5** (Einfache Optimierung2 ). Angenommen, wir möchten eine Lösung finden

$$\begin{aligned} &\text{Maximiere } xy, \\ &\text{sodass } x + yx, y \ddot{y}^{2} \ddot{y} 2 \\ &0 \end{aligned}$$

In diesem Fall haben wir keine ÿs und drei µs. Wir nehmen f(x, y) = ÿxy, h1(x, y) ÿ 2ÿ x ÿ y und h3(x, y) = y. Die $^{2}$, h2(x, y) = x, KKT-Bedingungen sind:

$$\begin{aligned} \text{Stationarität: } 0 &= \ddot{y}y + \mu 1 \ddot{y} \\ \mu 2\ 0 &= \ddot{y}x + 2\mu 1y \ddot{y} \mu 3 \\ \text{Ursprüngliche Machbarkeit: } x^{2} &\ \ddot{y}\ 2 \\ &+ yx, y \ddot{y} 0 \\ \text{Komplementäres Spiel: } \mu 1(2 \ddot{y} x \ddot{y} y\ \mu 2x = 0\ \mu 3y = ^{2}) &= 0 \\ &0 \end{aligned}$$

Duale Machbarkeit: µ1, µ2, µ3 ÿ 0

**Beispiel 9.6** (Lineare Programmierung). Betrachten Sie die Optimierung:

$$\begin{aligned} &\text{minimierex } b \cdot x \\ &\text{so dass } Ax \ddot{y} c \end{aligned}$$

Beachten Sie, dass Beispiel 9.2 auf diese Weise geschrieben werden kann. Die KKT-Bedingungen für dieses Problem sind:

$$\begin{aligned} \text{Stationarität: } &A\mu = b \\ \text{Ursprüngliche Machbarkeit: } &Ax \ddot{y} c \\ \text{Komplementärer Spielraum: } &\mu i(ai \cdot x \ddot{y} ci) = 0 \ \ddot{y}i, \text{ wobei } \quad \text{ist Zeile i von A} \\ \text{Doppelte Machbarkeit: } &\mu \ddot{y}0 \end{aligned}$$

Wie im Fall der Lagrange-Multiplikatoren können wir nicht davon ausgehen, dass jedes x ÿ , das die KKT-Bedingungen erfüllt, f vorbehaltlich der Einschränkungen automatisch minimiert, auch nicht lokal. Eine Möglichkeit, die lokale Optimalität zu überprüfen, besteht darin, die Hesse-Funktion von f zu untersuchen, die auf den Unterraum von Rn beschränkt ist , in dem sich x bewegen kann, ohne die Einschränkungen zu verletzen; Wenn dieser „reduzierte“ Hessesche Wert positiv definit ist, hat die Optimierung ein lokales Minimum erreicht.

---

2Von http://www.math.ubc.ca/~israel/m340/kkt2.pdf

# 9.3 Optimierungsalgorithmen

Eine sorgfältige Betrachtung von Algorithmen zur eingeschränkten Optimierung liegt außerhalb des Rahmens unserer Diskussion; Glücklicherweise gibt es viele stabile Implementierungen dieser Techniken und vieles kann als „Client“ dieser Software erstellt werden, anstatt sie von Grund auf neu zu schreiben. Dennoch ist es sinnvoll, einige mögliche Ansätze zu skizzieren, um einen Einblick in die Funktionsweise dieser Bibliotheken zu gewinnen.

## 9.3.1 Sequentielle quadratische Programmierung (SQP)

Ähnlich wie bei BFGS und anderen Methoden, die wir in unserer Diskussion der uneingeschränkten Optimierung betrachtet haben, besteht eine typische Strategie für die eingeschränkte Optimierung darin, f, g und h mit einfacheren Funktionen zu approximieren, die angenäherte Optimierung zu lösen und zu iterieren.

Angenommen, wir haben eine Schätzung xk der Lösung des eingeschränkten Optimierungsproblems. Wir könnten eine Taylor-Entwicklung zweiter Ordnung auf f und eine Näherung erster Ordnung auf g und h anwenden, um eine nächste Iteration wie folgt zu definieren:

$$xk+1 \ddot{y} xk+ \arg\min_{D} \frac{1}{2} dHf(xk)d + \ddot{y} f(xk) \cdot d + f(xk)$$

$$\text{so dass } gi(xk) + \ddot{y}gi(xk) \cdot d = 0 \; hi(xk)$$

$$+ \ddot{y}hi(xk) \cdot d \ddot{y} 0$$

Die Optimierung zum Finden von d hat ein quadratisches Ziel mit linearen Einschränkungen, für die die Optimierung mit einer von vielen Strategien erheblich einfacher sein kann. Es ist als quadratisches Programm bekannt.

Natürlich funktioniert diese Taylor-Näherung nur in der Umgebung des optimalen Punktes. Wenn eine gute anfängliche Schätzung x0 nicht verfügbar ist, werden diese Strategien wahrscheinlich scheitern.

**Gleichheitsbeschränkungen** Wenn die einzigen Beschränkungen Gleichheiten sind und h entfernt wird, weist das quadratische Programm für d die Lagrange-Multiplikator-Optimalitätsbedingungen auf, die wie folgt abgeleitet werden:

$$1 \ddot{y}(d,\ddot{y}) \ddot{y} \quad \frac{1}{2} dHf(xk)d + \ddot{y} f(xk) \cdot d + f(xk) + \ddot{y} (g(xk) + Dg(xk)d)$$

$$= \ddot{y} 0 = \ddot{y}d\ddot{y} = Hf(xk)d + \ddot{y} f(xk) + [Dg(xk)]\ddot{y}$$

Kombiniert man dies mit der Gleichheitsbedingung, erhält man ein lineares System:

$$\begin{bmatrix} Hf(xk) & [Dg(xk)] \\ Dg(xk) & 0 \end{bmatrix} \begin{matrix} D \\ \ddot{y} \end{matrix} = \begin{matrix} \ddot{y}\ddot{y} f(xk) \\ \ddot{y}g(xk) \end{matrix}$$

Somit kann jede Iteration der sequentiellen quadratischen Programmierung bei Vorhandensein nur von Gleichheitsbeschränkungen erreicht werden, indem dieses lineare System bei jeder Iteration gelöst wird, um xk+1 ÿ xk + d zu erhalten. Es ist wichtig zu beachten, dass das obige lineare System nicht positiv definit ist und daher im großen Maßstab schwierig zu lösen sein kann.

Erweiterungen dieser Strategie funktionieren als BFGS und ähnliche Näherungen funktionieren für eine uneingeschränkte Optimierung, indem Näherungen des Hessischen Hf eingeführt werden . Stabilität kann auch durch die Begrenzung der in einer einzelnen Iteration zurückgelegten Distanz erreicht werden.

**Ungleichungsbeschränkungen** Es gibt spezielle Algorithmen zum Lösen quadratischer Programme anstelle allgemeiner nichtlinearer Programme, und diese können zum Generieren von SQP-Schritten verwendet werden. Eine bemerkenswerte Strategie besteht darin, einen „aktiven Satz“ von Einschränkungen beizubehalten, die in Bezug auf d auf dem Minimum aktiv sind; Dann können die oben genannten gleichheitsbeschränkten Methoden angewendet werden, indem inaktive Einschränkungen ignoriert werden. Iterationen der Active-Set-Optimierung aktualisieren den aktiven Constraint-Satz, indem verletzte Constraints zum aktiven Set hinzugefügt und diejenigen Ungleichheits-Constraints hj entfernt werden , für die ÿ f · ÿhj ÿ 0 ist, wie in unserer Diskussion der KKT-Bedingungen.

### 9.3.2 Barrieremethoden

Eine weitere Möglichkeit zur Minimierung bei Vorliegen von Einschränkungen besteht darin, die Einschränkungen in Energieterme umzuwandeln. Im Fall der Gleichheitsbeschränkung könnten wir beispielsweise ein „erweitertes“ Ziel wie folgt minimieren:

$$f\ddot{y}(x) = f(x) + \ddot{y}g(x) \quad {}^{2}_{2}$$

Beachten Sie, dass die Annahme von *ÿ* ÿ ÿ dazu führt, dass g(x) so klein wie möglich ist, sodass wir schließlich g(x) ÿ0 erreichen. Daher wendet die Barrieremethode der eingeschränkten Optimierung iterative uneingeschränkte Optimierungstechniken auf *fÿ* an und prüft, wie gut die Einschränkungen erfüllt sind; Wenn sie nicht innerhalb einer bestimmten Toleranz liegen, wird *ÿ* erhöht und die Optimierung wird unter Verwendung der vorherigen Iteration als Ausgangspunkt fortgesetzt.

Barrieremethoden sind einfach zu implementieren und zu verwenden, können jedoch einige schädliche Fehlermodi aufweisen. Insbesondere nimmt mit zunehmendem *ÿ* der Einfluss von f auf die Zielfunktion ab und die Hesse-Funktion von *fÿ* wird immer schlechter konditioniert.

Barrieremethoden können auch auf Ungleichheitsbeschränkungen angewendet werden. Hier müssen wir sicherstellen, dass hi(x) ÿ 0 für alle i; Typische Optionen für Barrierefunktionen könnten 1/hi(x) (die „inverse Barriere“) oder ÿ log hi(x) (die „logarithmische Barriere“) sein.

## 9.4 Konvexe Programmierung

Im Allgemeinen bieten Methoden wie die, die wir für die eingeschränkte Optimierung beschrieben haben, nur wenige oder gar keine Garantien für die Qualität der Ausgabe. Sicherlich sind diese Methoden nicht in der Lage, globale Minima ohne eine gute anfängliche Schätzung x0 zu erhalten, und in bestimmten Fällen, z. B. wenn die Hesse-Methode verwendet wird
in der Nähe von $x^{\ddot{y}}$ nicht positiv definit ist, konvergieren sie möglicherweise überhaupt nicht.

Es gibt eine bemerkenswerte Ausnahme von dieser Regel, die in vielen wichtigen Optimierungen vorkommt: die konvexe Programmierung. Die Idee dabei ist, dass die Optimierung ein eindeutiges Minimum besitzt, wenn f eine konvexe Funktion ist und die zulässige Menge selbst konvex ist. Wir haben bereits eine konvexe Funktion definiert, müssen aber verstehen, was es für eine Reihe von Einschränkungen bedeutet
konvex:

**Definition 9.3** (Konvexe Menge). Eine Menge S ÿ Rn ist konvex, wenn für jedes x,y ÿ S der Punkt tx + (1 ÿ t)y auch für jedes t ÿ [0, 1] in S liegt.

Wie in Abbildung NUMMER gezeigt, ist eine Menge intuitiv konvex, wenn sich ihre Grenzform nicht sowohl nach innen als auch nach außen biegen kann.

**Beispiel 9.7** (Kreise). Die Scheibe {x ÿ Rn : x2 ÿ 1} ist konvex, der Einheitskreis {x ÿ Rn : x2 = 1} hingegen nicht.

Es ist leicht zu erkennen, dass eine konvexe Funktion ein eindeutiges Minimum hat, selbst wenn diese Funktion auf einen konvexen Bereich beschränkt ist. Insbesondere wenn die Funktion zwei lokale Minima hätte, dann darf die Punktlinie zwischen diesen Minima Werte von f ergeben, die nicht größer sind als die an den Endpunkten.

Für konvexe Optimierungen stehen starke Konvergenzgarantien zur Verfügung, die das Finden des globalen Minimums garantieren, solange f konvex ist und die Einschränkungen für g und h eine konvexe zulässige Menge ergeben. Daher ist es eine wertvolle Übung für fast jedes Optimierungsproblem, zu prüfen, ob es konvex ist, da eine solche Beobachtung das Vertrauen in die Lösungsqualität und die Erfolgsaussichten um einen großen Faktor erhöhen kann.

Ein neues Gebiet namens disziplinierte konvexe Programmierung versucht, einfache Regeln zur Konvexität zu verketten, um konvexe Optimierungen zu generieren (CITE CVX), wodurch der Endbenutzer einfache konvexe Energieterme und Einschränkungen kombinieren kann, solange sie Kriterien erfüllen, die die endgültige Optimierung konvex machen. Zu den nützlichen Aussagen zur Konvexität in diesem Bereich gehören die folgenden:

- Der Durchschnitt konvexer Mengen ist konvex; Daher ist das Hinzufügen mehrerer konvexer Einschränkungen eine zulässige Operation.
- Die Summe konvexer Funktionen ist konvex.
- Wenn f und g konvex sind, gilt auch h(x) ÿ max{ f(x), g(x)}.
- Wenn f eine konvexe Funktion ist, ist die Menge {x : f(x) ÿ c} konvex.

Tools wie die CVX-Bibliothek helfen dabei, die Implementierung verschiedener konvexer Ziele von deren Minimierung zu trennen.

**Beispiel 9.8** (Konvexe Programmierung).

- Das nichtnegative Kleinste-Quadrate-Problem in Beispiel 9.3 ist konvex, weil Ax ÿb2 konvex ist Funktion von x und die Menge x ÿ0 ist konvex.
- Das lineare Programmierproblem in Beispiel 9.6 ist konvex, weil es ein lineares Ziel hat und linear ist Einschränkungen.
- Wir können x1 in ein konvexes Optimierungsziel einbeziehen, indem wir eine Variable y einführen. Dazu fügen wir die Einschränkungen yi ÿ xi und yi ÿ ÿxi für jedes i und ein Ziel ÿi yi hinzu . Diese Summe hat Terme, die mindestens so groß sind wie |xi | und dass die Energie und die Beschränkungen konvex sind. Im Minimum müssen wir yi = |xi | haben da wir yi ÿ |xi | eingeschränkt haben und wir möchten den Energieverbrauch minimieren. „Disziplinierte“ konvexe Bibliotheken können solche Operationen im Hintergrund durchführen, ohne dass solche Ersetzungen dem Endbenutzer offengelegt werden.

Ein besonders wichtiges Beispiel für eine konvexe Optimierung ist die lineare Programmierung aus Beispiel 9.6.
Der berühmte Simplex-Algorithmus verfolgt die aktiven Einschränkungen, löst das resultierende x mithilfe eines linearen ÿ Systems auf und prüft, ob die aktive Menge aktualisiert werden muss. Es sind keine Taylor-Näherungen erforderlich, da die objektive und zulässige Menge durch lineare Maschinen gegeben sind. Für diese Probleme sind auch interne punktlineare Programmierstrategien wie die Barrieremethode erfolgreich. Aus diesem Grund können lineare Programme in großem Maßstab gelöst werden – bis zu Millionen oder Milliarden von Variablen! – und tauchen häufig bei Problemen wie Terminplanung oder Preisgestaltung auf.

## 9.5 Probleme

- Simplex ableiten?
- Dualität der linearen Programmierung

# Kapitel 10

# Iterative lineare Löser

In den beiden vorherigen Kapiteln haben wir Strategien zur Lösung einer neuen Klasse von Problemen entwickelt, bei denen es um die Minimierung einer Funktion f(x) mit oder ohne Einschränkungen für x geht. Dabei haben wir unseren Standpunkt von der numerischen linearen Algebra und insbesondere der Gaußschen Eliminierung, dass wir eine exakte Lösung für ein Gleichungssystem finden müssen, gelockert und uns stattdessen iterativen Schemata zugewandt, die das Minimum einer Funktion garantiert immer besser annähern iteriere immer mehr. Selbst wenn wir das Minimum nie genau finden, wissen wir, dass wir irgendwann ein x0 mit f(x0) ÿ0 mit beliebiger Ebenenqualität finden werden , abhängig von der Anzahl der von uns ausgeführten Iterationen.

Wir haben jetzt eine Wiederholung unseres Lieblingsproblems aus der numerischen linearen Algebra, nämlich die Lösung von Ax =b für x, wenden jedoch einen iterativen Ansatz an, anstatt zu erwarten, eine Lösung in geschlossener Form zu finden. Diese Strategie offenbart eine neue Klasse linearer Systemlöser, die in erstaunlich wenigen Iterationen zuverlässige Näherungen für x finden können. Wir haben bereits in unserer Diskussion der linearen Algebra vorgeschlagen, wie wir dies angehen können, indem wir vorgeschlagen haben, dass Lösungen für lineare Systeme unter anderem Minima der Energie Ax ÿb sind . 22,

Warum sollte man sich die Mühe machen, eine weitere Klasse linearer Systemlöser abzuleiten? Bisher erfordern die meisten unserer direkten Ansätze, dass wir A als vollständige n × n-Matrix darstellen , und Algorithmen wie LU, QR oder Cholesky-Faktorisierung erfordern alle etwa O(n) Gründe, iterative Schemata auszuprobieren: 3) Zeit. Bei Poten sind zwei Fälle zu beachten

1. Wenn A dünn besetzt ist, neigen Methoden wie die Gaußsche Eliminierung dazu, eine Füllung zu induzieren, was bedeutet, dass sogar ) Wenn A O(n) Werte ungleich Null enthält, können Zwischenschritte der Eliminierung O(n Werte ungleich Null einführen. Diese 2 Eigenschaft kann schnell dazu führen, dass lineare Algebrasysteme nicht mehr über genügend Speicher verfügen. Im Gegensatz dazu erfordern die Algorithmen in diesem Kapitel nur, dass Sie sie anwenden können A in Vektoren, was zeitlich proportional zur Anzahl der Werte ungleich Null in einer Matrix erfolgen kann.
2. Wir möchten vielleicht das O(n) besiegen 3) Laufzeit von Standardtechniken zur Matrixfaktorisierung. Insbesondere wenn ein iteratives Schema in wenigen Iterationen eine ziemlich genaue Lösung für Ax = b finden kann, können die Laufzeiten erheblich verkürzt werden.

Beachten Sie außerdem, dass viele der von uns besprochenen nichtlinearen Optimierungsmethoden, insbesondere diejenigen, die auf einem Newton-ähnlichen Schritt basieren, die Lösung eines linearen Systems in jeder Iteration erfordern! Daher kann die Formulierung des schnellstmöglichen Lösers einen erheblichen Unterschied machen, wenn umfangreiche Optimierungsmethoden implementiert werden, die eine oder mehrere lineare Lösungen pro Iteration erfordern. Tatsächlich könnte in diesem Fall eine ungenaue, aber schnelle Lösung eines linearen Systems akzeptabel sein, da sie ohnehin in eine größere iterative Technik einfließt.

Bitte beachten Sie, dass ein Großteil unserer Diskussion auf CITE zurückzuführen ist, auch wenn unsere Entwicklung dies sein kann angesichts der Entwicklung in den vorherigen Kapiteln etwas kürzer.

## 10.1 Gefälleabstieg

Wir werden unsere Diskussion auf die Lösung von Ax =b konzentrieren, wobei A drei Eigenschaften hat:

1. A ÿ Rn×n ist quadratisch
2. A ist symmetrisch, d. h. A = A
3. A ist positiv definit, d. h. für alle x = 0 ist x Ax > 0

Gegen Ende dieses Kapitels werden wir diese Annahmen lockern. Beachten Sie zunächst, dass wir Ax = b durch die Normalgleichungen A Ax = A b ersetzen können, um diese Kriterien zu erfüllen, obwohl diese Ersetzung, wie wir besprochen haben, zu numerischen Konditionierungsproblemen führen kann.

### 10.1.1 Ableitung des iterativen Schemas

In diesem Fall lässt sich leicht überprüfen, dass Lösungen von Ax = b Minima der durch die quadratische Form gegebenen Funktion f(x) sind

$$\frac{1}{2} f(x) \ddot{y} x Ax \ddot{y} bx + c \text{ für jedes } c$$

ÿ R. Insbesondere zeigt die Ableitung von f

$$\ddot{y} f(x) = Ax \ddot{y} b,$$

und das Setzen von ÿ f(x) =0 liefert das gewünschte Ergebnis.

Anstatt ÿ f(x) = 0 direkt zu lösen, wie wir es in der Vergangenheit getan haben, nehmen wir an, dass wir Folgendes anwenden: Gradientenabstiegsstrategie zu dieser Minimierung. Erinnern Sie sich an den grundlegenden Gradientenabstiegsalgorithmus:

1. Berechnen Sie die Suchrichtung d k ÿ ÿÿ f(xkÿ1) =b ÿ Axkÿ1 .
2. Definieren Sie xk ÿ xkÿ1 + ÿkd k , wobei ÿk so gewählt ist, dass f(xk) < f(xkÿ1)

Für eine generische Funktion f kann die Entscheidung über den Wert von ÿk ein schwieriges eindimensionales „Liniensuch“-Problem sein, das darauf hinausläuft, f(xkÿ1 + ÿkd k) als Funktion einer einzelnen Variablen ÿk ÿ 0 zu minimieren. Für Aufgrund unserer besonderen Wahl der quadratischen Form f(x) = x Ax ÿbx + c können wir jedoch eine Liniensuche in geschlossener Form durchführen. Insbesondere definieren

$$\begin{aligned} g(\ddot{y}) \ddot{y} f(x + \ddot{y}d) &= \frac{1}{2}(x + \ddot{y}d) A(x + \ddot{y}d) \ddot{y} b (x + \ddot{y}d) + c \\ &= \frac{1}{2}(x Ax + 2\ddot{y}x Ad + \ddot{y}^2 dAd) \ddot{y} bx \ddot{y} \ddot{y} b d + c \text{ aufgrund der Symmetrie von A} \\ &= \frac{1}{2} dAd + \ddot{y}(x Ad \ddot{y} b d) + \text{const.} \end{aligned}$$

$$\frac{dg}{d\ddot{y}}(\ddot{y}) = \ddot{y} dAd + d(Ax \ddot{y} b)$$

Wenn wir also g bezüglich ÿ minimieren wollen, wählen wir einfach

$$\ddot{y} = \frac{d(b \ddot{y} Ax)\, dAd}{}$$

Insbesondere für den Gradientenabstieg haben wir d gewählt $k = b \ddot{y} Axk$ , also nimmt ÿk tatsächlich eine schöne Form an:

$$\ddot{y}k = \frac{dd\ kk}{D \quad Ad_{kk}}$$

Am Ende ergibt unsere Formel für die Liniensuche das folgende iterative Gradientenabstiegsschema zum Lösen von Ax =b im symmetrischen positiv definiten Fall:

$$d_k = b \ddot{y} Axk\ddot{y}1$$

$$\ddot{y}k = \frac{dd\ kk}{d_k \ Ad_k}$$

$$xk = xk\ddot{y}1 + \ddot{y}kd\ k$$

## 10.1.2 Konvergenz

Konstruktionsbedingt verringert unsere Strategie für den Gradientenabstieg f(xk) , wenn k ÿ ÿ. Dennoch haben wir nicht gezeigt, dass der Algorithmus den minimal möglichen Wert von f erreicht, und wir konnten nicht charakterisieren, wie viele Iterationen wir ausführen sollten, um ein angemessenes Maß an Sicherheit zu erreichen, dass Axk ÿb .

Eine einfache Strategie zum Verständnis der Konvergenz des Gradientenabstiegsalgorithmus für unsere Wahl von f besteht darin, die Änderung des Rückwärtsfehlers von Iteration zu Iteration zu untersuchen.1 Angenommen, es handelt sich um die gesuchte Lösung, nämlich $\ddot{y}\ x$ Ax ÿ =b. Dann können wir das Rückwärtsverhältnis untersuchen Fehler von Iteration zu Iteration:

$$Rk \ddot{y} \frac{f(xk) \ddot{y} f(x \ddot{y})}{f(xk\ddot{y}1) \ddot{y} f(x \ddot{y})}$$

Offensichtlich zeigt die Begrenzung von Rk < $\ddot{y}$ < 1 für einige $\ddot{y}$ , dass der Gradientenabfall konvergiert.

Der Einfachheit halber können wir f(xk) erweitern:

$$\begin{aligned}
f(xk) &= f(xk\ddot{y}1 + \ddot{y}kd\ k) \text{ nach unserem iterativen Schema} \\
&= \frac{1}{2}(xk\ddot{y}1 + \ddot{y}kd\ k)\, A(xk\ddot{y}1 + \ddot{y}kd\ k) \ddot{y} b\ (xk\ddot{y}1 + \ddot{y}kd\ k) + c \\
&= f(xk\ddot{y}1) + \ddot{y}kd\ k\, Axk\ddot{y}1 + \ddot{y}\, k\, 2\, 1\, (b \ddot{y} d_k \ Ad_k\, k \ddot{y} \ddot{y}k\, bd\, k \text{ nach Definition von f} \\
&= f(xk\ddot{y}1) + \ddot{y}kd\ k \quad k) + \frac{1\ 2}{2}\ddot{y}\, D \quad Ad_k\, k \ddot{y} \ddot{y}k\, bd\, k \text{ seit } d\ k \quad k = b \ddot{y} Axk\ddot{y}1 \\
&= f(xk\ddot{y}1) \ddot{y} \ddot{y}kd \quad k + \ddot{y}\, kk\, 2 \frac{1\ 2\ d}{2} D \quad Ad_{kk}
\end{aligned}$$

1Dieses Argument wird z. B. in http://www-personal.umich.edu/~mepelman/teaching/IOE511/Handouts/ präsentiert. 511notes07-7.pdf.

$$= f(xk\ddot{y}1)\ \ddot{y}\ d\ \frac{dd\ kk}{k\ Ad\ k}\ 1\ \ddot{y}\ 2\frac{dd\ k + k}{} - \frac{dd\ kk}{D_k\ Ad\ k}^{2}\ D\ Ad_{k\ \text{nach Definition von } \ddot{y}k\ k}$$

$$= f(xk\ddot{y}1)\ \ddot{y}\ \frac{(d_k\ d\ k)^2}{D\ Ad_{kk}}\ 1\ \ddot{y}\ 2\ d\ \frac{(d\ _k\ d\ k)^2}{Ad_{kk}} = f(xk\ddot{y}1)\ \ddot{y}\ 2d\ \frac{(d\ _k\ d\ k)^2}{Ad_{kk}}$$

Somit können wir zu unserem Bruch zurückkehren:

$$Rk = \frac{f(xk\ddot{y}1)\ \ddot{y}\ \frac{(D_k\ d_k)^2}{2.\ _k\ Ad\ k}\ \ddot{y}\ f(x\ \ddot{y}\ )}{f(xk\ddot{y}1)\ \ddot{y}\ f(x\ \ddot{y})} \text{nach unserer Formel für } f(xk)$$

$$= 1\ \ddot{y}\ \frac{(D_k\ d\ k)^2}{2d\ k\quad Ad\ k(f(xk\ddot{y}1)\ \ddot{y}\ f(x\ \ddot{y}))}$$

Beachten Sie, dass $Ax\ \ddot{y}\ =b$, also können wir schreiben:

$$f(xk\ddot{y}1)\ \ddot{y}\ f(x\ \ddot{y}\ ) = \quad \frac{1}{2}\ x\ k\ddot{y}1Axk\ddot{y}1\ \ddot{y}b\ xk\ddot{y}1 + c\ \ddot{y}\ 2 \quad \frac{1}{2}\ (x\ \ddot{y}\ )\ b\ \ddot{y}bx\ 2^{\ \ddot{y}} + c$$

$$= \frac{1}{2}\ x\ k\ddot{y}1Axk\ddot{y}1\ \ddot{y}b\ xk\ddot{y}1\ \ddot{y}\ 2\ 1 \quad \frac{1}{2}\ b\ A\ \ddot{y}1b\ 2$$

$$= -\ (Axk\ddot{y}1\ \ddot{y}b)\ A\ 2\ 1\ d^{\ \ddot{y}1}\ (Axk\ddot{y}1\ \ddot{y}b) \text{ durch Symmetrie von A}$$

$$= -\frac{}{k\ 2}\ A^{\ \ddot{y}1}\ d\ k \text{ nach Definition von } d\ _k$$

Daher,

$$Rk = 1\ \ddot{y}\ 2d\ k\ \frac{(D_k\ d\ k)^2}{Ad\ k(f(xk\ddot{y}1)\ \ddot{y}\ f(x\ \ddot{y}))\ d\ k)}$$

$$= 1\ \ddot{y}\ d\ \frac{(D_k\ ^{2}}{Ad_{k\ \cdot\ d\ k\ k}\ A\ddot{y}1d\ k} \text{ durch unsere neueste Vereinfachung}$$

$$= 1\ \ddot{y}\ d\ \frac{D_k\ d\ k}{Ad\ k_k}\ \cdot\ \frac{D_k\ d\ k}{D\ \ A\ddot{y}1d\ k\ _k}$$

$$\ddot{y}\ 1\ \ddot{y}\ \min\ dAd\ \frac{1}{d=1} \qquad \min_{d=1}\ \frac{1}{dA\ddot{y}1d} \text{ da dadurch der zweite Term kleiner wird}$$

$$1\ \ddot{y}\ \max\ d=1 \qquad dAd = \qquad \max_{d=1}\ dA^{\ \ddot{y}1}\ D^{\ \ddot{y}1}$$

$$= 1\ \ddot{y}\ \frac{\ddot{y}\min}{\ddot{y}\max\ 1} \text{ wobei } \ddot{y}\min \text{ und } \ddot{y}\max \text{ die minimalen und maximalen Singulärwerte von A sind}$$

$$= 1\ \ddot{y}\ \frac{}{\text{cond A}}$$

Es erforderte einiges an Algebra, aber wir haben eine wichtige Tatsache bewiesen:

**Die Konvergenz des Gradientenabstiegs auf** f **hängt von der Konditionierung von A** ab.

Das heißt, je besser A konditioniert ist, desto schneller wird der Gradientenabfall konvergieren. Da cond A ÿ 1 ist, wissen wir außerdem, dass unsere obige Gradientenabstiegsstrategie bedingungslos gegen x konvergiert, ÿ , obwohl die Konvergenz langsam sein kann, wenn A schlecht konditioniert ist.

Abbildung NUMBER veranschaulicht das Verhalten des Gradientenabstiegs für gut und schlecht konditionierte Matrizen. Wie Sie sehen, kann es beim Gradientenabstieg schwierig sein, das Minimum unserer quadratischen Funktion f zu finden, wenn die Eigenwerte von A weit gestreut sind.

## 10.2 Konjugierte Farbverläufe

Denken Sie daran, dass das Lösen von Ax = b für A ÿ Rn×n die $3$) Zeit. Erneute Untersuchung des Gradientenabstiegs obige O(n- Strategie verwendet hat . Wir sehen, dass jede $2$) Zeit, da wir den Matrixvektor berechnen müssen Iteration O(n Produkte zwischen A, xkÿ1 und d k erfordert . Wenn also der Gradientenabstieg mehr als n Iterationen erfordert, müssen wir hätte genauso gut die Gaußsche Eliminierung anwenden können, die die exakte Lösung in der gleichen Zeitspanne wiederherstellt. Leider können wir nicht zeigen, dass der Gradientenabstieg eine endliche Anzahl von Iterationen erfordern muss, und in schlecht konditionierten Fällen tatsächlich auch Es kann eine große Anzahl von Iterationen erfordern, um das Minimum zu finden.

Aus diesem Grund werden wir einen Algorithmus entwerfen, der garantiert in höchstens n Schritten Unter Beibehaltung des $3$konvergiert ( Worst-Case-Timing für die Lösung linearer Systeme). Unterwegs werden wir finden O(n) weist dieser Algorithmus tatsächlich insgesamt bessere Konvergenzeigenschaften auf, was ihn zu einer vernünftigen Wahl macht, auch wenn wir ihn nicht vollständig ausführen.

### 10.2.1 Motivation

Unsere Ableitung des Algorithmus für konjugierte Gradienten basiert auf einer ziemlich einfachen Beobachtung. Angenommen, wir kennen die Lösung x auf $ÿ$ schreibÿn=b. Dann können wir unsere quadratische Form f in Ax andere Weise:

$$f(x) = x\,Ax - ÿbx + c \text{ per Definition } \frac{1}{2}(x ÿx ÿ)$$

$$= \frac{1}{2}A(x ÿx ÿ) + x\,Ax\,\frac{1}{2} \quad ÿ \quad \frac{1}{2}(x ÿ)\,Ax \quad ÿ \quad ÿbx + c$$

durch Addition und Subtraktion derselben Terme

$$= \frac{1}{2}(x ÿx ÿ)\,A(x ÿx ÿ) + xb ÿ \frac{1}{2}(x ÿ)\,b - ÿbx + c \text{ seit } Ax\,\frac{1}{2}\,\frac{1}{2} \quad - \quad ÿ = b$$

$$= \frac{1}{2}(x ÿx ÿ)\,A(x ÿx ÿ) + \text{const. da die xb-bedingungen ungültig werden}$$

zu einer konstanten Verschiebung, dass f das gleiche ist wie das $\frac{1}{2}(x ÿx ÿ)\,A(x ÿx ÿ)$. Natürlich wissen wir bis Produkt x, $ÿ$ aber diese Beobachtung zeigt uns die Natur von f; es misst einfach den Abstand ÿ v Av. von x nach x ÿ in Bezug auf die „A-Norm“ v Da A $\frac{2}{A}$

symmetrisch und positiv definit ist, wissen wir tatsächlich, dass es mithilfe der Cholesky-Strategie als faktorisiert werden kann, auch wenn die Umsetzung in der Praxis langsam sein könnte A = LL. Mit dieser Faktorisierung nimmt f eine noch schönere Form an:

$$f(x) = \frac{1}{2} \quad L\,(x ÿx ÿ) \qquad 2 + \text{Konst. } 2$$

Da L eine invertierbare Matrix ist, ist diese Norm tatsächlich ein Abstandsmaß zwischen x und x ŷ .

Definieren Sie y ÿ L x und y ŷ ÿ L x ŷ Dann minimieren wir von diesem neuen Standpunkt aus 2 . Natürlich, f(y) = .
y ÿy ŷ 2 wenn wir wirklich durch Cholesky-Faktorisierung und Optimierung an diesen Punkt gelangen könnten f
wäre außerordentlich einfach, aber um ein Schema für diese Minimierung ohne L abzuleiten, betrachten wir die Möglichkeit, f zu minimieren, indem wir nur die in §10.1.1 abgeleiteten Zeilensuchen verwenden.

Wir machen eine einfache Beobachtung zur Minimierung unserer vereinfachten Funktion f mithilfe einer solchen Strategie egy, dargestellt in Abbildung NUMMER:

**Vorschlag 10.1.** Angenommen {w 1, . . . , w n} sind orthogonal im Rn . Dann wird f in höchstens n Schritten durch Liniensuche in Richtung w 1, dann in Richtung w 2 usw. minimiert.

Nachweisen. Nehmen Sie an, dass die Spalten von Q ÿ Rn×n die Vektoren w i sind ; Q ist eine orthogonale Matrix. Da Q f(y) =
können wir 2 schreiben ; Mit anderen Worten, Wir haben ŷ 2 2 , dass Qw 1 ÿ der erste Standardbasisvektor ist, w 2 der zweite und so
weiter. Wenn wir z ÿ Qy schreiben und nach der zweiten Iteration
ŷ z ÿ Qy und ŷ , Dann müssen wir nach der ersten Iteration eindeutig z1 = z haben ŷ 1 ,
z2 = z 2 ŷ , so weiter. Nach n Schritten erreichen wir zn = z ŷ n , was das gewünschte Ergebnis liefert. □

Die Optimierung von f kann also immer durch n-zeilige Suchvorgänge erreicht werden, solange diese Suchvorgänge in orthogonalen Richtungen erfolgen.

Um von f nach f zu gelangen, haben wir lediglich die Koordinaten mit L gedreht . So eine lineare Transformation
mit geraden Linien zu geraden Linien, also führt eine Liniensuche zu einer f entlang eines Vektors w ist gleichbedeutend
Liniensuche entlang (L ) ÿ1w auf unserer ursprünglichen quadratischen Funktion f . Wenn wir umgekehrt n Liniensuchen auf f in Richtungen vi durchführen , so dass L vi ÿ w i orthogonal sind, dann müssen wir nach Proposition 10.1 x gefunden haben
ŷ . Beachten Sie, dass die Frage w i · w = 0 dasselbe ist wie die Frage j

$$0 = w_i \cdot w_j = (L v_i)(L v_j) = v_i (LL) v_j = v_i A v_j .$$

Wir haben gerade eine wichtige Folgerung zu Proposition 10.1 dargelegt. Definieren Sie konjugierte Vektoren wie folgt:

**Definition 10.1** (A-konjugierte Vektoren). Zwei Vektoren v, w sind A-konjugiert, wenn v Aw = 0.

Basierend auf unserer Diskussion haben wir dann gezeigt:

**Vorschlag 10.2.** Angenommen, {v1, . . . ,vn} sind A-konjugiert. Dann wird f in höchstens n Schritten durch Liniensuche in Richtung v1, dann in Richtung v2 usw. minimiert.

Auf einer hohen Ebene wendet der Algorithmus für konjugierte Gradienten diesen Vorschlag einfach an, indem er entlang A-konjugierter Richtungen generiert und sucht, anstatt sich entlang ŷÿ f zu bewegen. Beachten Sie, dass dieses Ergebnis möglicherweise etwas kontraintuitiv erscheint: Wir bewegen uns nicht unbedingt entlang der steilsten Abstiegsrichtung, sondern verlangen vielmehr, dass unsere Suchrichtungen ein globales Kriterium erfüllen, um sicherzustellen, dass wir die Arbeit nicht wiederholen. Dieser Aufbau garantiert Konvergenz in einer endlichen Anzahl von Iterationen und berücksichtigt die oben diskutierte Struktur von f in Bezug auf f.

Denken Sie daran, dass wir A-konjugierte Richtungen motiviert haben, indem wir festgestellt haben, dass sie orthogonal sind, nachdem L aus der Faktorisierung A = LL angewendet wurde . Von diesem Standpunkt aus haben wir es mit zwei Skalarprodukten zu tun: xi · xj und yi · yj ÿ (L xi) · (L xj) = x i LLxj = x i Axj . Diese beiden Produkte werden in gleicher Menge in unsere nachfolgende Diskussion einfließen, daher bezeichnen wir das „A-innere Produkt“ als

$$u,vA \ddot{y} (L u) \cdot (L v) = u Av.$$

### 10.2.2 Suboptimalität des Gradientenabstiegs

Bisher wissen wir, dass wir Ax =b in n Schritten über Liniensuchen entlang dieser Richtungen lösen können, wenn wir n A-konjugierte Suchrichtungen finden können. Was bleibt, ist, eine Strategie zu finden, um diese Richtungen möglichst effizient zu finden. Dazu werden wir eine weitere Eigenschaft des Gradientenabstiegsalgorithmus untersuchen, die zu einem verfeinerten Ansatz inspirieren wird.

Angenommen, wir befinden uns während einer iterativen Liniensuchmethode auf f(x) bei xk ; Wir nennen die Richtung des steilsten Abfalls von f bei xk das Residuum rk ÿ b ÿ Axk . Wir entscheiden uns möglicherweise nicht für eine Liniensuche entlang rk wie beim Gradientenabstieg, da die Gradientenrichtungen nicht unbedingt A-konjugiert sind. Wenn wir also etwas verallgemeinern, werden wir xk+1 über eine Liniensuche entlang einer noch unbestimmten Richtung vk+1 finden .

Aus unserer Ableitung des Gradientenabstiegs sollten wir xk+1 = xk + ÿk+1vk+1 wählen , wobei ÿk+1 gegeben ist durch

$$ÿk+1 = \frac{v_{k+1}\, rk}{v_{k+1} Avk+1}.$$

Wenn wir diese Erweiterung von xk+1 anwenden , können wir eine alternative Aktualisierungsformel für das Residuum schreiben:

$$\begin{aligned} rk+1 &= b ÿ Axk+1 = b ÿ \\ &\quad A(xk + ÿk+1vk+1) \text{ per Definition von } xk+1 = (b ÿ Axk) ÿ \\ &\quad ÿk+1Avk+1 = rk ÿ ÿk+1Avk+1 \text{ per} \\ &\quad \text{Definition ofrk} \end{aligned}$$

Diese Formel gilt unabhängig von unserer Wahl von vk+1 und kann auf jede iterative Liniensuchmethode angewendet werden.

Im Fall des Gradientenabstiegs haben wir jedoch vk+1 ÿ rk gewählt . Diese Auswahl ergibt eine Wiederholungsbeziehung:

$$rk+1 = rk ÿ ÿk+1Ark .$$

Diese einfache Formel führt zu einer lehrreichen Aussage:

**Vorschlag 10.3.** Wenn ein Gradientenabstieg auf f durchgeführt wird, span{r0, . . . ,rk} = span{r0, Ar0, . . . , Ein kr0}.

Nachweisen. Diese Aussage folgt induktiv aus unserer obigen Formel fork+1. □

Die Struktur, die wir entdecken, ähnelt stark den in Kapitel 5 erwähnten Krylov-Unterraummethoden: Das ist kein Fehler!

Der Gradientenabstieg erreicht xk , indem man sich entlang r0, dann r1 usw. über rk bewegt . Somit wissen wir am Ende, dass die Iteration xk des Gradientenabstiegs auf f irgendwo in der Ebene x0 + A kÿ1r0} liegt, nach Satz 10.3. Leider ist es span die Iteration xk in diesem Unterraum optimal ist , wenn wir {r0,r1, . . . ,rkÿ1} = x0 + span {r0, Ar0, . . . Es stimmt nicht, dass einen Gradientenabstieg ausführen . Mit anderen Worten: Im Allgemeinen kann es so sein

$$ÿx0 = vÿ\text{span}_{\{r0,Ar0,...,Akÿ1r0\}} \min_{xk} f(x0 + v) \arg$$

Im Idealfall würde die Umwandlung dieser Ungleichung in eine Gleichheit sicherstellen, dass die Generierung von xk+1 aus xk keine Arbeit „zunichtemacht“, die während der Iterationen 1 bis k ÿ 1 geleistet wurde.

Wenn wir unseren Beweis von Proposition 10.1 unter Berücksichtigung dieser Tatsache erneut prüfen, können wir eine Beobachtung machen, die darauf hindeutet, wie wir Konjugation nutzen könnten, um den Gradientenabstieg zu verbessern. Insbesondere wenn zi zu z wechselt, ändert sich der Wert in einer zukünftigen Iteration nie mehr. Mit anderen Worten, nach der Drehung von z zu i , x gilt folgender Satz:

**Vorschlag 10.4.** Nehmen Sie xk als die k-te Iteration des Prozesses aus Proposition 10.1 nach der Suche entlang vk an . Dann,

$$x_k = x_0 + \arg\min_{v \in \text{span}\{v_1,\ldots,v_k\}} f(x_0 + v)$$

In der besten aller möglichen Welten könnten wir also bei dem Versuch, den Gradientenabstieg zu übertreffen, hoffen, kÿ1r0} A-konjugierte Richtungen {v1, . . . ,vn}, so dass die Spanne {v1, . . . ,vk} = span {r0, Ar0, . . . , für jedes k; dass A Dann ist gewährleistet, dass unser iteratives Schema während einer bestimmten Iteration nicht schlechter abschneidet als der Gradientenabstieg. Aber wir möchten dies unbedingt tun, ohne Orthogonalisierung oder mehr als eine endliche Anzahl von Vektoren gleichzeitig zu speichern.

## 10.2.3 **A-konjugierte Richtungen erzeugen**

Natürlich können wir jede gegebene Richtungsmenge mit einer Methode wie der Gram-Schmidt-Orthogonalisierung A-orthogonal machen. Leider ist die Orthogonalisierung von {r0, Ar0, . . .} das Finden der Suchrichtungen ist kostspielig und würde erfordern, dass wir eine vollständige Liste der Richtungen führen vk ; Diese Konstruktion würde wahrscheinlich sogar den Zeit- und Speicherbedarf einer Gaußschen Eliminierung übersteigen.
Wir werden eine letzte Beobachtung über Gram-Schmidt offenbaren, die konjugierte Gradienten durch die Erzeugung konjugierter Richtungen ohne einen teuren Orthogonalisierungsprozess handhabbar macht.

Wenn wir diese Probleme ignorieren, könnten wir eine „Methode konjugierter Richtungen“ wie folgt schreiben:

**Suchrichtung aktualisieren (schlechter Gram-Schmidt-Schritt):** $v_k = A^{k-1} r_0 - \sum_{i<k} \frac{A^{k-1} r_0 \cdot v_i A}{v_i \cdot v_i A} v_i$

**Zeilensuche:** $\alpha_k = \frac{v_k \cdot r_{k-1}}{v_k \cdot A v_k}$

**Schätzung aktualisieren:** $x_k = x_{k-1} + \alpha_k v_k$

**Residuum aktualisieren:** $r_k = r_{k-1} - \alpha_k A v_k$

Hier berechnen wir die k-te Suchrichtung vk einfach durch Projizieren von v1, . . . ,vkÿ1 aus dem Vektor A kÿ1r0. Dieser Algorithmus hat offensichtlich die Eigenschaft span {v1, . . . ,vk} = span {r0, Ar0, . . . , A kÿ1r0} vorgeschlagen in §10.2.2, hat aber zwei Probleme:

1. Ähnlich wie bei der Potenziteration für Eigenvektoren sieht die Potenz A kÿ1r0 wahrscheinlich größtenteils wie der erste Eigenvektor von A aus, wodurch die Projektion immer schlechter konditioniert wird

2. Wir müssen v1 behalten . . . ,vkÿ1 um vk zu berechnen ; Daher benötigt jede Iteration dieses Algorithmus mehr Speicher und Zeit als die letzte.

Das erste Problem können wir relativ unkompliziert beheben. Insbesondere projizieren wir derzeit die vorherigen Suchrichtungen aus A kÿ1r0 heraus, aber in Wirklichkeit können wir vorherige Richtungen aus jedem Vektor w projizieren, solange

$$w \in \text{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\} \setminus \text{span}\{r_0, Ar_0, \ldots, A^{k-2} r_0\},$$

das heißt, solange w eine Komponente im neuen Teil des Raums hat.

Eine alternative Wahl von w mit dieser Eigenschaft ist das Residuum rkÿ1 . Diese Eigenschaft folgt aus der Restaktualisierung rk = rkÿ1 ÿ ÿkAvk ; In diesem Ausdruck multiplizieren wir vk mit A und führen so die neue Potenz von A ein, die wir benötigen. Diese Wahl ahmt auch den Gradientenabstiegsalgorithmus besser nach, der vk =rkÿ1 annahm . Somit können wir unseren Algorithmus etwas aktualisieren:

**Suchrichtung aktualisieren (schlechtes Gram-Schmidt auf Residuum):** $v_k = r_{k-1} - \sum_{i<k} \frac{r_{k-1}, Av_i}{v_i, Av_i} v_i$

**Zeilensuche:** $\alpha_k = \frac{v_k^T r_{k-1}}{v_k^T A v_k}$

**Schätzung aktualisieren:** $x_k = x_{k-1} + \alpha_k v_k$

**Residuum aktualisieren:** $r_k = r_{k-1} - \alpha_k A v_k$

Jetzt führen wir keine Arithmetik mit dem schlecht konditionierten Vektor A kÿ1r0 durch , haben aber immer noch das obige „Speicherproblem“.

Tatsächlich ist die überraschende Beobachtung zum obigen Orthogonalisierungsschritt, dass die meisten Terme in der Summe genau Null sind! Diese erstaunliche Beobachtung ermöglicht es, jede Iteration konjugierter Gradienten durchzuführen, ohne den Speicherverbrauch zu erhöhen. Wir halten dieses Ergebnis in einem Satz fest:

**Vorschlag 10.5.** In der oben genannten Methode der „konjugierten Richtung“ ist $r_k, Av_\ell = 0$ für alle $\ell < k$.

Nachweisen. Wir gehen induktiv vor. Für den Basisfall k = 1 gibt es nichts zu beweisen. Nehmen Sie also an, dass k > 1 ist und dass das Ergebnis für alle $\ell < k$ gilt. Durch die Restaktualisierungsformel wissen wir:

$$r_k, Av_\ell = r_{k-1}, Av_\ell - \alpha_k \langle Av_k, Av_\ell \rangle = r_{k-1}, Av_\ell - \alpha_k \langle v_k, Av_\ell \rangle,$$

wobei die zweite Gleichheit aus der Symmetrie von A folgt.

Nehmen wir zunächst an, dass $\ell < k - 1$. Dann ist der erste Term der obigen Differenz durch Induktion Null. Darüber hinaus ist $Av_\ell \in \text{span}\{v_1, \ldots, v_{\ell+1}\}$, da wir also unsere Suchrichtungen A-konjugiert konstruiert haben, wissen wir, dass der zweite Term ebenfalls Null sein muss.

Um den Beweis abzuschließen, betrachten wir den Fall $\ell = k - 1$. Mithilfe der Restaktualisierungsformel wissen wir:

$$Av_{k-1} = \frac{1}{\alpha_{k-1}} (r_{k-2} - r_{k-1})$$

Die Vormultiplikation von $r_k$ zeigt:

$$r_k, Av_{k-1} = \frac{1}{\alpha_{k-1}} r_k^T (r_{k-2} - r_{k-1})$$

Die Differenz $r_{k-2} - r_{k-1}$ lebt im Bereich $\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, nach der Restaktualisierungsformel. Satz 10.4 zeigt, dass $x_k$ in diesem Unterraum optimal ist. Da $r_k = -\nabla f(x_k)$, impliziert dies $r_k \perp \{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, da es sonst eine Richtung gäbe, die wir haben müssten $r_k \in \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$ im Unterraum, um von $x_k$ zu einer Verringerung von f zu gelangen. Insbesondere zeigt dies wie gewünscht das innere Produkt über $r_k, Av_{k-1} = 0$. □

Somit zeigt unser obiger Beweis, dass wir eine neue Richtung vk wie folgt finden können:

$$vk = rk\ddot{y}1 \ddot{y} \ddot{y} \sum_{i<k} \frac{rk\ddot{y}1 , viA}{vi , viA} vi \text{ nach der Gram-Schmidt-Formel}$$

$$= rk\ddot{y}1 \ddot{y} \frac{rk\ddot{y}1 , vk\ddot{y}1A}{} vk\ddot{y}1 \text{ , weil die restlichen Terme } vk\ddot{y}1 , vk\ddot{y}1A \text{ verschwinden}$$

Da die Summierung über i verschwindet, hängen die Kosten für die Berechnung von vk nicht von k ab.

## 10.2.4 Formulieren des Algorithmus für konjugierte Gradienten

Da wir nun über eine Strategie verfügen, die A-konjugierte Suchrichtungen mit relativ geringem Rechenaufwand liefert, wenden wir diese Strategie einfach an, um den Algorithmus für konjugierte Gradienten zu formulieren.

Nehmen wir insbesondere an, dass x0 eine anfängliche Schätzung der Lösung von Ax = b ist und dass r0 ÿ b ÿ Ax0 gilt.
Nehmen Sie der Einfachheit halber v0 ÿ 0. Dann aktualisieren wir xkÿ1 iterativ auf xk , indem wir eine Reihe von Schritten für k = 1, 2, ... verwenden. . .:

**Suchrichtung aktualisieren:** $vk = rk\ddot{y}1 \ddot{y} \frac{rk\ddot{y}1 , vk\ddot{y}1A}{vk\ddot{y}1 , vk\ddot{y}1A} vk\ddot{y}1$

**Zeilensuche:** $\ddot{y}k = \frac{v_k \, rk\ddot{y}1}{v_k \, Avk}$

**Schätzung aktualisieren:** $xk = xk\ddot{y}1 + \ddot{y}kvk$

**Residuum aktualisieren:rk** $= rk\ddot{y}1 \ddot{y} \ddot{y}kAvk$

Dieses iterative Schema stellt nur eine geringfügige Anpassung des Gradientenabstiegsalgorithmus dar, weist jedoch konstruktionsbedingt viele wünschenswerte Eigenschaften auf:

- f(xk) wird nach oben durch die k-te Iteration des Gradientenabstiegs begrenzt
- Der Algorithmus konvergiert gegen $x^{\ddot{y}}$ in n Schritten
- Bei jedem Schritt ist die Iteration xk in dem von den ersten k Suchrichtungen aufgespannten Unterraum optimal

Um die maximale numerische Qualität aus konjugierten Gradienten herauszuholen, können wir versuchen, die Zahlen der obigen Ausdrücke zu vereinfachen. Wenn wir beispielsweise die Aktualisierung der Suchrichtung in die Formel für ÿk einbauen , können wir durch Orthogonalität schreiben:

$$\ddot{y}k = \frac{R_{k\ddot{y}1} \, rk\ddot{y}1}{v_k \, Avk}$$

Der Zähler dieses Bruchs ist nun ohne numerische Genauigkeit garantiert nicht negativ
Probleme.

Ebenso können wir eine Konstante ÿk definieren, um die Aktualisierung der Suchrichtung in zwei Schritte aufzuteilen:

$$\ddot{y}k \ddot{y} \ddot{y} \frac{rk\ddot{y}1 , vk\ddot{y}1A}{vk\ddot{y}1 , vk\ddot{y}1A}$$

$$vk = rk\ddot{y}1 + \ddot{y}kvk\ddot{y}1$$

Wir können unsere Formel für ÿk vereinfachen :

$$\ddot{y}k \ddot{y} \ddot{y} \frac{rk\ddot{y}1 ,vk\ddot{y}1A}{vk\ddot{y}1 ,vk\ddot{y}1A}$$

$$\frac{rk\ddot{y}1Avk\ddot{y}1}{v\ k\ddot{y}1Avk\ddot{y}1} \text{ nach Definition von } \cdot, \cdot A$$

$$\frac{r_{k\ddot{y}1}\ (rk\ddot{y}2\ \ddot{y}rk\ddot{y}1)}{\ddot{y}k\ddot{y}1v\ k\ddot{y}1Avk\ddot{y}1} \text{dark} = rk\ddot{y}1\ \ddot{y}\ \ddot{y}kAvk$$

$$= \frac{R_{k\ddot{y}1}\ rk\ddot{y}1}{k\ddot{y}1Avk\ddot{y}1} \text{durch eine Berechnung unten } \ddot{y}k\ddot{y}1v$$

$$= \frac{R_{k\ddot{y}1}\ rk\ddot{y}1}{R_{k\ddot{y}2}\ rk\ddot{y}2} \text{nach unserer letzten Formel für } \ddot{y}k$$

Dieser Ausdruck zeigt, dass ÿk ÿ 0 ist, eine Eigenschaft, die nach Problemen mit der numerischen Genauigkeit möglicherweise nicht gilt. Wir haben unten noch eine verbleibende Berechnung: (rkÿ2 ÿ

$R_{k\ddot{y}2}\ rk\ddot{y}1 = r\ k\ddot{y}2 \quad \ddot{y}k\ddot{y}1Avk\ddot{y}1)$ durch unsere Restaktualisierungsformel

$$= r \quad rk\ddot{y}2\ \ddot{y}\ v \frac{R_{k\ddot{y}2}\ rk\ddot{y}2}{k\ddot{y}1Avk\ddot{y}1} r\ k\ddot{y}2Avk\ddot{y}1 \text{ nach unserer Formel für } \ddot{y}k\ k\ddot{y}2$$

$$= r \quad \ddot{y}\ v\ k\ddot{y}1Avk\ddot{y}1 \frac{R_{k\ddot{y}2}\ rk\ddot{y}2}{v\ k\ddot{y}1Avk\ddot{y}1} \text{ durch die Aktualisierung für vk und A-Konjugation der vk 's } k\ddot{y}2\ rk\ddot{y}2$$

$$= 0, \text{ je nach Bedarf.}$$

Mit diesen Vereinfachungen haben wir eine alternative Version konjugierter Gradienten:

**Suchrichtung aktualisieren:** $\ddot{y}k = \frac{R_{k\ddot{y}1}\ rk\ddot{y}1}{R_{k\ddot{y}2}\ rk\ddot{y}2}$

$$vk = rk\ddot{y}1 + \ddot{y}kvk\ddot{y}1$$

**Zeilensuche:** $\ddot{y}k = \frac{R_{k\ddot{y}1}\ rk\ddot{y}1}{v_k\ Avk}$

**Schätzung aktualisieren:** $xk = xk\ddot{y}1 + \ddot{y}kvk$

**Residuum aktualisieren:rk** $= rk\ddot{y}1\ \ddot{y}\ \ddot{y}kAvk$

Aus numerischen Gründen ist es gelegentlich ratsam, anstelle der Aktualisierungsformel forrk die Restformel mark =b ÿ Axk zu verwenden . Diese Formel erfordert eine zusätzliche Matrix-Vektor-Multiplikation, repariert jedoch die numerische „Drift“, die durch das Runden mit endlicher Genauigkeit verursacht wird. Beachten Sie, dass es nicht erforderlich ist, eine lange Liste vorheriger Residuen oder Suchrichtungen zu speichern: Konjugierte Gradienten nehmen von Iteration zu Iteration eine konstante Menge an Platz ein.

## 10.2.5 Konvergenz- und Stoppbedingungen

Durch die Konstruktion wird garantiert, dass der Algorithmus für konjugierte Gradienten (CG) nicht langsamer konvergiert als der Gradientenabstieg auf f , während er nicht schwieriger zu implementieren ist und eine Reihe anderer aufweist

positive Eigenschaften. Eine detaillierte Diskussion der CG-Konvergenz würde den Rahmen unserer Diskussion sprengen, aber im Allgemeinen verhält sich der Algorithmus am besten bei Matrizen mit gleichmäßig verteilten Eigenwerten über einen kleinen Bereich. Eine grobe Schätzung parallel zu unserer Schätzung in §10.1.2 zeigt, dass der CG-Algorithmus Folgendes erfüllt:

$$\frac{f(xk)\ \ddot{y}\ f(x\ \ddot{y}\ )}{f(x0)\ \ddot{y}\ f(x\ \ddot{y})}\ddot{y}\ 2 \qquad \frac{\ddot{y}\ \bar{\ddot{y}}\ddot{y}\ 1\ \ddot{y}}{\bar{\ddot{y}}+1}^{k}$$

wobei $\bar{\ddot{y}}\ \ddot{y}$ cond A. Allgemeiner kann die Anzahl der Iterationen, die für den konjugierten Gradienten erforderlich sind, um einen gegebenen Fehlerwert zu erreichen, normalerweise durch eine Funktion von $\ddot{y}\ \ddot{y}$ begrenzt werden, wohingegen die Grenzen für die Konvergenz des Gradientenabfalls proportional zu $\ddot{y}$ sind.

Wir wissen, dass konjugierte Gradienten garantiert genau in n Schritten gegen x $\ddot{y}$ konvergieren , aber wenn n groß ist, kann es besser sein, früher aufzuhören. Tatsächlich dividiert die Formel für $\ddot{y}k$ durch Null, wenn das Residuum sehr kurz wird, was zu Problemen mit der numerischen Genauigkeit in der Nähe des Minimums von f führen kann. Daher wird in der Praxis der Schwerpunkt normalerweise dann angehalten, wenn das Verhältnis rk/r0 ausreichend klein ist.

## 10.3 Vorkonditionierung

Wir verfügen nun über zwei leistungsstarke iterative Schemata, um Lösungen für Ax = b zu finden, wenn A symmetrisch und positiv definit ist: Gradientenabstieg und konjugierte Gradienten. Beide Strategien konvergieren bedingungslos, was bedeutet, dass wir unabhängig von der anfänglichen Schätzung x0 mit genügend Iterationen in einer endlichen Anzahl von Iterationen genau in einer endlichen Anzahl von Iterationen $^{\ddot{y}}$ ; Tatsächlich garantieren konjugierte Gradienten, dass wir x erreichen $^{\ddot{y}}$ beliebig nahe an die wahre Lösung x herankommen. Natürlich ist die Zeit, die für beide Methoden benötigt wird, um eine Lösung von Ax = b zu erreichen, direkt proportional zur Anzahl der Iterationen, die erforderlich sind, um x innerhalb einer $^{\ddot{y}}$ akzeptablen Toleranz zu erreichen. Daher ist es sinnvoll, eine Strategie so weit wie möglich abzustimmen, um die Anzahl der Iterationen für die Konvergenz zu minimieren.

Zu diesem Zweck stellen wir fest, dass wir in der Lage sind, die Konvergenzraten beider Algorithmen und vieler weiterer verwandter iterativer Techniken anhand der Bedingungszahl cond A zu charakterisieren. Das heißt, je kleiner der Wert von cond A ist, desto weniger Zeit sollte dafür benötigt werden um Ax = b zu lösen. Beachten Sie, dass diese Situation bei der Gaußschen Eliminierung etwas anders ist, da diese unabhängig von A die gleiche Anzahl an Schritten erfordert. Mit anderen Worten: Die Konditionierung von A beeinflusst nicht nur die Qualität der Ausgabe iterativer Methoden, sondern auch die Geschwindigkeit, mit der x $^{\ddot{y}}$ wird angefahren.

Natürlich gilt für jede invertierbare Matrix P, dass die Lösung von PAx = Pb äquivalent zur Lösung von Ax = b ist. Der Trick besteht jedoch darin, dass die Konditionsnummer von PA nicht mit der von A identisch sein muss; Im (unerreichbaren) Extrem würden wir natürlich, wenn wir P = A nehmen würden, Konditionierungsprobleme vollständig $^{\ddot{y}1}$ beseitigen! Allgemeiner angenommen, P $\ddot{y}$ A cond A, und daher kann es ratsam sein, P vor der Lösung des $^{\ddot{y}1}$ . Dann erwarten wir cond PA linearen Systems anzuwenden. In diesem Fall nennen wir P einen Vorkonditionierer.

Obwohl die Idee der Vorkonditionierung attraktiv erscheint, bleiben zwei Probleme bestehen:

1. Während A symmetrisch und positiv definit sein kann, wird das Produkt PA im Allgemeinen nicht gefallen diese Eigenschaften.
2. Wir müssen P $\ddot{y}$ A finden $^{\ddot{y}1}$ das ist einfacher zu berechnen als A $^{\ddot{y}1}$ selbst.

Auf diese Probleme gehen wir in den folgenden Abschnitten ein.

## 10.3.1 CG mit Vorkonditionierung

Wir werden unsere Diskussion auf konjugierte Gradienten konzentrieren, da diese über bessere Konvergenzeigenschaften verfügen, obwohl sich die meisten unserer Konstruktionen auch relativ leicht auf den Gradientenabstieg anwenden lassen. Wenn wir von diesem Standpunkt aus unsere Konstruktionen in §10.2.1 betrachten, wird klar, dass unsere Konstruktion von CG stark sowohl von der Symmetrie als auch der positiven Bestimmtheit von A abhängt, sodass die Ausführung von CG auf PA normalerweise nicht sofort konvergiert.

Nehmen wir jedoch an, dass der Vorkonditionierer P selbst symmetrisch und positiv definit ist. Dies ist eine vernünftige Annahme, da A diese Eigenschaften erfüllen muss. Dann können wir wieder $a = EE^{\ddot{y}1}$ schreiben. Wir machen folgende Beobachtung: Cholesky-Faktorisierung des inversen P- **Satz 10.6.** $^{\ddot{y}1}$

Die Konditionszahl von PA ist die gleiche wie die von $E\ddot{y}1AE\ddot{y}$.

Nachweisen. Wir zeigen, dass PA und $E\ddot{y}1AE\ddot{y}$ die gleichen Singulärwerte haben; Die Bedingungszahl ist das Verhältnis des maximalen zum minimalen Singulärwert, daher ist diese Aussage mehr als ausreichend. Insbesondere ist klar, dass $E\ddot{y}1AE\ddot{y}$ symmetrisch und positiv definit ist, sodass seine Eigenvektoren seine Singulärwerte sind. Nehmen wir also an, dass $E\ddot{y}1AE\ddot{y}x = \ddot{y}x$ ist. Wir kennen P. Wenn wir also beide Seiten unseres Eigenvektorausdrucks vorab mit $E\ddot{y}$ multiplizieren $^{\ddot{y}1}$, finden wir $PAE\ddot{y}x = \ddot{y}E\ddot{y}x$. $^{\ddot{y}1}$

Die Definition von $y \ddot{y} E \ddot{y}x$ zeigt $PAy = \ddot{y}y$. Somit haben PA und $E\ddot{y}1AE\ddot{y}$ beide vollständige Eigenräume und identische Eigenwerte. □

Diese Aussage impliziert, dass wir, wenn wir CG auf der symmetrischen positiv definiten Matrix $E\ddot{y}1AE\ddot{y}$ durchführen, die gleichen Konditionierungsvorteile erhalten, die wir hätten, wenn wir auf PA iterieren könnten. Wie in unserem Beweis von Proposition 10.6 könnten wir unsere neue Lösung für $y = E\,x$ in zwei Schritten durchführen:

1. Lösen Sie $E\ddot{y}1AE\ddot{y}y = E\ddot{y}1b$.
2. Lösen Sie $x = E\ddot{y}y$.

Das Finden von E wäre ein wesentlicher Bestandteil dieser Strategie, ist aber wahrscheinlich schwierig, aber wir werden in Kürze beweisen, dass es unnötig ist.

Wenn wir die Berechnung von E ignorieren, könnten wir Schritt 1 mit CG wie folgt ausführen:

**Suchrichtung aktualisieren:** $\ddot{y}k = \dfrac{^{R}k\ddot{y}1\ r k\ddot{y}1}{^{R}k\ddot{y}2\ r k\ddot{y}2}$

$$vk = rk\ddot{y}1 + \ddot{y}kvk\ddot{y}1$$

**Zeilensuche:** $\ddot{y}k = \dfrac{^{R}k\ddot{y}1\ r k\ddot{y}1}{^{v}{}_{k}\ E\ddot{y}1AE\ddot{y}vk}$

**Schätzung aktualisieren:** $yk = yk\ddot{y}1 + \ddot{y}kvk$

**Residuum aktualisieren:rk** $= rk\ddot{y}1 \ddot{y}\ \ddot{y}kE\ddot{y}1AE\ddot{y}vk$

Dieses iterative Schema wird entsprechend der Konditionierung unserer Matrix $E\ddot{y}1AE\ddot{y}$ konvergieren.

Definieren Sie $r\tilde{}k \ddot{y} Erk$, $v\tilde{}k \ddot{y} E\ddot{y}vk$ und $xk \ddot{y} Eyk$. Wenn wir uns an die Beziehung $P = E\ddot{y}E$ erinnern, $^{\ddot{y}1}$, wir können schreiben wir unsere vorkonditionierte konjugierte Gradienteniteration unter Verwendung dieser neuen Variablen neu:

**Suchrichtung aktualisieren:** $\ddot{y}k = \dfrac{r\tilde{}\ k\ddot{y}1Pr\tilde{}k\ddot{y}1}{r\tilde{}\ k\ddot{y}2Pr\tilde{}k\ddot{y}2}$

$$\tilde{v}_k = Pr\tilde{}_{k\ddot{y}1} + \ddot{y}k\tilde{v}_{k\ddot{y}1}$$

**Zeilensuche:** $\ddot{y}k = \frac{r_{k\ddot{y}1} \quad Prk\ddot{y}1}{\tilde{v}_k \; A\tilde{v}_k}$

**Schätzung aktualisieren:** $xk = xk\ddot{y}1 + \ddot{y}k\tilde{v}_k$

**Residuum aktualisieren:** $r\tilde{}_k = r\tilde{}_{k\ddot{y}1} \ddot{y} \ddot{y}kA\tilde{v}_k$

Diese Iteration hängt nicht von der Cholesky-Faktorisierung von P ab, die ausschließlich $\ddot{y}1$ überhaupt, sondern kann unter Verwendung von P und A durchgeführt wird. Es ist leicht zu erkennen, dass das xk ÿ x-Schema $\ddot{y}$, tatsächlich so sein die Vorteile der Vorkonditionierung genießt, ohne dass der Vorkonditionierer faktorisiert werden muss.

Nebenbei bemerkt: Eine noch effektivere Vorkonditionierung kann durchgeführt werden, indem A durch PAQ für eine zweite Matrix Q ersetzt wird, obwohl für die Anwendung dieser zweiten Matrix zusätzliche Berechnungen erforderlich sind. Dieses Beispiel stellt einen häufigen Kompromiss dar: Wenn die Anwendung eines Vorkonditionierers selbst in einer einzelnen Iteration von CG oder einer anderen Methode zu lange dauert, lohnt sich die reduzierte Anzahl von Iterationen möglicherweise nicht.

### 10.3.2 Gemeinsame Vorkonditionierer

In der Praxis gute Vorkonditionierer zu finden, ist ebenso eine Kunst wie eine Wissenschaft. Das Finden des Besten hängt Näherung P von A usw. $\ddot{y}1$ von der Struktur von A, der jeweiligen Anwendung usw. ab
Selbst grobe Näherungen können die Konvergenz jedoch erheblich verbessern, so dass CG-Anwendungen, die keinen Vorkonditionierer verwenden, selten vorkommen.

Die beste Strategie zur Formulierung von P ist oft anwendungsspezifisch, und ein interessantes technisches Näherungsproblem besteht darin, verschiedene Ps für den besten Vorkonditionierer zu entwerfen und zu testen.
Nachfolgend finden Sie zwei gängige Strategien:

- Ein diagonaler (oder „Jacobi“) Vorkonditionierer nimmt einfach P als die Matrix an, die man durch Invertieren diagonaler Elemente von A erhält; das heißt, P ist die Diagonalmatrix mit den Einträgen 1/aii. Diese Strategie kann eine ungleichmäßige Skalierung von Zeile zu Zeile verringern, die eine häufige Ursache für schlechte Konditionierung ist.

- Der spärliche approximative inverse Vorkonditionierer wird durch Lösen eines Teilproblems minPÿS AP ÿ IFro formuliert, wobei P darauf beschränkt ist, in einer Menge S von Matrizen zu sein, über die es weniger schwierig ist, ein solches Ziel zu optimieren. Eine häufige Einschränkung besteht beispielsweise darin, ein Sparsity-Muster für P vorzuschreiben, z. B. nur auf der Diagonale ungleich Nullen oder wenn A ungleich Nullen hat.

- Die unvollständigen Cholesky-Vorbedingungsfaktoren A ÿ LÿL $\ddot{y}$ und nähert sich dann A an $\ddot{y}1$ durch Lösen der entsprechenden Vorwärts- und Rücksubstitutionsprobleme. Eine beliebte Strategie besteht beispielsweise darin, die Schritte der Cholesky-Faktorisierung durchzuführen, die Ausgabe jedoch nur an den Positionen (i, j) zu speichern, an denen aij = 0 ist.

- Die Werte ungleich Null in A können als Diagramm betrachtet werden, und das Entfernen von Kanten im Diagramm oder das Gruppieren von Knoten kann dazu führen, dass verschiedene Komponenten getrennt werden. Das resultierende System ist nach dem Permutieren von Zeilen und Spalten blockdiagonal und kann daher mithilfe einer Folge kleinerer Lösungen gelöst werden. Eine solche Domänenzerlegungsstrategie kann für lineare Systeme effektiv sein, die aus Differentialgleichungen wie den in Kapitel NUMBER betrachteten entstehen.

Einige Vorkonditionierer verfügen über Grenzen, die Änderungen an der Konditionierung von A nach dem Ersetzen durch PA beschreiben. In den meisten Fällen handelt es sich jedoch um heuristische Strategien, die getestet und verfeinert werden sollten.

## 10.4 Andere iterative Schemata

Die Algorithmen, die wir in diesem Kapitel ausführlich entwickelt haben, gelten für die Lösung von Ax = b, wenn A quadratisch, symmetrisch und positiv definit ist. Wir haben uns auf diesen Fall konzentriert, weil er in der Praxis so häufig vorkommt, aber es gibt Fälle, in denen A asymmetrisch, unbestimmt oder sogar rechteckig ist. Es liegt außerhalb des Rahmens unserer Diskussion, in jedem Fall iterative Algorithmen abzuleiten, da viele eine spezielle Analyse oder fortgeschrittene Entwicklung erfordern. Wir fassen hier jedoch einige Techniken auf hoher Ebene zusammen (JEDEN zitieren):

- Aufteilungsmethoden zerlegen A = M ÿ N und beachten, dass Ax = b äquivalent zu Mx = Nx +b ist. Wenn M leicht zu invertieren ist, kann ein Festkommaschema abgeleitet werden, indem man schreibt: Mxk = Nxkÿ1 +b (CITE); Diese Techniken sind einfach zu implementieren, weisen jedoch eine Konvergenz auf, die vom Spektrum der Matrix G = Mÿ1N abhängt , und können insbesondere divergieren, wenn der Spektralradius von G größer als eins ist. Eine beliebte Wahl für M ist die Diagonale von A. Methoden wie die sukzessive Überrelaxation (SOR) gewichten diese beiden Terme für eine bessere Konvergenz.

- Die Methode der konjugierten Gradientennormalgleichung (CGNR) wendet einfach den CG-Algorithmus auf die Normalengleichungen A Ax = A b an. Diese Methode ist einfach zu implementieren und konvergiert garantiert, solange A den vollen Rang hat. Die Konvergenz kann jedoch aufgrund der schlechten Konditionierung von AA langsam sein, wie in Kapitel NUMBER erläutert.

- Die Methode des konjugierten Gradientennormalgleichungsfehlers (CGNE) löst auf ähnliche Weise AAy =b; Dann die Lösung von Ax =b ist einfach A y.

- Methoden wie MINRES und SYMMLQ gelten für symmetrische, aber nicht unbedingt positiv definite Matrizen A, indem wir unsere quadratische Form f(x) durch g(x) ÿ b ÿ Ax2 ersetzen; Diese Funktion g wird bei Lösungen für Ax =b minimiert, unabhängig von der Bestimmtheit von A.

- Angesichts der schlechten Konditionierung von CGNR und CGNE minimieren die Algorithmen LSQR und LSMR auch g(x) mit weniger Annahmen zu A, was insbesondere die Lösung von Systemen der kleinsten Quadrate ermöglicht.

- Verallgemeinerte Methoden einschließlich GMRES, QMR, BiCG, CGS und BiCGStab lösen Ax =b mit der einzigen Einschränkung, dass A quadratisch und invertierbar ist. Sie optimieren ähnliche Energien, müssen jedoch häufig mehr Informationen über frühere Iterationen speichern und möglicherweise Zwischenmatrizen faktorisieren, um die Konvergenz mit dieser Allgemeingültigkeit zu gewährleisten.

- Schließlich kehren die Fletcher-Reeves-, Polak-Ribi`ere- und andere Methoden zum allgemeineren Problem der Minimierung einer nichtquadratischen Funktion f zurück, indem sie konjugierte Gradientenschritte anwenden, um neue Liniensuchrichtungen zu finden. Funktionen f, die durch Quadrate gut approximiert werden, können mit diesen Strategien sehr effektiv minimiert werden, obwohl sie nicht unbedingt die Hesse-Funktion nutzen; Beispielsweise ersetzt die Fletcher-Reeves-Methode einfach das Residuum in CG-Iterationen durch den negativen Gradienten ÿÿ f . Es ist möglich, die Konvergenz dieser Methoden zu charakterisieren, wenn sie von ausreichend effektiven Liniensuchstrategien begleitet werden.

Viele dieser Algorithmen sind fast so einfach zu implementieren wie CG oder Gradient Descent, und es gibt viele Implementierungen, die lediglich die Eingabe von A und b erfordern. Viele der oben aufgeführten Algorithmen erfordern die Anwendung von A und $A^n$, was in manchen Fällen eine technische Herausforderung darstellen kann. Als Faustregel gilt: Je allgemeiner eine Methode ist – d. h. je weniger Annahmen sie über die Struktur der Matrix A trifft –, desto mehr Iterationen ist wahrscheinlich erforderlich, um diesen Mangel an Annahmen auszugleichen. Allerdings gibt es keine festen Regeln, wenn man sich einfach nur das erfolgreichste iterative Schema anschaut, obwohl es nur begrenzte theoretische Diskussionen zum Vergleich der Vor- und Nachteile jeder dieser Methoden gibt (CITE).

### 10.5 Probleme

- Leiten Sie CGNR und/oder CGNE ab
- MINRES ableiten
- Leiten Sie Fletcher-Reeves ab
- Folie 13 von http://math.ntnu.edu.tw/~min/matrix_computation/Ch4_Slide4_CG_2011.pdf

# Teil IV

# Funktionen, Ableitungen und Integrale

# Kapitel 11

# Interpolation

Bisher haben wir Methoden zur Analyse von Funktionen f abgeleitet, z. B. zum Finden ihrer Minima und Wurzeln. Die Auswertung von f(x) für ein bestimmtes x ÿ Rn könnte teuer sein, aber eine Grundannahme der Methoden, die wir in den vorherigen Kapiteln entwickelt haben, ist, dass wir f(x) unabhängig von x erhalten können, wenn wir es wollen.

Es gibt viele Kontexte, in denen diese Annahme nicht realistisch ist. Wenn wir beispielsweise ein Foto mit einer Digitalkamera aufnehmen, erhalten wir ein n × m-Raster aus Pixelfarbwerten, das das Kontinuum des Lichts abtastet, das in ein Kameraobjektiv einfällt. Wir könnten uns ein Foto als eine kontinuierliche Funktion von der Bildposition (x, y) bis zur Farbe (r, g, b) vorstellen, aber in Wirklichkeit kennen wir den Bildwert nur an nm getrennten Orten auf der Bildebene. In ähnlicher Weise erhalten wir beim maschinellen Lernen und in der Statistik oft nur Stichproben einer Funktion an den Punkten, an denen wir Daten gesammelt haben, und wir müssen sie interpolieren, um an anderer Stelle Werte zu haben; In einem medizinischen Umfeld überwachen wir möglicherweise die Reaktion eines Patienten auf unterschiedliche Dosierungen eines Arzneimittels, können jedoch nur vorhersagen, was bei einer Dosierung passieren wird, die wir nicht explizit ausprobiert ha

Bevor wir in diesen Fällen eine Funktion minimieren, ihre Wurzeln finden oder sogar Werte f(x) an beliebigen Orten x berechnen können, benötigen wir ein Modell für die Interpolation von f(x) auf den gesamten Rn (oder eine Teilmenge davon) bei gegebenem a Sammlung von Proben f(xi). Natürlich sind Techniken zur Lösung dieses Interpolationsproblems von Natur aus Näherungsmethoden, da wir die wahren Werte von f nicht kennen. Stattdessen streben wir danach, dass die interpolierte Funktion glatt ist und als „vernünftige“ Vorhersage der Funktionswerte dient.

In diesem Kapitel gehen wir davon aus, dass die Werte f(xi) mit völliger Sicherheit bekannt sind; In diesem Fall könnten wir uns das Problem auch so vorstellen, dass f auf den Rest der Domäne ausgedehnt wird, ohne den Wert an einer der Eingabestellen zu stören. In Kapitel NUMBER (SCHREIBEN SIE MICH IN 2014) werden wir das Regressionsproblem betrachten, bei dem der Wert f(xi) mit einer gewissen Unsicherheit bekannt ist. In diesem Fall können wir auf die Anpassung von f(xi) vollständig verzichten und stattdessen f glatter machen .

## 11.1 Interpolation in einer einzelnen Variablen

Bevor wir den allgemeinsten Fall betrachten, werden wir Methoden zur Interpolation von Funktionen einer einzelnen Variablen f : **R** ÿ R entwerfen. Als Eingabe nehmen wir eine Menge von k Paaren (xi , yi) mit der Annahme f(xi) = yi ; Unsere Aufgabe ist es, f(x) für x ÿ {x1, . . . , xk}.

Unsere Strategie in diesem und anderen Abschnitten lässt sich von der linearen Algebra inspirieren, indem wir f(x) in eine Basis schreiben. Das heißt, die Menge aller möglichen Funktionen f : **R** ÿ **R** ist viel zu groß, um damit zu arbeiten, und enthält viele Funktionen, die in einer rechnerischen Umgebung nicht praktikabel sind. Daher vereinfachen wir den Suchraum, indem wir erzwingen, dass f als Linearkombination einfacherer Bausteine geschrieben wird

Basisfunktionen. Diese Strategie ist bereits aus der Grundrechnung bekannt: Die Taylor-Entwicklung schreibt Funktionen auf der Basis von Polynomen, während Fourier-Reihen Sinus und Cosinus verwenden.

### 11.1.1 Polynominterpolation

Der vielleicht einfachste Interpolant besteht darin, anzunehmen, dass f(x) in R[x], der Menge der Polynome, liegt . Polynome sind glatt und es ist einfach, ein Polynom vom Grad k ÿ 1 durch k Abtastpunkte zu finden.

Tatsächlich werden in Beispiel 3.3 bereits die Details einer solchen Interpolationstechnik erläutert. Als ein Zur Erinnerung: Angenommen, wir möchten f(x) ÿ a0 + a1x + a2x finden $^{2}$ + . . . + akÿ1x $^{kÿ1}$ ; hier sind unsere Unbekannten die Werte a0, . . . , akÿ1 . Das Einsetzen des Ausdrucks yi = f(xi) für jedes i zeigt, dass der Vektora das k × k Vandermonde-System erfüllt:

$$\begin{array}{ccccccc} ÿ & 1x1 & x_1^2 & \cdots x_1^{kÿ1} & ÿ & a0 & ÿ \\ & 1x2 & x_2^2 & x\,2^{kÿ1} & ÿ & a1 & ÿ \\ & \vdots & \vdots & \vdots & & \vdots & \\ ÿÿÿÿ & 1\ xkÿ1\ x & {}^{2}_{kÿ1} & \cdots x\ kÿ1^{kÿ1} & ÿÿÿÿ & akÿ1 & ÿ \end{array} \quad ÿÿÿÿ = \begin{array}{c} y0 \\ y1 \\ \vdots \\ ykÿ1 \end{array} \quad ÿÿÿÿ$$

Somit kann die Durchführung einer Grad-k-Polynominterpolation mithilfe der linearen ak × k-Lösung unter Anwendung unserer generischen Strategien aus den vorherigen Kapiteln durchgeführt werden, aber tatsächlich können wir es besser machen.

Eine Möglichkeit, über unsere Form für f(x) nachzudenken, besteht darin, dass sie in einer Basis geschrieben ist. Genauso wie eine Basis für Rn eine Menge von n linear unabhängigen Vektoren v1, . . . ,vn, hier der Raum der Polynome für k ÿ 1 sein, die in der Spanne der Monome {1, x, x R[x] $^{2}$ , . . . , vom Grad x kÿ1}. Es mag die offensichtlichste Basis geschrieben wird, aber unsere aktuelle Wahl hat nur wenige Eigenschaften, die sie für das Interpolationsproblem nützlich machen. Eine Möglichkeit, dieses Problem zu erkennen, besteht darin, die Folge der Funktionen 1, x, x k $^{2}$ , $^{3x}$ − , . . . für x ÿ [0, 1]; in diesem aussehen. Intervall ist es leicht zu erkennen, dass die Funktionen x größer werden, je größer k ist, die alle ähnlich

Wenn wir weiterhin unsere Intuition aus der linearen Algebra anwenden, können wir uns dafür entscheiden, unser Polynom auf einer Basis zu schreiben, die für das vorliegende Problem besser geeignet ist. Denken Sie dieses Mal daran, dass wir k Paare (x1, y1) erhalten . . . ,(xk , yk). Wir werden diese (Fix-)Punkte verwenden, um die Lagrange-Interpolationsbasis ÿ1, . zu definieren . . , ÿk indem man schreibt:

$$ÿi(x) ÿ \frac{ÿj=i\ (x ÿ xj)}{ÿj=i\ (xi ÿ xj)}$$

Obwohl es nicht in der Basis 1 geschrieben ist, x, $^{2}$ , . . . , $x^{kÿ1}$ , Es ist leicht zu erkennen, dass jedes ÿi immer noch ein Poly ist x Nominal vom Grad k ÿ 1. Darüber hinaus hat die Lagrange-Basis die folgende wünschenswerte Eigenschaft:

$$ÿi(x) = \begin{cases} 1 \text{ wenn} = i \\ 0 \text{ sonst.} \end{cases}$$

Daher ist es auf der Lagrange-Basis einfach , das eindeutige Polynom vom Grad k ÿ 1 zu finden, das zu unseren (xi , yi) -Paaren passt:

$$f(x) ÿ ÿ yiÿi(x)$$

Insbesondere wenn wir x = xj einsetzen, finden wir: f(xj)

$$= ÿ yiÿi(xj)$$

= yj durch unseren obigen Ausdruck für ÿi(x) .

Somit können wir in der Lagrange-Basis eine geschlossene Formel für f(x) schreiben , die keine Lösung des Vandermonde-Systems erfordert. Der Nachteil besteht jedoch darin, dass jedes ÿi(x) O(k) Zeit benötigt , um unter Verwendung der obigen Formel für ein gegebenes x ausgewertet zu werden. Für die Ermittlung von f(x) $^2$) Zeit; wenn wir die Koeffizienten finden wird also explizit O(n ai ) aus dem Vandermonde-System benötigt , jedoch die Auswertung Die Zeit kann auf O(n) reduziert werden .

Abgesehen von der Rechenzeit hat die Lagrange-Basis einen zusätzlichen numerischen Nachteil. Beachten Sie, dass der Nenner das Produkt mehrerer Terme ist. Wenn die xi nahe beieinander liegen, enthält das Produkt möglicherweise viele Terme nahe Null, sodass wir durch eine potenziell kleine Zahl dividieren. Wie wir gesehen haben, kann dieser Vorgang zu numerischen Problemen führen, die wir vermeiden möchten.

Eine Basis für Polynome vom Grad k ÿ 1, die versucht, einen Kompromiss zwischen der numerischen Qualität der Monome und der Effizienz der Lagrange-Basis zu finden, ist die Newton-Basis, die wie folgt definiert ist:

$$\ddot{y}i(x) = (x \ddot{y} \ddot{y}xj)$$
$$j=1$$

Wir definieren ÿ1(x) ÿ 1. Beachten Sie, dass ÿi(x) ein Polynom vom Grad i ÿ 1 ist. Durch die Definition von ÿi ist klar, dass ÿi(x) = 0 für alle < i. Wenn wir f(x) = ÿi ciÿi(x) schreiben und diese Beobachtung expliziter formulieren wollen, finden wir:

$$\begin{aligned} f(x1) &= c1\ddot{y}1(x1) \; f(x2) \\ &= c1\ddot{y}1(x2) + c2\ddot{y}2(x2) \; f(x3) = \\ &c1\ddot{y}1(x3) + c2\ddot{y}2(x3) + c3\ddot{y}3(x3) \\ &\vdots \quad \vdots \end{aligned}$$

Mit anderen Worten, wir können die folgende untere Dreieckssystemkraft lösen:

$$\ddot{y} \begin{matrix} \ddot{y}1(x1) & 0 & \ddot{y}1(x2) & \ddot{y}2 & \cdots & 0 \\ (x2) & 0 & \ddot{y}1(x3) & \ddot{y}2(x3) & \cdots & 0 \\ \ddot{y}3(x3) & \cdots & & & & 0 \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ \ddot{y}1(xk) & \ddot{y}2(xk) & \ddot{y}3(xk) & \cdots & \ddot{y}k(xk) & \end{matrix} \ddot{y} \begin{matrix} c1 \\ c2 \\ \vdots \\ ck \end{matrix} \ddot{y} = \ddot{y} \begin{matrix} y1 \\ y2 \\ \vdots \\ Ja \end{matrix} \ddot{y}$$

Dieses System kann in O(n gelöst werden, $^2$) Zeit unter Verwendung der Vorwärtssubstitution anstelle der O(n $^3$) Zeit die zur Lösung des Vandermonde-Systems benötigt werden.

Wir haben jetzt drei Strategien zum Interpolieren von k Datenpunkten unter Verwendung eines Polynoms vom Grad k ÿ 1, indem wir es auf der Monom-, Lagrange- und Newton-Basis schreiben. Alle drei stellen unterschiedliche Kompromisse zwischen numerischer Qualität und Geschwindigkeit dar. Eine wichtige Eigenschaft ist jedoch, dass die resultierende interpolierte Funktion f(x) in jedem Fall dieselbe ist. Genauer gesagt gibt es genau ein Polynom vom Grad k ÿ 1, das durch eine Menge von k Punkten geht. Da also alle unsere Interpolanten vom Grad k ÿ 1 sind, müssen sie die gleiche Ausgabe haben.

### 11.1.2 Alternative Basen

Obwohl sich Polynomfunktionen besonders gut für die mathematische Analyse eignen, gibt es keinen grundsätzlichen Grund, warum unsere Interpolationsbasis nicht aus verschiedenen Arten von Funktionen bestehen kann. Ein krönendes Ergebnis der Fourier-Analyse impliziert beispielsweise, dass eine große Klasse von Funktionen durch Summen der trigonometrischen Funktionen cos(kx) und sin(kx) für k ÿ N gut angenähert werden kann. Eine Konstruktion

wie das Vandermonde-System gilt in diesem Fall immer noch, und tatsächlich zeigt der Fast-Fourier-Transformationsalgorithmus (der eine ausführlichere Diskussion verdient) wie man eine solche Interpolation noch schneller durchführen kann.

Eine kleinere Erweiterung der Entwicklung in §11.1.1 betrifft rationale Funktionen der Form:

$$f(x) = \frac{p0 + p1x + p2x^2 + \cdots + pmx^M}{q0 + q1x + q2x^2 + \cdots + qnx^N}$$

Beachten Sie, dass wir bei gegebenen k Paaren (xi , yi) m + n + 1 = k benötigen , damit diese Funktion wohldefiniert ist. Es muss ein zusätzlicher Freiheitsgrad festgelegt werden, um der Tatsache Rechnung zu tragen, dass dieselbe rationale Funktion durch identische Skalierung von Zähler und Nenner auf mehrere Arten ausgedrückt werden kann.

Rationale Funktionen können Asymptoten und andere Muster aufweisen, die mit nur Polynomen nicht erreichbar sind, daher können sie wünschenswerte Interpolanten für Funktionen sein, die sich schnell ändern oder Pole haben. Sobald m und n festgelegt sind, können die Koeffizienten pi und qi tatsächlich immer noch mithilfe linearer Techniken ermittelt werden, indem beide Seiten mit dem Nenner multipliziert werden:

$$yi(q0 + q1xi + q2x^2_{\_} + \cdots + qnx^N_{\_}) = p0 + p1xi + p2x^2_{\_} + \cdots + pmx^M_{\_}$$

Auch hier sind die Unbekannten in diesem Ausdruck die ps und qs.

Die Flexibilität rationaler Funktionen kann jedoch einige Probleme verursachen. Betrachten Sie zum Beispiel das folgende Beispiel:

**Beispiel 11.1** (Fehler der rationalen Interpolation, Bulirsch-Stoer §2.2). Angenommen, wir möchten f(x) mit den folgenden Datenpunkten finden : (0, 1), (1, 2), (2, 2). Wir könnten m = n = 1 wählen. Dann werden unsere linearen Bedingungen zu:

$$q0 = p0$$

$$2(q0 + q1) = p0 + p1$$

$$2(q0 + 2q1) = p0 + 2p1$$

Eine nicht triviale Lösung für dieses System ist:

$$p0 = 0$$

$$p1 = 2$$

$$q0 = 0$$

$$q1 = 1$$

Dies impliziert die folgende Form für f(x):

$$f(x) = \frac{2x}{x}$$

Diese Funktion weist bei x = 0 eine Entartung auf, und tatsächlich führt die Streichung des x im Zähler und Nenner nicht zu f(0) = 1, wie wir es uns wünschen.

Dieses Beispiel veranschaulicht ein größeres Phänomen. Unser lineares System zum Finden der p- und q-Werte kann auf Probleme stoßen, wenn der resultierende Nenner ÿ px eine Wurzel bei einem der festen xi hat .
Es kann gezeigt werden, dass in diesem Fall keine rationale Funktion mit der festen Wahl von m und n existiert, die die gegebenen Werte interpolieren. Eine typische Teillösung für diesen Fall wird in (CITE) dargestellt, bei der m und n abwechselnd erhöht werden, bis eine nichttriviale Lösung vorliegt. Aus praktischer Sicht ist die Spezialisierung dieser Methoden jedoch ein guter Indikator dafür, dass alternative Interpolationsstrategien vorzuziehen sein könnten, wenn die grundlegenden rationalen Methoden versagen.

## 11.1.3 Stückweise Interpolation

Bisher haben wir unsere Interpolationsstrategien durch die Kombination einfacher Funktionen für ganz R konstruiert.
Wenn die Anzahl k der Datenpunkte jedoch hoch wird, werden viele Entartungen sichtbar.
Abbildung NUMBER zeigt beispielsweise Beispiele, in denen die Anpassung von Polynomen höheren Grades an Eingabedaten zu unerwarteten Ergebnissen führen kann. Darüber hinaus zeigt Abbildung NUMBER, dass diese Strategien nichtlokal sind, was bedeutet, dass die Änderung eines einzelnen Werts yi in den Eingabedaten das Verhalten von f für alle x ändern kann, auch für diejenigen, die weit vom entsprechenden xi entfernt sind . Irgendwie ist diese Eigenschaft unrealistisch: Wir erwarten, dass nur die Eingabedaten in der Nähe eines bestimmten x den Wert von f(x) beeinflussen , insbesondere wenn es eine große Wolke von Eingabepunkten gibt.

Wenn wir aus diesen Gründen einen Satz von Basisfunktionen ÿ1 entwerfen, . . . , ÿk , eine wünschenswerte Eigenschaft ist nicht nur, dass sie einfach zu bearbeiten sind, sondern auch, dass sie eine kompakte Unterstützung haben:

**Definition 11.1** (Kompakte Unterstützung). Eine Funktion g(x) hat kompakte Unterstützung, wenn es C ÿ **R** gibt, so dass g(x) = 0 für jedes x mit |x| > C.

Das heißt, kompakt unterstützte Funktionen haben nur einen endlichen Bereich von Punkten, in denen sie Werte ungleich Null annehmen können.

Eine übliche Strategie zur Konstruktion interpolierender Basen mit kompakter Unterstützung besteht darin, dies stückweise zu tun. Insbesondere ein Großteil der Literatur zur Computergrafik basiert auf der Konstruktion stückweiser Polynome, die definiert werden, indem **R** in eine Reihe von Intervallen aufgeteilt und in jedes Intervall ein anderes Polynom geschrieben wird. Dazu ordnen wir unsere Datenpunkte so an, dass x1 < x2 < · · · < xk . Dann sind zwei einfache Beispiele für stückweise Interpolanten die folgenden:

- Stückweise Konstante (Abbildung NUMBER): Finden Sie für ein gegebenes x den Datenpunkt xi , der |x ÿ minimiert xi | und definiere f(x) = yi .

- Stückweise linear (Abbildung NUMBER): Wenn x < x1 , ist f(x) = y1, und wenn x > xk , ist f(x) = yk . Andernfalls suchen Sie ein Intervall mit x ÿ [xi , xi+1] und definieren Sie es

$$f(x) = y_{i+1} \cdot \frac{x \ddot{y} x_i}{x_{i+1} \ddot{y} x_i} + y_{i-1} \ddot{y} \frac{x \ddot{y} x_i}{x_{i+1} \ddot{y} x_i}.$$

Allgemeiner gesagt können wir in jedem Intervall [xi , xi+1] ein anderes Polynom schreiben. Beachten Sie unser bisheriges Muster: Stückweise konstante Polynome sind diskontinuierlich, während stückweise lineare Funktionen stetig sind. Es ist leicht zu erkennen, dass stückweise Quadrate C1 und stückweise Kubik C2 sein können. Diese erhöhte Kontinuität und Differenzierbarkeit tritt auf, obwohl jedes Yi lokale Unterstützung hat; Diese Theorie wird im Detail bei der Konstruktion von „Splines“ oder Kurven, die zwischen Punkten mit gegebenen Funktionswerten und Tangenten interpolieren, ausgearbeitet.

Diese erhöhte Kontinuität hat jedoch ihre eigenen Nachteile. Mit jedem zusätzlichen Differenzierbarkeitsgrad setzen wir eine stärkere Glätteannahme für f. Diese Annahme kann unrealistisch sein: Viele physikalische Phänomene sind tatsächlich verrauscht oder diskontinuierlich, und diese erhöhte Glätte kann sich negativ auf die Interpolationsergebnisse auswirken. Ein Bereich, in dem dieser Effekt besonders deutlich wird, ist die Verwendung der Interpolation in Verbindung mit physikalischen Simulationswerkzeugen. Durch die Simulation turbulenter Flüssigkeitsströme mit überglätteten Funktionen können diskontinuierliche Phänomene wie Stoßwellen, die als Ausgabe wünschenswert sind, entfernt werden.

Abgesehen von diesen Problemen können stückweise Polynome immer noch als lineare Kombinationen von Basisfunktionen geschrieben werden. Als Basis für die stückweise konstanten Funktionen dienen beispielsweise folgende Funktionen:

$$\ddot{y}i(x) = \begin{cases} 1 \text{ wenn } xi\ddot{y}1 + xi\ddot{y}x < \frac{xi+xi+1}{2} \\ 0 \text{ sonst} \end{cases}$$

Diese Basis setzt einfach die Konstante 1 in die Nähe von xi und 0 anderswo; Die stückweise konstante Interpolation einer Menge von Punkten (xi , yi) wird geschrieben als f(x) = ÿi yiÿi(x). In ähnlicher Weise umfasst die in Abbildung NUMBER gezeigte sogenannte „Hut“-Basis die Menge stückweise linearer Funktionen mit scharfen Kanten an unseren Datenpunkten xi :

$$\ddot{y}i(x) = \begin{cases} \ddot{y} \\ \\ \ddot{y}\ddot{y} \\ \\ \ddot{y}\ddot{y} \end{cases} \begin{array}{ll} \frac{x\ddot{y}xi\ddot{y}1}{xi\ddot{y}xi\ddot{y}1} & \text{wenn } xi\ddot{y}1 < x \ddot{y} xi \text{ wenn} \\ \frac{xi+1\ddot{y}x}{xi+1\ddot{y}xi} & xi < x \ddot{y} xi+1 \text{ sonst} \\ 0 & \end{array}$$

Auch hier ist die stückweise lineare Interpolation der gegebenen Datenpunkte konstruktionsbedingt f(x) = ÿi yiÿi(x).

### 11.1.4 Gaußsche Prozesse und Kriging

Nicht abgedeckt in CS 205A, Herbst 2013.

## 11.2 Multivariable Interpolation

Es gibt viele Erweiterungen der oben genannten Strategien zum Interpolieren einer Funktion mit gegebenen Datenpunkten (xi , yi) , bei denen xi ÿ Rn nun mehrdimensional sein kann. Allerdings sind die Interpolationsstrategien in diesem Fall nicht ganz so klar, da es weniger offensichtlich ist, Rn in eine kleine Anzahl von Regionen um xi herum zu unterteilen . Aus diesem Grund besteht ein gängiges Muster darin, mit Funktionen relativ niedriger Ordnung zu interpolieren, d. h. einfache und effiziente Interpolationsstrategien gegenüber solchen zu bevorzugen, die C ÿ -Funktionen ausgeben.

Wenn alles, was uns gegeben wird, die Menge der Ein- und Ausgänge (xi , yi) ist, dann ist eine stückweise konstante Strategie für die Interpolation die Verwendung der Interpolation des nächsten Nachbarn. In diesem Fall nimmt f(x) einfach den Wert yi an, der xi entspricht und x ÿ xi2 minimiert; Einfache Implementierungen iterieren über alle i, um diesen Wert zu finden, obwohl Datenstrukturen wie KD-Bäume die nächsten Nachbarn schneller finden können. So wie stückweise konstante Interpolationen **R** in Intervalle um die Datenpunkte xi aufteilen , teilt die Nächste-Nachbarn-Strategie Rn in eine Menge von Voronoi-Zellen:

**Definition 11.2** (Voronoi-Zelle). Gegeben sei eine Menge von Punkten S = {x1,x2, . . . ,xk} ÿ Rn , die Voronoi-Zelle entsprechend einem bestimmten xi ist die Menge Vi ÿ {x : x ÿxi2 < x ÿxj2 für alle j = i}. Das heißt, es ist die Menge der Punkte, die näher an xi liegen als an jedem anderen xj in S.

Abbildung NUMBER zeigt ein Beispiel dafür, dass die Voronoi-Zellen über eine Reihe von Datenpunkten in R2- . Diese Zellen viele günstige Eigenschaften haben; Beispielsweise handelt es sich um konvexe Polygone, die um jedes xi herum lokalisiert sind . Tatsächlich ist die Konnektivität von Voronoi-Zellen ein gut untersuchtes Problem der Computergeometrie, das zur Konstruktion der berühmten Delaunay-Triangulation führte.

Es gibt viele Optionen für die kontinuierliche Interpolation von Funktionen auf Rn, jede mit ihren eigenen Vor- und Nachteilen. Wenn wir beispielsweise unsere obige Nächste-Nachbarn-Strategie erweitern möchten, könnten wir mehrere nächste Nachbarn von x berechnen und f(x) basierend auf x ÿ interpolieren

xi2 für jeden nächsten Nachbarn xi . Bestimmte „k-nächste Nachbar“-Datenstrukturen können Abfragen beschleunigen, bei denen Sie mehrere Punkte in einem Datensatz finden möchten, die einem bestimmten x am nächsten liegen.

Eine weitere Strategie, die in der Literatur zur Computergrafik häufig vorkommt, ist die baryzentrische Interpolation. Angenommen, wir haben genau n + 1 Abtastpunkte (x1, y1), . . . ,(xn+1, yn+1), wobei xi ÿ Rn und wir wie immer die y-Werte , auf das gesamte Rn interpolieren möchten ; Auf der Ebene würden uns beispielsweise drei Werte gegeben, die den Eckpunkten eines Dreiecks zugeordnet sind. Jeder Punkt x ÿ Rn kann eindeutig als lineare Kombination x = ÿ n+1 i=1 aixi mit der zusätzlichen Einschränkung geschrieben werden, dass ÿi ai = 1; mit anderen Worten, wir schreiben x als gewichteten Durchschnitt der Punkte xi . Die baryzentrische Interpolation schreibt in diesem Fall einfach f(x) = ÿi ai(x)yi .

Auf der Ebene R2 weist die baryzentrische Interpolation eine direkte geometrische Interpolation in umlaufenden Dreiecksflächen auf, wie in Abbildung NUMBER dargestellt. Darüber hinaus lässt sich leicht überprüfen, ob die resultierende interpolierte Funktion f(x) affin ist, was bedeutet, dass sie für ein c ÿ **R** und d ÿ Rn als f(x) = c + d · x geschrieben werden kann .

Im Allgemeinen ist das Gleichungssystem, das wir für eine baryzentrische Interpolation lösen möchten, bei einigen x ÿ Rn ist:

$$\ddot{y}\ a_i x_i = x$$

$$\ddot{y}\ a_i = 1$$

In Abwesenheit von Entartungen ist dieses System fora invertierbar, wenn es n + 1 Punkte xi gibt . Wenn jedoch mehr xi vorhanden sind , wird das System für a unterbestimmt. Das bedeutet, dass es mehrere Möglichkeiten gibt, ein gegebenes x als gewichteten Durchschnitt der xi zu schreiben .

Eine Lösung dieses Problems besteht darin, weitere Bedingungen für den Vektor der Durchschnittsgewichte hinzuzufügensa. Diese Strategie führt zu verallgemeinerten baryzentrischen Koordinaten, einem Forschungsthema in der modernen Mathematik und Technik. Typische Einschränkungen erfordern, dass es als Funktion auf Rn glatt und im Inneren der Menge von xi nicht negativ ist , wenn diese Punkte ein Polygon oder Polyeder definieren. Abbildung NUMBER zeigt ein Beispiel für verallgemeinerte Schwerpunktkoordinaten, die aus Datenpunkten auf einem Polygon mit mehr als n + 1 Punkten berechnet wurden.

Eine alternative Lösung des unterbestimmten Problems für baryzentrische Koordinaten bezieht sich auf die Idee, stückweise Funktionen für die Interpolation zu verwenden; Der Einfachheit halber beschränken wir unsere Diskussion hier auf xi ÿ R2 , obwohl die Erweiterungen auf höhere Dimensionen relativ offensichtlich sind.
Oftmals erhalten wir nicht nur die Menge der Punkte xi , sondern auch eine Zerlegung des Bereichs, um den wir uns kümmern (in diesem Fall eine Teilmenge von R2 ), in n + 1-dimensionale Objekte, wobei diese Punkte als Eckpunkte verwendet werden. Abbildung NUMBER zeigt beispielsweise eine solche Tessellation eines Teils von R2 in Dreiecke.
Die Interpolation ist in diesem Fall unkompliziert: Das Innere jedes Dreiecks wird mithilfe baryzentrischer Koordinaten interpoliert.

**Beispiel 11.2** (Schattierung). In der Computergrafik ist eine der häufigsten Darstellungen einer Form eine Reihe von Dreiecken in einem Netz. Im Schattierungsmodell pro Scheitelpunkt wird eine Farbe für jeden Scheitelpunkt auf einem Netz berechnet. Um das Bild dann auf dem Bildschirm darzustellen, werden diese Werte pro Scheitelpunkt mithilfe einer baryzentrischen Interpolation in die Innenräume der Dreiecke interpoliert. Ähnliche Strategien werden für die Texturierung und andere häufige Aufgaben verwendet. Abbildung NUMBER zeigt ein Beispiel für dieses einfache Schattierungsmodell.
Nebenbei bemerkt ist das Zusammenspiel von Perspektiventransformationen und Interpolationsstrategien ein spezifisches Problem der Compute
Die baryzentrische Interpolation von Farben auf einer 3D-Oberfläche und die anschließende Projektion dieser Farbe auf die Bildebene ist nicht dasselbe wie die Projektion von Dreiecken auf die Bildebene und die anschließende Interpolation von Farben in das Innere des Dreiecks. Daher müssen Algorithmen in diesem Bereich eine perspektivische Korrektur anwenden, um diesen Fehler zu berücksichtigen.

Bei einer Menge von Punkten in‚R2 ist das Problem der Triangulation alles andere als trivial, und Algorithmen für diese Art von Berechnung lassen sich oft nur schlecht auf Rn übertragen . Daher werden in höheren Dimensionen Next-Neighbor- oder Regressionsstrategien vorzuziehen (siehe Kapitel NUMMER).

Die baryzentrische Interpolation führt zu einer Verallgemeinerung der stückweise linearen Hutfunktionen aus §11.1.3, dargestellt in Abbildung NUMBER. Denken Sie daran, dass unsere interpolatorische Ausgabe vollständig durch die Werte yi an den Eckpunkten der Dreiecke bestimmt wird. Tatsächlich können wir uns f(x) als eine lineare Kombination ÿi yiÿi(x) vorstellen, wobei jedes ÿi(x) die stückweise Schwerpunktfunktion ist, die man erhält, indem man auf xi eine 1 und überall sonst eine 0 setzt, wie in Abbildung NUMMER . Diese dreieckigen Hutfunktionen bilden die Grundlage der „Finite-Elemente-Methode erster Ordnung“, die wir in zukünftigen Kapiteln untersuchen werden; Spezielle Konstruktionen, die Polynome höherer Ordnung verwenden, werden als „Elemente höherer Ordnung“ bezeichnet und können verwendet werden, um die Differenzierbarkeit entlang der Dreieckskanten zu gewährleisten.

Eine alternative und ebenso wichtige Zerlegung des Bereichs von f erfolgt, wenn die Punkte xi auf einem regelmäßigen Gitter in Rn liegen . Die folgenden Beispiele veranschaulichen Situationen, in denen dies der Fall ist:

**Beispiel 11.3** (Bildverarbeitung). Wie in der Einleitung erwähnt, wird ein typisches digitales Foto als m × n-Raster aus roten, grünen und blauen Farbintensitäten dargestellt. Wir können uns diese Werte so vorstellen, als ob sie auf einem Gitter in **Z** × Z leben. Angenommen, wir möchten das Bild um einen Winkel drehen, der jedoch kein Vielfaches von 90ÿ ‚ ist. Dann müssen wir, wie in Abbildung NUMBER dargestellt, Bildwerte an möglicherweise nicht ganzzahligen Positionen nachschlagen, was die Interpolation der Farben auf **R** × R erfordert.

**Beispiel 11.4** (Medizinische Bildgebung). Die typische Ausgabe eines Magnetresonanztomographiegeräts (MRT) ist ein Werteraster von einem × n × p, das die Gewebedichte an verschiedenen Punkten darstellt. Theoretisch ist das typische Modell für diese Funktion f : R3 ÿ R. Wir können die äußere Oberfläche eines bestimmten Organs extrahieren, wie in Abbildung NUMMER gezeigt, indem wir den Ebenensatz {x : f(x) = c} für ein c ermitteln. Um diesen Ebenensatz zu finden, müssen wir f auf das gesamte Voxelgitter erweitern, um genau zu finden, wo es c kreuzt.

Gitterbasierte Interpolationsstrategien wenden typischerweise die eindimensionalen Formeln aus §11.1.3 Dimension für Dimension an. Beispielsweise interpolieren bilineare Interpolationsschemata in R2 jeweils eine Dimension linear, um den Ausgabewert zu erhalten:

**Beispiel 11.5** (Bilineare Interpolation). Angenommen, f nimmt die folgenden Werte an:

- f(0, 0) = 1
- f(0, 1) = ÿ3
- f(1, 0) = 5
- f(1, 1) = ÿ11

und dass dazwischen f durch bilineare Interpolation erhalten wird. Um f( zu finden $\frac{1}{4}, \frac{1}{2}$ ), interpolieren wir zunächst in x1 , um Folgendes zu finden:

$$F\left(\frac{1}{4}, 0\right) = \frac{3}{4} f(0,0) + \frac{1}{4} f(1,0) = \frac{2}{4}$$

$$F\left(\frac{1}{4}, 1\right) = \frac{3}{4} f(0,1) + \frac{1}{4} f(1,1) = \frac{-5}{4}$$

Als nächstes interpolieren wir in x2:

$$F\left(\frac{1}{4}, \frac{1}{2}\right) = \frac{1}{2} f\left(\frac{1}{4}, 0\right) + 2 \cdot f\left(\frac{1}{4}, 3\right) = 2 -$$

Eine wichtige Eigenschaft der bilinearen Interpolation besteht darin, dass wir die gleiche Ausgabe erhalten, wenn wir zuerst in x2 und dann in x1 interpolieren .

Methoden höherer Ordnung wie die bikubische Interpolation und die Lanczos-Interpolation verwenden wiederum mehr Polynomterme, sind jedoch langsamer zu berechnen. Insbesondere bei der Interpolation von Bildern erfordern bikubische Strategien mehr Datenpunkte als das Quadrat der Funktionswerte, die einem Punkt x am nächsten liegen; Dieser zusätzliche Aufwand kann grafische Tools verlangsamen, für die jede Suche im Speicher zusätzliche Rechenzeit verursacht.

# 11.3 Theorie der Interpolation

Bisher war unsere Behandlung der Interpolation ziemlich heuristisch. Während wir uns größtenteils auf unsere Intuition verlassen, was eine „vernünftige“ Interpolation für eine Reihe von Funktionswerten eine akzeptable Strategie ist, können bei verschiedenen Interpolationsmethoden subtile Probleme auftreten, die es zu berücksichtigen gilt.

## 11.3.1 Lineare Algebra von Funktionen

Wir begannen unsere Diskussion damit, verschiedene Interpolationsstrategien als unterschiedliche Grundlagen für die Menge der Funktionen f : **R** ÿ R vorzustellen. Diese Analogie zu Vektorräumen erstreckt sich auf eine vollständige geometrische Funktionstheorie und im Wesentlichen auf frühe Arbeiten auf dem Gebiet der Funktionalanalyse erweitert die Geometrie von Rn auf Mengen von Funktionen. Hier werden wir Funktionen einer Variablen diskutieren, obwohl viele Aspekte der Erweiterung auf allgemeinere Funktionen einfach durchzuführen sind.

So wie wir für Funktionen die Begriffe Spanne und Linearkombination definieren können, können wir für feste a, b ÿ **R** ein inneres Produkt der Funktionen f(x) und g(x) wie folgt definieren:

$$\langle f, g \rangle \ddot{y} \int_A^B f(x)g(x)\,dx.$$

So wie das A-innere Produkt von Vektoren uns bei der Ableitung des Algorithmus für konjugierte Gradienten half und viel mit dem Skalarprodukt gemeinsam hatte, kann das funktionale innere Produkt verwendet werden, um Methoden der linearen Algebra für den Umgang mit Funktionsräumen und das Verständnis ihrer Spanne zu definieren. Wir definieren eine Norm einer Funktion auch als $\|f\| \ddot{y} \sqrt{\langle f, f \rangle}$.

**Beispiel 11.6.** Funktion Inneres Produkt Nehmen wir $p_n(x) = x^n$ das n-te Monom sein. Dann gilt für a = 0 und xb = 1, wir haben:

$$\begin{aligned} \langle p_n, p_m \rangle &= \int_0^1 x^n \cdot x^m\,dx \\ &= \int_0^1 x^{n+m}\,dx \\ &= \frac{1}{n+m+1} \end{aligned}$$

Beachten Sie, dass dies Folgendes zeigt:

$$\frac{pn}{pn}, \frac{Uhr}{Uhr} = \frac{pn,}{pm} = \frac{pnpm\ (2n + 1)(2m}{+ 1)\ n + m + 1}$$

Dieser Wert beträgt ungefähr 1, wenn n ÿ m, aber n = m, was unsere frühere Behauptung untermauert, dass sich die Monome auf [0, 1] erheblich „überlappen“.

Angesichts dieses inneren Produkts können wir den Gram-Schmidt-Algorithmus anwenden, um eine Orthonormalbasis für die Menge der Polynome zu finden. Wenn wir a = ÿ1 und b = 1 nehmen, erhalten wir die Legendre-Polynome, dargestellt in Abbildung NUMBER:

$$P0(x) = 1$$
$$P1(x) = x$$
$$P2(x) = \frac{1}{2}(3x^2 \text{ ÿ } 1)$$
$$P3(x) = (5x^3 \text{ ÿ } 3x)$$
$$P4(x) = \frac{1}{8}(35x^4 \text{ ÿ } 30x^2 + 3)$$
$$\vdots \quad \vdots$$

Diese Polynome haben aufgrund ihrer Orthogonalität viele nützliche Eigenschaften. Angenommen, wir möchten f(x) mit einer Summe ÿi aiPi(x) approximieren. Wenn wir f ÿ ÿi aiPi in der Funktionsnorm minimieren wollen , ist dies ein Problem der kleinsten Quadrate! Aufgrund der Orthogonalität der Legendre-Basis für R[x] zeigt eine einfache Erweiterung unserer Projektionsmethoden:

$$ai = \frac{f\ ,\ Pi}{Pi\ ,\ Pi}$$

Somit kann die Approximation von f mithilfe von Polynomen einfach durch Integration von f gegen die Mitglieder der Legendre-Basis erreicht werden; Im nächsten Kapitel werden wir erfahren, wie dieses Integral näherungsweise ausgeführt werden könnte.

Bei einer gegebenen positiven Funktion w(x) können wir durch Schreiben ein allgemeineres inneres Produkt ·, ·w definieren

$$f\ ,\ gw = \int_A^B w(x)f(x)g(x)\ dx.$$

Wenn wir $w(x) = \text{ÿ}\ _{1\text{ÿ}x^2}^{\frac{1}{\phantom{x}}}$ mit a = ÿ1 und b = 1, dann ergibt die Anwendung von Gram-Schmidt den Tschebyscheff Polynome nehmen:

$$T0(x) = 1$$
$$T1(x) = x$$
$$T2(x) = 2x^2 \text{ ÿ } 1$$
$$T3(x) = 4x^3 \text{ ÿ } 3x$$
$$T4(x) = 8x^4 \text{ ÿ } 8x^2 + 1$$
$$\vdots \quad \vdots$$

Tatsächlich gilt für diese Polynome eine überraschende Identität:

$$T_k(x) = \cos(k \arccos(x)).$$

Diese Formel kann überprüft werden, indem man sie explizit auf T0 und T1 überprüft und dann die Beobachtung induktiv anwendet:

$$\begin{aligned} T_{k+1}(x) &= \cos((k+1)\arccos(x)) \\ &= 2x\cos(k\arccos(x)) - \cos((k-1)\arccos(x)) \text{ durch die} \\ &\qquad \text{Identität } \cos((k+1)\theta) = 2\cos(k\theta)\cos(\theta) - \cos((k \\ &- 1)\theta) = 2xT_k(x) - T_{k-1}(x) \end{aligned}$$

Diese „Drei-Term-Rekurrenz“-Formel bietet auch eine einfache Möglichkeit, die Tschebyscheff-Polynome zu generieren.

Wie in Abbildung NUMBER dargestellt, ist dank der trigonometrischen Formel für die Tschebyscheff-Polynome leicht zu erkennen, dass die Minima und Maxima von Tk zwischen +1 und −1 schwanken .
Darüber hinaus liegen diese Extrema bei cos(iπ/k) (die sogenannten „Tschebyscheff-Punkte“) für i von 0 bis k; Diese schöne Verteilung der Extrema vermeidet Oszillationsphänomene wie die in Abbildung NUMBER gezeigten, wenn eine endliche Anzahl von Polynomtermen zur Approximation einer Funktion verwendet wird. Tatsächlich empfehlen technischere Behandlungen der Polynominterpolation, xi für die Interpolation in der Nähe von Tschebyscheff-Punkten zu platzieren, um eine gleichmäßige Ausgabe zu erhalten.

## 11.3.2 Approximation über stückweise Polynome

Angenommen, wir möchten eine Funktion f(x) mit einem Polynom vom Grad n auf einem Intervall [a, b] approximieren . Definieren Sie Δx als den Abstand b − a. Ein Maß für den Fehler einer Näherung ist die Funktion von Δx, die verschwinden sollte, wenn Δx → 0. Wenn wir dann f mit stückweisen Polynomen approximieren, sagt uns diese Art der Analyse, wie weit wir die Polynome voneinander entfernen sollten, um dies zu erreichen ein gewünschtes Maß an Annäherung.

Angenommen, wir approximieren f mit einer konstanten c = f( $\frac{a+b}{2}$ -Interpolation. ), wie in stückweise konstant Wenn wir | f (x)| < M für alle x ∈ [a, b] annehmen, haben wir:

$$\max |f(x) - c| \le \Delta x \max_{x\in[a,b]} M \text{ nach dem Mittelwertsatz } x\in[a,b]$$

$$\le M\Delta x$$

Daher erwarten wir einen O(Δx) -Fehler, wenn wir die stückweise konstante Interpolation verwenden.

Angenommen, wir approximieren stattdessen f durch stückweise lineare Interpolation, also durch Nehmen

$$\tilde{f}(x) = f(a) + \frac{b - x}{b - a} f(b).$$

Durch den Mittelwertsatz wissen wir f (x) = f *(θ)* für ein *θ* ∈ [a, b]. Das Schreiben der Taylor-Entwicklung über *x* zeigt $f(x) = f(\theta) + f'(\theta)(x - \theta) + O(\Delta x^2)$ auf [a, b], während wir unsere lineare $\tilde{f}(x) = f(\theta) + f'(\theta)(x - \theta)$ umschreiben können. Die Subtraktion dieser beiden Ausdrücke zeigt also, dass ).
Näherungsfehler von f auf $O(\Delta x^{n+1})$ mit einem Polynom vom Es ist nicht schwer, diesen Näherungsfehler vorherzusagen , Grad n abnimmt, ist O(Δx) für stückweise lineare obwohl in der Praxis die quadratische Konvergenz vorliegt Approximationen für die meisten Anwendungen ausreichend.

## 11.4 Probleme

Ideen:

- Horners Methode zur Auswertung von Polynomen
- Rekursive Strategie für Newton-Polynomkoeffizienten.
- Splines, deCasteljeau
- Prüfen Sie die Dreieckflächeninterpolation der baryzentrischen Interpolation

# Kapitel 12

# Numerische Integration und Differenzierung

Im vorherigen Kapitel haben wir Werkzeuge entwickelt, um sinnvolle Werte einer Funktion f(x) anhand einer Stichprobe von Werten (xi , f(xi)) im Bereich von f einzugeben. Offensichtlich ist dieses Interpolationsproblem an sich nützlich, um Funktionen zu vervollständigen, von denen bekannt ist, dass sie stetig oder differenzierbar sind, deren Werte jedoch nur an einer Reihe isolierter Punkte bekannt sind. In einigen Fällen möchten wir jedoch die Eigenschaften dieser Funktionen untersuchen. Insbesondere wenn wir Werkzeuge aus der Analysis auf f anwenden möchten, müssen wir in der Lage sein, seine wir müssen Integrale und Ableitungen zu approximieren.

Tatsächlich gibt es viele Anwendungen, bei denen numerische Integration und Differentiation eine Schlüsselrolle bei der Berechnung spielen. Im einfachsten Fall werden einige bekannte Funktionen als Integrale definiert. Beispielsweise wird die „Fehlerfunktion“, die als kumulative Verteilung einer Gauß- oder Glockenkurve verwendet wird, wie folgt geschrieben:

$$\text{erf}(x) \ddot{y} \frac{2}{\ddot{y} \ddot{y}} \int_0^x \ddot{y} t^2 e \, dt$$

Näherungen von erf(x) werden in vielen statistischen Zusammenhängen benötigt, und ein sinnvoller Ansatz zur Ermittlung dieser Werte besteht darin, das obige Integral numerisch auszuführen.

In anderen Fällen sind numerische Näherungen von Ableitungen und Integralen Teil eines größeren Systems. Beispielsweise werden Methoden, die wir in zukünftigen Kapiteln zur Approximation von Lösungen für Differentialgleichungen entwickeln, stark von diesen Approximationen abhängen. In ähnlicher Weise erscheinen in der rechnerischen Elektrodynamik Integralgleichungen, die nach einer unbekannten Funktion *ÿ* bei gegebenem Kern K und Ausgabe f aufgelöst werden, in der Beziehung:

$$f(x) = \int_{Rn} K(x,y)\ddot{y}(y) \, dy.$$

Diese Arten von Gleichungen müssen gelöst werden, um elektrische und magnetische Felder abzuschätzen. Wenn *ÿ* und K jedoch nicht sehr speziell sind, können wir nicht hoffen, ein solches Integral in geschlossener Form zu finden, und lösen diese Gleichung allein für die unbekannte Funktion ÿ.

In diesem Kapitel werden wir verschiedene Methoden zur numerischen Integration und Differenzierung anhand einer Stichprobe von Funktionswerten entwickeln. Bei diesen Algorithmen handelt es sich in der Regel um relativ einfache Annäherungen. Um sie zu vergleichen, werden wir auch einige Strategien entwickeln, die bewerten, wie gut wir die Leistung verschiedener Methoden erwarten.

## 12.1 Motivation

Es ist nicht schwer, einfache Anwendungen der numerischen Integration und Differentiation zu formulieren, wenn man bedenkt, wie oft die Werkzeuge der Analysis in den Grundformeln und Techniken der Physik, Statistik und anderen Bereichen vorkommen. Hier schlagen wir einige weniger offensichtliche Orte vor, an denen Integration und Differenzierung auftreten.

**Beispiel 12.1** (Stichprobe aus einer Verteilung). Angenommen, wir erhalten eine Wahrscheinlichkeitsverteilung p(t) für das Intervall [0, 1]; Das heißt, wenn wir zufällig Werte gemäß dieser Verteilung abtasten, erwarten wir, dass p(t) proportional zu der Häufigkeit ist, mit der wir einen Wert in der Nähe von t zeichnen. Eine häufige Aufgabe besteht darin, Zufallszahlen zu generieren, die wie p(t) verteilt sind.

Anstatt jedes Mal, wenn wir ein neues p(t) erhalten, eine spezielle Methode zu entwickeln, ist es möglich, eine nützliche Beobachtung zu machen. Wir definieren die kumulative Verteilungsfunktion von p als

$$F(t) = \int_0^t p(x)\,dx.$$

Wenn X dann eine gleichmäßig in [0, 1] verteilte Zufallszahl ist , kann man zeigen, dass F ÿ 1 (X) wie p verteilt ist, wobei F ÿ 1 die Umkehrung von F ist Fÿ1 können wir Zufallszahlen gemäß einer beliebigen Verteilung p generieren; Diese Näherung läuft auf die Integration von p hinaus, was möglicherweise numerisch erfolgen muss, wenn die Integrale nicht in geschlossener Form bekannt sind.

**Beispiel 12.2** (Optimierung). Denken Sie daran, dass die meisten unserer Methoden zum Minimieren und Finden von Wurzeln einer Funktion f davon abhingen, nicht nur Werte f(x) , sondern auch ihren Gradienten ÿ f(x) und sogar Hssian Hf zu haben . Wir haben gesehen, dass Algorithmen wie BFGS und die Broyden-Methode während des Optimierungsprozesses grobe Näherungen der Ableitungen von f aufbauen. Wenn f jedoch hohe Frequenzen aufweist, ist es möglicherweise besser, ÿ f in der Nähe der aktuellen Iteration xk zu approximieren , anstatt Werte von möglicherweise weit entfernten Punkten x für < k zu verwenden.

**Beispiel 12.3** (Rendering). Die Rendering-Gleichung von Raytracing und anderen Algorithmen für hochwertiges Rendering ist ein Integral, das besagt, dass das eine Oberfläche verlassende Licht gleich dem Integral des Lichts ist, das über alle möglichen Einfallsrichtungen in die Oberfläche eindringt, nachdem es reflektiert und gestreut wurde; Im Wesentlichen heißt es, dass die Lichtenergie vor und nach der Wechselwirkung von Licht mit einem Objekt erhalten bleiben muss. Algorithmen zum Rendern müssen dieses Integral annähern, um die Lichtmenge zu berechnen, die von einer Oberfläche emittiert wird, die Licht in einer Szene reflektiert.

**Beispiel 12.4** (Bildverarbeitung). Angenommen, wir stellen uns ein Bild als Funktion zweier Variablen I(x, y) vor. Viele Filter, einschließlich Gaußscher Unschärfen, können als Faltungen betrachtet werden, gegeben durch

$$(I \ast g)(x, y) = \int I(u, v)\,g(x - u, y - v)\,du\,dv.$$

Um beispielsweise ein Bild unscharf zu machen, könnten wir g als Gaußsche Funktion betrachten; in diesem Fall kann man sich (I ÿ g)(x, y) als gewichteten Durchschnitt der Farben von I in der Nähe des Punktes (x, y) vorstellen . In der Praxis handelt es sich bei Bildern um diskrete Pixelgitter, daher muss dieses Integral angenähert werden.

**Beispiel 12.5** (Bayes-Regel). Angenommen, X und Y sind Zufallsvariablen mit kontinuierlichem Wert; Wir können P(X) und P(Y) verwenden , um die Wahrscheinlichkeiten auszudrücken, dass X und Y bestimmte Werte annehmen. Manchmal kann die Kenntnis von X unser Wissen über Y beeinflussen. Wenn X beispielsweise der Blutdruck eines Patienten und Y das Gewicht eines Patienten ist,

Dann kann das Wissen, dass ein Patient ein hohes Gewicht hat, darauf hindeuten, dass er auch einen hohen Blutdruck hat. Wir können daher auch bedingte Wahrscheinlichkeitsverteilungen P(X|Y) (sprich „die Wahrscheinlichkeit von X bei gegebenem Y“) schreiben, die solche Beziehungen ausdrücken.

Eine Grundlage der modernen Wahrscheinlichkeitstheorie besagt, dass P(X|Y) und P(Y|X) wie folgt

$$P(X|Y) = \frac{\text{zusammenhängen: } P(Y|X)P(X)}{P(Y|X)P(X)\,dY}$$

Das Schätzen des Integrals im Nenner kann bei maschinellen Lernalgorithmen, bei denen die Wahrscheinlichkeitsverteilungen komplexe Formen annehmen, ein ernstes Problem darstellen. Daher werden für Algorithmen zur Parameterauswahl, die diesen Wert als Teil einer größeren Optimierungstechnik verwenden, approximative und oft randomisierte Integrationsschemata benötigt.

# 12.2 Quadratur

Wir beginnen mit der Betrachtung des Problems der numerischen Integration oder Quadratur. Dieses Problem – in einer einzelnen Variablen – kann ausgedrückt werden als: „Gegeben eine Stichprobe von n Punkten aus einer Funktion f(x), f(x) Finden Sie eine Annäherung an $\int_A^B$ dx.“ Im vorherigen Abschnitt haben wir mehrere Situationen vorgestellt genau diese Technik.

Es gibt einige Variationen des Problems, die eine leicht unterschiedliche Behandlung oder Anpassung erfordern tion:

- Die Endpunkte a und b können fest sein, oder wir möchten möglicherweise ein Quadraturschema finden, das Integrale für viele ( a, b) -Paare effizient approximieren kann.
- Möglicherweise können wir f(x) an jedem x abfragen, möchten aber das Integral mit relativ wenigen Stichproben annähern, oder wir erhalten möglicherweise eine Liste vorberechneter Paare (xi , f(xi)) und sind auf die Verwendung dieser Daten beschränkt Punkte in unserer Näherung.

Diese Überlegungen sollten im Hinterkopf behalten werden, wenn wir verschiedene Algorithmen für das Quadraturproblem entwerfen.

## 12.2.1 Interpolatorische Quadratur

Viele der im vorherigen Kapitel entwickelten Interpolationsstrategien können mithilfe einer sehr einfachen Beobachtung auf Methoden für die Quadratur erweitert werden. Angenommen, wir schreiben eine Funktion f(x) anhand einer Menge von Basisfunktionen ÿi(x):

$$f(x) = \ddot{y}\ a_i\ddot{y}_i(x).$$

Dann können wir das Integral von f wie folgt finden:

$$\int_A^B f(x)\,dx = \int_A^B \ddot{y}\ a_i\ddot{y}_i(x)\,dx \quad \text{nach Definition von } f$$

$$= \ddot{y}\ a_i \int_A^B \ddot{y}_i(x)\,dx$$

$$= \ddot{y}\ c_i a_i \text{, wenn wir die Definition } c_i \ddot{y} \text{ machen } \int_A^B \ddot{y}_i(x)\,dx$$

Mit anderen Worten: Bei der Integration von f werden einfach die Integrale der Basisfunktionen, aus denen f besteht, linear kombiniert.

**Beispiel 12.6** (Monome). Angenommen, wir schreiben f(x) = ÿk akx k . Wir wissen

$$\int_0^1 x^k \, dx = \frac{1}{k+1},$$

Wenn wir also die obige Ableitung anwenden, wissen wir es

$$\int_0^1 f(x) \, dx = \ddot{y}_k \frac{a_k}{k+1}.$$

Mit anderen Worten, in unserer obigen Notation haben wir ck = definiert $\frac{1}{k+1}$ .

Schemata, bei denen wir eine Funktion durch Interpolation von Abtastwerten und Integration der interpolierten Funktion integrieren, werden als interpolatorische Quadraturregeln bezeichnet. Fast alle Methoden, die wir im Folgenden vorstellen, können auf diese Weise geschrieben werden. Natürlich kann es zu einem Henne-Ei-Problem kommen, wenn das Integral ÿi(x) dx selbst nicht in geschlossener Form bekannt ist. Bestimmte Methoden in finiten Elementen höherer Ordnung lösen dieses Problem, indem sie zusätzliche Rechenzeit in die Erstellung einer qualitativ hochwertigen numerischen Näherung des Integrals eines einzelnen ÿi investieren und dann, da alle ÿs eine ähnliche Form haben, Koordinatenänderungsformeln anwenden Schreiben Sie Integrale für die verbleibenden Basisfunktionen. Dieses kanonische Integral kann mithilfe eines hochpräzisen Schemas offline angenähert und dann wiederverwendet werden.

### 12.2.2 Quadraturregeln

Wenn wir eine Menge von (xi , f(xi)) -Paaren erhalten , schlägt unsere obige Diskussion die folgende Form für eine Quadraturregel zur Approximation des Integrals von f in einem bestimmten Intervall vor:

$$Q[f] \ddot{y} \ddot{y} w_i f(x_i).$$

Unterschiedliche Gewichte ergeben unterschiedliche Näherungen des Integrals, von denen wir hoffen, dass sie immer ähnlicher werden, je dichter wir die xi abtasten.

Tatsächlich legt sogar die klassische Integrationstheorie nahe, dass diese Formel ein vernünftiger Ausgangspunkt ist. Beispielsweise hat das Riemann-Integral, das in vielen Einführungskursen in die Analysis vorgestellt wird, die Form:

$$\int_A^B f(x) = \lim \ddot{y}x_k \ddot{y}0 \ddot{y}_k f(\tilde{x}_k)(x_{k+1} \ddot{y} x_k)$$

Hier wird das Intervall [a, b] in Teile a = x1 < x2 < · · · < xn = b unterteilt , wobei ÿxk = xk+1 ÿ xk und x˜k ein beliebiger Punkt in [xk , xk+1 ist ]. Für eine feste Menge von xk vor der Grenzziehung kann dieses Integral eindeutig in der obigen Q[ f ] -Form geschrieben werden.

Aus dieser Perspektive bestimmen die Entscheidungen von {xi} und {wi} vollständig eine Strategie für die Quadratur. Es gibt viele Möglichkeiten, diese Werte zu bestimmen, wie wir im nächsten Abschnitt sehen werden und wie wir bereits für die interpolatorische Quadratur gesehen haben.

**Beispiel 12.7** (Methode unbestimmter Koeffizienten). Angenommen, wir reparieren x1, . . . , xn und möchten einen sinnvollen Satz begleitender Gewichte wi finden, sodass ÿi wi f(xi) eine geeignete Näherung für das Integral ist

aus . Eine Alternative zur oben aufgeführten Basisfunktionsstrategie ist die Verwendung der Methode unbestimmter Koeffizienten. In dieser Strategie wählen wir n Funktionen f1(x), . . . , fn(x) , deren Integrale bekannt sind, und verlangen, dass unsere Quadraturregel die Integrale dieser Funktionen genau wiederherstellt:

$$\begin{aligned}
&\int_A^B f1(x)\,dx = w1\,f1(x1) + w2\,f1(x2) + \cdots + wn\,f1(xn)\\
&\int_A^B f2(x)\,dx = w1\,f2(x1) + w2\,f2(x2) + \cdots + wn\,f2(xn)\\
&\qquad\vdots \qquad\qquad \vdots\\
&\int_A^B fn(x)\,dx = w1\,fn(x1) + w2\,fn(x2) + \cdots + wn\,fn(xn)
\end{aligned}$$

Dadurch entsteht ein n × n lineares Gleichungssystem für die wi .

Eine häufige Wahl besteht darin, fk(x) = x als Integrale$^{k\ddot{y}1}$ , das heißt, um sicherzustellen, dass das Quadraturschema wiederhergestellt wird von Polynomen niedriger Ordnung zu verwenden. Wir wissen

$$\int_A^B k\,x\,k+1\,dx = \frac{b \quad \ddot{y}\ a^{\,k+1}}{k+1}.$$

Somit erhalten wir das folgende lineare Gleichungssystem für die wi :

$$\begin{aligned}
&w1 + w2 + \cdots + wn = b\ \ddot{y}\ a\\
&b\ x1w1 + x2w2 + \cdots + xnwn = \frac{2\,2\,\ddot{y}\,a}{2}\\
&{}^{2x}1w1 + x\,2^{2}w2 + \cdots + x\quad {}^{2}nwn = \frac{b\,3\ 3\,\ddot{y}\,a}{2}\\
&\qquad\qquad\qquad\qquad\qquad\vdots \qquad \vdots\\
&{}^{n\ddot{y}1\,x}1\,w1 + x\,2^{\,n\ddot{y}1}w2 + \cdots + x\quad {}^{n\ddot{y}1}N\quad wn = \frac{2\,2\,b\,\ddot{y}\,a}{2}
\end{aligned}$$

Dieses System ist genau das in §11.1.1 diskutierte Vandermonde-System.

## 12.2.3 Newton-Cotes-Quadratur

Quadratur regiert, wenn das x regiert. s, die gleichmäßig in [a, b] verteilt sind , werden als Newton-Cotes-Quadratur bezeichnet Wie in Abbildung NUMMER dargestellt, gibt es zwei sinnvolle Möglichkeiten für gleichmäßig verteilte Proben:

- Geschlossene Newton-Cotes-Quadratur platziert xi an a und b. Insbesondere für k ÿ {1, . . . , n} wir nehmen

$$xk\ \ddot{y}\ a + \frac{(k\ \ddot{y}\ 1)(b\ \ddot{y}\ a)}{n\ \ddot{y}\ 1}.$$

• Die offene Newton-Cotes-Quadratur platziert kein xi an a oder b:

$$xk \ddot{y} a + \frac{k(b \ddot{y} a)}{n + 1}.$$

Nachdem diese Wahl getroffen wurde, wenden die Newton-Cotes-Formeln einfach eine Polynominterpolation an, um das Integral von a nach b anzunähern; Der Grad des Polynoms muss offensichtlich n ÿ 1 sein, um die Quadraturregel wohldefiniert zu halten.

Im Allgemeinen werden wir n relativ klein halten. Auf diese Weise vermeiden wir Oszillations- und Rauschphänomene, die beim Anpassen von Polynomen höheren Grades an einen Satz von Datenpunkten auftreten. Wie bei der stückweisen Polynominterpolation verketten wir dann kleine Teile zu zusammengesetzten Regeln, wenn wir über ein großes Intervall [a, b] integrieren.

**Geschlossene Regeln.** Geschlossene Newton-Cotes-Quadraturstrategien erfordern n ÿ 2, um eine Division durch Null zu vermeiden. In der Praxis tauchen häufig zwei Strategien auf:

• Die Trapezregel erhält man für n = 2 (also x1 = a und x2 = b) durch lineare Interpolation von f(a) nach f(b). Es sagt, dass

$$\int_A^B dx \ddot{y} (b \ddot{y} a) \frac{f(a) + f(b)}{2} f(x).$$

• Die Simpson-Regel ergibt sich aus der Annahme von n = 3, also gilt jetzt

$$x1 = a$$
$$x2 = \frac{a + b}{2}$$
$$x3 = b$$

Integriert man die Parabel, die durch diese drei Punkte geht, erhält man

$$\int_A^B f(x) dx \ddot{y} \frac{b \ddot{y} a}{6} f(a) + 4 f \frac{a + b}{2} + f(b).$$

**Offene Regeln.** Offene Regeln für die Quadratur ermöglichen die Möglichkeit von n = 1, was die vereinfachte Mittelpunktsregel ergibt:

$$\int_A^B f(x) dx \ddot{y} (b \ddot{y} a) f \frac{a + b}{2}.$$

Größere Werte von n ergeben Regeln ähnlich der Simpson-Regel und der Trapezregel.

**Zusammengesetzte Integration.** Im Allgemeinen möchten wir möglicherweise f(x) mit mehr als einem, zwei oder drei Werten xi integrieren . Es ist offensichtlich, wie man eine zusammengesetzte Regel aus den obigen Mittelpunkt- oder Trapezregeln konstruieren kann, wie in Abbildung NUMMER dargestellt; Summieren Sie einfach die Werte entlang jedes Intervalls. Wenn wir beispielsweise [a, b] in k Intervalle unterteilen, können wir ÿx ÿ und $\overline{\text{xi ÿ a + iÿx}}^{\text{bÿa k}}$ annehmen. Dann lautet die zusammengesetzte Mittelpunktregel:

$$\int_A^B f(x) dx \ddot{y} f \sum_{i=1}^{k} \ddot{y} \frac{xi+1 + xi}{2} \ddot{y}x$$

Ebenso lautet die zusammengesetzte Trapezregel:

$$\int_A^B f(x)\,dx \approx \sum_{i=1}^{k} \frac{f(x_i) + f(x_{i+1})}{2}\,\Delta x$$

$$= \Delta x \left[\frac{1}{2} f(a) + f(x_1) + f(x_2) + \cdots + f(x_{k-1}) + \frac{1}{2} f(b)\right]$$

durch Trennung der beiden gemittelten Werte von f in der ersten Zeile und Neuindizierung

Eine alternative Behandlung der zusammengesetzten Mittelpunktsregel besteht darin, die interpolatorische Quadraturformel aus §12.2.1 auf die stückweise lineare Interpolation anzuwenden; In ähnlicher Weise beruht die zusammengesetzte Version der Trapezregel auf einer stückweisen linearen Interpolation.

Die zusammengesetzte Version der Simpson-Regel, dargestellt in Abbildung NUMMER, verkettet jeweils drei Punkte, um parabolische Näherungen vorzunehmen. Benachbarte Parabeln treffen sich an geraden xi -Achsen und dürfen keine gemeinsamen Tangenten haben. Diese Summation, die nur existiert, wenn n gerade ist, wird zu:

$$\int_A^B f(x)\,dx \approx \frac{\Delta x}{3}\left[f(a) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) + 4\sum_{i=1}^{n/2} f(x_{2i-1}) + f(b)\right]$$

$$= \frac{\Delta x}{3}\left[f(a) + 4 f(x_1) + 2 f(x_2) + 4 f(x_3) + 2 f(x_4) + \cdots + 4 f(x_{n-1}) + f(b)\right]$$

**Genauigkeit.** Bisher haben wir eine Reihe von Quadraturregeln entwickelt, die denselben Satz von f(xi) auf unterschiedliche Weise effektiv kombinieren, um unterschiedliche Näherungen des Integrals von f zu erhalten. Jede Näherung basiert auf einer anderen technischen Annahme, daher ist unklar, ob eine dieser Regeln besser ist als jede andere. Daher müssen wir Fehlerschätzungen entwickeln, die ihr jeweiliges Verhalten charakterisieren. Wir werden unsere oben genannten Newton-Cotes-Integratoren verwenden, um zu zeigen, wie solche Vergleiche durchgeführt werden könnten, wie in CITE dargestellt.

Betrachten Sie zunächst die Mittelpunktquadraturregel für ein einzelnes Intervall [a, b]. Definieren Sie $c = \frac{1}{2}(a + b)$. Der Die Taylorreihe von f über c ist:

$$f(x) = f(c) + f'(c)(x - c) + \frac{1}{2} f''(c)(x - c)^2 + \frac{1}{6} f'''(c)(x - c)^3 + \frac{1}{24} f^{(4)}(c)(x - c)^4 + \cdots$$

Aufgrund der Symmetrie um c fallen also die ungeraden Terme weg:

$$\int_A^B f(x)\,dx = (b - a) f(c) + \frac{1}{24}(b - a) f''(c) + \frac{1}{1920}(b - a)^5 f^{(4)}(c) + \cdots$$

Beachten Sie, dass der erste Term dieser Summe genau die Schätzung der $\int_A^B f(x)\,dx$ vom Mittelpunkt bereitgestellt Regel ist, sodass diese Regel bis zu $O(\Delta x^3)$ genau ist).

Wenn man nun a und b in unsere Taylor-Reihe für f über c einfügt, zeigt sich:

$$f(a) = f(c) + f'(c)(a - c) + \frac{1}{2} f''(c)(a - c)^2 + \frac{1}{6} f'''(c)(a - c)^3 + \cdots$$

$$f(b) = f(c) + f'(c)(b - c) + \frac{1}{2} f''(c)(b - c)^2 + \frac{1}{6} f'''(c)(b - c)^3 + \cdots$$

Addiert man diese und multipliziert beide Seiten mit $b - a/2$, erhält man:

$$\frac{f(a) + f(b)}{2}(b - a) = f(c)(b - a) + \frac{1}{4} f''(c)\left((a - c)^2 + (b - c)^2\right) + \cdots$$

Der f (c) -Term verschwindet per Definition von c. Beachten Sie, dass die linke Seite die Integralschätzung der Trapezregel ist und die rechte Seite bis auf den kubischen Term mit unserer Taylor-Reihe für $\int_a^b f(x)\,dx$ übereinstimmt. Mit anderen Worten, die Trapezregel lautet auch $O(\ddot{y}x^3)$ in einem einzigen Intervall genau.

Wir machen hier eine Pause, um ein zunächst überraschendes Ergebnis zu bemerken: Die Trapez- und Mittelpunktregeln haben die gleiche Genauigkeitsreihenfolge! Tatsächlich zeigt die Untersuchung des Termes dritter Ordnung, dass die Mittelpunktsregel etwa zweimal genauer ist als die Trapezregel. Dieses Ergebnis scheint kontraintuitiv zu sein, da die Trapezregel eine lineare Näherung verwendet, während die Mittelpunktsregel konstant ist.
Wie in Abbildung NUMBER dargestellt, stellt die Mittelpunktregel jedoch tatsächlich das Integral linearer Funktionen wieder her, was ihren zusätzlichen Grad an Genauigkeit erklärt.

Ein ähnliches Argument gilt für die Suche nach einer Fehlerschätzung für die Simpson-Regel. [ERKLÄRUNG SCHREIBEN $O(\ddot{y}x^5)$. NATION HIER; AUS 205A WENDEN]. Am Ende stellen wir fest, dass die Simpson-Regel einen Fehler wie $O(\ddot{y}x)$ hat

Für diese Art der Analyse gilt ein wichtiger Vorbehalt. Im Allgemeinen gilt der Satz von Taylor nur, wenn ÿx ausreichend klein ist. Wenn die Proben weit voneinander entfernt sind, gelten die Nachteile der Polynominterpolation, und die in Abschnitt NUMBER erläuterten Oszillationsphänomene können bei Integrationsschemata höherer Ordnung zu instabilen Ergebnissen führen.

Kehren wir also zu dem Fall zurück, in dem a und b weit voneinander entfernt sind, unterteilen wir nun [a, b] in Intervalle der Breite ÿx und wenden innerhalb dieser Intervalle eine unserer Quadraturregeln an. Beachten Sie, dass unsere Gesamtzahl an Intervallen $b\ddot{y}a/\ddot{y}x$ beträgt, daher müssen wir in diesem Fall unsere Fehlerschätzungen mit 1/ÿx multiplizieren. Insbesondere gelten folgende Genauigkeitsordnungen:

- Zusammengesetzter Mittelpunkt: $O(\ddot{y}x^2)$
- Zusammengesetztes Trapez: $O(\ddot{y}x^2)$
- Zusammengesetzter Simpson: $O(\ddot{y}x^4)$

## 12.2.4 Gaußsche Quadratur

In einigen Anwendungen können wir die Orte xi auswählen , an denen f abgetastet wird. In diesem Fall können wir nicht nur die Gewichte für die Quadraturregel, sondern auch die Orte xi optimieren , um die höchste Qualität zu erzielen. Diese Beobachtung führt zu anspruchsvollen, aber theoretisch ansprechenden Quadraturregeln.

Die Einzelheiten dieser Technik liegen außerhalb des Rahmens unserer Diskussion, wir bieten jedoch einen einfachen Weg zu ihrer Ableitung. Nehmen wir insbesondere wie in Beispiel 12.7 an, dass wir x1 optimieren möchten . . . , xn und w1, . . . , wn gleichzeitig die Ordnung eines Integrationsschemas erhöhen. Jetzt haben wir 2n statt n Bekannte, sodass wir Gleichheit für 2n Beispiele erzwingen können:

$$\int_A^B f1(x)\,dx = w1\,f1(x1) + w2\,f1(x2) + \cdots + wn\,f1(xn)$$
$$\int_A^B f2(x)\,dx = w1\,f2(x1) + w2\,f2(x2) + \cdots + wn\,f2(xn)$$
$$\vdots \qquad\qquad \vdots$$
$$\int_A^B f2n(x)\,dx = w1\,fn(x1) + w2\,fn(x2) + \cdots + wn\,fn(xn)$$

Jetzt sind sowohl die xi als auch die wi unbekannt, sodass dieses Gleichungssystem nicht mehr linear ist. Wenn wir diese Werte beispielsweise für Polynome im Intervall [ÿ1, 1] optimieren möchten , würden wir dies tun

müssen das folgende Polynomsystem (CITE) lösen:

$$w1 + w2 = \int_{-1}^{1} 1\,dx = 2$$

$$w1x1 + w2x2 = \int_{-1}^{1} x\,dx = 0$$

$$2\,w1x\,1^2 + w2x\,2 = \int_{-1}^{1} 2\,2\,dx = x\,3$$

$$3\,3 + w2x\,w1x\,1 \quad 2 = \int_{-1}^{1} {}_{3x}\,dx = 0$$

Es kann vorkommen, dass Systeme wie dieses mehrere Wurzeln und andere Entartungen haben, die nicht nur von der Wahl der fi (typischerweise Polynome) abhängen, sondern auch vom Intervall, über das wir ein Integral approximieren. Darüber hinaus sind diese Regeln nicht progressiv in dem Sinne, dass die Menge von xi für n Datenpunkte nichts mit denen für k Datenpunkte gemeinsam hat, wenn k = n, sodass es schwierig ist, Daten wiederzuverwenden, um eine bessere Schätzung zu erzielen. Wenn sie jedoch anwendbar sind, hat die Gaußsche Quadratur den höchstmöglichen Grad für festes n. Die Kronrod-Quadraturregeln versuchen, dieses Problem zu vermeiden, indem sie die Quadratur mit 2n + 1 Punkten optimieren und gleichzeitig die Gaußschen Punkte wiederverwenden.

## 12.2.5 Adaptive Quadratur

Wie wir bereits gezeigt haben, gibt es bestimmte Funktionen f, deren Integrale besser mit einer gegebenen Quadraturregel angenähert werden können als andere; Beispielsweise integrieren die Mittelpunkt- und Trapezregeln lineare Funktionen mit voller Genauigkeit, während es zu Abtastproblemen und anderen Problemen kommen kann, wenn f schnell oszilliert.

Denken Sie daran, dass die Gaußsche Quadraturregel darauf hindeutet, dass die Platzierung der xi einen Einfluss auf die Qualität eines Quadraturschemas haben kann. Eine Information haben wir allerdings noch nicht genutzt: die Werte f(xi). Schließlich bestimmen diese die Qualität unseres Quadraturschemas.

Vor diesem Hintergrund untersuchen adaptive Quadraturstrategien die aktuelle Schätzung und generieren neue xi , wenn der Integrand komplizierter ist. Strategien zur adaptiven Integration vergleichen oft die Ausgabe mehrerer Quadraturtechniken, z. B. Trapez und Mittelpunkt, mit der Annahme, dass sie dort übereinstimmen, wo die Abtastung von f ausreichend ist (siehe Abbildung NUMBER). Wenn sie mit einer bestimmten Toleranz in einem bestimmten Intervall nicht übereinstimmen, wird ein zusätzlicher Stichprobenpunkt generiert und die Integralschätzungen werden aktualisiert.

WEITERE DETAILS ODER EIN BEISPIEL HINZUFÜGEN; Diskutieren Sie den rekursiven Algorithmus; GÄNSERICH UND GAUTSCHI

**12.2.6 Mehrere Variablen** Oft möchten

wir Funktionen f(x) integrieren , wobei x ÿ Rn ist . Wenn beispielsweise n = 2 ist, könnten wir durch Berechnen über ein Rechteck integrieren

$$\int_{A}^{B}\int_{C}^{D} f(x, y)\,dx\,dy.$$

Allgemeiner ausgedrückt möchten wir, wie in Abbildung NUMBER¡ dargestellt, möglicherweise ein Integral $\int_{\ddot{y}} f(x)\,dx$, finden, bei dem ÿ eine Teilmenge von Rn ist

Ein „Fluch der Dimensionalität“ erschwert die Integration mit zunehmender Dimension exponentiell. Insbesondere nimmt die Anzahl der Abtastwerte von f, die erforderlich sind, um eine vergleichbare Quadraturgenauigkeit für ein Integral in Rk zu erreichen, wie O(n ) zu. Diese Beobachtung k mag entmutigend sein, ist aber einigermaßen vernünftig: Je mehr Eingabedimensionen für f vorhanden sind, desto mehr Stichproben werden benötigt, um sein Verhalten in allen Dimensionen zu verstehen.

Die einfachste Strategie zur Integration in Rk ist das integrierte Integral. Wenn f beispielsweise eine Funktion ist Nehmen wir an, wir möchten f(x, y) dx dy finden, um zwei Variablen zu betrachten $\int_A^B \int_C^D$. Für festes y können wir das innere Integral über x mithilfe einer eindimensionalen Quadraturregel annähern; Anschließend integrieren wir diese Werte über y mithilfe einer anderen Quadraturregel. Offensichtlich verursachen beide Integrationsschemata einen gewissen Fehler, sodass wir möglicherweise xi dichter als in einer Dimension abtasten müssen, um die gewünschte Ausgabequalität zu erzielen.

Alternativ können wir, genau wie wir [a, b] in Intervalle unterteilt haben, ÿ in Dreiecke und Rechtecke in 2D, Polyeder oder Kästchen in 3D usw. unterteilen und in jedem Teil einfache interpolatorische Quadraturregeln verwenden. Eine beliebte Option besteht beispielsweise darin, die Ausgabe der baryzentrischen Interpolation innerhalb von Polyedern zu integrieren, da dieses Integral in geschlossener Form bekannt ist.

Wenn n jedoch hoch ist, ist es nicht praktikabel, die Domäne wie vorgeschlagen zu teilen. In diesem Fall können wir die randomisierte Monte-Carlo-Methode verwenden. In diesem Fall generieren wir einfach k Zufallspunkte xi ÿ ÿ mit beispielsweise gleichmäßiger Wahrscheinlichkeit. Die Mittelung der Werte f(xi) ergibt eine Näherung von f(x) dx, die $\bar{y}$ wie 1/ ÿ k konvergiert – unabhängig von der Dimension von ÿ! Also in großen Dimensionen Die Monte-Carlo-Schätzung ist den oben genannten deterministischen Quadraturmethoden vorzuziehen.

WEITERE DETAILS ZUR MONTE-CARLO-KONVERGENZ UND DER AUSWAHL DER VERTEILUNGEN ÜBER ÿ

## 12.2.7 Konditionierung

Bisher haben wir die Qualität einer Quadraturmethode unter Verwendung der Genauigkeitswerte O(ÿx) betrachtet. $^{k}$); Aufgrund dieser Metrik ist offensichtlich ein Satz Quadraturgewichte mit großem k vorzuziehen.

Ein anderes Maß gleicht jedoch die mithilfe von Taylor-Argumenten ermittelten Genauigkeitsmessungen aus. Erinnern Sie sich insbesondere daran, dass wir unsere Quadraturregel als Q[ f ] ÿ ÿi wi f(xi) geschrieben haben. Angenommen, wir stören f in ein anderes f. Definieren Sie $\hat{f}$ ÿ f ÿ $\max_{x\in[a,b]}$ | f(x) ÿ f(x)|. Dann,

$$\frac{|Q[f] - \hat{Q}[f]|}{\|f - \hat{f}\|} = \frac{|\sum_i w_i (f(x_i) - \hat{f}(x_i))|}{\|f - \hat{f}\|} \le \frac{\sum_i |w_i| \, |f(x_i) - \hat{f}(x_i)|}{\|f - \hat{f}\|}$$

durch die Dreiecksungleichung

da $|f(x_i) - \hat{f}(x_i)| \le \|f - \hat{f}\|$ per Definition.

$\le \|w\|_1$

Somit hängt die Stabilität oder Konditionierung einer Quadraturregel von der Norm des Gewichtssatzes ab w .

Im Allgemeinen lässt sich leicht überprüfen, dass die Konditionierung w schlechter wird, wenn wir die Ordnung der Quadraturgenauigkeit erhöhen, da die wi große negative Werte annehmen; Dies steht im Gegensatz zum vollständig positiven Fall, bei dem die Konditionierung durch b ÿ a begrenzt ist, da ÿi wi = b ÿ a für Polynome in Terpolationsschemata gilt und die meisten Methoden niedriger Ordnung nur positive Koeffizienten haben (CHECK). Diese Tatsache spiegelt die gleiche Intuition wider, dass wir Funktionen nicht mit Polynomen höherer Ordnung interpolieren sollten. Daher bevorzugen wir in der Praxis in der Regel die zusammengesetzte Quadratur gegenüber Methoden höherer Ordnung, die möglicherweise bessere Schätzungen liefern, bei numerischen Störungen jedoch instabil sein können.

## 12.3 Differenzierung

Die numerische Integration ist ein relativ stabiles Problem. , dass der Einfluss eines einzelnen Werts f(x) auf f(x) dx auf Null schrumpft, wenn a und b weit auseinander liegen. Die Approximation der Ableitung einer Funktion f (x) hingegen weist keine solche Stabilitätseigenschaft auf. Aus der Perspektive der Fourier-Analyse kann man zeigen, dass das Integral f(x) im Allgemeinen niedrigere Frequenzen als f hat, während die Differenzierung zur Erzeugung von f die hohen Frequenzen von f verstärkt, was Stichprobenbeschränkungen, Konditionierung und Stabilität für die Approximation von f besonders schwierig macht.

Trotz der schwierigen Umstände sind Approximationen von Ableitungen in der Regel relativ einfach zu berechnen und können abhängig von der jeweiligen Funktion stabil sein. Tatsächlich haben wir bei der Entwicklung der Sekantenregel, der Broyden-Methode usw. einfache Näherungen von Ableitungen und Gradienten verwendet, um Optimierungsroutinen zu leiten.

Hier konzentrieren wir uns auf die Approximation von f für f : **R** ÿ R. Das Finden von Gradienten und Jacobi-Funktionen erfolgt häufig durch Differenzieren in jeweils einer Dimension, wodurch wir uns effektiv auf das eindimensionale Problem reduzieren, das wir hier betrachten.

### 12.3.1 Basisfunktionen differenzieren

Der einfachste Fall der Differenzierung liegt bei Funktionen vor, die mithilfe von Interpolationsroutinen erstellt werden. Wenn wir wie in §12.2.1 f(x) = ÿi aiÿi(x) schreiben können , dann wissen wir es durch Linearität

$$f(x) = \ddot{y}(x).\ a_i\ddot{y}_i$$

Mit anderen Worten: Wir können uns die Funktionen $\ddot{y}_i$ als Basis für Ableitungen von Funktionen vorstellen, die in der ÿi- Basis geschrieben sind!

Ein Beispiel für dieses Verfahren ist in Abbildung NUMBER dargestellt. Dieses Phänomen verbindet häufig verschiedene Interpolationsschemata. Beispielsweise haben stückweise lineare Funktionen stückweise konstante Ableitungen, Polynomfunktionen haben Polynomableitungen niedrigeren Grades und so weiter; Wir werden auf diese Struktur zurückkommen, wenn wir Diskretisierungen partieller Differentialgleichungen betrachten. In diesem Fall ist es jedoch wertvoll zu wissen, dass f mit voller Sicherheit bekannt ist, obwohl seine Ableitungen, wie in Abbildung NUMMER, unerwünschte Diskontinuitäten aufweisen können.

### 12.3.2 Endliche Differenzen

Ein häufigerer Fall ist, dass wir eine Funktion f(x) haben , die wir abfragen können, deren Ableitungen jedoch unbekannt sind. Dies geschieht häufig, wenn f eine komplexe Form annimmt oder wenn ein Benutzer f(x) als Unterprogramm ohne analytische Informationen über seine Struktur bereitstellt.

Die Definition des Derivats legt einen sinnvollen Ansatz nahe:

$$\lim_{h\ddot{y}0} h \quad \frac{f(x+h)\ \ddot{y}\ f(x)\ f\ (x)\ \ddot{y}}{}$$

Wie zu erwarten ist, gilt für ein endliches h > 0 mit kleinem |h| Der Ausdruck im Grenzwert liefert einen möglichen Wert, der f (x) annähert.

Um diese Intuition zu untermauern, können wir Taylor-Reihen verwenden, um zu schreiben:

$$1\ f(x+h) = f(x) + f\ (x)h + f\ (x)h\ 2 \quad - \qquad 2 + \cdots$$

Das Umordnen dieses Ausdrucks zeigt:

$$= \frac{f(x + h) \ddot{y} f(x)}{f (x)} + O(h) \; h$$

Somit weist die folgende Vorwärtsdifferenznäherung von f eine lineare Konvergenz auf:

$$\ddot{y} \; h \quad \frac{f(x + h) \ddot{y} f(x)}{f (x)}$$

In ähnlicher Weise zeigt das Umdrehen des Vorzeichens von h, dass auch Rückwärtsdifferenzen eine lineare Konvergenz aufweisen:

$$\ddot{y} \; h \quad \frac{f(x) \ddot{y} f(x \ddot{y} h)}{f (x)}$$

Mit einem Trick können wir tatsächlich die Konvergenz unserer Näherung verbessern. Von Taylor Satz können wir schreiben:

$$f(x + h) = f(x) + f (x)h + \frac{1}{2} f (x)h^{2} + \frac{1}{6} f (x)h^{3} + \cdots$$

$$f(x \ddot{y} h) = f(x) \ddot{y} f (x)h + \frac{1}{2} f (x)h^{2} - \frac{1}{6} f (x)h^{3} + \cdots$$

$$1 = \ddot{y} f(x + h) \ddot{y} f(x \ddot{y} h) = 2 f (x)h + f (x)h^{3} - \cdots$$

$$= \ddot{y} \quad \frac{f(x + h) \ddot{y} f(x \ddot{y} h) = f}{(x) + O(h^{2})} \; 2h$$

Somit ergibt diese zentrierte Differenz eine Näherung von f (x) mit quadratischer Konvergenz; Dies ist die höchste Konvergenzordnung, die wir mit einer geteilten Differenz erwarten können. Wir können jedoch eine höhere Genauigkeit erreichen, indem wir f an anderen Punkten auswerten, z. B. x + 2h, obwohl diese Näherung in der Praxis nicht oft zugunsten einer einfachen Verringerung von h verwendet wird.

Die Erstellung von Schätzungen für Ableitungen höherer Ordnung kann durch ähnliche Konstruktionen erfolgen. Für Wenn wir beispielsweise die Taylor-Entwicklungen von f(x + h) und f(x ÿ h) addieren, sehen wir

$$f(x + h) + f(x \ddot{y} h) = 2 f(x) + f (x)h \; f(x + h) \ddot{y}^{2} + O(h^{3})$$

$$= \ddot{y} \quad \frac{2 f(x) + f(x \ddot{y} h) = f (x) + O(h \; h \; 2}{2})$$

Um ähnliche Kombinationen für höhere Ableitungen vorherzusagen, besteht ein Trick darin, unseren zweiten zu beachten Die Ableitungsformel kann unterschiedlich faktorisiert werden:

$$\frac{f(x + h) \ddot{y} \; 2 \; f(x) + f(x \ddot{y} h) \; h \; 2}{} = \frac{\frac{f(x+h)\ddot{y}f(x) \; h}{} \quad \frac{f(x)\ddot{y}f(x\ddot{y}h) \; h}{}}{H}$$

Das heißt, unsere Näherung der zweiten Ableitung ist eine „endliche Differenz endlicher Differenzen“.
Eine Möglichkeit, diese Formel zu interpretieren, ist in Abbildung NUMBER dargestellt. Wenn wir die Vorwärtsdifferenznäherung von f zwischen x und x + h berechnen, können wir uns vorstellen, dass diese Steigung bei x + h/2 liegt; In ähnlicher Weise können wir Rückwärtsdifferenzen verwenden, um eine Steigung bei x ÿ h/2 zu platzieren. Durch Ermitteln der Steigung zwischen diesen Werten wird die Näherung auf x zurückgesetzt.

Eine Strategie, die die Konvergenz der oben genannten Näherungen verbessern kann, ist die Richardson-Extrapolation. Als Beispiel für ein allgemeineres Muster nehmen wir an, wir möchten Vorwärtsdifferenzen verwenden, um f anzunähern. Definieren

$$D(h) \ddot{y} h \frac{f(x + h) \ddot{y} f(x)}{\quad}.$$

Offensichtlich nähert sich D(h) f (x) , wenn h ÿ 0. Genauer gesagt wissen wir jedoch aus unserer Diskussion in §12.3.2, dass D(h) die Form annimmt:

$$D(h) = f(x) + \frac{1}{2} f''(x)h + O(h^2)$$

Angenommen, wir kennen D(h) und *D(ÿh)* für 0 < *ÿ* < 1. Wir wissen:

$$D(\ddot{y}h) = f(x) + \frac{1}{2} f''(x)\ddot{y}h + O(h^2)$$

Wir können diese beiden Beziehungen in einer Matrix schreiben:

$$\begin{matrix} 1 & \frac{1}{2}H \\ 1 & \frac{1}{2}\ddot{y}h \end{matrix} \begin{matrix} f(x) \\ f''(x) \end{matrix} = \begin{matrix} D(h) \\ D(\ddot{y}h) \end{matrix} + O(h^2)$$

Oder gleichwertig,

$$\begin{matrix} f(x) \\ f''(x) \end{matrix} = \begin{matrix} \frac{1}{2}H & \ddot{y}1 \\ 1\ 1 & \frac{1}{2}\ddot{y}h \end{matrix} \begin{matrix} D(h) \\ D(\ddot{y}h) \end{matrix} + O(h^2)$$

Das heißt, wir haben eine O(h) -Näherung von f (x) unter Verwendung von D(h) genommen und $^2$) ca daraus eine O(h -Imitation gemacht ! Diese clevere Technik ist eine Methode zur Sequenzbeschleunigung, da sie die Konvergenzordnung der verbessert Näherung D(h). Der gleiche Trick lässt sich allgemeiner auf viele andere Probleme anwenden, indem man eine Näherung D(h) = a + bhn + O(hm ) schreibt, wobei m > n ist, wobei a die Größe ist, die wir schätzen möchten und b ist der nächste Term in der Taylor-Entwicklung. Tatsächlich kann die Richardson-Extrapolation sogar rekursiv angewendet werden, um Näherungen immer höherer Ordnung zu erstellen.

## 12.3.3 Auswahl der Schrittgröße

Im Gegensatz zur Quadratur hat die numerische Differentiation eine merkwürdige Eigenschaft. Es scheint, dass jede Methode, die wir wählen, beliebig genau sein kann, indem wir einfach ein ausreichend kleines h wählen. Diese Beobachtung ist aus der Perspektive interessant, dass wir ohne zusätzliche Rechenzeit qualitativ hochwertigere Näherungen erzielen können. Der Haken ist jedoch, dass wir durch h dividieren und immer mehr ähnliche Werte f(x) und f(x + h) vergleichen müssen; In der Arithmetik mit endlicher Genauigkeit führt das Addieren und/oder Dividieren durch Werte nahe Null zu numerischen Problemen und Instabilitäten. Somit gibt es einen Bereich von h-Werten, die nicht groß genug sind, um einen signifikanten Diskretisierungsfehler hervorzurufen, und nicht klein genug, um numerische Probleme zu verursachen; Abbildung NUMBER zeigt ein Beispiel für die Differenzierung einer einfachen Funktion in der IEEE-Gleitkomma-Arithmetik.

## 12.3.4 Integrierte Größen

Nicht abgedeckt in CS 205A, Herbst 2013.

## 12.4 Probleme

- Gaußsche Quadratur – enthält immer Mittelpunkte, Strategie unter Verwendung orthogonaler Polynome
- Adaptive Quadratur
- Anwendungen der Richardson-Extrapolation an anderer Stelle

# Kapitel 13

# Gewöhnliche Differentialgleichungen

Wir haben das Problem der Interpolation in Kapitel 11 motiviert, indem wir vom Analysieren zum Finden von Funktionen übergegangen sind. Das heißt, bei Problemen wie Interpolation und Regression ist die Unbekannte eine Funktion f und die Aufgabe des Algorithmus besteht darin, fehlende Daten zu ergänzen.

Wir setzen diese Diskussion fort, indem wir ähnliche Probleme beim Ausfüllen von Funktionswerten betrachten. Hier ist unsere Unbekannte weiterhin eine Funktion f, aber anstatt einfach fehlende Werte zu erraten, möchten wir komplexere Designprobleme lösen. Betrachten Sie beispielsweise die folgenden Probleme:

- Finden Sie f, das eine andere Funktion f0 annähert , aber zusätzliche Kriterien erfüllt (Glattheit, Kontinuität, Niederfrequenz usw.).
- Simulieren Sie eine dynamische oder physikalische Beziehung als f(t), wobei t die Zeit ist.
- Finden Sie f mit ähnlichen Werten wie f0 , aber bestimmten Eigenschaften, die mit einer anderen Funktion gemeinsam sind g0.

In jedem dieser Fälle ist unsere Unbekannte eine Funktion f , aber unser Erfolgskriterium ist komplexer als „passt zu einem bestimmten Satz von Datenpunkten“.

Die Theorien der gewöhnlichen Differentialgleichungen (ODEs) und der partiellen Differentialgleichungen (PDEs) untersuchen den Fall, dass wir eine Funktion f(x) basierend auf Informationen über oder Beziehungen zwischen ihren Ableitungen finden möchten. Beachten Sie, dass wir in unserer Diskussion der Quadratur bereits eine einfache Version dieses Problems gelöst haben: Bei gegebenem f (t) bieten Methoden für die Quadratur Möglichkeiten, f(t) durch Integration zu approximieren.

In diesem Kapitel betrachten wir den Fall gewöhnlicher Differentialgleichungen und insbesondere Anfangswertprobleme. Hier ist die Unbekannte eine Funktion f(t) : **R** ÿ Rn und wir haben eine Gleichung gegeben, die durch f und seine Ableitungen sowie f(0) erfüllt ist; Unser Ziel ist es, f(t) für t > 0 vorherzusagen. Wir werden mehrere Beispiele für ODEs liefern, die in der Informatikliteratur vorkommen, und dann mit der Beschreibung gängiger Lösungstechniken fortfahren.

Als Hinweis verwenden wir die Notation f, um die Ableitung d f/dt von f : [0, ÿ) ÿ Rn zu bezeichnen . Unser Ziel wird es sein , f(t) gegebene Beziehungen zwischen t, f(t), f (t), f (t) usw. zu finden .

## 13.1 Motivation

ODEs tauchen in nahezu jedem Teil wissenschaftlicher Beispiele auf, und es ist nicht schwer, auf praktische Situationen zu stoßen, die eine Lösung erfordern. Die Grundgesetze der physikalischen Bewegung werden beispielsweise durch eine ODE angegeben:

**Beispiel 13.1** (Newtons zweites Gesetz). Erinnern Sie sich in Fortsetzung von §5.1.2 daran, dass Newtons zweites Bewegungsgesetz besagt, dass F = ma, das heißt, die Gesamtkraft auf ein Objekt ist gleich seiner Masse mal seiner Beschleunigung. Wenn wir n Teilchen gleichzeitig simulieren, können wir uns vorstellen, alle ihre Positionen in einem Vektor x ÿ R3n zusammenzufassen . In ähnlicher Weise können wir eine Funktion F(t,x,x ) ÿ R3n schreiben , die die Zeit, die Positionen der Teilchen und ihre Geschwindigkeiten annimmt und die Gesamtkraft auf jedes Teilchen zurückgibt. Diese Funktion kann Wechselbeziehungen zwischen Partikeln (z. B. Gravitationskräfte oder Federn), externe Effekte wie Windwiderstand (der von x abhängt), externe Kräfte, die sich mit der Zeit t ändern, usw. berücksichtigen.

Um dann die Positionen aller Teilchen als Funktionen der Zeit zu ermitteln, möchten wir die Gleichung x = F(t,x,x )/m lösen. Als Ausgangsbedingung werden uns üblicherweise die Positionen und Geschwindigkeiten aller Teilchen zum Zeitpunkt t = 0 gegeben.

**Beispiel 13.2** (Proteinfaltung). Im kleineren Maßstab sind die Gleichungen, die die Bewegung von Molekülen regeln, auch gewöhnliche Differentialgleichungen. Ein besonders anspruchsvoller Fall ist die Proteinfaltung, bei der die geometrische Struktur eines Proteins durch die Simulation intermolekularer Kräfte über die Zeit vorhergesagt wird. Diese Kräfte nehmen viele oft nichtlineare Formen an, die Forscher in der Computerbiologie weiterhin vor Herausforderungen stellen.

**Beispiel 13.3** (Gradientenabstieg). Angenommen, wir möchten eine Energiefunktion E(x) über alle x minimieren. Wir haben in Kapitel 8 gelernt, dass ÿÿE(x) in die Richtung zeigt, in der E bei einem gegebenen x am stärksten abnimmt, also haben wir eine Liniensuche entlang dieser Richtung von x durchgeführt, um E lokal zu minimieren. Eine alternative Option, die in bestimmten Theorien beliebt ist, besteht darin, eine ODE der Form x = ÿÿE(x) zu lösen; Mit anderen Worten: Stellen Sie sich x als eine Funktion der Zeit x(t) vor , die versucht, E zu verringern, indem sie bergab geht.

Nehmen wir zum Beispiel an, wir möchten Ax =b für das symmetrische positiv definite A auflösen. Wir wissen aus §10.1.1 x dass dies äquivalent zur Minimierung von E(x) ÿ x = $\frac{1}{2}$Ax ÿbx + c. Somit könnten wir versuchen, die ODE zu lösen ÿÿ f(x) =b ÿ Ax ist. Für t ÿ ÿ erwarten wir, dass x(t) das lineare System immer besser erfüllt.

**Beispiel 13.4** (Massensimulation). Angenommen, wir schreiben Videospielsoftware, die eine realistische Simulation virtueller Menschenmengen, Tiere, Raumschiffe und dergleichen erfordert. Eine Strategie zur Erzeugung plausibler Bewegungen, dargestellt in Abbildung NUMMER, ist die Verwendung von Differentialgleichungen. Hierbei wird die Geschwindigkeit eines Mitglieds der Menschenmenge als Funktion seiner Umgebung bestimmt; In Menschenmengen können beispielsweise die Nähe anderer Menschen, die Entfernung zu Hindernissen usw. die Bewegungsrichtung eines bestimmten Agenten beeinflussen. Diese Regeln können einfach sein, aber insgesamt ist ihr Zusammenspiel komplex. Dieser Maschinerie liegen stabile Integratoren für Differentialgleichungen zugrunde, da wir kein merklich unrealistisches oder unphysikalisches Verhalten wünschen.

## 13.2 Theorie der ODEs

Eine vollständige Behandlung der Theorie gewöhnlicher Differentialgleichungen liegt außerhalb des Rahmens unserer Diskussion und wir verweisen den Leser für weitere Einzelheiten auf CITE. Abgesehen davon erwähnen wir hier einige Highlights, die für unsere Entwicklung in zukünftigen Abschnitten relevant sein werden.

### 13.2.1 Grundbegriffe

Das allgemeinste ODE-Anfangswertproblem hat die folgende Form:

$$\text{Finden Sie } f(t) : \mathbf{R} \ddot{y} \; ^{N}$$

$$\mathbf{R}, \text{ das } F[t, f(t), f'(t), f''(t), \text{ erfüllt } \ldots, f^{(k)}(t)] = 0 \text{ Gegeben}$$

$$f(0), f'(0), f''(0), \ldots, f^{(k\ddot{y}1)}(0)$$

Hier ist F eine Beziehung zwischen f und allen seinen Ableitungen; Wir verwenden $f^{()}$, um die -te Ableitung von f zu bezeichnen. Wir können uns ODEs als bestimmende Entwicklung von f über die Zeit t vorstellen; Wir kennen f und seine Ableitungen zum Zeitpunkt Null und möchten es für die Zukunft vorhersagen.

ODEs nehmen selbst in einer einzigen Variablen viele Formen an. Bezeichnen Sie zum Beispiel y = f(t) und nehmen Sie an, dass y ÿ R1 . Dann sind Beispiele für ODEs:

- y = 1 + cos t: Diese ODE kann durch Integration beider Seiten, z. B. mittels Quadratur, gelöst werden Methoden
- y = ay: Diese ODE ist linear in y
- $y = ay + e^{t}$: Diese ODE ist zeit- und ortsabhängig
- y + 3y ÿ y = t: Diese ODE beinhaltet mehrere Ableitungen von y
- $y \sin y = e^{ty}$ : Diese ODE ist in y und t nichtlinear.

Offensichtlich kann es schwierig sein, die allgemeinsten ODEs zu lösen. Wir werden den Großteil unserer Diskussion auf den Fall expliziter ODEs beschränken, bei denen die Ableitung höchster Ordnung isoliert werden kann:

**Definition 13.1** (Explizite ODE). Eine ODE ist explizit, wenn sie in der Form geschrieben werden kann

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \ldots, f^{(k\ddot{y}1)}(t)].$$

Eine explizite Form des zweiten Newtonschen Gesetzes ist beispielsweise $x''(t) = \frac{1}{M} a(t, x(t), x'(t))$.

Überraschenderweise lässt sich bei Verallgemeinerung des Tricks in §5.1.2 tatsächlich jede explizite ODE in eine Gleichung erster Ordnung f (t) = F[t, f(t)] umwandeln, wobei f eine mehrdimensionale Ausgabe hat. Diese Beobachtung impliziert, dass wir bei unserer Behandlung von ODE-Algorithmen nicht mehr als eine Ableitung benötigen. Um diese Beziehung zu sehen, erinnern wir uns einfach daran, dass d 2y/dt2 = d/dt(dy/dt). Somit können wir eine Zwischenvariable z ÿ dy/dt definieren und d 2y/dt2 als dz/dt mit der Einschränkung z = dy/dt verstehen . Allgemeiner gesagt, wenn wir das explizite Problem lösen wollen

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \ldots, f^{(k\ddot{y}1)}(t)],$$

wobei $f : \mathbf{R} \ddot{y} R^{n}$ , dann definieren wir $g(t) : \mathbf{R} \ddot{y} R^{kn}$ unter Verwendung der ODE erster Ordnung:

$$\frac{D}{dt} \begin{pmatrix} g_1(t) \\ g_2(t) \\ \vdots \\ g_{k\ddot{y}1}(t) \\ g_k(t) \end{pmatrix} \ddot{y} \begin{pmatrix} g_2(t) \\ g_3(t) \\ \vdots \\ g_k(t) \\ F[t, g_1(t), g_2(t), \ldots, g_{k\ddot{y}1}(t)] \end{pmatrix}$$

Hier bezeichnen wir gi(t) : **R** ÿ Rn , um n Komponenten von g zu enthalten. Dann erfüllt g1(t) die ursprüngliche ODE. Um dies zu sehen, überprüfen wir einfach, ob unsere obige Gleichung g2(t) = g 1 (t), g3(t) = g 2 (t) = g 1 (t) usw. impliziert. Somit zeigt die Durchführung dieser Ersetzungen, dass die letzte Zeile die ursprüngliche ODE kodiert.

Der obige Trick wird unsere Notation vereinfachen, es sollte jedoch darauf geachtet werden, dass dieser Ansatz die Berechnungen nicht trivialisiert. Insbesondere wird unsere Funktion f(t) in vielen Fällen nur eine einzige Ausgabe haben, die ODE jedoch in mehreren Ableitungen vorliegen. Wir ersetzen diesen Fall durch eine Ableitung und mehrere Ausgaben.

**Beispiel 13.5** (ODE-Erweiterung). Angenommen, wir möchten y = 3y ÿ 2y + y lösen, wobei y(t) : **R** ÿ R. Diese Gleichung entspricht:

$$\frac{D}{dt}\begin{pmatrix} y \\ y \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & ÿ2 & 3 \end{pmatrix}\begin{pmatrix} y \\ y \\ y \end{pmatrix}$$

So wie unser obiger Trick es uns ermöglicht, nur ODEs erster Ordnung zu berücksichtigen, können wir unsere Notation noch stärker auf autonome ODEs beschränken. Diese Gleichungen haben die Form f (t) = F[ f(t)], das heißt, F hängt nicht mehr von t ab. Dazu könnten wir definieren

$$g(t) ÿ \begin{matrix} f(t) \\ g^{-}(t) \end{matrix} .$$

Dann können wir stattdessen die folgende ODE für g lösen:

$$g (t) = \begin{matrix} f (t) \\ g^{-} (t) \end{matrix} = \begin{matrix} F[ f(t), g^{-}(t)] \\ 1 \end{matrix} .$$

Insbesondere gilt g¯(t) = t unter der Annahme, dass g¯(0) = 0 ist.

Es gibt viele Möglichkeiten, das Verhalten von ODEs zu visualisieren, wie in Abbildung NUMBER dargestellt. Wenn beispielsweise die Unbekannte f(t) eine Funktion einer einzelnen Variablen ist, können wir uns F[ f(t)] als die Steigung von f(t) vorstellen, wie in Abbildung NUMMER dargestellt. Alternativ können wir, wenn f(t) eine Ausgabe in , R2 hat , die Abhängigkeit von der Zeit t nicht mehr visualisieren, aber wir können einen Phasenraum zeichnen, der den Tangens von f(t) an jedem (x, y) ÿ R2 zeigt

## 13.2.2 Existenz und Einzigartigkeit

Bevor wir mit der Diskretisierung des Anfangswertproblems fortfahren, sollten wir kurz anerkennen, dass nicht alle Differentialgleichungsprobleme lösbar sind. Darüber hinaus lassen einige Differentialgleichungen mehrere Lösungen zu.

**Beispiel 13.6** (Unlösbare ODE). Betrachten Sie die Gleichung y = 2y/t, wobei y(0) = 0 gegeben ist; Beachten Sie, dass wir nicht durch Null dividieren, da y(0) vorgeschrieben ist. Umschreiben als

$$\frac{1}{y}\frac{dy}{dt} = \frac{2}{t}$$

und die Integration bezüglich t auf beiden Seiten zeigt:

$$\ln |y| = 2 \ln t + c$$

Oder äquivalent: y = Ct2 für ein C ÿ R. Beachten Sie, dass y(0) = 0 in diesem Ausdruck ist, was unseren Anfangsbedingungen widerspricht. Somit hat diese ODE keine Lösung mit den gegebenen Anfangsbedingungen.

**Beispiel 13.7** (Nichteindeutige Lösungen). Betrachten Sie nun dieselbe ODE mit y(0) = 0. Betrachten Sie y(t), gegeben durch y(t) = Ct2 für jedes C ÿ R. Dann ist y (t) = 2Ct. Daher,

$$\frac{2}{\text{Jahre t}} = \frac{2Ct2}{T} = 2Ct = y\,(t),$$

Dies zeigt, dass die ODE unabhängig von C durch diese Funktion gelöst wird. Daher sind Lösungen dieses Problems nicht eindeutig.

Zum Glück gibt es eine umfangreiche Theorie, die das Verhalten und die Stabilität von Lösungen von Differentialgleichungen charakterisiert. Unsere Entwicklung im nächsten Kapitel wird einen stärkeren Satz von Bedingungen haben, die für die Existenz einer Lösung erforderlich sind, aber tatsächlich ist es unter schwachen Bedingungen für f möglich zu zeigen, dass eine ODE f (t) = F[f( t ) ] a hat Lösung. Ein solcher Satz garantiert beispielsweise die lokale Existenz einer Lösung:

**Satz 13.1** (Lokale Existenz und Einzigartigkeit). Angenommen, F ist stetig und Lipschitz, das heißt F[y] ÿ F[x]2 ÿ Ly ÿx2 für ein L. Dann lässt die ODE f (t) = F[ f(t)] genau eine Lösung für alle zu t ÿ 0 unabhängig von den Anfangsbedingungen.

In unserer weiteren Entwicklung gehen wir davon aus, dass die ODE, die wir zu lösen versuchen, die Bedingungen eines solchen Theorems erfüllt; Diese Annahme ist insofern ziemlich realistisch, als es zumindest lokal ein ziemlich degeneriertes Verhalten geben müsste, um solche schwachen Annahmen zu durchbrechen.

## 13.2.3 Modellgleichungen

Eine Möglichkeit, das Verhalten von ODEs zu verstehen, besteht darin, das Verhalten von Lösungen einiger einfacher Modellgleichungen zu untersuchen, die in geschlossener Form gelöst werden können. Diese Gleichungen stellen Linearisierungen praktischerer Gleichungen dar und modellieren somit lokal die Art von Verhalten, die wir erwarten können.

Wir beginnen mit ODEs in einer einzelnen Variablen. Angesichts unserer Vereinfachungen in §13.2.1 wäre die einfachste Gleichung, mit der wir arbeiten könnten, y = F[y], wobei y(t) : **R** ÿ R. Eine lineare Näherung würde Gleichungen vom Typ y = ay + ergeben B. Das Einsetzen von y¯ ÿ y + b/a ergibt: y¯ = y = ay + b = a(y¯ ÿ b/a) + b = ay¯. Somit induziert die Konstante b in unseren Modellgleichungen einfach eine Verschiebung, und für unsere phänomenologische Untersuchung in diesem Abschnitt können wir b = 0 annehmen.

Mit dem obigen Argument können wir das Verhalten von y = F[y] lokal verstehen , indem wir die Linearität untersuchen Gleichung y = ay. Tatsächlich zeigt die Anwendung von Standardargumenten aus der Infinitesimalrechnung dies

$$y(t) = Ceat\ .$$

Offensichtlich gibt es drei Fälle, die in Abbildung NUMMER dargestellt sind:

1. a > 0: In diesem Fall werden die Lösungen immer größer; in der Tat, wenn y(t) und yˆ(t) beide die erfüllen ODE mit leicht unterschiedlichen Startbedingungen, da sie bei t ÿ ÿ divergieren.

2. a = 0: Das System wird in diesem Fall durch Konstanten gelöst; Lösungen mit unterschiedlichen Startpunkten bleiben im gleichen Abstand voneinander.

3. a < 0: Dann nähern sich alle Lösungen der ODE 0, wenn t ÿ ÿ.

Wir sagen, dass die Fälle 2 und 3 stabil sind, in dem Sinne, dass eine leichte Störung von y(0) zu Lösungen führt, die sich mit der Zeit immer weiter annähern; Fall 1 ist instabil, da ein kleiner Fehler bei der Angabe des Eingabeparameters y(0) mit fortschreitender Zeit t verstärkt wird. Instabile ODEs erzeugen schlecht gestellte Rechenprobleme; Ohne sorgfältige Überlegung können wir in diesem Fall nicht erwarten, dass numerische Methoden brauchbare Lösungen liefern, da selbst theoretische Ausgaben sehr empfindlich auf Störungen der Eingabe reagieren.
Andererseits sind stabile Probleme gut gestellt, da kleine Fehler in y(0) mit der Zeit abnehmen.

Wenn wir zu mehreren Dimensionen vordringen, könnten wir die linearisierte Gleichung untersuchen

$$y = Ja.$$

Wie in §5.1.2 erläutert, sind y1, · · · ,yk Eigenvektoren von A mit den Eigenwerten ÿ1, . . . , ÿk und y(0) = dann c1y1 + · · · ckyk , +

$$y(t) = c1e^{ÿ1t}ÿk\ t\ y1 + \cdots + cke^{yk}.$$

Mit anderen Worten, die Eigenwerte von A treten in unserem eindimensionalen Beispiel an die Stelle von a. Aus diesem Ergebnis lässt sich leicht ableiten, dass ein multivariables System genau dann stabil ist, wenn sein Spektralradius kleiner als eins ist.

In Wirklichkeit möchten wir y = F[y] für allgemeine Funktionen F lösen . Unter der Annahme, dass F differenzierbar ist, können wir F[y] ÿ F[y0] + JF(y0)(y ÿ y0) schreiben, was die obige Modellgleichung ergibt nach einer Schicht. Daher erwarten wir für kurze Zeiträume ein ähnliches Verhalten wie die Modellgleichung. Darüber hinaus können die Bedingungen in Satz 13.1 als eine Grenze für das Verhalten von JF angesehen werden, was eine Verbindung zu weniger lokalisierten ODE-Theorien herstellt.

# 13.3 Zeitschrittschemata

Wir beschreiben nun mehrere Methoden zur Lösung der nichtlinearen ODE y = F[y] für potenziell nichtlineare Funktionen F. Im Allgemeinen werden unsere Methoden bei einem gegebenen „Zeitschritt“ h verwendet, um Schätzungen von y(t + h) zu generieren ) gegeben y(t). Durch die Anwendung dieser Methoden werden iterativ Schätzungen von y0 ÿ y(t), y1 ÿ y(t + h), y2 ÿ y(t + 2h), y3 ÿ y(t + 3h) usw. generiert. Beachten Sie, dass der Mechanismus zur Generierung jedes zusätzlichen Schritts derselbe ist wie der erste, da F keine t-Abhängigkeit aufweist, sodass wir größtenteils nur einen einzelnen Schritt dieser Methoden beschreiben müssen. Wir nennen Methoden zum Erzeugen von Näherungen für y(t) Integratoren, was die Tatsache widerspiegelt, dass sie die Ableitungen in der Eingabegleichung herausintegrieren.

Von zentraler Bedeutung für unsere Überlegungen ist der Gedanke der Stabilität. So wie ODEs stabil oder instabil sein können, können auch Diskretisierungen stabil oder instabil sein. Wenn h beispielsweise zu groß ist, häufen sich bei einigen Schemata Fehler exponentiell an; Im Gegensatz dazu sind andere Methoden insofern stabil, als die Lösungen auch dann begrenzt bleiben, wenn h groß ist. Stabilität kann jedoch mit Genauigkeit konkurrieren; Oft sind zeitstabile Schemata schlechte Annäherungen an y(t), auch wenn garantiert ist, dass sie kein wildes Verhalten aufweisen.

## 13.3.1 Vorwärts-Euler

Unsere erste ODE-Strategie ergibt sich aus unserer Konstruktion des Vorwärtsdifferenzierungsschemas in §12.3.2:

$$F[yk] = y(t) = \frac{yk+1\ ÿyk}{} + O(h)\ h$$

Das Auflösen dieser Beziehung für yk+1 zeigt

$$y_{k+1} = y_k + hF[y_k] + O(h^2) \ddot{y}\ y_k + hF[y_k].$$

Daher wendet das Vorwärts-Euler-Schema die Formel rechts an, um yk+1 zu schätzen . Dies ist eine der effizientesten Strategien für die Zeitmessung, da sie einfach F auswertet und ein Vielfaches des Ergebnisses zu yk addiert . Aus diesem Grund nennen wir es eine explizite Methode, das heißt, es gibt eine explizite Formel für yk+1 in Bezug auf yk und F.

Die Analyse der Genauigkeit dieser Methode ist ziemlich einfach. Beachten Sie, dass unsere Näherung ), sodass jeder von yk+1 ist O(h $^2$ Schritt einen quadratischen Fehler induziert. Wir nennen diesen Fehler lokalisierten Kürzungsfehler, da es sich um den Fehler handelt, der durch einen einzelnen Schritt verursacht wird. Das Wort „Trunkierung“ bezieht sich auf die Tatsache, dass wir eine Taylor-Reihe gekürzt haben, um diese Formel zu erhalten. Natürlich kann es sein, dass unser Iterationsk aufgrund angesammelter Kürzungsfehler aus früheren Iterationen bereits ungenau ist. Wenn wir von t0 bis t mit O(1/h) Schritten integrieren, dann sieht unser Gesamtfehler wie folgt aus: O(h); Diese Schätzung stellt einen globalen Kürzungsfehler dar, und daher schreiben wir normalerweise, dass das Vorwärts-Euler-Schema „genau erster Ordnung“ ist.

Die Stabilität dieser Methode erfordert etwas mehr Überlegung. In unserer Diskussion werden wir die Stabilität von Methoden im Fall y = ay mit einer Variablen ausarbeiten , mit der Intuition, dass ähnliche Aussagen auf mehrdimensionale Gleichungen übertragen werden können, indem wir a durch den Spektralradius ersetzen. In diesem Fall wissen wir es

$$y_{k+1} = y_k + ahy_k = (1 + ah)y_k .$$

Mit anderen Worten: yk = (1 + ah) ky0. Somit ist der Integrator stabil, wenn |1 + ah| ÿ 1, da sonst |yk | ÿ ÿ exponentiell. Unter der Annahme, dass a < 0 ist (andernfalls ist das Problem schlecht gestellt), können wir Folgendes vereinfachen:

$$|1 + ah| \ddot{y}\ 1 \ddot{y}\ddot{y}\ \ddot{y}1 \ddot{y}\ 1 + ah \ddot{y}\ 1 \ddot{y}\ddot{y}\ \ddot{y}2 \ddot{y}\ ah \ddot{y}$$

$$0$$

$$\ddot{y}\ddot{y}\ 0 \ddot{y}\ h \ddot{y}\ |a| \qquad \frac{2}{}, \text{ da } a < 0$$

Somit lässt Vorwärts-Euler eine Zeitschrittbeschränkung für die Stabilität zu, die durch unsere letzte Bedingung für h gegeben ist. Mit anderen Worten: Die Ausgabe von Vorwärts-Euler kann explodieren, selbst wenn y = ay stabil ist, wenn h nicht klein genug ist. Abbildung NUMMER zeigt, was passiert, wenn diese Bedingung befolgt oder verletzt wird.
In mehreren Dimensionen können wir diese Einschränkung durch eine analoge ersetzen, indem wir den Spektralradius von A verwenden. Für nichtlineare ODEs gibt diese Formel einen Leitfaden für die Stabilität zumindest lokal in der Zeit; Global muss h möglicherweise angepasst werden, wenn die Jacobi-Funktion von F schlechter konditioniert wird.

## 13.3.2 Rückwärts-Euler

In ähnlicher Weise hätten wir das Rückwärtsdifferenzierungsschema bei yk+1 anwenden können , um einen ODE-Integrator zu entwerfen:

$$F[y_{k+1}] = y\ (t) = h \qquad \frac{y_{k+1} \ddot{y} y_k}{} + O(h)$$

Somit lösen wir das folgende potenziell nichtlineare Gleichungssystem für yk+1 :

$$y_k = y_{k+1} \ddot{y}\ hF[y_{k+1}].$$

Da wir diese Gleichung nach yk+1 lösen müssen , ist der Rückwärts-Euler ein impliziter Integrator.

Diese Methode ist durch einen identischen Beweis erster Ordnung genau wie Forward Euler. Die Stabilität dieser Methode steht jedoch im erheblichen Gegensatz zur Vorwärts-Euler-Methode. Unter erneuter Betrachtung der Modellgleichung y = ay schreiben wir:

$$yk\ yk = yk+1 \ddot{y}\ hayk+1 = \ddot{y}\ yk+1 = 1\ \frac{}{\ddot{y}\ ha}.$$

Parallel zu unserem vorherigen Argument ist der Rückwärts-Euler unter der folgenden Bedingung stabil:

$$\frac{1}{} \ddot{y}\ 1\ \ddot{y}\ddot{y}\ |1 \ddot{y}\ ha|\ \ddot{y}\ 1\ |1 \ddot{y}\ ha|$$

$$\ddot{y}\ddot{y}\ 1 \ddot{y}\ ha \ddot{y}\ \ddot{y}1 \text{ oder } 1 \ddot{y}\ ha \ddot{y}\ 1$$

$$2 \ddot{y}\ddot{y}\ h \ddot{y} \text{ oder } -\frac{h}{A} \ddot{y}\ 0, \text{ für } a < 0$$

Offensichtlich nehmen wir immer h ÿ 0 an, sodass Rückwärts-Euler bedingungslos stabil ist.

Selbst wenn der Rückwärts-Euler stabil ist, ist er natürlich nicht unbedingt genau. Wenn h zu groß ist, geht yk viel zu schnell gegen Null. Bei der Simulation von Stoffen und anderen physikalischen Materialien, die viele Hochfrequenzdetails erfordern, um realistisch zu sein, ist die Rückwärts-Euler-Methode möglicherweise keine effektive Wahl.
Darüber hinaus müssen wir F[·] invertieren, um nach yk+1 aufzulösen .

**Beispiel 13.8** (Rückwärts-Euler). Angenommen, wir möchten y = Ay für A ÿ Rn×n lösen . Um dann yk+1 zu finden, lösen wir das folgende System:

$$yk = yk+1 \ddot{y}\ hAyk+1 = \ddot{y}\ yk+1 = (In\times n \ddot{y}\ hA)^{\ddot{y}1}{}_{yk}.$$

## 13.3.3 Trapezmethode

Angenommen , yk ist zum Zeitpunkt tk bekannt und yk+1 stellt den Wert zum Zeitpunkt tk+1 = tk + h dar. Angenommen, wir kennen auch yk+1/2 auf halbem Weg zwischen diesen beiden Schritten. Dann wissen wir durch unsere Ableitung der zentrierten Differenzierung:

$$yk+1 = yk + hF[yk+1/2\ ] + O(h^{\ 3})$$

Aus unserer Ableitung der Trapezregel:

$$\frac{F[yk+1\ ] + F[yk\ ]}{2} = F[yk+1/2\ ] + O(h^{\ 2})$$

Das Ersetzen dieser Beziehung ergibt unser erstes Integrationsschema zweiter Ordnung, die Trapezmethode zur Integration von ODEs:

$$yk+1 = yk + h\,\frac{F[yk+1\ ] + F[yk\ ]}{2}$$

Wie die Rückwärts-Euler-Methode ist diese Methode implizit, da wir diese Gleichung nach yk+1 lösen müssen .

Wenn wir erneut eine Stabilitätsanalyse für y = ay durchführen , finden wir in diesem Fall Zeitschritte der Trapezmethode

$$yk+1 = yk + \frac{1}{2}\,ha(yk+1 + yk)$$

Mit anderen Worten,

$$yk = \frac{1 + \frac{1}{2}\,ha}{1 \ddot{y}\ \frac{1}{2}\,hak}\ y0.$$

Die Methode ist somit stabil, wenn

$$\frac{ha\ 1 + \frac{1}{2}}{1 \ddot{y}\ \frac{2\ 1}{ha\ 2}} < 1.$$

Es ist leicht zu erkennen, dass diese Ungleichung immer dann gilt, wenn a < 0 und h > 0, was zeigt, dass die Trapezmethode bedingungslos stabil ist.

Trotz ihrer höheren Genauigkeit bei gleichbleibender Stabilität weist die Trapezmethode jedoch einige Nachteile auf, die sie weniger beliebt machen als die Rückwärts-Euler-Methode. Berücksichtigen Sie insbesondere die
Verhältnis

$$R \ddot{y} \quad \frac{yk+1}{Ja} = \frac{1 + \frac{1}{2}Ha}{2\ 1\ 1 \ddot{y} \frac{1}{2}Ha}$$

Wenn a < 0 ist und h groß genug ist, wird dieses Verhältnis schließlich negativ; Tatsächlich gilt für h ÿ ÿ R ÿ ÿ1. Wie in Abbildung NUMBER dargestellt, neigt die trapezförmige Integrationsmethode bei zu großen Zeitschritten dazu, ein unerwünschtes Oszillationsverhalten zu zeigen, das überhaupt nicht dem entspricht, was wir für Lösungen von y = ay erwarten würden .

## 13.3.4 Runge-Kutta-Methoden

Eine Klasse von Integratoren kann durch die folgende Beobachtung abgeleitet werden:

$$yk+1 = yk + \int_{tk}^{tk+ÿt} y\ (t)\ dt \text{ nach dem Fundamentalsatz der Analysis}$$

$$= yk + \int_{Tk}^{tk\ tk+ÿt} F[y(t)]\ dt$$

Natürlich funktioniert die direkte Verwendung dieser Formel nicht zur Formulierung einer Methode zur Zeitschrittmessung, da wir y(t) nicht kennen, aber eine sorgfältige Anwendung unserer Quadraturformeln aus dem vorherigen Kapitel kann praktikable Strategien hervorbringen.

Angenommen, wir wenden zur Integration die Trapezmethode an. Dann finden wir:

$$h\ yk+1 = yk + \frac{}{2}\ (F[yk\ ] + F[yk+1\ ]) + O(h^{3})$$

Dies ist die Formel, die wir für die Trapezmethode in §13.3.3 geschrieben haben.

Wenn wir jedoch nicht implizit nach yk+1 auflösen wollen , müssen wir einen Ausdruck für approxi finden . Herstellung Kumpel F[yk+1 ]. Mit der Euler-Methode wissen wir jedoch, dass yk+1 = yk + hF[yk ] + O(h . Diese $^{2}$ Substitution für yk+1 hat keinen Einfluss auf die Reihenfolge der Approximation des trapezförmigen Zeitschritts oben, sodass wir schreiben können:

$$= yk + 2 \quad \frac{H}{}\ (F[yk\ ] + F[yk + hF[yk\ ]]) + O(h\ yk+1^{3})$$

Ignorieren des O(h) $^{3}$) Begriffe ergeben eine neue Integrationsstrategie, die als Heun-Methode bekannt ist zweiter Ordnung genau und explizit.

Wenn wir das Stabilitätsverhalten der Heun-Methode für y = ay für a < 0 untersuchen, wissen wir:

$$h\,y_{k+1} = y_k + \frac{h}{2}(ay_k + a(y_k + hay_k))$$

$$+ 2 \qquad h\,a(2 + ha)\,y_k = 1$$

$$1 = 1 + ha + h^2 \frac{2^2 a}{2} \qquad \text{Ja}$$

Somit ist die Methode stabil, wenn

$$1 \ddot{y} 1 \ddot{y} 1 + ha + h^2 \frac{2^2 a}{2} \ddot{y} 1$$

$$\ddot{y}\ddot{y}\ddot{y}4 \ddot{y} 2ha + h^2 \ddot{y} 0$$

Die Ungleichung auf der rechten Seite zeigt $h \ddot{y} a < \frac{2}{|a|}$ und die auf der linken Seite gilt immer für $h > 0$ und $|a|$ , 0, daher ist die Stabilitätsbedingung $h \ddot{y}$ Die Heun- $\frac{2}{|a|}$ .

Methode ist ein Beispiel für eine Runge-Kutta-Methode, die durch die Anwendung von Quadraturmethoden auf das obige Integral und das Einsetzen von Euler-Schritten in F[·] abgeleitet wird. Forward Euler ist eine genaue Runge-Kutta-Methode erster Ordnung, und Heuns Methode ist zweiter Ordnung. Eine beliebte Runge-Kutta-Methode vierter Ordnung (abgekürzt „RK4“) ist gegeben durch:

$$= y_k + \frac{h}{6}(k1 + 2k2 + 2k3 + k4)\,y_{k+1}$$

wobei $k1 = F[y_k]$

$$k2 = F\left(y_k + \frac{1}{2}hk1\right)$$

$$k3 = F\left(y_k + \frac{1}{2}hk2\right)$$

$$k4 = F(y_k + hk3)$$

Diese Methode kann durch Anwendung der Simpson-Regel für die Quadratur abgeleitet werden.

Runge-Kutta-Methoden sind beliebt, weil sie explizit und daher leicht auszuwerten sind und gleichzeitig ein hohes Maß an Genauigkeit bieten. Der Preis dieser Genauigkeit besteht jedoch darin, dass F[·] mehrmals ausgewertet werden muss. Darüber hinaus können Runge-Kutta-Strategien auf implizite Methoden erweitert werden, die steife Gleichungen lösen können.

**13.3.5 Exponentielle Integratoren** Eine Klasse

von Integratoren, die eine hohe Genauigkeit erreichen, wenn F[·] annähernd linear ist, besteht darin, unsere Lösung der Modellgleichung explizit zu verwenden. Insbesondere wenn wir die ODE y = Ay unter Verwendung von Eigenvektoren von A (oder einer anderen Methode) lösen würden, könnten wir eine explizite Lösung y(t) finden , wie in §13.2.3 erläutert. Normalerweise schreiben wir $y_{k+1} = e^{Ah} y_k$ , kodiert unsere Potenzierung der Eigenwerte (tatsächlich können wir jetzt eine Matrix e finden, wenn wir schreiben $e^{Ah}$ aus diesem Ausdruck, der die ODE zur Zeit h löst).

$$y = Ay + G[y],$$

Wenn G eine nichtlineare, aber kleine Funktion ist, können wir eine ziemlich hohe Genauigkeit erreichen, indem wir den A-Teil explizit integrieren und dann den nichtlinearen G-Teil separat approximieren. Beispielsweise wendet der Exponentialintegrator erster Ordnung Vorwärts-Euler auf den nichtlinearen G-Term an:

$$Ah\ y_{k+1} = e\ y_k \ddot{y} A^{\ddot{y}1}\ (1 \ddot{y} e\ Ah)G[y_k]$$

Die Analyse, die die Vorteile dieser Methode aufzeigt, ist komplexer als das, was wir geschrieben haben, aber intuitiv ist klar, dass sich diese Methoden besonders gut verhalten, wenn G klein ist.

## 13.4 Mehrwertige Methoden

Die Transformationen in §13.2.1 ermöglichten es uns, die Notation im vorherigen Abschnitt erheblich zu vereinfachen, indem wir alle expliziten ODEs auf die Form y = F[y] reduzierten. Obwohl alle expliziten ODEs auf diese Weise geschrieben werden können, ist nicht klar, ob dies immer der Fall sein sollte.

Insbesondere als wir ODEs k-ter Ordnung auf ODEs erster Ordnung reduzierten, führten wir eine Reihe von Variablen ein, die die ersten bis k ÿ 1 -ten Ableitungen der gewünschten Ausgabe darstellen. Tatsächlich kümmern wir uns in unserer endgültigen Lösung nur um die nullte Ableitung, also um die Funktion selbst, sodass Genauigkeitsreihenfolgen für die temporären Variablen weniger wichtig sind.

Betrachten Sie aus dieser Perspektive die Taylor-Reihe

$$2\ y(t_k + h) = y(t_k) + hy\ (t_k) + y\ (t_k) + O(h^{3})$$

Wenn wir y nur bis O(h $^{2}$), hat dies keinen Einfluss auf unsere Näherung, da y mit h multipliziert wird. Wenn wir y nur bis zu O(h) kennen, hat diese Näherung ebenfalls keinen Einfluss auf die obigen Taylor-Reihenterme, da sie mit h 2/2 multipliziert werden. Daher betrachten wir nun „mehrwertiges“ Meth (k) (t) = F[t,y (t),y (t), . . . ,y (kÿ1) (t)] mit Genauigkeit Funktion y zu integrieren. unterschiedlicher Ordnung für ods, entworfen, um y verschiedene Ableitungen der

Angesichts der Bedeutung von Newtons zweitem Gesetz F = ma beschränken wir uns auf den Fall y = F[t,y,y ]; Für den weniger verbreiteten Fall k-ter Ordnung gibt es viele Erweiterungen. Wir führen einen „Geschwindigkeits“-Vektor v(t) = y (t) und einen „Beschleunigungs“-Vektora ein. Mit unserer bisherigen Reduktion wollen wir das folgende System erster Ordnung lösen:

$$\begin{aligned} y\ (t) &= v(t)\ v\ (t) \\ &= a(t)\ a(t) = \\ &\quad F[t,y(t),v(t)] \end{aligned}$$

Unser Ziel ist es, einen speziell auf dieses System zugeschnittenen Integrator abzuleiten.

### 13.4.1 Newmark-Systeme

Wir beginnen mit der Ableitung der berühmten Klasse der Newmark-Integratoren.1 Bezeichnen Sie yk , vk und ak als Positions-, Geschwindigkeits- und Beschleunigungsvektoren zum Zeitpunkt tk ; Unser Ziel ist es, zur Zeit tk+1 ÿ tk + h vorzurücken .

1Wir verfolgen die Entwicklung unter http://www.stanford.edu/group/frg/course_work/AA242B/CA-AA242B-Ch7.pdf.

Verwenden Sie y(t), v(t) und a(t), um die Funktionen der Zeit zu bezeichnen, vorausgesetzt, wir beginnen bei tk . Dann natürlich wir können schreiben

$$v_{k+1} = v_k + \int_{t_k}^{t_{k+1}} a(t)\,dt$$

Wir können yk+1 auch als Integral mit a(t) schreiben, indem wir ein paar Schritte befolgen:

$$\begin{aligned}
y_{k+1} &= y_k + \int_{t_k}^{t_{k+1}} v(t)\,dt \\
&= y_k + [tv(t)]_{t_k}^{t_{k+1}} - \int_{t_k}^{t_{k+1}} ta(t)\,dt \text{ nach partieller Integration} \\
&= y_k + t_{k+1}v_{k+1} - t_k v_k - \int_{t_k}^{t_{k+1}} ta(t)\,dt \text{ durch Erweiterung des Differenzterms} \\
&= y_k + hv_k + t_{k+1}v_{k+1} - t_{k+1}v_k - \int_{t_k}^{t_{k+1}} ta(t)\,dt \text{ durch Addieren und Subtrahieren von } hv_k \\
&= y_k + hv_k + t_{k+1}(v_{k+1} - v_k) - \int_{t_k}^{t_{k+1}} ta(t)\,dt \text{ nach Faktorisierung} \\
&= y_k + hv_k + t_{k+1}\int_{t_k}^{t_{k+1}} a(t)\,dt - \int_{t_k}^{t_{k+1}} ta(t)\,dt \text{ da } v'(t) = a(t) \\
&= y_k + hv_k + \int_{t_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt
\end{aligned}$$

Angenommen, wir wählen $\bar{y} \in [t_k, t_{k+1}]$. Dann können wir Ausdrücke forak undak+1 unter Verwendung der Taylor-Reihe über $\bar{y}$ schreiben:

$$\begin{aligned}
a_k &= a(\bar{y}) + a'(\bar{y})(t_k - \bar{y}) + O(h^2) \\
a_{k+1} &= a(\bar{y}) + a'(\bar{y})(t_{k+1} - \bar{y}) + O(h^2)
\end{aligned}$$

Wenn wir für jede Konstante $\bar{y} \in \mathbb{R}$ die erste Gleichung mit $1 - \bar{y}$ und die zweite mit $\bar{y}$ skalieren und die Ergebnisse summieren, finden wir:

$$\begin{aligned}
a(\bar{y}) &= (1 - \bar{y})a_k + \bar{y}a_{k+1} + a'(\bar{y})((\bar{y} - 1)(t_k - \bar{y}) - \bar{y}(t_{k+1} - \bar{y})) + O(h^2) \\
&= (1 - \bar{y})a_k + \bar{y}a_{k+1} + a'(\bar{y})(\bar{y} - h\bar{y} - t_k) + O(h^2) \text{ nach Einsetzen von } t_{k+1} = t_k + h
\end{aligned}$$

Am Ende möchten wir von tk bis tk+1 integrieren , um die Geschwindigkeitsänderung zu erhalten. Somit berechnen wir:

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} a(\bar{y})\,d\bar{y} &= (1 - \bar{y})ha_k + \bar{y}ha_{k+1} + \int_{t_k}^{t_{k+1}} a'(\bar{y})(\bar{y} - h\bar{y} - t_k)\,d\bar{y} + O(h^3) \\
&= (1 - \bar{y})ha_k + \bar{y}ha_{k+1} + O(h^2),
\end{aligned}$$

wobei der zweite Schritt gilt, weil der Integrand O(h) ist und das Integrationsintervall die Breite h hat. Mit anderen Worten, wir wissen jetzt:

$$v_{k+1} = v_k + (1 - \bar{y})ha_k + \bar{y}ha_{k+1} + O(h^2)$$

Um eine ähnliche Näherung für yk+1 zu erreichen , können wir schreiben

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt &= \int_{t_k}^{t_{k+1}} (t_{k+1} - \bar{y})((1 - \bar{y})a_k + \bar{y}a_{k+1} + a'(\bar{y})(\bar{y} - h\bar{y} - t_k))\,d\bar{y} + O(h^3) \\
&= \frac{TK\,1}{2}(1 - \bar{y})h^2 a_k + \frac{1}{2}\bar{y}h^2 a_{k+1} + O(h^2) \text{ mit einem ähnlichen Argument.}
\end{aligned}$$

Somit können wir unsere frühere Beziehung verwenden, um zu zeigen:

$$y_{k+1} = y_k + hv_k + \int_{T_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt \text{ von vorher}$$

$$= y_k + hv_k + \left(\frac{1}{2} - \beta\right)h^2 a_k + \beta h^2 a_{k+1} + O(h^2)$$

Hier verwenden wir $\beta$ anstelle von $\beta$ (und absorbieren dabei einen Faktor zwei), da das $\beta$, das wir für die Approximation von $y_{k+1}$ wählen , nicht dasselbe sein muss wie das, das wir für die Approximation von $v_{k+1}$ wählen .

Nach all dieser Integration haben wir die Klasse der Newmark-Schemata mit zwei Eingaben abgeleitet Parameter $\gamma$ und $\beta$, die durch den obigen Beweis eine Genauigkeit erster Ordnung haben:

$$y_{k+1} = y_k + hv_k + \left(\frac{1}{2} - \beta\right)h^2 a_k + \beta h^2 a_{k+1}$$

$$v_{k+1} = v_k + (1 - \gamma)ha_k + \gamma h a_{k+1}$$

$$a_k = F[t_k, y_k, v_k]$$

Unterschiedliche Wahlen von $\gamma$ und $\beta$ führen zu unterschiedlichen Schemata. Betrachten Sie beispielsweise die folgenden Beispiele:

- $\gamma = \beta = 0$ ergibt den Konstantbeschleunigungsintegrator:

$$y_{k+1} = y_k + hv_k + h\,a_k\frac{1}{2}$$

$$v_{k+1} = v_k + ha_k$$

  Dieser Integrator ist explizit und gilt genau dann, wenn die Beschleunigung eine konstante Funktion ist.

- $\gamma = 1/2$, $\beta = 1$ ergibt den konstanten impliziten Beschleunigungsintegrator:

$$y_{k+1} = y_k + hv_k + h\,a_{k+1}\frac{1}{2}$$

$$v_{k+1} = v_k + ha_{k+1}$$

  Hier wird die Geschwindigkeit implizit mithilfe der Rückwärts-Euler-Methode schrittweise erhöht, was eine Genauigkeit erster Ordnung ergibt. Das Update von y kann jedoch geschrieben werden

$$y_{k+1} = y_k + \frac{1}{2}h(v_k + v_{k+1}),$$

  zeigt, dass es lokal durch die Mittelpunktregel aktualisiert wird; Dies ist unser erstes Beispiel für ein Schema, bei dem die Geschwindigkeits- und Positionsaktualisierungen unterschiedliche Genauigkeitsordnungen haben. Dennoch lässt sich zeigen, dass diese Technik in y global genau erster Ordnung ist.

- $\gamma = 1/4$, $\beta = 1/2$ ergibt nach etwas Algebra das folgende Trapezschema zweiter Ordnung:

$$x_{k+1} = x_k + \frac{1}{2}h(v_k + v_{k+1})$$

$$v_{k+1} = v_k + \frac{1}{2}h(a_k + a_{k+1})$$

- $\ddot{y} = 0$, $\ddot{y} = 1/2$ ergibt ein genaues zentrales Differenzierungsschema zweiter Ordnung. Im Kanonischen Form, wir haben

$$x_{k+1} = x_k + h v_k + h \, a_k \frac{1}{2}$$

$$v_{k+1} = v_k + \frac{1}{2} h(a_k + a_{k+1})$$

Die Methode trägt ihren Namen, weil die Vereinfachung der obigen Gleichungen zu der alternativen Form führt:

$$v_{k+1} = \frac{y_{k+2} - \ddot{y} y_k}{2h \, y_{k+1} \ddot{y}}$$

$$\underline{2y_{k+1} + y_k \, a_{k+1} = h^2}$$

Es lässt sich zeigen, dass Newmarks Methoden bedingungslos stabil sind, wenn $4\ddot{y} > 2\ddot{y} > 1$, und dass Genauigkeit zweiter Ordnung genau dann auftritt, wenn $\ddot{y} = 1/2$ (CHECK).

## 13.4.2 Versetztes Raster

Eine andere Möglichkeit, eine Genauigkeit zweiter Ordnung iny zu erreichen, besteht darin, zentrierte Differenzen über die Zeit $t_{k+1/2} \ddot{y} t_k + h/2$ zu verwenden:

$$y_{k+1} = y_k + h v_{k+1/2}$$

Anstatt Taylor-Argumente zu verwenden, um zu versuchen, $v_{k+1/2}$ zu bewegen , können wir Geschwindigkeiten v einfach an halben Punkten auf dem Gitter der Zeitschritte speichern.

Dann können wir ein ähnliches Update verwenden, um die Geschwindigkeiten zu erhöhen:

$$v_{k+3/2} = v_{k+1/2} + h a_{k+1} .$$

Beachten Sie, dass diese Aktualisierung tatsächlich auch für x zweiter Ordnung genau ist, denn wenn wir unsere Ausdrücke für $v_{k+1/2}$ und $v_{k+3/2}$ ersetzen , können wir schreiben:

$$a_{k+1} = \frac{1}{h^2} (y_{k+2} \ddot{y} 2y_{k+1} + y_k)$$

Schließlich genügt für den Beschleunigungsterm eine einfache Näherung, da es sich um einen Term höherer Ordnung handelt:

$$a_{k+1} = F\left[t_{k+1}, x_{k+1}, \frac{1}{2} \dagger (v_{k+1/2} + v_{k+3/2})\right]$$

Dieser Ausdruck kann in den Ausdruck für $v_{k+3/2}$ eingesetzt werden .

Wenn $F[\cdot]$ keine Abhängigkeit von v hat, ist die Methode vollständig explizit:

$$y_{k+1} = y_k + h v_{k+1/2}$$

$$a_{k+1}$$

$$= F[t_{k+1}, y_{k+1}]$$

$$v_{k+3/2} = v_{k+1/2} + h a_{k+1}$$

Aufgrund des gestaffelten Zeitrasters und der Tatsache, dass jeder Mittelpunkt zur Aktualisierung der nächsten Geschwindigkeit oder Position verwendet wird, ist dies als Leapfrog-Integrationsmethode bekannt.

Andernfalls wird die Methode implizit, wenn die Geschwindigkeitsaktualisierung von v abhängt.
Oftmals ist die Abhängigkeit von der Geschwindigkeit symmetrisch; Beispielsweise ändert der Windwiderstand einfach das Vorzeichen, wenn Sie die Bewegungsrichtung umkehren. Diese Eigenschaft kann im impliziten Schritt zur Geschwindigkeitsaktualisierung zu symmetrischen Matrizen führen, wodurch es möglich wird, konjugierte Gradienten und verwandte schnelle iterative Methoden zur Lösung zu verwenden.

### 13.5 Zu erledigen

- Definieren Sie steife ODE
- Geben Sie eine Tabelle mit Zeitschrittmethoden für F[t;y] an.
- Verwenden Sie die y-Notation konsequenter

## 13.6 Probleme

- TVD RK
- Mehrschritt-/Mehrwertmethoden a la Heath
- Verlet-Integration
- Symplektische Integratoren

# Kapitel 14

# Partielle Differentialgleichungen

Unsere Intuition für gewöhnliche Differentialgleichungen beruht im Allgemeinen auf der zeitlichen Entwicklung physikalischer Systeme. Gleichungen wie das zweite Newtonsche Gesetz, das die Bewegung physikalischer Objekte über die Zeit bestimmt, dominieren die Literatur zu solchen Anfangswertproblemen; Weitere Beispiele sind chemische Konzentrationen, die im Laufe der Zeit reagieren, Populationen von Raubtieren und Beutetieren, die von Saison zu Saison interagieren, und so weiter. In jedem Fall ist die Anfangskonfiguration – z. B. die Positionen und Geschwindigkeiten von Teilchen in einem System zum Zeitpunkt Null – bekannt, und die Aufgabe besteht darin, das Verhalten im Laufe der Zeit vorherzusagen.

In diesem Kapitel beschäftigen wir uns jedoch mit der Möglichkeit, Beziehungen zwischen verschiedenen Ableitungen einer Funktion zu koppeln. Es ist nicht schwer, Beispiele zu finden, bei denen diese Kopplung notwendig ist. Wenn Sie beispielsweise Rauch- oder Gasgrößen wie „Druckgradienten“, die Ableitung des Drucks eines Gases im Raum, simulieren, können Sie ermitteln, wie sich das Gas im Laufe der Zeit bewegt. Diese Struktur ist sinnvoll, da Gas auf natürliche Weise von Hochdruckregionen in Tiefdruckregionen diffundiert. Bei der Bildverarbeitung koppeln Ableitungen noch natürlicher, da Messungen an Bildern tendenziell gleichzeitig in x- und y-Richtung erfolgen.

Gleichungen, die Ableitungen von Funktionen miteinander verknüpfen, werden als partielle Differentialgleichungen bezeichnet. Sie sind Gegenstand einer reichhaltigen, aber stark nuancierten Theorie, die einer umfassenderen Behandlung würdig ist. Daher wird es hier unser Ziel sein, Schlüsselideen zusammenzufassen und ausreichend Material zur Lösung von Problemen bereitzustellen, die in der Praxis häufig auftreten.

## 14.1 Motivation

Partielle Differentialgleichungen (PDEs) entstehen, wenn die Unbekannte eine Funktion f : Rn ÿ Rm ist. Wir erhalten eine oder mehrere Beziehungen zwischen den partiellen Ableitungen von f und das Ziel besteht darin, ein f zu finden, das die Kriterien erfüllt. PDEs kommen in fast allen Bereichen der angewandten Mathematik vor, und wir listen nachfolgend nur einige auf.

Abgesehen davon sollten wir vor der Einführung spezifischer PDEs einige Notationen einführen. Insbesondere gibt es einige Kombinationen partieller Ableitungen, die in der Welt der PDEs häufig vorkommen. Wenn f : R3 ÿ **R** und v : R3 ÿ R3 , Dann sollten Sie sich die folgenden Operatoren merken:

$$\text{Steigung: } \ddot{y}\, f\, \ddot{y} \qquad \frac{\ddot{y}}{f\, \ddot{y}x1}, \frac{\ddot{y}\, f}{\ddot{y}x2}, \frac{\ddot{y}\, f}{\ddot{y}x3}$$

$$\text{Divergenz: } \ddot{y} \cdot v \ddot{y} + + \ddot{y}x1 \ddot{y}x2 \ddot{y}v1 \ddot{y}v2 \ddot{y}v3 \ddot{y}x2 \ddot{y}x3 \ddot{y}v3 \ddot{y}v2 \ddot{y}x2 \ddot{y}x3 \ddot{y} 2 \ddot{y} 2 \ddot{y} 2 \text{ fff}$$

$$\text{Locken: } \ddot{y} \times v \ddot{y} \quad \frac{\ddot{y}v1}{\ddot{y}x3} \quad \frac{\ddot{y}v3}{\ddot{y}x1}, \frac{\ddot{y}v2}{\ddot{y}x1} \quad \frac{\ddot{y}v1}{\ddot{y}x2}$$

$$\text{Laplace-Operator: } \ddot{y}2 \text{ f } \ddot{y} + + \ddot{y}x 2 \ddot{y}x 2 \ddot{y}x 2 1$$

**Beispiel 14.1** (Flüssigkeitssimulation). Der Fluss von Flüssigkeiten wie Wasser und Rauch wird durch die Navier-Stokes-Gleichungen bestimmt, ein System von PDEs mit vielen Variablen. Nehmen wir insbesondere an, dass sich eine Flüssigkeit in einem Bereich ÿ ÿ R3 bewegt . Wir definieren die folgenden Variablen, dargestellt in Abbildung NUMBER:

- t ÿ [0, ÿ): Zeit
- v(t) : ÿ ÿ R3 : Die Geschwindigkeit der Flüssigkeit
- ÿ(t) : ÿ ÿ R: Die Dichte der Flüssigkeit
- p(t) : ÿ ÿ R: Der Druck der Flüssigkeit
- f(t) : ÿ ÿ R3 : Äußere Kräfte wie Schwerkraft auf die Flüssigkeit

Wenn die Flüssigkeit die Viskosität μ hat und wir davon ausgehen, dass sie inkompressibel ist, lauten die Navier-Stokes-Gleichungen:

$$\ddot{y} \quad \frac{\ddot{y}v}{\ddot{y}t} \quad +v \cdot \ddot{y}v = \ddot{y}\ddot{y}p + \mu\ddot{y}2 \, v + f$$

Hier ist ÿ2v = $\ddot{y}$ 2v1/ÿx 2 + $\ddot{y}$ 2v2/ÿx 2 + $\ddot{y}$ 2v3/ÿx 2 ; Wir stellen uns den Gradienten ÿ als einen Gradienten im Raum vor und nicht als $\ddot{y}$ f $\ddot{y}$ f ÿx1 , ÿx2 , Zeit, also ÿ f = (, kann $\frac{\ddot{y}\,f}{\ddot{y}x3}$ ). Dieses Gleichungssystem bestimmt die Zeitdynamik der Flüssigkeitsbewegung und tatsächlich konstruiert werden, indem man Newtons zweites Gesetz auf die Verfolgung von „Teilchen“ einer Flüssigkeit anwendet. Seine Aussage beinhaltet jedoch nicht nur Ableitungen in der Zeit $\ddot{y}\,\ddot{y}t$ aber auch Ableitungen im Raum ÿ, was es zu einer PDE macht.

**Beispiel 14.2** (Maxwell-Gleichungen). Maxwells Gleichungen bestimmen die Wechselwirkung von elektrischen Feldern E und magnetischen Feldern B über die Zeit. Wie bei den Navier-Stokes-Gleichungen stellen wir uns Gradient, Divergenz und Curl als partielle Ableitungen im Raum (und nicht in der Zeit t) vor. Dann kann das Maxwell-System (in „starker“ Form) geschrieben werden:

$$\text{Gaußsches Gesetz für elektrische Felder: } \ddot{y} \cdot E = \frac{\ddot{y}}{\ddot{y}0}$$

$$\text{Gaußsches Gesetz für Magnetismus: } \ddot{y} \cdot B = 0$$

$$\text{Faradaysches Gesetz: } \ddot{y} \times E = \ddot{y} \frac{\ddot{y}B}{\ddot{y}t}$$

$$\text{Amperesches Gesetz: } \ddot{y} \times B = \mu 0 \, J + \ddot{y}0 \, \ddot{y}t \quad \frac{\ddot{y}E}{}$$

Hier sind ÿ0 und μ0 physikalische Konstanten und J kodiert die Dichte des elektrischen Stroms. Genau wie die Navier-Stokes-Gleichungen verknüpften die Maxwell-Gleichungen die Ableitungen physikalischer Größen in der Zeit t mit ihren Ableitungen im Raum (gegeben durch Curl- und Divergenzterme).

**Beispiel 14.3** (Laplace-Gleichung). Angenommen, ÿ ist ein Bereich in R2 mit der Grenze ÿÿ und wir erhalten eine Funktion g: ÿÿ ÿ R, dargestellt in Abbildung NUMBER. Möglicherweise möchten wir g in das Innere von ÿ interpolieren. Wenn ÿ jedoch eine unregelmäßige Form hat, können unsere Interpolationsstrategien aus Kapitel 11 scheitern.

Angenommen, wir definieren f(x) : ÿ ÿ **R** als Interpolationsfunktion. Dann besteht eine von unserem Ansatz zur Methode der kleinsten Quadrate inspirierte Strategie darin, ein Energiefunktional zu definieren:

$$E[\,f\,] = \int_{\ddot{y}} \ddot{y}\, f(x) \;\; {}^{2}_{2}\, dx$$

Das heißt, E[ f ] misst die „Gesamtableitung“ von f, gemessen durch Nehmen der Norm seines Gradienten und Integrieren dieser Größe über ganz ÿ. Stark schwankende Funktionen f werden hohe Werte von E[ f ] haben , da die Steigung ÿ f an vielen Stellen groß sein wird; glatte und niederfrequente Funktionen f haben dagegen ein kleines E[ f ] , da ihre Steigung überall klein ist.1 Dann könnten wir verlangen, dass f g interpoliert und dabei im Inneren von ÿ möglichst glatt ist folgende Optimierung:

$$\begin{aligned} &\text{minimiere } f \; E[\,f\,] \\ &\text{mit } f(x) = g(x) \;\ddot{y}x \;\ddot{y}\; \ddot{y}\ddot{y} \end{aligned}$$

Dieser Aufbau sieht aus wie Optimierungen, die wir in früheren Beispielen gelöst haben, aber jetzt ist unsere Unbekannte eine Funktion f und kein Punkt in Rn !

Wenn f E minimiert, dann ist E[ f + h] ÿ E[ f ] für alle Funktionen h(x). Diese Aussage gilt auch für kleine Störungen E[ f + ÿh] , wenn *ÿ* ÿ 0. Wenn wir durch *ÿ* dividieren und den Grenzwert als *ÿ* ÿ 0 annehmen, müssen wir $\frac{d}{d\ddot{y}} E[\,f + \ddot{y}h]|_{\ddot{y}=0} = 0$ haben; Dies ist so, als würde man Richtungsableitungen einer Funktion gleich Null setzen, um deren Minima zu ermitteln. Wir können Folgendes vereinfachen:

$$\begin{aligned} E[\,f + \ddot{y}h] &= \int_{\ddot{y}} \ddot{y}\, f(x) + \ddot{y}\ddot{y}h(x) \;\; {}^{2}_{2}\, dx \\ &= \int_{\ddot{y}} (\ddot{y}\, f(x) \;\; {}^{2}_{2} + 2\ddot{y}\ddot{y}\, f(x) \cdot \ddot{y}h(x) + \ddot{y} \;\; {}^{2} \;\; \ddot{y}h(x) \;\; {}^{2}_{2})\, dx \end{aligned}$$

Differenzierende Shows:

$$\begin{aligned} \frac{D}{} E[\,f + \ddot{y}h] &= d\ddot{y} \int_{\ddot{y}} (2\ddot{y}\, f(x) \cdot \ddot{y}h(x) + 2\ddot{y}\ddot{y}h(x) \;\; {}^{2}_{2})\, dx \\ \frac{D}{d\ddot{y}} E[\,f + \ddot{y}h]|_{\ddot{y}=0} &= 2 \int_{\ddot{y}} [\ddot{y}\, f(x) \cdot \ddot{y}h(x)]\, dx \end{aligned}$$

Diese Ableitung muss für alle h gleich Null sein, daher können wir insbesondere h(x) = 0 für alle x ÿ ÿÿ wählen. Wenn wir dann die partielle Integration anwenden, erhalten wir:

$$\frac{D}{} E[\,f + \ddot{y}h]|_{\ddot{y}=0} = \ddot{y}2\, d\ddot{y} \int_{\ddot{y}} h(x)\ddot{y}2\, f(x)\, dx$$

Dieser Ausdruck muss für alle (alle!) Störungen h gleich Null sein, also müssen wir ÿ2 f(x) = 0 für alle x ÿ *ÿ\ÿÿ* haben (ein formaler Beweis liegt außerhalb des Rahmens unserer Diskussion). Das heißt, das obige Interpolationsproblem

1Die hier verwendete Notation E[·] steht nicht für „Erwartung“, wie es in der Wahrscheinlichkeitstheorie der Fall wäre, sondern ist einfach so ein „Energie“-Funktional; Es ist die Standardnotation in Bereichen der Funktionsanalyse.

kann mit der folgenden PDE gelöst werden: ÿ2

$$f(x) = 0 \; f(x) =$$

$$g(x) \; ÿx \; ÿ \; ÿÿ$$

Diese Gleichung ist als Laplace-Gleichung bekannt und kann mit dünn besetzten positiv definiten linearen Methoden gelöst werden, wie wir sie in Kapitel 10 behandelt haben. Wie wir gesehen haben, kann sie auf Interpolationsprobleme für unregelmäßige Domänen ÿ angewendet werden; Darüber hinaus kann E[ f ] erweitert werden, um andere Eigenschaften von f zu messen, z. B. wie gut f eine verrauschte Funktion f0 annähert, um verwandte PDEs durch Parallelisierung des obigen Arguments abzuleiten.

**Beispiel 14.4** (Eikonal-Gleichung). Angenommen, ÿ ÿ Rn sei ein geschlossener Raumbereich. Dann könnten wir d(x) als eine Funktion betrachten, die den Abstand von einem Punkt x0 zu x vollständig innerhalb von ÿ misst. Wenn ÿ konvex ist, können wir d in geschlossener Form schreiben:

$$d(x) = x \; ÿx02.$$

Wie in Abbildung NUMBER dargestellt, wird es jedoch schwieriger, Abstände zu berechnen, wenn ÿ nicht konvex ist oder ein komplizierterer Bereich wie eine Oberfläche ist. In diesem Fall erfüllen die Distanzfunktionen d die lokalisierte Bedingung, die als Eikonalgleichung bekannt ist:

$$ÿd2 = 1.$$

Wenn wir es berechnen können, kann d für Aufgaben wie die Planung von Roboterpfaden verwendet werden, indem die Distanz, die sie zurücklegen müssen, minimiert wird, mit der Einschränkung, dass sie sich nur in ÿ bewegen können.

Spezielle Algorithmen, sogenannte Fast-Marching-Methoden, werden verwendet, um Schätzungen von d bei gegebenem x0 und ÿ durch Integration der Eikonalgleichung zu finden. Diese Gleichung ist in der Ableitung ÿd nichtlinear, daher sind die Integrationsmethoden für diese Gleichung etwas spezialisiert und der Nachweis ihrer Wirksamkeit ist komplex. Interessanterweise, aber nicht überraschend, haben viele Algorithmen zur Lösung der Eikonalgleichung eine ähnliche Struktur wie Dijkstras Algorithmus zur Berechnung kürzester Pfade entlang von Graphen.

**Beispiel 14.5** (Harmonische Analyse). Verschiedene Objekte reagieren unterschiedlich auf Vibrationen, und diese Reaktionen hängen zum großen Teil von der Geometrie der Objekte ab. Beispielsweise können Celli und Klaviere die gleiche Note spielen, aber selbst ein unerfahrener Musiker kann die von ihnen erzeugten Klänge leicht unterscheiden. Aus mathematischer Sicht können wir ÿ ÿ R3 als eine Form annehmen, die entweder als Oberfläche oder als Volumen dargestellt wird.
Wenn wir die Kanten der Form einklemmen, ist ihr Frequenzspektrum durch Lösungen des folgenden Differentialeigenwertproblems gegeben:

$$ÿ2 \; f = ÿ \; f \; f(x)$$

$$= 0 \; ÿx \; ÿ \; ÿÿ, \text{ wobei } ÿ2$$

der Laplace-Operator von ÿ und ÿÿ der Rand von ÿ ist. Abbildung NUMBER zeigt Beispiele dieser Funktionen auf verschiedenen Domänen ÿ.

Es lässt sich leicht überprüfen, dass sin kx dieses Problem löst, wenn ÿ das Intervall [0, 2ÿ] für k ÿ Z ist. Insbesondere ist der Laplace-Operator in einer Dimension $ÿ 2/ÿx 2$ und so können wir überprüfen:

$$\frac{ÿ 2}{ÿx} \; ÿ \sin kx = k \; \cos kx \; ÿx \; 2$$

$$2 = ÿk \sin kx$$

$$\sin k \cdot 0 = 0$$

$$\sin k \cdot 2ÿ = 0$$

Somit sind die Eigenfunktionen sin kx mit Eigenwerten ÿk 2 .

## 14.2 Grundlegende Definitionen

Unter Verwendung der CITE-Notation gehen wir davon aus, dass unsere Unbekannte eine Funktion f : Rn ÿ R ist. Für Gleichungen mit bis zu drei Variablen verwenden wir die Indexnotation, um partielle Ableitungen zu bezeichnen:

$$f_x \ddot{y} \quad \frac{\ddot{y} f}{\ddot{y} x},$$

$$f_y \ddot{y} \quad \frac{\ddot{y} f}{\ddot{y} y \ddot{y}},$$

$$f_{xy} \ddot{y} \quad \frac{2 f}{\ddot{y} x \ddot{y} y},$$

und so weiter.

Partielle Ableitungen werden im Folgenden üblicherweise als Beziehungen zwischen zwei oder mehr als Ableitungen von f angegeben:

- Linear, homogen: fxx + fxy ÿ fy = 0 •
- Linear: fxx ÿ y fyy + f = xy2
- Nichtlinear: $f_{xx}^{2} = f_{xy}$

Im Allgemeinen möchten wir f : ÿ ÿ **R** für ein ÿ ÿ Rn finden . So wie ODEs als Anfangswertprobleme angegeben wurden, werden wir die meisten PDEs als Randwertprobleme bezeichnen. Das heißt, unsere Aufgabe besteht darin, f im Inneren von ÿ mit gegebenen Werten an seinem Rand auszufüllen. Tatsächlich können wir uns das ODE-Anfangswertproblem folgendermaßen vorstellen: Die Domäne ist ÿ = [0, ÿ) mit der Grenze ÿÿ = {0}, wo wir Eingabedaten bereitstellen. Abbildung NUMBER veranschaulicht komplexere Beispiele. Randbedingungen für diese Probleme können viele Formen annehmen:

- Dirichlet-Bedingungen geben einfach den Wert von f(x) auf ÿÿ an
- Neumann-Bedingungen geben die Ableitungen von f(x) auf ÿÿ an
- Gemischte oder Robin-Bedingungen kombinieren diese beiden

## 14.3 Modellgleichungen

Erinnern Sie sich an das vorherige Kapitel, dass wir viele Eigenschaften von ODEs verstehen konnten, indem wir eine Modellgleichung y = ay untersuchten. Wir können versuchen, einen ähnlichen Ansatz für PDEs zu verfolgen, werden jedoch feststellen, dass die Geschichte nuancierter ist, wenn Derivate miteinander verknüpft werden.

Wie bei der Modellgleichung für ODEs werden wir den linearen Fall mit einer Variablen untersuchen. Wir beschränken uns auch auf Systeme zweiter Ordnung, also Systeme, die höchstens die zweite Ableitung von u enthalten; Die Modell-ODE war erster Ordnung, aber hier benötigen wir mindestens zwei Ordnungen, um zu untersuchen, wie Derivate auf nichttriviale Weise interagieren.

Eine lineare PDE zweiter Ordnung hat die folgende allgemeine Form:

$$= 0 \quad \sum_{ij} a_{ij} \frac{\ddot{y} f}{\ddot{y} x_i \ddot{y} x_j} \ddot{y} x_i + \ddot{y} b_i + c$$

Formal könnten wir den „Gradientenoperator“ wie folgt definieren:

$$\ddot{y}\ \ddot{y} \qquad \frac{}{\ddot{y}\,\ddot{y}x1}, \frac{}{\ddot{y}\,\ddot{y}x2}, \ldots, \frac{}{\ddot{y}\,\ddot{y}xn} \quad .$$

Sie sollten überprüfen, ob dieser Operator eine sinnvolle Schreibweise hat, da Ausdrücke wie ÿ f , ÿ · v und ÿ ×v alle die richtigen Ausdrücke liefern. In dieser Notation kann man sich die PDE als eine Matrixform vorstellen:

$$(\ddot{y}A\ddot{y} + \ddot{y} \cdot b + c)f = 0.$$

Diese Form hat viel mit unserer Untersuchung quadratischer Formen in konjugierten Gradienten gemeinsam, und tatsächlich charakterisieren wir PDEs normalerweise durch die Struktur von A:

- Wenn A positiv oder negativ definit ist, ist das System elliptisch.
- Wenn A positiv oder negativ semidefinit ist, ist das System parabolisch.
- Wenn A nur einen Eigenwert mit unterschiedlichem Vorzeichen vom Rest hat, ist das System hyperbolisch.
- Wenn A keines der Kriterien erfüllt, ist das System ultrahyperbolisch.

Diese Kriterien sind ungefähr in der Reihenfolge des Schwierigkeitsgrads der Lösung jedes Gleichungstyps aufgeführt. Im Folgenden betrachten wir die ersten drei Fälle und geben Beispiele für entsprechendes Verhalten; Ultrahyperbolische Gleichungen kommen in der Praxis nicht so häufig vor und erfordern hochspezialisierte Techniken zu ihrer Lösung.

TODO: Reduktion auf kanonische Form über Eigenmaterial von A (nicht in 205A)

### 14.3.1 Elliptische PDEs

So wie positiv definite Matrizen spezielle Algorithmen wie die Cholesky-Zerlegung und konjugierte Gradienten ermöglichen, die ihre Inversion vereinfachen, haben elliptische PDEs eine besonders starke Struktur, die zu effektiven Lösungstechniken führt.

Die elliptische Modell-PDE ist die Laplace-Gleichung, gegeben durch ÿ2 f = g für eine gegebene Funktion g as im Beispiel 14.3. Beispielsweise ergibt sich für zwei Variablen die Laplace-Gleichung

$$f_{xx} + f_{yy} = g.$$

Abbildung NUMBER zeigt einige Lösungen der Laplace-Gleichung für verschiedene Auswahlmöglichkeiten von u und f in einem zweidimensionalen Bereich.

Elliptische Gleichungen haben viele wichtige Eigenschaften. Von besonderer theoretischer und praktischer Bedeutung ist die Idee der elliptischen Regularität, dass Lösungen elliptischer PDGs automatisch glatte Funktionen in C ÿ(ÿ) sind. Diese Eigenschaft ist nicht sofort offensichtlich: Eine PDE zweiter Ordnung in f erfordert nur, dass f zweimal differenzierbar ist, um einen Sinn zu ergeben, aber tatsächlich sind sie unter schwachen Bedingungen automatisch unendlich differenzierbar. Diese Eigenschaft unterstützt die physikalische Intuition, dass elliptische Gleichungen stabile physikalische Gleichgewichte darstellen, wie die Ruhelage einer ausgestreckten Gummiplatte.
Auch elliptische Gleichungen zweiter Ordnung in der obigen Form lassen im Gegensatz zu PDEs in einigen anderen Formen garantiert Lösungen zu.

**Beispiel 14.6** (Poisson in einer Variablen). Die Laplace-Gleichung mit g = 0 erhält den speziellen Namen Poisson-Gleichung. In einer Variablen kann f (x) = 0 geschrieben werden, was trivialerweise durch f(x) = ÿx + ÿ gelöst wird. Diese Gleichung reicht aus, um mögliche Randbedingungen auf [a, b] zu untersuchen:

- Dirichlet-Bedingungen für diese Gleichung geben einfach f(a) und f(b) an; Es gibt offensichtlich eine eindeutige Linie, die durch (a, f(a)) und (b, f(b)) geht und die Lösung der Gleichung liefert.

- Neumann-Bedingungen würden f (a) und f (b) spezifizieren. Aber f (a) = f (b) = *ÿ* für f(x) = ÿx + ÿ. Auf diese Weise können Grenzwerte für Neumann-Probleme Kompatibilitätsbedingungen unterliegen, die für eine Lösung erforderlich sind. Darüber hinaus hat die Wahl von *ÿ* keinen Einfluss auf die Randbedingungen, sodass die Lösung nicht eindeutig ist, wenn diese erfüllt sind.

### 14.3.2 Parabolische PDEs

In Anlehnung an die Struktur der linearen Algebra sind positiv semidefinite Gleichungssysteme nur geringfügig schwieriger zu handhaben als positiv definite. Insbesondere positive semidefinite Matrizen lassen einen Nullraum zu, mit dem sorgfältig umgegangen werden muss, aber in den übrigen Richtungen verhalten sich die Matrizen genauso wie im definiten Fall.

Die Wärmegleichung ist die modellparabolische PDE. Angenommen, f(0; x, y) ist eine Wärmeverteilung in einem Bereich ÿ ÿ R2 zum Zeitpunkt t = 0. Dann bestimmt die Wärmegleichung, wie die Wärme über die Zeit t als Funktion f(t; x, y) diffundiert. :

$$\frac{\ddot{y}\, f}{} = \ddot{y}\ddot{y}2\ f\ ,\ \ddot{y}t$$

wobei *ÿ* > 0 und wir uns ÿ2 wieder als den Laplace-Operator in den Raumvariablen x und y vorstellen, also ÿ2 = *ÿ 2/ÿx* 2 + *ÿ 2/ÿy* 2 . Diese Gleichung muss parabolisch sein, da vor fxx und fyy derselbe Koeffizient *ÿ* steht , ftt jedoch nicht in die Gleichung einfließt.

Abbildung NUMBER veranschaulicht eine phänomenologische Interpretation der Wärmegleichung. Wir können uns ÿ2 f als Maß für die Konvexität von f vorstellen, wie in Abbildung NUMBER(a). Somit nimmt die Wärmegleichung u mit der Zeit zu, wenn ihr Wert nach oben „erhöht“ wird, andernfalls nimmt sie ab. Diese negative Rückkopplung ist stabil und führt zu einem Gleichgewicht, wenn t ÿ ÿ.

Für die Wärmegleichung sind zwei Randbedingungen erforderlich, die beide mit einhergehen einfache physikalische Interpretationen:

- Die Wärmeverteilung f(0; x, y) zum Zeitpunkt t = 0 an allen Punkten (x, y) ÿ ÿ

- Verhalten von f, wenn t > 0 an Punkten (x, y) ÿ ÿÿ. Diese Randbedingungen beschreiben das Verhalten an der Grenze der Domäne. Dirichlet-Bedingungen liefern hier f(t; x, y) für alle t ÿ 0 und (x, y) ÿ ÿÿ, entsprechend der Situation, in der ein äußerer Agent die Temperaturen an der Grenze der Domäne festlegt. Diese Bedingungen können auftreten, wenn ÿ ein Stück Folie ist, das neben einer Wärmequelle liegt, deren Temperatur durch die Folie nicht wesentlich beeinflusst wird, wie z. B. ein großer Kühlschrank oder Ofen. Im Gegensatz dazu geben Neumann-Bedingungen die Ableitung von f in der Richtung normal zur Grenze ÿÿ an, wie in Abbildung NUMBER; Sie entsprechen der Festlegung des Wärmeflusses aus ÿ, der durch verschiedene Arten der Isolierung verursacht wird.

### 14.3.3 Hyperbolische PDEs

Die endgültige Modellgleichung ist die Wellengleichung, die dem Fall der unbestimmten Matrix entspricht:

$$\frac{\ddot{y}\, 2}{\ddot{y}t\ 2} f\ \ddot{y}\ c\ 2\ddot{y}2\ f = 0$$

Die Wellengleichung ist hyperbolisch, da die zweite zeitliche Ableitung das entgegengesetzte Vorzeichen der beiden räumlichen Ableitungen hat. Diese Gleichung bestimmt die Bewegung von Wellen über ein elastisches Medium wie eine Gummiplatte; Es kann beispielsweise abgeleitet werden, indem das zweite Newtonsche Gesetz auf Punkte auf einem Gummistück angewendet wird, wobei x und y Positionen auf dem Blatt sind und f(t; x, y ) die Höhe des Gummistücks zum Zeitpunkt t ist.

Abbildung NUMBER zeigt eine eindimensionale Lösung der Wellengleichung. Das Wellenverhalten steht in erheblichem Gegensatz zur Wärmediffusion, da bei t ÿ ÿ die Energie möglicherweise nicht diffundiert. Insbesondere können Wellen innerhalb einer Domäne unbegrenzt hin und her springen. Aus diesem Grund werden wir sehen, dass implizite Integrationsstrategien möglicherweise nicht für die Integration hyperbolischer PDEs geeignet sind, da sie dazu neigen, Bewegungen zu dämpfen.

Die Randbedingungen für die Wellengleichung ähneln denen der Wärmegleichung, jedoch jetzt wir müssen sowohl f(0; x, y) als auch ft(0; x, y) zum Zeitpunkt Null angeben:

- Die Bedingungen bei t = 0 geben die Position und Geschwindigkeit der Welle zum Anfangszeitpunkt an.
- Randbedingungen an ÿ bestimmen, was an den Enden des Materials passiert. Dirichlet-Bedingungen entsprechen dem Fixieren der Seiten der Welle, z. B. dem Zupfen einer Cellosaite, die an ihren beiden Enden flach am Instrument gehalten wird. Neumann-Bedingungen entsprechen dem Unberührtlassen der Enden der Welle wie dem Ende einer Peitsche.

## 14.4 Derivate als Operatoren

In PDEs und anderswo können wir uns Ableitungen als Operatoren vorstellen, die auf Funktionen genauso einwirken wie Matrizen auf Vektoren. Unsere Wahl der Notation spiegelt oft diese Parallele wider: Die Ableitung d f/dx sieht aus wie das Produkt eines Operators d/dx und einer Funktion f . Tatsächlich ist die Differentiation ein linearer Operator, genau wie die Matrixmultiplikation, da für alle f , g : **R** ÿ **R** und a, b ÿ **R** gilt

$$\frac{d}{dx}(a f(x) + b g(x)) = a \frac{d}{dx} f(x) + b \frac{d}{dx} g(x).$$

Wenn wir PDEs für numerische Lösungen diskretisieren, können wir diese Analogie tatsächlich vollständig ausführen. Betrachten Sie beispielsweise eine Funktion f auf [0, 1] , die mit n + 1 gleichmäßig verteilten Stichproben diskretisiert wurde, wie in Abbildung NUMBER. Denken Sie daran, dass der Abstand zwischen zwei Stichproben h = 1/n beträgt. In Kapitel 12 haben wir eine Näherung für die zweite Ableitung f (x) entwickelt :

$$f(x) = \frac{f(x + h) \ddot{y} 2 f(x) + f(x \ddot{y} h)}{h} + O(h)$$

Angenommen, unsere n Stichproben von f(x) auf [0, 1] sind y0 ÿ f(0), y1 ÿ f(h), y2 ÿ f(2h), . . . , yn = f(nh).
Dann ergibt die Anwendung unserer Formel oben eine Strategie zur Annäherung von f an jedem Gitterpunkt:

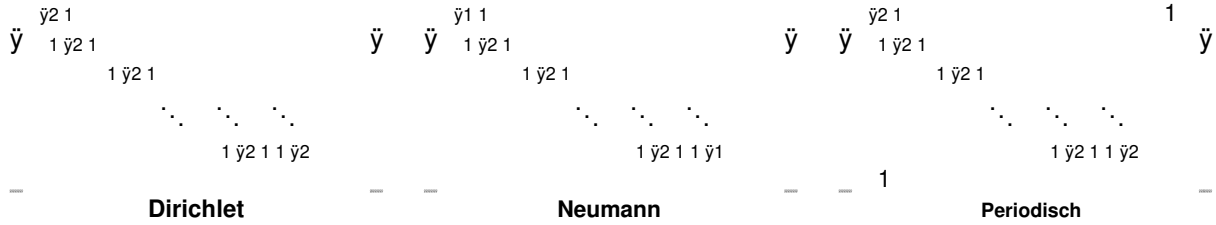
$$\text{jk}_{\ddot{y}} \quad \frac{yk+1 \ddot{y} 2yk + yk\ddot{y}1}{h^2}$$

Das heißt, die zweite Ableitung einer Funktion auf einem Punktgitter kann mit der in Abbildung NUMBER(a) dargestellten 1— ÿ 2—1-Schablone berechnet werden.

Eine Feinheit, auf die wir nicht eingegangen sind, ist, was bei y 0 und yn passiert , da die obige Formel yÿ1 und yn+1 erfordern würde. Tatsächlich kodiert diese Entscheidung die in §14.2 eingeführten Randbedingungen.
Unter Berücksichtigung der Tatsache, dass y0 = f(0) und yn = f(1), sind Beispiele für mögliche Randbedingungen für f:

- Dirichlet-Randbedingungen: yÿ1 = yn+1 = 0, das heißt, legen Sie einfach den Wert von y über dem fest Endpunkte
- Neumann-Randbedingungen: yÿ1 = y0 und yn+1 = yn, Kodierung der Randbedingung f (0) = f (1) = 0.
- Periodische Randbedingungen: yÿ1 = yn und yn+1 = y0, was die Identifikation f(0) = ergibt f(1)

Angenommen, wir stapeln die Stichproben yk in einen Vektor y ÿ Rn+1 und den Stichproben y k- in eine Sekunde Vektor w ÿ Rn+1 . Dann ist aus unserer obigen Konstruktion leicht zu erkennen, dass h 2w = L1y, wobei L1 eine der folgenden Optionen ist:




Das heißt, die Matrix L kann als diskretisierte Version des Operators y ÿ Rn+1 und nicht als Funktion $\frac{d}{2\,dx2}$ Einwirken auf f : [0, 1] ÿ R betrachtet werden.

Wir können eine ähnliche Näherung für ÿ2 f schreiben, wenn wir f : [0, 1] × [0, 1] ÿ **R** mit einem Wertegitter abtasten, wie in Abbildung NUMBER. Denken Sie insbesondere daran, dass in diesem Fall ÿ2 f = fxx + fyy ist, sodass wir insbesondere die zweiten Ableitungen von x und y aufsummieren können, wie wir es im obigen eindimensionalen Beispiel getan haben. Dies führt zu einer verdoppelten 1- ÿ 2-1-Schablone, wie in Abbildung NUMMER. Wenn wir unsere Stichproben als yk, ÿ f(kh, h) nummerieren, lautet unsere Formel für den Laplace-Operator von f in diesem Fall:

$$(\ddot{y}2\ y)k,\ \ddot{y}\ h\frac{1}{2}\ (y(k\ddot{y}1),\ +\ yk,(\ddot{y}1)\ +\ y(k+1),\ +\ yk,(+1)\ \ddot{y}\ 4yk,)$$

Wenn wir unsere Stichproben von y und y noch einmal zu y und w kombinieren, können wir mit einer ähnlichen Konstruktion und Wahl der Randbedingungen wieder h 2w = L2y schreiben. Dieser zweidimensionale Gitter-Laplace-Operator L2 erscheint in vielen Bildverarbeitungsanwendungen, in denen (k, ) zum Indizieren von Pixeln in einem Bild verwendet wird.

Nach der obigen Diskussion stellt sich natürlich die Frage, warum wir in unserer obigen Diskussion zur zweiten Ableitung des Laplace-Operators gesprungen sind, anstatt die erste Ableitung f ( x) zu diskretisieren. Im Prinzip gibt es keinen Grund, warum wir nicht ähnliche Matrizen D erstellen könnten, die Vorwärts-, Rückwärts- oder symmetrische Differenznäherungen von f implementieren. Einige technische Details machen diese Aufgabe jedoch etwas schwieriger, wie weiter unten beschrieben.

Am wichtigsten ist, dass wir entscheiden müssen, welche Näherung der ersten Ableitung wir verwenden möchten. 1 als Vorwärtsdifferenz h − Wenn wir beispielsweise y k (yk+1 ÿ yk) schreiben, befinden wir uns in der unnatürlich asymmetrischen Situation, dass wir bei yn eine Randbedingung benötigen, bei y0 jedoch nicht . Im Gegensatz dazu könnten wir die symmetrische Differenz (yk+1 ÿ ykÿ1) verwenden, aber diese Diskretisierung leidet unter einem subtileren Zaunpfahl f2h Problem, das in Abbildung NUMMER dargestellt ist. Insbesondere ignoriert diese Version von y k den Wert von yk und betrachtet nur seine Nachbarn ykÿ1 und yk+1 , was zu Artefakten führen kann, da jede Zeile von D nur yk für entweder gerades oder ungerades k, aber nicht beides beinhaltet.

Wenn wir Vorwärts- oder Rückwärtsableitungen verwenden, um die Zaunpfahlprobleme zu vermeiden, verlieren wir eine Größenordnung an Genauigkeit und leiden außerdem unter den oben beschriebenen Asymmetrien. Wie bei der Leapfrog-Integration

Algorithmus in §13.4.2 besteht eine Möglichkeit, diese Probleme zu vermeiden, darin, sich die Ableitungen als auf halben Gitterpunkten lebend vorzustellen, wie in Abbildung NUMMER dargestellt. Im eindimensionalen Fall entspricht diese Änderung der Differenz an Scheitelpunkten, $\frac{1}{\Delta x}(y_{k+1} - y_k)$ als $y_{k+1/2}$ . Diese Technik, bei der verschiedene Ableitungen zur Kennzeichnung Kanten und Mittelpunkten von Gitterzellen platziert werden, kommt besonders häufig in der Flüssigkeitssimulation zum Einsatz, bei der Drücke, Flüssigkeitsgeschwindigkeiten usw. an Stellen beibehalten werden, die die Berechnungen vereinfachen.

Abgesehen von diesen Feinheiten ist unsere wichtigste Schlussfolgerung aus dieser Diskussion, dass, wenn wir eine Funktion f(x) diskretisieren, indem wir die Stichproben (xi , yi) verfolgen, die meisten vernünftigen Näherungen von Ableitungen von f als Produkt Lx für eine Matrix berechenbar sind L. Diese Beobachtung vervollständigt die Analogie: „Ableitungen wirken auf Funktionen wie Matrizen auf Vektoren.“ Oder in standardisierter Prüfungsnotation: Ableitungen: Funktionen:: Matrizen: Vektoren

# 14.5 PDEs numerisch lösen

Über die Theorie der PDEs bleibt noch viel zu sagen. Fragen der Existenz und Einzigartigkeit sowie der Möglichkeit, Lösungen für verschiedene PDEs zu charakterisieren, führen zu differenzierten Diskussionen unter Verwendung fortgeschrittener Aspekte der realen Analyse. Während ein vollständiges Verständnis dieser Eigenschaften erforderlich ist, um die Wirksamkeit von PDE-Diskretisierungen rigoros nachzuweisen, verfügen wir bereits über genügend Kenntnisse, um einige Techniken vorzuschlagen, die in der Praxis eingesetzt werden.

## 14.5.1 Elliptische Gleichungen lösen

Die meiste Arbeit zur Lösung elliptischer PDEs haben wir bereits in §14.4 erledigt. Nehmen wir insbesondere an, wir möchten eine lineare elliptische PDE der Form $Lf = g$ lösen . Hier ist L ein Differentialoperator; Um beispielsweise die Laplace-Gleichung zu lösen, würden wir $L \equiv \nabla^2$ als Laplace-Operator nehmen. Dann haben wir in §14.4 gezeigt, dass, wenn wir f diskretisieren, indem wir eine Menge von Stichproben in einem Vektor y mit $y_i = f(x_i)$ nehmen, dann eine entsprechende Näherung von Lf für eine Matrix L als Ly geschrieben werden kann. Wenn wir auch g diskretisieren Unter Verwendung von Stichproben in einem Vektorb wird die Lösung der elliptischen PDE $Lf = g$ durch die Lösung des linearen Systems $Ly = b$ angenähert.

**Beispiel 14.7** (Elliptische PDE-Diskretisierung). Angenommen, wir möchten Lösungen für $f''(x) = g(x)$ auf [0, 1] mit Randbedingungen $f(0) = f(1) = 0$ approximieren. Wir werden f(x) mit einem Vektor $y \in \mathbb{R}^n$ approximieren Probenahme f wie folgt:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \approx \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f(nh) \end{bmatrix}$$

wobei $h = 1/n+1$. Wir fügen keine Stichproben bei $x = 0$ oder $x = 1$ hinzu, da dort die Randbedingungen die Werte bestimmen. Wir werden b verwenden, um einen analogen Satz von Werten für g(x) zu halten.

Unter Berücksichtigung unserer Randbedingungen diskretisieren wir f (x) als $h^{-2}Ly$, wobei L gegeben ist durch:

$$L = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 \end{bmatrix}$$

Somit ist unsere Näherungslösung der PDE gegeben durch $y = h^2 L^{-1} b$.

So wie elliptische PDEs die am einfachsten zu lösenden PDEs sind, sind ihre Diskretisierungen mithilfe von Matrizen wie im obigen Beispiel am einfachsten zu lösen. Tatsächlich ist die Diskretisierung eines elliptischen Operators L im Allgemeinen eine positiv definite und dünn besetzte Matrix, die sich perfekt für die in Kapitel 10 abgeleiteten Lösungstechniken eignet.

**Beispiel 14.8** (Elliptische Operatoren als Matrizen). Betrachten Sie die Matrix L aus Beispiel 14.7. Wir können zeigen, dass L negativ definit ist (und daher das positiv definite System $-Ly = -h^2 b$ unter Verwendung konjugierter Gradienten gelöst werden kann), indem wir beachten, dass $-L = D^T D$ für die Matrix $D \in \mathbb{R}^{(n+1)\times n}$ gegeben durch:

$$D = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 \end{bmatrix}$$

Diese Matrix ist nichts anderes als die endliche differenzierte erste Ableitung, daher entspricht diese Beobachtung der Tatsache, dass $d^2 f/dx^2 = d/dx(df/dx)$. Somit ist $x^T Lx = -x^T D^T Dx = -\|Dx\|_2^2 \leq 0$, was zeigt, dass L negativ semidefinit ist. Es ist leicht zu erkennen, dass $Dx = 0$ genau dann ist, wenn $x = 0$, was den Beweis vervollständigt, dass L tatsächlich negativ definit ist.

### 14.5.2 Parabolische und hyperbolische Gleichungen lösen

Parabolische und hyperbolische Gleichungen führen im Allgemeinen eine Zeitvariable in die Formulierung ein, die ebenfalls differenziert ist, jedoch möglicherweise in niedrigerer Ordnung. Da Lösungen parabolischer Gleichungen viele Stabilitätseigenschaften zulassen, sind numerische Techniken zum Umgang mit dieser Zeitvariablen oft stabil und gut konditioniert; Im Gegensatz dazu muss mehr Sorgfalt darauf verwendet werden, hyperbolisches Verhalten zu behandeln und eine Dämpfung der Bewegung im Laufe der Zeit zu verhindern.

**Semidiskrete Methoden** Der wahrscheinlich einfachste Ansatz zur Lösung einfacher zeitabhängiger Gleichungen ist die Verwendung einer semidiskreten Methode. Hier diskretisieren wir den Bereich, aber nicht die Zeitvariable, was zu einer ODE führt, die mit den Methoden aus Kapitel 13 gelöst werden kann.

**Beispiel 14.9** (Semidiskrete Wärmegleichung). Betrachten Sie die Wärmegleichung in einer Variablen, gegeben durch $f_t = f_{xx}$, wobei $f(t; x)$ die Wärme an Position x und Zeit t darstellt. Als Grenzdaten stellt der Benutzer Folgendes bereit:

Funktion f0(x) mit f(0; x) ÿ f0(x); wir verbinden auch den Rand x ÿ {0, 1} mit einem Kühlschrank und erzwingen damit f(t; 0) = f(t; 1) = 0.

Angenommen, wir diskretisieren die x-Variable, indem wir Folgendes definieren:

$$\begin{aligned} f_1(t) &\ddot{y} f(h; t) \\ f2(t) &\ddot{y} f(2h; t) \\ &\vdots \quad \vdots \\ fn(t) &\ddot{y} f(nh; t), \end{aligned}$$

wobei wir wie in Beispiel 14.7 h = 1/n+1 nehmen und Stichproben bei x ÿ {0, 1} weglassen , da sie durch die Randbedingungen bereitgestellt werden.

Durch die Kombination dieser fi können wir f(t) : **R** ÿ Rn als die semidiskrete Version von f definieren, bei der wir im Raum, aber nicht in der Zeit abgetastet haben. Nach unserer Konstruktion ist die semidiskrete PDE-Näherung die durch f (t) = L f(t) gegebene ODE .

Das vorherige Beispiel zeigt ein Beispiel eines sehr allgemeinen Musters für parabolische Gleichungen. Wenn wir kontinuierliche Phänomene simulieren, wie etwa Wärme, die sich über eine Domäne bewegt, oder Chemikalien, die durch eine Membran diffundieren, gibt es normalerweise eine Zeitvariable und dann mehrere räumliche Variablen, die auf elliptische Weise differenziert sind. Wenn wir dieses System semidiskret diskretisieren, können wir ODE-Integrationsstrategien für ihre Lösung verwenden. Genauso wie die Matrix, die zur Lösung einer linearen elliptischen Gleichung wie in §14.5.1 verwendet wird, im Allgemeinen positiv oder negativ definit ist, ist die Matrix L normalerweise negativ definit, wenn wir eine semidiskrete parabolische PDE f = L f schreiben. Diese Beobachtung impliziert, dass die Lösung dieser kontinuierlichen ODE durch f bedingungslos stabil ist, da negative Eigenwerte mit der Zeit gedämpft werden.

Wie im vorherigen Kapitel dargelegt, haben wir aufgrund einer räumlichen Diskretisierung viele Möglichkeiten, die ODE zeitlich zu lösen. Wenn die Zeitschritte klein und begrenzt sind, können explizite Methoden akzeptabel sein. Implizite Löser werden häufig zur Lösung parabolischer PDGs eingesetzt; Das diffusive Verhalten des impliziten Eulers kann zu Ungenauigkeiten führen, aber verhaltenstechnisch gesehen ähnelt es der durch die Wärmegleichung bereitgestellten Diffusion und kann selbst bei relativ großen Zeitschritten akzeptabel sein. Hyperbolische PDEs erfordern möglicherweise implizite Schritte für die Stabilität, aber fortgeschrittene Integratoren wie „symplektische Integratoren“ können eine durch diese Art von Schritten verursachte Überglättung verhindern.

Ein kontrastierender Ansatz besteht darin, Lösungen semidiskreter Systeme f = L f in Form von Eigenvektoren von L zu schreiben. Angenommen, v1, . . . ,vn sind Eigenvektoren von L mit den Eigenwerten ÿ1, . . . , ÿn und dass wir f(0) = c1v1 + · · · + cnvn wissen. Denken Sie dann daran, dass die Lösung von f = L f gegeben ist durch:

$$f(t) = \ddot{y} \quad cie\, vi^{\ddot{y}i\, t}$$

Diese Formel ist nichts Neues über §5.1.2 hinaus, den wir während unserer Diskussion über Eigenvektoren und Eigenwerte eingeführt haben. Die Eigenvektoren von L können jedoch im Fall einer semidiskreten PDE eine physikalische Bedeutung haben, wie in Beispiel 14.5, das zeigte, dass Eigenvektoren der Laplace-Operatoren L verschiedenen Resonanzschwingungen der Domäne entsprechen. Somit kann dieser Eigenvektoransatz angewendet werden, um beispielsweise „Niederfrequenznäherungen“ der Anfangswertdaten zu entwickeln, indem die obige Summe über i gekürzt wird, mit dem Vorteil, dass die t-Abhängigkeit ohne Zeitschritt genau bekannt ist.

**Beispiel 14.10** (Eigenfunktionen des Laplace-Operators). Abbildung NUMBER zeigt Eigenvektoren der Matrix L aus Beispiel 14.7. Eigenvektoren mit niedrigen Eigenwerten entsprechen Niederfrequenzfunktionen auf [0, 1] mit an den Endpunkten festgelegten Werten und können gute Annäherungen an f(x) sein , wenn es relativ glatt ist.

**Vollständig diskrete Methoden** Alternativ könnten wir die Raum- und Zeitvariablen demokratischer behandeln und beide gleichzeitig diskretisieren. Diese Strategie ergibt ein zu lösendes Gleichungssystem, das eher §14.5.1 ähnelt. Diese Methode lässt sich leicht formulieren, indem man den elliptischen Fall parallelisiert, aber die resultierenden linearen Gleichungssysteme können groß sein, wenn die Abhängigkeit zwischen Zeitschritten eine globale Reichweite hat.

**Beispiel 14.11** (Vollständig diskrete Wärmediffusion). Explizit, implizit, Crank-Nicolson. Nicht in CS 205A abgedeckt.

Es ist wichtig zu beachten, dass letztendlich sogar semidiskrete Methoden als vollständig diskret betrachtet werden können, da die zeitschrittbasierte ODE-Methode immer noch die t-Variable diskretisiert; Der Unterschied besteht hauptsächlich in der Klassifizierung der Art und Weise, wie Methoden abgeleitet wurden. Ein Vorteil semidiskreter Techniken besteht jedoch darin, dass sie den Zeitschritt für t abhängig von der aktuellen Iteration anpassen können. Wenn sich Objekte beispielsweise in einer physikalischen Simulation schnell bewegen, kann es sinnvoll sein, mehr Zeitschritte zu verwenden, um diese Bewegung aufzulösen. Einige Methoden passen sogar die Diskretisierung des Bereichs der x-Werte an, falls in der Nähe lokaler Diskontinuitäten oder anderer Artefakte eine höhere Auflösung erforderlich ist.

## 14.6 Methode der Finiten Elemente

Nicht in 205A abgedeckt.

## 14.7 Beispiele aus der Praxis

Anstelle einer strengen Behandlung aller häufig verwendeten PDE-Techniken stellen wir in diesem Abschnitt Beispiele dafür vor, wo sie in der Informatik in der Praxis vorkommen.

### 14.7.1 Bildverarbeitung im Gradientenbereich

### 14.7.2 Kantenerhaltende Filterung

### 14.7.3 Gitterbasierte Flüssigkeiten

## 14.8 Zu erledigen

- Mehr zum Thema Existenz/Einzigartigkeit
- CFL-Bedingungen
- Laxer Äquivalenzsatz
- Beständigkeit, Stabilität und Freunde

## 14.9 Probleme

- Zeigen Sie, dass der 1d-Laplace-Operator als DD für die erste Ableitungsmatrix D faktorisiert werden kann
- Lösen Sie die PDE erster Ordnung

# Danksagungen

[Diskussion geht hier]

Ich schulde den Studenten des Stanford CS 205A, Herbst 2013, viel Dank dafür, dass sie bei der Entwicklung dieses Buches zahlreiche Tippfehler und Fehler entdeckt haben. Das Folgende ist zweifellos eine unvollständige Liste der Studenten, die zu diesem Projekt beigetragen haben: Tao Du, Lennart Jansson, Miles Johnson, Luke Knepper, Minjae Lee, Nisha Masharani, John Reyna, William Song, Ben-Han Sung, Martina Troesch, Ozhan Turgut, Patrick Ward, Joongyeub Yeo und Yang Zhao.

Besonderer Dank geht an Jan Heiland und Tao Du für ihre Hilfe bei der Klärung der Ableitung des BFGS-Algorithmus.

[Weitere Diskussion hier]