FORGOTTEN FORGE

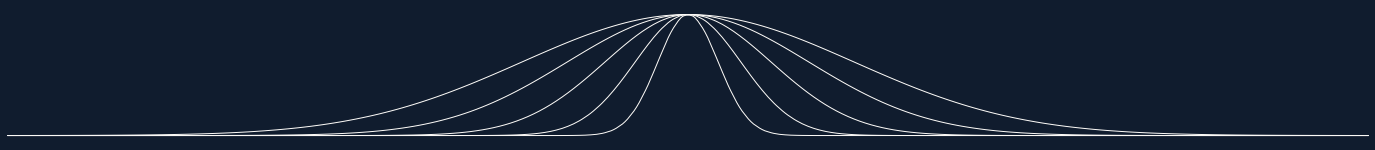
Ein Lehrbuch in sechs Teilen

Der Gipfel

ein universelles Rezept für Phasenübergänge, in zwölf Welten

Quantenhardware — Magnetismus — Finanzmärkte — Maschinelles Lernen
Seismologie — Klima — GPU — Proteine — Sprache
Zahlentheorie — Edge — LLM-Ökonomie

M. C. Wurm
ForgottenForge | Buckenhof | 2026



Begleitband zu SIGMA-C-FRAMEWORK v3.0.0
Empirischer Anker: AVS Quantum Science **8**, 013804 (2026)

Der Gipfel

ein universelles Rezept für Phasenübergänge,
in zwölf Welten

M. C. Wurm

ForgottenForge.xyz | 2026

Vorwort

σ_c ist die Stelle, an der das System kippt. $\Pi = D\,\gamma$ ist der Grund, warum es kippt. Der meiste Teil dieses Buches handelt vom Ersten. Das mittlere Drittel handelt vom Zweiten.

Wir versprechen Ihnen keine Revolution. Revolutionen sind laut, unordentlich und schlecht organisiert; am Morgen danach ist niemand für das Aufräumen zuständig. Wir versprechen Ihnen stattdessen etwas Leiseres. Ein Rezept. Drei Zeilen Python, die ein System mit einem einstellbaren Knopf und einer messbaren Antwort nehmen und Ihnen sagen, an welcher Stelle es umschlägt.

Einen Parameter durchfahren. Messen. Ableiten. Den Gipfel finden. Vier Schritte; die letzten drei ändern sich zwischen den Welten nicht, nur der erste tut es. Dieselben vier Schritte finden die Curie-Temperatur eines Eisenmagneten, die operationale Rauschschwelle eines Quantenprozessors, den Regimewechsel an einer Börse und das Erkrankungsalter einer erblichen Proteinkrankheit. Zwölf Domänen, ein Rezept. Das klingt verdächtig nach Marktschreierei. Wir wissen es. Die erste Hälfte dieses Buches ist deshalb für die Misstrauischen geschrieben — ihre Leser bleiben einem treuer.

Eine gute Methode ist wie ein guter Butler: man bemerkt sie nicht. Sie verrichtet ihre Arbeit so leise, wie sich der Deutsche einen Butler vorstellt, ohne je einen gesehen zu haben. Sie erfindet keine Probleme und löst auch nicht mehr, als von ihr verlangt worden ist. An ihren schlechten Tagen teilt sie Ihnen mit, dass sie nicht weiterhelfen kann, und macht sich auf, jemanden zu suchen, der es vermag. Wenn sich der Suszeptibilitäts-Rahmen anständig benimmt, ist er genau diese Sorte Butler. Benimmt er sich daneben, dann nennt er Ihnen, welche von fünf Vertrauensbedingungen verletzt worden ist — und genau das zugehörige Kapitel steht aus gutem Grund am Anfang von Teil II.

Wie es dazu kam. Es fing damit an, dass ich zusehen musste.

Nicht ganz freiwillig. Wenn man ein gewisses Alter erreicht hat, hat man bereits eine erhebliche Anzahl Mitmenschen kennengelernt, und ein bestimmter Teil davon scheint sich darauf spezialisiert zu haben, in einer Weise zu handeln, die nachträglich die Frage aufwirft: wann eigentlich ist das passiert? Der Kollege, der seinen sicheren Posten kündigt, um Kryptowährungen zu züchten. Der Bekannte, der nach drei Jahren des Schimpfens über seine Verlobte plötzlich heiratet. Die Patientin, die in der Sprechstunde verkündet, sie habe die Operation nun doch nicht vor, sie habe online etwas Besseres gefunden. Der Bundestag, der eine Kommission einsetzt. Der Hausbesitzer, der die Küche zum elften Mal renoviert.

Ich war nicht dabei, als diese Leute ihre Entscheidung trafen. Aber ich war dabei kurz davor und kurz danach, und in jedem dieser Fälle gab es einen Moment — einen schmalen

Moment, oft kein ganzer Tag —, in dem die Person noch auf der einen Seite stand und kurz darauf, unwiderruflich, auf der anderen.

Diesen Moment nenne ich seither den *Kipppunkt*. Es ist ein bescheidener Name für etwas, das mir, ehrlich gesagt, die Sprache verschlägt.

Eine Zeit lang habe ich gehofft, das wäre eine soziale Beobachtung. Etwas, das nur Menschen betrifft, und nur die unwilligen unter ihnen. Es stellte sich heraus, dass das nicht stimmt. Magnete haben einen Kipppunkt. Aktienmärkte haben einen Kipppunkt. Quantencomputer haben einen Kipppunkt. Erdbeben, das Klima, Stromnetze, Proteine — alle haben Kipppunkte. Es ist die häufigste Eigenschaft, die ich an Systemen kenne, gleich nach der Eigenschaft, dass sie kaputtgehen können.

Das wäre nun an sich nicht schlimm. Schlimm ist nur, dass ich, nachdem ich einige Jahre damit verbracht habe, diese Kipppunkte zu identifizieren — was übrigens, wenn man weiß, wonach man sucht, gar nicht so schwer ist —, immer noch nicht weiß, wie man von der unangenehmen Seite zurück auf die angenehme kommt.

Das ist der zweite Teil. Der erste Teil — „hier ist der Kipppunkt“ — ist mathematisch lösbar. Das ist dieses Buch.

Der zweite Teil — „wie kommt man dorthin zurück, wo es noch nett war“ — ist nicht mathematisch lösbar. Vielleicht ist er gar nicht lösbar. Vielleicht ist er nur jenseits dessen, was ich als Einzelner in einer Lebenszeit klären kann. Ich wäre froh, wenn jemand, der dies liest, mir dazu eine Idee schickt.

Bis dahin: das Verfahren, das im Folgenden beschrieben wird, sagt einem wenigstens, dass es einen Kipppunkt gibt. Es sagt, wo er ist. Es sagt sogar, wie scharf er ist, was sich als überraschend nützlich herausgestellt hat. Was es nicht sagt, ist, was man tun soll, wenn man ihn gefunden hat. Das müssen die Leser entscheiden.

Es ist ein bescheidenes Buch. Es kann eine Sache. Diese eine Sache hatte ich nicht erfinden wollen — sie hat sich mir, durch das Hinsehen über mehrere Jahrzehnte, aufgedrängt. Ich überreiche sie hier in der Hoffnung, dass das nächste Mal vielleicht jemand zur rechten Zeit hinguckt und sagt: „Moment, halt — das kippt gleich. Wolltest du das wirklich?“ Auch das wäre eine Verbesserung gegenüber dem jetzigen Zustand, wenn auch eine kleine.

Eines noch. Wir setzen voraus, dass Sie addieren, subtrahieren, multiplizieren und dividieren können. Wir setzen voraus, dass Sie lesen können. Wir setzen *nicht* voraus, dass Sie schon einen griechischen Buchstaben gesehen, einem Erwartungswert begegnet sind oder ein Python-Skript ausgeführt haben. Wenn ein Eintrag im Inhaltsverzeichnis Sie einschüchtert, ignorieren Sie ihn; bis Sie am Kapitel ankommen, wird er es nicht mehr tun.

Der erste Entwurf dieses Buches war 78 Seiten lang und unlesbar. Der zweite war 130 Seiten lang und unehrlich. Dies ist der dritte, und er ist ehrlich auf die Art, auf die ein guter Butler ehrlich ist — indem er meldet, was tatsächlich im Zimmer steht.

Über dieses Buch

Dieses Buch hat eine einzige Aufgabe. Es soll eine Leserin oder einen Leser, die rechnen können, an den Punkt bringen, an dem sie den *Suszeptibilitäts-Rahmen* — die σ_c-Methode — auf ihre eigenen Daten anwenden können. Quantencomputer oder Aktienmarkt, magnetisches Material oder GPU-Benchmark. Das Rezept ist *ein* Rezept.

Für wen dieses Buch geschrieben ist. Für jeden, der schon einmal etwas gemessen hat, während er an einem Knopf drehte, und sich gefragt hat: *„Wo kippt das hier um?“* Ein Studienabschluss ist nicht nötig. Vier Grundrechenarten und etwas Geduld für ein durchgerechnetes Beispiel reichen.

Was dieses Buch nicht ist. Eine Forschungsmonographie. Wir setzen nicht voraus, dass Sie Aufsätze über Phasenübergänge, Fisher-Information oder Nichtgleichgewichts-Statistik gelesen haben. Wo diese Ideen auftauchen, bauen wir sie vom Boden auf.

Das Versprechen. Am Ende von Teil II betreiben Sie die σ_c-Methode in fünf Zeilen Python. Am Ende von Teil III verstehen Sie, *warum* sie funktioniert. Am Ende von Teil IV wissen Sie, welcher der zwölf mitgelieferten Domänen-Adapter zu Ihrem Problem passt — und wie Sie den Rahmen um einen dreizehnten erweitern, der ganz Ihrer ist.

Wie man liest. Die Teile I bis III sind eine Kette. Wer Glieder überspringt, hinterlässt Lücken, in die spätere Kapitel leise hineinfallen. Ab Teil IV steht jedes Kapitel für sich. Wählen Sie die Domäne, die zu Ihrer Arbeit passt; die anderen sind da, wenn Sie sie brauchen.

Konventionen.

- Jedes neue Symbol wird bei seinem ersten Erscheinen definiert. Griechische Buchstaben kriegen keinen Freifahrtschein.
- Jeder Formel folgt mindestens ein numerisches Beispiel. Wenn Sie die Zahl auf Papier nicht nachrechnen können, ist die Erklärung noch nicht zu Ende.
- Jedes Kapitel endet mit einer kleinen Übung. Sie dauert fünf bis dreißig Minuten und ist die Zeit wert.
- Der Code ist in Python geschrieben und läuft in jeder Umgebung, in der NumPy und SciPy vorhanden sind. Die Bibliothek `sigma_c` ist eine Bequemlichkeit, keine Voraussetzung — der fünfzeilige Kern passt auf einen Bierdeckel.

- Sätze, die ein Kapitel überleben, ohne durch einen besseren ersetzt zu werden, stehen am Ende in einem farbigen Kasten. Vier Farben, vier Zwecke: *blau* für das, was man behalten soll, *orange* für das, was man meiden soll, *grün* für das, was man probieren soll, und *rot* für ausdrückliche Warnungen (klinisch, finanziell, sicherheitsrelevant).
- **Das Symbol σ wird in jedem Kapitel neu belegt.** In jedem Kapitel bezeichnet σ die Kontrollachse *dieses Kapitels*, nicht ein universelles physikalisches Sigma. Der griechische Buchstabe ist kurz und traditionell; eine eigene physikalische Bedeutung trägt er nicht von einer Domäne zur nächsten. Für die Kernbreite schreiben wir σ_{ker} und für die Volatilität im Finanzkapitel σ_t, um Kollisionen zu vermeiden.

Der zugehörige Rahmen. Die begleitende quelloffene Bibliothek heißt `sigma-c-framework` (Version 3.0.0). Installation:

```
pip install sigma-c-framework
```

Sie können dieses Buch zu Ende lesen, ohne sie je zu installieren; der fünfzeilige Kern braucht nur NumPy und SciPy. Mit dem Rahmen ist es bequemer, so wie es mit einem Butler bequemer ist als ohne.

Zitierweise.

> M. C. Wurm, *Der Gipfel: ein universelles Rezept für Phasenübergänge, in zwölf Welten*, ForgottenForge, Buckenhof, 2026.

(Lizenz- und Auflagedetails stehen im Kolophon am Schluss.)

Drei Lesepfade

In diesem Buch finden sich ungefähr vierhunderttausend Dinge zu lesen, und Sie werden mit großer Sicherheit nicht alle davon lesen wollen. Wir schlagen Ihnen daher drei Lesepfade vor, nach Tiefe gestaffelt:

Pfad A — „ich will es nur anwenden“ (1–2 Abende).

- Vorrede und Griechisch-Hilfe (15 min)
- Kapitel 1–7 *im Überflug* (Grundlagen; überblicken Sie, wenn Sie je Differentialrechnung hatten)
- Kapitel 9 (**das universelle Rezept**)
- Kapitel 10 (**wenn die Methode versagt**)
- Kapitel 12 (**κ-Schwellen**)
- Ein Kapitel aus Teil IV, das zu Ihrer Domäne passt
- Teil V (**Validierung**) kurz bevor Sie veröffentlichen

Pfad B — „ich will verstehen, warum es funktioniert“ (eine Woche). Pfad A, plus den gesamten Teil III (Kontraktionsgeometrie) und den gesamten Teil V (Validierung). Kapitel aus Teil IV außerhalb Ihrer Domäne können überflogen werden.

Pfad C — „ich will den Rahmen erweitern“ (zwei Wochen). Pfad B, plus die Rezepte in Teil VI und die Anhänge, plus die Umsetzung eines eigenen Adapters (Kapitel 46). Ab hier sind Sie Mitwirkende, nicht mehr Leser.

Was dieses Buch *nicht* verspricht. Es verspricht keine Theorie von allem. Es verspricht nicht, dass jedes System einen Gipfel hat; manche haben keinen, und Kapitel 10 hilft Ihnen, sie zu erkennen. Es verspricht nicht, dass dasselbe numerische σ_c über Hardware-Plattformen hinweg gilt; die Nachfolgearbeit zum Magnetismuspapier auf anderer Hardware zeigt, dass sich der Wert um ± 3–$5\,\%$ verschiebt, wenn man das Rauschmodell wechselt. Es verspricht nicht, dass die Methode domänenspezifisches Fachwissen ersetzt; die Methode findet, wo man hinschauen muss — was man dort sieht, müssen Sie noch immer selbst deuten.

Was wir tatsächlich versprechen: wenn Ihr System einen einstellbaren Knopf und eine ungefähr monotone Antwort hat, lokalisiert das Rezept aus Kapitel 9 die operationale Schwelle, *ohne* dass Sie vorher ein Modell erfinden müssen.

Inhaltsverzeichnis

Die sieben Symbole, die Sie wirklich brauchen

Wenn Sie aus diesem Buch nichts anderes mitnehmen — behalten Sie die folgenden sieben Symbole. Alles andere wird definiert, wenn es auftaucht.

Symbol	in einer Zeile	taucht zuerst auf
σ	der Kontrollparameter, an dem Sie drehen — die Temperatur, der Rauschpegel, die Lernrate	Kapitel 2
O	die beobachtbare Größe, die Sie dabei messen	Kapitel 2
$\chi(\sigma) = \lvert \mathrm{d}O/\mathrm{d}\sigma \rvert$	die Suszeptibilität: wie schnell sich O ändert	Kapitel 11
σ_c	der Ort des Gipfels von χ — *wo das System kippt*	Kapitel 9
κ	die Gipfelschärfe: $\chi_{\max}$ geteilt durch den mittleren χ	Kapitel 12
D	der Kontraktionsdefekt einer Abbildung: $\lvert S \rvert / \lvert f(S) \rvert$	Kapitel 17
γ	die Drift einer Abbildung: geometrisches Mittel von $f(x)/x$ (*nicht* dasselbe γ wie die Quantenrauschstärke!)	Kapitel 18
$\Pi = D \cdot \gamma$	der Bonus, das achte Symbol: das Kontraktionsprodukt — *warum* das System kippt	Kapitel 19

Die Ein-Satz-Zusammenfassung des ganzen Buches:

MERKE

σ_c ist, wo es kippt. $\Pi = D\gamma$ ist, warum es kippt.

Die volle Aussprachehilfe für die griechischen Buchstaben folgt auf der nächsten Seite (die klassischen Buchstaben muss man sich einprägen, aber jeder ist nur ein Buchstabe).

Teil I

Grundlagen — von der Arithmetik zur Ableitung

Eine Lesehilfe für griechische Buchstaben

Dieses Buch verwendet griechische Buchstaben. Schullehrpläne gehen mit ihnen sparsam um, und vielen Lesern wirken sie einschüchternder, als sie verdienen. Ein griechischer Buchstabe ist ein Buchstabe. Er wird ausgesprochen, wie unten angegeben, und verhält sich sonst genau wie ein gewöhnlicher Buchstabe — man kann ihn multiplizieren, addieren, Gleichungen mit ihm schreiben.

Symbol	Name	wo im Buch
σ	„sigma“	Kontrollparameter (Kapitel 2–7)
σ_c	„sigma-zeh“	Ort des Suszeptibilitätsgipfels
χ	„chi“ (gesprochen „ki“)	die Suszeptibilität selbst
κ	„kappa“	Gipfelschärfe (wie spitz der Gipfel ist)
γ	„gamma“	Drift in der Zahlentheorie, Rauschen in der Quantenwelt
Δ	„delta“ (groß)	Änderung, Differenz
ρ	„rho“ (gesprochen „ro“)	Dichtematrix, Korrelation
ξ	„xi“ (gesprochen „ksi“)	Korrelationslänge
π	„pi“	die Konstante 3,14… und die GARCH-Persistenz
β, α	„beta“, „alpha“	kritische Exponenten bei Magneten, GARCH-Parameter
∂	„del“, „partiell“	eine besondere Art von Ableitung
$\sim$	„geht wie“, „skaliert wie“	ungefähre Proportionalität
$\propto$	„proportional zu“	strenge Proportionalität
$\langle \cdot \rangle$	„spitze Klammern“	Mittelwert über viele Versuche
$\lvert \cdot \rvert$	„Betrag“	Vorzeichen weg, Größe behalten

Wenn Sie eines davon vergessen, kehren Sie auf diese Seite zurück. Sie läuft nicht weg.

Kapitel 1

Was ist eine Zahl, was ist ein Verhältnis?

Dieses Kapitel setzt nichts voraus. Wer $7 + 3 = 10$ und $10 \div 2 = 5$ beherrscht, ist bereits überqualifiziert. Wer hingegen schon weiß, was eine Ableitung ist, darf das Kapitel überfliegen — wer es noch nicht weiß, sollte das nicht tun. Alles, was auf den verbleibenden 300 Seiten folgt, ruht auf den vier Grundrechenarten, sorgfältig ausgeführt. Beginnen wir also damit, sie sorgfältig auszuführen.

1.1 Zahlen, in der einzigen Form, in der wir sie brauchen

Eine *Zahl* ist in diesem Buch eines der folgenden:

- eine ganze Zahl wie 0, 1, 2, 3, -4, 17;
- eine Zahl mit Nachkommastellen wie 0,5, 3,14, $-2{,}718$;
- eine sehr kleine positive Zahl wie 0,001 oder 10^{-6}, gelesen als „zehn hoch minus sechs“. Das Symbol 10^{-6} ist eine kurze Schreibweise für 0,000001.

Die Menge all dieser Zahlen schreiben wir $\mathbb{R}$ und nennen sie „die reellen Zahlen“. Eine tiefere Definition werden Sie nicht brauchen.

1.2 Subtraktion sagt Ihnen eine Änderung

War eine Größe zuvor 4 und ist jetzt 7, dann ist die Änderung $7 - 4 = 3$. Wir sagen, die Größe ist *um* 3 *gestiegen*.

War sie zuvor 4 und ist jetzt 1, dann ist die Änderung $1 - 4 = -3$, also negativ. Wir sagen, die Größe ist *um* 3 *gefallen*. Allein das Minuszeichen sagt Ihnen die Richtung der Änderung.

Definition 1.1 (Änderung)**.** Die Änderung einer Größe O zwischen zwei Messungen ist $\Delta O = O_{\text{neu}} - O_{\text{alt}}$. Der griechische Buchstabe Δ (Delta) wird „delta“ gesprochen und bedeutet von jetzt an immer „Änderung von“.

Beispiel 1.2 (Die Kaffeetasse, Teil 1: die Änderung)**.** Eine Kaffeetasse hatte vor einem Augenblick noch 80 °C, inzwischen zeigt das Thermometer nur noch 77 °C. Die Temperaturänderung beträgt also $\Delta T = T_{\text{neu}} - T_{\text{alt}} = 77 - 80 = -3\,^{\circ}\text{C}$. Das negative Vorzeichen verrät uns, dass die Temperatur um 3 Grad *gefallen* ist. Im weiteren Verlauf dieses Kapitels werden wir noch öfter auf diese Tasse zurückkommen.

1.3 Division sagt Ihnen eine Rate

Die Tasse aus Beispiel 1.2 verlor ihre drei Grad nicht von einem Moment auf den nächsten. Nehmen wir an, sie brauchte dafür 60 Sekunden. Dann ist die *Rate* der Abkühlung

$$\frac{\Delta T}{\Delta t} = \frac{-3\,^\circ\mathrm{C}}{60\,\mathrm{s}} = -0{,}05\,^\circ\mathrm{C} \text{ pro Sekunde.}$$

Zweierlei fällt hier auf: *erstens* haben wir eine Division durchgeführt, und *zweitens* — was nicht weniger wichtig ist — sind die Einheiten brav mitgegangen: Grad im Zähler, Sekunden im Nenner, zusammen also „Grad pro Sekunde“.

Hätte sie nur 5 Sekunden gebraucht, wäre die Rate $-3/5 = -0{,}6\,^\circ\mathrm{C/s}$ — eine deutlich schnellere Abkühlung. Eine große Rate bedeutet eine schnelle Änderung, eine kleine Rate eine langsame. *Dieser eine Gedanke ist der Keim für alles, was in diesem Buch noch folgen wird.*

MERKE

Eine *Rate* ist eine Änderung geteilt durch eine andere Änderung. Sie sagt Ihnen, wie schnell sich eine Größe bewegt, wenn sich eine andere bewegt. Alles an der σ_c-Methode kommt aus *einer* bestimmten Rate: wie schnell sich eine Messung ändert, wenn man am Kontrollparameter dreht.

1.4 Verhältnisse mit Einheiten — und warum Einheiten zählen

Die Rate $-0{,}05\,^\circ\mathrm{C/s}$ hing von den Einheiten ab. Hätten wir die Zeit in Minuten statt Sekunden gemessen, wäre dieselbe Abkühlung $-3\,^\circ\mathrm{C} \div 1\,\mathrm{min} = -3\,^\circ\mathrm{C/min}$. *Derselbe physikalische Vorgang, eine andere Zahl — weil andere Einheit.*

STOLPERFALLE

Wenn Sie zwei Raten vergleichen, müssen Sie dieselben Einheiten benutzen. Eine Rate von $-3\,^\circ\mathrm{C/min}$ ist *nicht* größer als eine Rate von $-0{,}05\,^\circ\mathrm{C/s}$; sie sind gleich. Prüfen Sie immer die Einheiten, bevor Sie schließen.

1.5 Eines hängt vom anderen ab

Bis hierher haben wir eine Größe (die Tassentemperatur) und eine „Achse“ verfolgt, längs derer sie variiert (die Zeit). Das ist die einfachst denkbare Beziehung zwischen zwei Größen — eine Eingabe, eine Ausgabe. Dieses Muster ist so verbreitet, dass die Mathematik ihm einen Namen gegeben hat: eine *Funktion*.

Das nächste Kapitel handelt von Funktionen. Fürs Erste genügt das folgende Bild: eine Funktion ist ein Rezept, das eine Zahl hineinnimmt (die *Eingabe*) und eine Zahl zurückgibt (die *Ausgabe*). Auch unsere Abkühlgeschichte ist eine Funktion: zu jedem Zeitpunkt t gehört genau eine Temperatur T, und wir schreiben $T = T(t)$.

PROBIER ES

Ein Auto war um 14:00 Uhr bei Kilometer 200 einer Straße und um 14:30 bei Kilometer 260. Berechnen Sie die Durchschnittsgeschwindigkeit in km/h. War das Auto schneller oder langsamer als 100 km/h?

Lösung, Schritt für Schritt.

Schritt 1: die beiden Größen und ihre Änderungen benennen. Was sich ändert, ist die Position x. Die Achse, längs derer sie sich ändert, ist die Zeit t (dasselbe Muster wie bei

der Kaffeetasse aus Beispiel 1.2, nur räumlich statt thermisch).

$$\Delta x \;=\; x_{\text{neu}} - x_{\text{alt}} \;=\; 260\,\text{km} - 200\,\text{km} \;=\; 60\,\text{km}.$$

$$\Delta t \;=\; t_{\text{neu}} - t_{\text{alt}} \;=\; 14{:}30 - 14{:}00 \;=\; 30\,\text{min} \;=\; 0{,}5\,\text{h}.$$

Wir rechnen die 30 Minuten in 0,5 Stunden um, damit Zähler (km) und Nenner (h) am Ende in der gefragten Einheit km/h zusammenkommen.

Schritt 2: die Rate bilden.

$$\text{Geschwindigkeit} \;=\; \frac{\Delta x}{\Delta t} \;=\; \frac{60\,\text{km}}{0{,}5\,\text{h}} \;=\; 120\,\text{km/h}.$$

Die Arithmetik ist eine Division: $60 \div 0{,}5 = 120$. (Falls Sie das Teilen durch einen Dezimalwert nervös macht: $60/0{,}5 = 60 \cdot (1/0{,}5) = 60 \cdot 2 = 120$.)

Schritt 3: mit der gefragten Schwelle vergleichen. $120 > 100$, also fuhr das Auto *schneller* als 100 km/h.

Plausibilitätsprüfung. Wäre das Auto die 60 km in einer ganzen Stunde gefahren, wäre die Geschwindigkeit 60 km/h gewesen — langsamer als 100. Wir haben dieselbe Strecke in nur einer *halben* Stunde zurückgelegt, also erwarten wir etwa das Doppelte. Genau das kam heraus.

Was diese Übung lehrt. Sie haben jetzt alle drei Zutaten an einem Ort: eine Größe, die sich ändert (x), eine Achse, längs derer sie sich ändert (t), und eine Rate (ihren Quotienten). Die σ_c-Methode später im Buch ist genau dasselbe Muster, angewandt auf eine Größe, die sich ändert (eine beobachtbare Größe O), wenn man an einem Knopf dreht (einem Kontrollparameter σ).

Kapitel 2

Was ist eine Funktion?

In Beispiel 1.2 hatten wir eine Kaffeetasse. Sie kühlte von 80 °C auf 77 °C ab. Eine Zahl (Temperatur) hing von einer anderen (Zeit) ab. Diese kleine Beobachtung ist wichtiger, als sie klingt. Mathematiker haben darauf dreihundert Jahre Analysis gebaut. Sie gaben dem Muster einen Namen — *Funktion* — und eine Schreibweise: $f(x)$. Die dreihundert Jahre sind, woher die Ableitungen kommen; daher kommen die Gipfel; und hier fängt dieses Buch an, nützlich zu werden.

2.1 Die Rezeptsicht auf eine Funktion

Eine *Funktion* f ist eine Regel, die zu einer Eingabe x genau eine Ausgabe zurückgibt, geschrieben $f(x)$. Wir lesen $f(x)$ als „f von x“. Das Wort „genau eine“ arbeitet hart in diesem Satz: eine Funktion muss bei derselben Eingabe stets dieselbe Ausgabe liefern. Eine Regel, die „irgendeine Zahl zwischen 3 und 5“ zurückgibt, ist keine Funktion. „Die heutige Temperatur in Berlin“ ist auch keine — andere Tage, andere Antworten. Eine Funktion ist ein Vertrag zwischen Eingabe und Ausgabe, und der Vertrag hält.

Beispiel 2.1 (Verdoppeln)**.** Die Regel „verdopple die Eingabe“ ist eine Funktion. Wenn wir sie f nennen, dann ist $f(3) = 6$, $f(0) = 0$, $f(-1{,}5) = -3$, $f(100) = 200$. Wir können das Rezept knapp schreiben: $\boxed{f(x) = 2x}$. Was immer hineingeht, das Doppelte kommt heraus.

Beispiel 2.2 (Quadrieren)**.** Die Regel „multipliziere die Eingabe mit sich selbst“ ist eine Funktion. Knapp: $\boxed{f(x) = x \cdot x = x^2}$. Also $f(2) = 4$, $f(3) = 9$ und $f(-2) = 4$ ebenso (weil $(-2) \cdot (-2) = +4$ — zwei Negative multipliziert ergeben ein Positives). Wir verwenden diese Funktion als durchgehendes Beispiel für die nächsten zwei Kapitel.

Manchmal schreiben wir $f\colon x \mapsto x^2$, was nur eine vornehmere Notation für dasselbe Rezept aus Beispiel 2.2 ist. Der Pfeil $\mapsto$ wird „bildet ab auf“ gelesen. Lesen Sie ihn als: „f schickt die Eingabe x auf ihr Quadrat“.

2.2 Die Tabellensicht

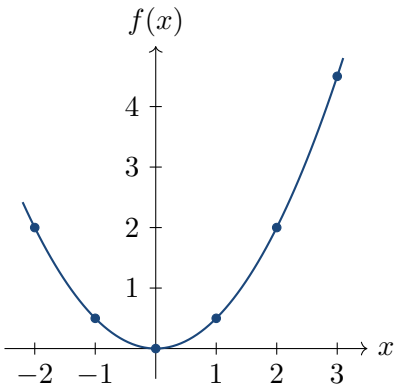
Jede Funktion lässt sich *tabellieren*: wähle einige Eingaben, schreibe jede Eingabe neben ihre Ausgabe. Für die Quadratfunktion aus Beispiel 2.2:

Eingabe x	Ausgabe $f(x) = x^2$
-2	4
-1	1
0	0
1	1
2	4
3	9

Genau diese Art Tabelle erzeugen Sie, wenn Sie in einem echten Experiment einen Kontrollparameter *durchfahren* — ein vornehmes Wort für „lass ihn einen Wertebereich durchlaufen und schreib jedes Mal auf, was du misst“.

2.3 Die Plotsicht

Tragen wir dieselbe Tabelle ein, indem wir jedes Paar $(x, f(x))$ als Punkt in einem Diagramm mit x von links nach rechts und $f(x)$ von unten nach oben zeichnen, erhalten wir einen *Plot*.



(Wir haben hier $f(x) = x^2/2$ statt x^2 geplottet — nur, damit die Abbildung auf eine Seite passt; durch zwei zu teilen staucht die Parabel vertikal. Die Form ist dieselbe: eine U-Kurve, eine *Parabel*.)

2.4 Funktionen im Labor

In einem echten Experiment notieren wir kein Rezept wie „x^2“. Wir *kennen* das Rezept nicht. Wir kennen nur die Tabelle: wir stellen einen Kontrollparameter ein, wir messen eine beobachtbare Größe, wir notieren das Paar. Die Funktion ist implizit durch die Daten gegeben.

Anschauung. Im gesamten Buch ist die Funktion, die uns interessiert, die Antwort auf die Frage: *„Wenn ich den Kontrollparameter auf σ einstelle, welchen Wert hat meine Messung O?“* Wir schreiben das $O(\sigma)$. Das Rezept ist unbekannt. Die Tabelle ist, was wir haben. Alles Weitere wird aus dieser Tabelle berechnet.

2.5 Der Kontrollparameter σ

Für den Kontrollparameter verwenden wir den griechischen Buchstaben σ (sigma). In verschiedenen Domänen bedeutet σ verschiedene Dinge — und das ist der ganze Witz des Rahmens. Fünf echte Beispiele:

- In einem Magnetexperiment ist σ die Temperatur T der Probe.
- Auf einem Quantencomputer ist σ die Stärke γ des injizierten Rauschens.
- Im Finanzwesen ist σ eine Zeitskala, etwa die Anzahl der Tage, über die wir die Volatilität berechnen.
- Bei einem Proteinexperiment ist σ eine Änderung der freien Energie $\Delta\Delta G$ infolge einer Mutation.
- Im Maschinellen Lernen ist σ ein Hyperparameter wie die Lernrate.

Dem Rahmen ist es gleich, welches davon Sie vor sich haben. Das Rezept ist identisch. *Das* meinen wir mit „universell“.

PROBIER ES

Wählen Sie ein messbares Phänomen aus Ihrem Alltag — etwa, wie laut ein Radio klingt, wenn Sie am Lautstärkeregler drehen. Identifizieren Sie: Was ist σ? Was ist O? Stellen Sie sich nun vor, Sie notieren das Paar (σ, O) für zehn verschiedene Knopfstellungen. Skizzieren Sie die Tabelle, die Sie erhalten würden.

Ausgearbeitete Lösung.

Schritt 1: den Kontrollparameter σ identifizieren. Der Kontrollparameter ist das, was man *einstellen* kann. Beim Radio ist das die Stellung des Lautstärkereglers. Wir messen sie als Zahl von 0 (lautlos) bis 10 (maximal), so wie es die meisten Skalen tun.

Schritt 2: die beobachtbare Größe O identifizieren. Die beobachtbare Größe ist das, was man als Antwort *misst*. Bei Schall ist das die Lautstärke, gemessen in Dezibel (dB) mit einer Handy-App oder einem SPL-Messgerät. Dezibel ist eine logarithmische Einheit; das spielt fürs Rezept keine Rolle. Wir notieren einfach die angezeigte Zahl.

Schritt 3: zehn Knopfstellungen wählen. Wir wollen einen gleichmäßigen Sweep über den ganzen Bereich. Am einfachsten: $\sigma = 1, 2, 3, \ldots, 10$. ($\sigma = 0$ lassen wir aus: ein lautloses Radio liefert nur das Hintergrundrauschen des Raums, ein entarteter Punkt.)

Schritt 4: die erwartete Tabelle skizzieren. Ein Radio ist in dB ungefähr linear in der Knopfstellung; jeder Schritt sollte also ähnlich viele Dezibel hinzufügen. Das Hintergrundgeräusch in einem ruhigen Zimmer liegt um die 30 dB:

σ (Knopfstellung)	O (dB)
1	≈ 35
2	≈ 42
3	≈ 50
4	≈ 57
5	≈ 63
6	≈ 70
7	≈ 76
8	≈ 82
9	≈ 88
10	≈ 94

(Ihr echtes Radio wird leicht andere Werte liefern; das ist in Ordnung.)

Schritt 5: was für eine Funktion ist das? Ungefähr linear: jeder Knopfschritt addiert 6–7 dB. Der Plot von O gegen σ wäre also nahezu eine Gerade. *Kein Übergang in diesem System.* Das Rezept aus Kapitel 9 würde hier keinen interessanten σ_c finden, weil es

keinen operationalen „Umschlagpunkt“ gibt. Eine ganz legitime Plausibilitätsprüfung: *ein System ohne Übergang erzeugt korrekterweise auch keinen Gipfel.*

Was diese Übung lehrt. Nicht jedes System hat ein interessantes σ_c. Der Rahmen ist für Systeme gedacht, die mindestens einen qualitativen Regime-Wechsel aufweisen. Eine monoton-lineare Reaktion — wie der Lautstärkeregler — ist das einfachste Negativbeispiel.

Kapitel 3

Steigung: die mittlere Änderungsrate

3.1 Zwei Punkte auf einer Kurve

Wählen Sie aus der Tabelle einer Funktion f zwei beliebige Paare: $(x_1, f(x_1))$ und $(x_2, f(x_2))$. Die *mittlere Änderungsrate* zwischen ihnen ist

$$\text{Steigung} \;=\; \frac{f(x_2) - f(x_1)}{x_2 - x_1}.$$

Die Formel hat dieselbe Gestalt wie die Abkühlrate der Kaffeetasse: *Änderung der Ausgabe geteilt durch Änderung der Eingabe.*

Die Steigung sagt Ihnen: *im Mittel steigt die Ausgabe $f(x)$ pro Einheit, um die x zwischen x_1 und x_2 wächst, um „Steigung“ Einheiten.* Ist die Steigung $+0{,}5$, steigt die Kurve um eine halbe Einheit pro Einheit; ist sie -2, fällt sie um zwei Einheiten pro Einheit.

3.2 Die Steigung einer Geraden

Die einfachsten Funktionen sind die *linearen*, mit der Form

$$f(x) \;=\; a \cdot x + b.$$

Die beiden Buchstaben a und b sind feste Zahlen (die *Koeffizienten*); x ist die Eingabe. „Linear“ heißt: der Plot ist eine Gerade. Der Koeffizient a steuert, wie steil die Gerade ist; der Koeffizient b steuert, wo sie bei $x = 0$ sitzt (weil $f(0) = a \cdot 0 + b = b$). Die Verdoppelungsfunktion aus dem vorigen Kapitel ist die lineare Funktion mit $a = 2$ und $b = 0$.

Für *jede* lineare Funktion ist die Steigung gleich, egal welche zwei Punkte Sie wählen: sie ist immer die Zahl a. Sie können das einmal nachrechnen und dann für immer darauf vertrauen:

$$\frac{(a \cdot x_2 + b) - (a \cdot x_1 + b)}{x_2 - x_1} \;=\; \frac{a(x_2 - x_1)}{x_2 - x_1} \;=\; a.$$

Die b's heben sich auf; der Faktor $a(x_2 - x_1)$ im Zähler kürzt sich gegen $(x_2 - x_1)$ im Nenner heraus, übrig bleibt a. *Auf einer Geraden überall dieselbe Steigung.* Im Nachhinein offensichtlich; einmal in der Algebra sehenswert.

Beispiel 3.1 (Eimer unter dem Wasserhahn)**.** Ein Wasserhahn füllt einen Eimer nach t Minuten auf $f(t) = 2t + 3$ Liter. Die Konstante $b = 3$ ist die Wassermenge, die zu Beginn ($t = 0$) schon im Eimer ist. Der Koeffizient $a = 2$ ist die *Flussrate*: pro Minute kommen 2 Liter dazu. Prüfung, dass a wirklich die Steigung ist: zwei Zeitpunkte wählen. Bei $t_1 = 1$: $f = 5$. Bei $t_2 = 4$: $f = 11$. Steigung $= (11 - 5)/(4 - 1) = 6/3 = 2$. Wie angekündigt.

3.3 Die Steigung einer gekrümmten Kurve — sie hängt davon ab, wo man hinschaut

Betrachten Sie nun $f(x) = x^2$. Zwei Paare:

- Zwischen $x_1 = 1$ und $x_2 = 2$: Steigung $= (4 - 1)/(2 - 1) = 3$.
- Zwischen $x_1 = 3$ und $x_2 = 4$: Steigung $= (16 - 9)/(4 - 3) = 7$.

Die Steigung einer gekrümmten Kurve ist nicht konstant. Sie hängt davon ab, *wo* auf der Kurve man misst. Wo die Kurve steil steigt (rechts an der Parabel), ist die Steigung groß. Wo sie flach ist (am Boden), ist sie nahe null. Wo sie fällt (links), ist sie negativ.

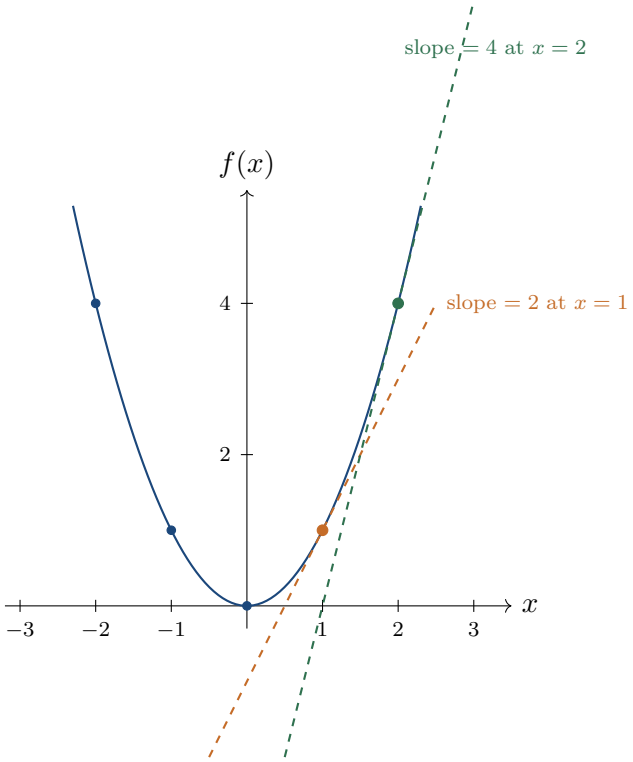


Abbildung 3.1: Die Steigung von $f(x) = x^2$ ist keine einzige Zahl; sie ist an jedem Punkt eine andere. Bei $x = 1$ ist sie 2; bei $x = 2$ ist sie 4. Mit wachsendem x wird die Kurve steiler. Die beiden gestrichelten Linien sind *Tangenten*: Geraden, die die Kurve in genau einem Punkt berühren und dort ihre Steigung teilen.

MERKE

Die Steigung einer Funktion ist eine *lokale* Eigenschaft. Eine gekrümmte Funktion mit einer einzigen Zahl zu beschreiben, geht nicht — man braucht an jeder Stelle einen eigenen Steigungswert. Genau das liefert uns die Ableitung, die im nächsten Kapitel definiert wird.

3.4 Die Steigung ist selbst eine Funktion

Schauen Sie sich die Parabel $f(x) = x^2$ an. Wählen Sie einen Basispunkt x und einen kleinen Schritt $h > 0$. Die Steigung zwischen x und $x + h$ ist

$$\frac{(x+h)^2 - x^2}{h} \;=\; \frac{x^2 + 2xh + h^2 - x^2}{h} \;=\; \frac{2xh + h^2}{h} \;=\; 2x + h.$$

(Wenn Sie noch nicht wissen, warum $(x+h)^2 = x^2 + 2xh + h^2$ ist, multiplizieren Sie es selbst aus: $(x+h)(x+h) = x \cdot x + x \cdot h + h \cdot x + h \cdot h$.)

Stellen Sie sich nun vor, h schrumpft gegen null — was wir im nächsten Kapitel präzise machen —, dann strebt die Steigung gegen $2x$ am Basispunkt. Bei $x = 1$ ist die Steigung 2; bei $x = 3$ ist sie 6; bei $x = 4$ ist sie 8. *Die Steigung ist selbst eine Funktion von x.* Wir nennen sie die *Ableitung* von f.

(Diese kleine Rechnung habe ich an einem Märzmorgen 2026 auf einem Bahnsteig in Erlangen auf der Rückseite eines Kaffeebons gemacht, weil ich mein Notizbuch zu Hause gelassen hatte. Der Bon liegt irgendwo in einer Schublade; das Ergebnis ist immer noch $2x$.)

PROBIER ES

Berechnen Sie die Steigung von $f(x) = x^2$ zwischen $x_1 = 1{,}99$ und $x_2 = 2{,}01$. Sehen Sie etwas in der Nähe der Formel $2x = 4$ bei $x = 2$?

Lösung, Schritt für Schritt.

Schritt 1: die beiden Punktpaare auf der Kurve identifizieren. Wir brauchen $(x_1, f(x_1))$ und $(x_2, f(x_2))$. Die Eingaben sind gegeben: $x_1 = 1{,}99$ und $x_2 = 2{,}01$. Die Ausgaben kommen aus dem Funktionsrezept $f(x) = x \cdot x$. Also:

$$f(x_1) \;=\; f(1{,}99) \;=\; 1{,}99 \cdot 1{,}99.$$

Schritt 2: $f(1{,}99) = 1{,}99 \cdot 1{,}99$ auf Papier ausrechnen. Das ist die Art Quadrierung, die die meisten ohne nachzudenken machen; rechnen wir es einmal transparent. Zwei saubere Wege.

Weg 1 (schriftliche Multiplikation). Zuerst die Kommas weglassen; eines setzen wir am Ende wieder. Wir rechnen 199×199:

199	
× 199	
1791	(199 × 9)
17910	(199 × 90, eine Stelle versetzt)
19900	(199 × 100, zwei Stellen versetzt)
39601	(die Summe)

Jeder Faktor 1,99 hat zwei Nachkommastellen, das Ergebnis braucht also vier. Komma vier Stellen von rechts setzen: $39601 \to 3{,}9601$. Damit ist $1{,}99 \cdot 1{,}99 = 3{,}9601$.

Weg 2 (algebraischer Trick, schneller). Verwenden Sie die Identität $(a-b)^2 = a^2 - 2ab + b^2$ mit $a = 2$ und $b = 0{,}01$:

$$1{,}99^2 \;=\; (2 - 0{,}01)^2 \;=\; 4 - 2 \cdot 2 \cdot 0{,}01 + 0{,}01^2 \;=\; 4 - 0{,}04 + 0{,}0001 \;=\; 3{,}9601.$$

Dasselbe Ergebnis, viel weniger Arithmetik. Diese Abkürzung benutzen wir auch für den anderen Punkt.

Schritt 3: $f(2{,}01) = 2{,}01 \cdot 2{,}01$ ausrechnen. Jetzt mit $a = 2$, $b = 0{,}01$ und $(a+b)^2 = a^2 + 2ab + b^2$:

$$2{,}01^2 \;=\; (2 + 0{,}01)^2 \;=\; 4 + 2 \cdot 2 \cdot 0{,}01 + 0{,}01^2 \;=\; 4 + 0{,}04 + 0{,}0001 \;=\; 4{,}0401.$$

Schritt 4: die Steigungsformel anwenden. Die Steigung zwischen $(x_1, f(x_1)) = (1{,}99, 3{,}9601)$ und $(x_2, f(x_2)) = (2{,}01, 4{,}0401)$ ist

$$\text{Steigung} = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{4{,}0401 - 3{,}9601}{2{,}01 - 1{,}99} = \frac{0{,}0800}{0{,}02}.$$

Die Division: $0{,}0800/0{,}02 = 800/200 = 4{,}00$ (Zähler und Nenner mit 10 000 multiplizieren, um Dezimalen loszuwerden, dann kürzen).

Schritt 5: mit der Formelvorhersage vergleichen. Weiter oben haben wir hergeleitet, dass die Steigung von $f(x) = x^2$ am Punkt x gleich $f'(x) = 2x$ ist. Bei $x = 2$ ergibt das $2 \cdot 2 = 4$. Unsere numerische Schätzung ist 4,00. Übereinstimmung auf vier signifikante Stellen.

Warum so genau? Erinnern Sie sich an die algebraische Antwort: die Steigung zwischen x und $x + h$ ist für $f = x^2$ gleich $2x + h$. Hier haben wir $x = 2$ symmetrisch umrahmt, also sollte die Steigungsformel am Mittelpunkt genau $2x$ ergeben — die h-Beiträge heben sich zwischen linker und rechter Seite auf. Genau das ist passiert: wir bekamen 4,00, nicht 4,02 oder 3,98. Genau deshalb verwendet der Rahmen *zentrale Differenzen* (x_{i+1} gegen x_{i-1}, x_i in der Mitte), nicht Vorwärts- oder Rückwärtsdifferenzen.

Was diese Übung lehrt. Drei Dinge: (i) Die Steigungsformel liefert auf einer glatten Kurve eine Zahl, die beliebig nahe an die wahre Ableitung kommt, sobald die beiden Punkte näher beieinander liegen; (ii) symmetrische (zentrale) Stichprobennahme löscht den Fehler erster Ordnung; (iii) man kann eine Ableitungsformel an einem einzelnen Punkt mit zwei Rechenoperationen und einer Division verifizieren — was bedeutet, dass man eine Ableitung auch aus Daten *entdecken* kann, ohne die Formel zu kennen. Genau das tun wir im ganzen Buch.

Kapitel 4

Die Ableitung: die Steigung an einem einzelnen Punkt

4.1 Die Idee

An jedem Punkt x hätten wir gerne eine einzige Zahl, die uns sagt, wie schnell sich f dort ändert. Das Rezept ist uns schon bekannt: man berechne die Steigung zwischen x und $x + h$ und lasse anschließend h *schrumpfen*.

Definition 4.1 (Ableitung)**.** Die *Ableitung* von f am Punkt x, geschrieben $f'(x)$ oder $\frac{df}{dx}$, ist der Grenzwert

$$f'(x) \;=\; \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

sofern dieser Grenzwert existiert.

Das Symbol $\lim_{h \to 0}$ „Grenzwert für h gegen null“ liest sich als: *der Wert, dem sich die Steigung annähert, wenn die Schrittweite beliebig klein wird.* Wir teilen nicht wirklich durch null. Wir sehen nur nach, wohin sich die Steigung einpendelt, wenn h winzig wird.

4.2 Drei Ableitungen, die Sie auf einen Blick kennen sollten

Konstante: Ist $f(x) = c$ konstant, so ist die Steigung stets null, also $f'(x) = 0$. Eine flache Funktion hat keine Änderungsrate.

Gerade: Ist $f(x) = a \cdot x + b$, so ist die Steigung überall a, also $f'(x) = a$.

Parabel: Für $f(x) = x^2$ haben wir oben $f'(x) = 2x$ ausgerechnet.
Für unsere Zwecke genügen diese drei Muster. Mathematiker haben die Ableitung praktisch jeder klassischen Funktion tabelliert (Exponentialfunktion, Sinus, Logarithmus etc.), aber wir brauchen davon nichts. Warum? Weil wir im Labor nie eine saubere Formel haben; wir haben nur eine Tabelle.

4.3 Die Ableitung, wenn man nur eine Tabelle hat

In einem echten Experiment ist f eine Liste gemessener Paare:

$$(\sigma_1, O_1),\ (\sigma_2, O_2),\ \ldots,\ (\sigma_n, O_n).$$

Die σ_i sind die Kontrollwerte, die wir eingestellt haben; die O_i ist das Gemessene. Die Ableitung am mittleren Punkt eines Tripels $(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ wird durch die *zentrale Differenz* approximiert:

$$O'(\sigma_i) \;\approx\; \frac{O_{i+1} - O_{i-1}}{\sigma_{i+1} - \sigma_{i-1}}.$$

Das ist genau die Steigungsformel, angewandt zwischen den beiden Nachbarn von σ_i.

Beispiel 4.2. Sie haben die Magnetisierung eines Magneten bei drei Temperaturen gemessen:

T (K)	M
2,20	0,55
2,27	0,41
2,34	0,18

Die zentrale Differenz für dM/dT bei $T = 2{,}27$ ist

$$\frac{0{,}18 - 0{,}55}{2{,}34 - 2{,}20} \;=\; \frac{-0{,}37}{0{,}14} \;=\; -2{,}64 \;\; \text{[Einheiten von } M \text{ pro K]}.$$

Vorzeichen: negativ, weil M fällt. *Betrag:* groß, weil M in diesem Temperaturbereich schnell abfällt. Genau dieses „großer Betrag, dort wo die Funktion schnell fällt“ ist das Signal, das die σ_c-Methode sucht.

4.4 Einseitige Differenzen am Rand

Am ersten ($i = 1$) und am letzten ($i = n$) Punkt der Tabelle können wir keine zentrale Differenz bilden, weil ein Nachbar fehlt. Wir benutzen:

$$O'(\sigma_1) \approx \frac{O_2 - O_1}{\sigma_2 - \sigma_1}, \qquad O'(\sigma_n) \approx \frac{O_n - O_{n-1}}{\sigma_n - \sigma_{n-1}}.$$

Das sind *Vorwärts-* und *Rückwärtsdifferenzen*. Sie sind etwas weniger genau als die zentrale Differenz, genügen aber am Rand.

4.5 Im Code: numerische Ableitung in drei Zeilen

NumPy bringt das fertig mit:

```
import numpy as np
sigma = np.array([2.20, 2.27, 2.34])
O     = np.array([0.55, 0.41, 0.18])
dO_dsigma = np.gradient(O, sigma)
print(dO_dsigma)   # [-2.0, -2.64..., -3.29...]
```

`np.gradient` verwendet am Rand einseitige Differenzen und in der Mitte zentrale Differenzen — genau wie eben definiert.

MERKE

Die Ableitung ist nur eine Steigung, lokal angewandt. Mit drei Zeilen NumPy verwandeln Sie jede Tabelle (σ_i, O_i) in eine neue Tabelle $(\sigma_i, O'(\sigma_i))$. Den Rest des Buches verbringen wir damit, die Stelle zu finden, an der diese Ableitung am größten ist.

PROBIER ES

Nehmen Sie irgendeine Funktion, die Ihnen einfällt — den Füllstand einer Tasse, während Sie sie auskippen, den Kurs einer Aktie der letzten zehn Tage, die Temperatur eines Raumes über einen Abend hinweg. Notieren Sie fünf Punkte mit der Hand. Berechnen Sie an den drei mittleren die zentralen Differenzen. Welcher Punkt hat den größten Steigungsbetrag?

Ausgearbeitete Lösung (mit einem Aktienkurs-Beispiel über zehn Tage; das Rezept ist für jede andere Wahl identisch).

Schritt 1: fünf (σ_i, O_i)-Paare notieren. Wir wählen fünf gleich beabstandete Tage einer hypothetischen Aktie, die ihre Handelswoche mit einem steilen Absturz beendet hat. Die σ-Achse ist die Tagesnummer; die O-Achse ist der Schlusskurs in Dollar.

Tag σ_i	Kurs O_i \$
1	100,0
2	99,5
3	98,0
4	90,5
5	85,0

Wir haben dieses Beispiel gewählt, weil der steilste Abfall offenbar zwischen Tag 3 und Tag 4 liegt — ein sichtbarer Regimewechsel. Unsere Aufgabe ist, das auch *numerisch* sichtbar zu machen.

Schritt 2: welche Punkte sind „die drei mittleren“? Zentrale Differenzen brauchen einen Nachbarn auf jeder Seite. Bei fünf Punkten $1, 2, 3, 4, 5$ sind die inneren Punkte $2, 3, 4$. Die Punkte 1 und 5 sind die Ränder und bräuchten einseitige Differenzen (Abschnitt 4.4).

Schritt 3: an jedem inneren Punkt die zentrale Differenz anwenden. Die Formel lautet $O'(\sigma_i) \approx (O_{i+1} - O_{i-1})/(\sigma_{i+1} - \sigma_{i-1})$. Hier ist $\sigma_{i+1} - \sigma_{i-1} = 2$ Tage an jedem inneren Punkt, weil das Gitter gleichmäßig mit Schrittweite 1 ist.

$$O'(2) \approx \frac{O_3 - O_1}{\sigma_3 - \sigma_1} = \frac{98{,}0 - 100{,}0}{3 - 1} = \frac{-2{,}0}{2} = -1{,}0 \text{ \$/Tag},$$
$$O'(3) \approx \frac{O_4 - O_2}{\sigma_4 - \sigma_2} = \frac{90{,}5 - 99{,}5}{4 - 2} = \frac{-9{,}0}{2} = -4{,}5 \text{ \$/Tag},$$
$$O'(4) \approx \frac{O_5 - O_3}{\sigma_5 - \sigma_3} = \frac{85{,}0 - 98{,}0}{5 - 3} = \frac{-13{,}0}{2} = -6{,}5 \text{ \$/Tag}.$$

Alle drei Steigungen sind negativ, weil der Kurs fällt.

Schritt 4: Beträge bilden, um Größenordnungen zu vergleichen. Die Frage war „welcher hat den größten Steigungsbetrag“. Das Rahmenrezept benutzt aus genau diesem Grund auch Beträge.

$$|O'(2)| = 1{,}0, \quad |O'(3)| = 4{,}5, \quad |O'(4)| = 6{,}5.$$

Schritt 5: Antwort ablesen. Der größte Steigungsbetrag liegt am Tag 4, $|O'(4)| = 6{,}5$ \$/Tag. Das ist der σ_c-Kandidat dieser kleinen Tabelle: der Moment der steilsten Änderung. *Beachten Sie*, dass der steilste Abfall eigentlich zwischen Tag 3 und Tag 4 lag — die zentrale Differenz an Tag 4 „spürt“ diesen Abfall am stärksten, weil sowohl Tag 3 (noch hoch bei \$98) als auch Tag 5 (sehr tief bei \$85) in ihrem Fenster liegen.

Was diese Übung lehrt.

- Zentrale Differenzen verteilen die Wirkung jedes Schrittes über *zwei* innere Punkte, nicht nur einen. Deshalb findet der Rahmen einen Gipfel statt einer einzelnen gezackten Stufe.

- Schon mit fünf Punkten und je einer Division pro Punkt hat die Methode den Regimewechsel in den Daten lokalisiert.
- Dieselbe Arithmetik funktioniert für Füllstand gegen Zeit, Temperatur gegen Zeit oder jede andere „eine-Größe-variiert-mit-einer-anderen“-Situation.

Kapitel 5

Der Gipfel: wo die Steigung am größten ist

5.1 Betrag: die Richtung ignorieren

In Beispiel 4.2 war die Ableitung negativ ($-2{,}64$). Für die σ_c-Methode ist es uns meist egal, *in welche Richtung* sich die Funktion bewegt; uns interessiert *wie schnell*. Das Standardwerkzeug für „Vorzeichen weg, Größe behalten“ ist der *Betrag*, geschrieben $|x|$:

$$|x| \;=\; \begin{cases} +x & \text{falls } x \geq 0, \\ -x & \text{falls } x < 0. \end{cases}$$

Also $|3| = 3$, $|-2{,}64| = 2{,}64$, $|0| = 0$.

Anschauung. Eine Funktion mit Steigung $+2$ an einer Stelle und -2 an einer anderen ändert sich an beiden Stellen „gleich schnell“, nur in entgegengesetzte Richtungen. Die σ_c-Methode sucht die Stelle, wo die Änderung in Größe am größten ist. Wir arbeiten also stets mit $|O'(\sigma)|$, dem Steigungsbetrag.

5.2 Argmax: der Ort des Maximums

Angenommen, Sie haben eine Tabelle der Werte $|O'(\sigma_1)|$, $|O'(\sigma_2)|, \ldots$ und fragen: welche Zeile hat den größten Wert? Die σ-Koordinate dieser Zeile heißt das *argmax* der Funktion:

$$\sigma_c \;=\; \arg\max_{\sigma} \; |O'(\sigma)|.$$

Gelesen: „σ_c ist derjenige σ-Wert, der $|O'(\sigma)|$ am größten macht.“

Beispiel 5.1. Angenommen

σ	O	$\lvert O'\rvert$
1,00	0,90	0,20
1,25	0,85	0,40
1,50	0,70	1,60
1,75	0,35	1,20
2,00	0,20	0,30

Der größte Steigungsbetrag ist 1,60, bei $\sigma = 1{,}50$. Also $\sigma_c = 1{,}50$. Das ist die Stelle, an der sich die beobachtbare Größe am schnellsten ändert — der Kandidat für die kritische Skala des Systems.

5.3 Im Code: argmax in einer Zeile

```
sigma_c = sigma[np.argmax(np.abs(dO_dsigma))]
```

Dieser eine Befehl ist, im Kern, was der ganze σ_c-Rahmen tut — nur mit einer Menge Verfeinerungen, die die restlichen Kapitel erklären.

5.4 Warum das überhaupt interessant ist

In jeder Domäne, die wir uns ansehen werden, erweist sich die Stelle, an der sich eine beobachtbare Größe am schnellsten ändert, als bedeutsame physikalische oder operationale Schwelle. Ein paar Vorschauen:

- In einem Magneten liegt der größte Steigungsbetrag der Magnetisierung gegen die Temperatur am *Curie-Punkt* — der Temperatur, oberhalb derer das Material seinen Dauermagnetismus verliert.
- Auf einem Quantenprozessor liegt der größte Steigungsbetrag der Verschränkung gegen das Rauschen an der *operationalen Dekohärenzschwelle* — dem Rauschpegel, oberhalb dessen Quanteninformation den Schaltkreis nicht überlebt.
- In einem Markt liegt der größte Steigungsbetrag der Korrelation gegen die zeitliche Verzögerung am *Regimewechsel-Horizont*.
- In einem Protein liefert der größte Steigungsbetrag der Faltungsstabilität gegen die mutationsbedingte freie Energie die *Toleranzschwelle*, oberhalb derer das Protein amyloidogen wird.

Zwölf solche Beispiele arbeiten wir in Teil IV im Detail aus. Das Muster ist immer dasselbe: *einen Knopf durchfahren, messen, eine Steigung bilden, den Gipfel finden.*

PROBIER ES

Berechnen Sie für die folgenden (σ, O)-Daten $|O'|$ mit `np.gradient` und lesen Sie σ_c ab:

$$\sigma = [0, 1, 2, 3, 4, 5], \qquad O = [1{,}00, 0{,}98, 0{,}93, 0{,}60, 0{,}20, 0{,}15].$$

Lösung, Schritt für Schritt.

Schritt 1: vor jedem Rechnen die Daten ansehen. O bleibt von $\sigma = 0$ bis $\sigma = 2$ nahe 1,0, fällt zwischen $\sigma = 2$ und $\sigma = 4$ scharf, pendelt sich dann um 0,15 ein. Mit dem Auge: der steilste Abfall liegt in der Mitte. Die Aufgabe von $|O'|$ ist, dieses „mit dem Auge“ in eine Zahl zu verwandeln.

Schritt 2: zentrale Differenzen an den inneren Punkten. Bei gleichmäßiger Schrittweite $\Delta\sigma = 1$ zwischen aufeinanderfolgenden Gitterpunkten ist der Nenner der zentralen Differenz überall $\sigma_{i+1} - \sigma_{i-1} = 2$:

$$\begin{aligned} O'(1) &\approx \frac{O(2) - O(0)}{2} = \frac{0{,}93 - 1{,}00}{2} = -0{,}035, \\ O'(2) &\approx \frac{O(3) - O(1)}{2} = \frac{0{,}60 - 0{,}98}{2} = -0{,}190, \\ O'(3) &\approx \frac{O(4) - O(2)}{2} = \frac{0{,}20 - 0{,}93}{2} = -0{,}365, \\ O'(4) &\approx \frac{O(5) - O(3)}{2} = \frac{0{,}15 - 0{,}60}{2} = -0{,}225. \end{aligned}$$

Schritt 3: Ränder (einseitige Differenzen). Bei $\sigma = 0$ und $\sigma = 5$ fehlt ein Nachbar, daher Vorwärts- bzw. Rückwärtsdifferenz:

$$O'(0) \approx \frac{O(1) - O(0)}{1 - 0} = \frac{0{,}98 - 1{,}00}{1} = -0{,}02,$$
$$O'(5) \approx \frac{O(5) - O(4)}{5 - 4} = \frac{0{,}15 - 0{,}20}{1} = -0{,}05.$$

Schritt 4: Beträge bilden und argmax suchen.

$$|O'(\sigma)| = [0{,}02,\ 0{,}035,\ 0{,}190,\ \mathbf{0{,}365},\ 0{,}225,\ 0{,}05]$$

Der größte Eintrag ist 0,365 bei $\sigma = 3$. Also $\sigma_c = 3$.

Schritt 5: mit einer Zeile NumPy verifizieren.

```
import numpy as np
sigma = np.array([0, 1, 2, 3, 4, 5])
O     = np.array([1.00, 0.98, 0.93, 0.60, 0.20, 0.15])
chi   = np.abs(np.gradient(O, sigma))
sigma_c = sigma[np.argmax(chi)]
print(chi)         # [0.02, 0.035, 0.19, 0.365, 0.225, 0.05]
print(sigma_c)     # 3
```

Die Gipfelschärfe κ ausrechnen.

$$\bar{\chi} = \frac{0{,}02 + 0{,}035 + 0{,}19 + 0{,}365 + 0{,}225 + 0{,}05}{6} = \frac{0{,}885}{6} \approx 0{,}148,$$

$$\kappa = \chi_{\max}/\bar{\chi} = 0{,}365/0{,}148 \approx 2{,}5.$$

Nach den Schwellen aus Kapitel 12 ist das grenzwertig ($1{,}5 \le \kappa < 3$). Bei echten Daten würden wir, bevor wir $\sigma_c = 3$ selbstbewusst nennen, einen Permutationstest ergänzen.

Was diese Übung lehrt. Das ganze Rezept — durchfahren, messen, differenzieren, Gipfel — in fünf Zeilen Arithmetik. Sie brauchten keinen Computer, um σ_c zu finden; der Computer macht es nur unermüdlich.

Kapitel 6

Daten sind verrauscht: glätten

6.1 Warum rohe Ableitungen sich danebenbenehmen

In jedem echten Experiment liefern zwei Messungen bei demselben σ leicht unterschiedliche O-Werte. Das ist *Rauschen*: Schuss-zu-Schuss-Fluktuation, Sensor-Jitter, Endliche-Statistik-Varianz. Rauschen hat einen unangenehmen Effekt auf Ableitungen.

Stellen Sie sich vor, die wahre zugrundeliegende Funktion ist glatt, aber jede Messung wird von einem kleinen zufälligen Schubs verunreinigt. Ein kleiner Schubs auf O bleibt klein — aber wenn man ableitet, teilt man durch ein kleines $\Delta\sigma$, und das Rauschen wird *verstärkt*. Die naive Ableitung verrauschter Daten sieht aus wie Gras auf einem Rasen: zackig, sprunghaft, verbirgt die wahre Struktur.

6.2 Glätten: jeden Wert durch einen lokalen Mittelwert ersetzen

Die Abhilfe: *glätten* Sie die O-Werte, bevor Sie ableiten. Der einfachste Glätter ist der *gleitende Mittelwert*: ersetze jedes O_i durch den Mittelwert aus sich selbst und seinen Nachbarn.

Beispiel 6.1. Rohes O: [0,50, 0,60, 0,40, 0,30, 0,45].
Der dreipunktige gleitende Mittelwert am Index i ersetzt O_i durch $(O_{i-1} + O_i + O_{i+1})/3$. Am Index 2 (von 1 gezählt): $(0{,}50 + 0{,}60 + 0{,}40)/3 = 0{,}50$.
Geglättetes O (innen): [?, 0,50, 0,43, 0,38, ?]. Die Ränder, ohne Nachbar, behalten den rohen Wert.

6.3 Gauß-Glättung: ein schlauerer gewichteter Mittelwert

Der gleitende Mittelwert behandelt alle Nachbarn gleich. Ein *Gauß*-Glätter gewichtet nahe Punkte stärker als ferne — nach einer Glockenkurve. Die Glockenform wird vom Parameter σ_{ker} beschrieben, der *Kernbreite*, nicht zu verwechseln mit σ, dem Kontrollparameter; wir hängen den Index „ker“ an, um Verwechslungen zu vermeiden.

Definition 6.2 (Gauß-Glätter)**.** Für eine Folge $O_1, \ldots, O_n$ ist der geglättete Wert am Index i

$$\tilde{O}_i \;=\; \sum_{j=1}^{n} w_{ij}\, O_j, \qquad w_{ij} \;=\; \frac{\exp\!\left(-\frac{(i-j)^2}{2\sigma_{\text{ker}}^2}\right)}{\sum_{k=1}^{n} \exp\!\left(-\frac{(i-k)^2}{2\sigma_{\text{ker}}^2}\right)}.$$

Der Nenner zwingt die Gewichte, sich zu eins aufzusummieren; der Zähler ist die Glockenkurve, am schmalsten bei kleinem σ_{ker}.

Sie müssen diese Gewichte nicht selbst ausrechnen. SciPy liefert den Gauß-Glätter in einer Zeile. Die Standard-Kernbreite des Rahmens ist $\sigma_{\text{ker}} = 0{,}6$ — gerade genug, um einzelne Spitzen zu beseitigen, aber nicht so viel, dass es echte Gipfel verschmiert.

```
from scipy.ndimage import gaussian_filter1d
O_smooth = gaussian_filter1d(O, sigma=0.6)
```

Einheiten von σ_{ker}: Indexraum, nicht σ-Raum

Dies ist der am meisten missverstandene Parameter des Rahmens. Das `sigma`-Argument von `gaussian_filter1d`, also unser σ_{ker}, wird in *Array-Indizes* gemessen, nicht in den Einheiten Ihres Kontrollparameters σ. „0,6“ heißt also „0,6 Indexpositionen“, ungefähr ein Nachbar auf jeder Seite. Nicht „0,6 K“, nicht „0,6 Qubits“, und nicht „0,6 von dem, was gerade auf der x-Achse steht“.

STOLPERFALLE

Unregelmäßige Gitter zerstören das. Sind Ihre σ_i nicht gleichmäßig verteilt — etwa logarithmisch verteilte Lernraten, adaptive Stichprobennahme nahe einem vermuteten Übergang oder eine Renditen-Reihe mit Lücken —, so übersetzt sich „Kernbreite 0,6 Indizes“ in verschiedenen Teilen der Messreihe zu unterschiedlichen σ-Breiten. Der Rahmen mittelt dann still über unterschiedliche physikalische Skalen.
Zwei sichere Wege.

- *Auf ein gleichmäßiges σ-Gitter umtasten* vor dem Glätten (`numpy.interp`); O entsprechend mit umstasten.
- *Savitzky–Golay mit explizitem physikalischen Fenster* via `sigma_c.core.derivatives.savitzky_golay_derivative`. Sie übergeben Fensterlänge und Polynomgrad; den Schrittabstand berechnet der Rahmen aus Ihren echten σ_i.

Wann der Standardwert 0,6 vernünftig ist. Indexraum-„0,6“ ist in Ordnung, wenn die Messreihe ungefähr gleichmäßig ist und der Übergang mindestens drei bis fünf Gitterpunkte überspannt. Für Reihen unter 20 Punkten verwenden Sie 0,3; für Reihen über 100 Punkten mögen Sie 1,0 oder größer, prüfen Sie mit dem Kern-Sweep-Rezept aus Kapitel 45, dass σ_c stabil ist.

6.4 Die geglättete Pipeline

Jetzt haben wir das vollständige Rezept in fünf Zeilen:

```
import numpy as np
from scipy.ndimage import gaussian_filter1d

O_smooth = gaussian_filter1d(O, sigma=0.6)
chi      = np.abs(np.gradient(O_smooth, sigma))
sigma_c  = sigma[np.argmax(chi)]
kappa    = chi.max() / chi.mean()
```

Die dritte Zeile: glätten. Die vierte: absolute Ableitung. Die fünfte: den Gipfel lokalisieren. Die sechste: ein Maß für die Schärfe des Gipfels (genannt κ, im nächsten Kapitel definiert). *Das ist die σ_c-Methode in voller Gestalt.*

6.5 Wie viel glätten?

Zu wenig lässt Spitzen stehen; zu viel wäscht echte Gipfel heraus. Der Standardwert $\sigma_{\text{ker}} = 0{,}6$ ist konservativ. Zwei Faustregeln:

- Hat Ihr Datensatz nur $n < 20$ Punkte, hilft Glättung kaum; nehmen Sie einen kleineren Kern ($\sigma_{\text{ker}} = 0{,}3$) oder lassen Sie sie weg.
- Hat Ihr Datensatz Hunderte von Punkten, probieren Sie σ_{ker} zwischen 0,5 und 2,0 und prüfen, dass σ_c sich kaum bewegt. Diese Stabilität ist selbst schon eine Plausibilitätsprüfung.

STOLPERFALLE

Verwenden Sie nie einen Gauß-Filter, dessen Breite mit der Breite des gesuchten Gipfels vergleichbar ist. Sie verschmieren ihn weg. Wenn Sie einen scharfen Gipfel vermuten, nutzen Sie statt dessen die Savitzky–Golay-Ableitung (`sigma_c.core.derivatives.savitzky_golay_derivative`).

Savitzky–Golay in einem Absatz. Der Gauß-Glätter passt einen impliziten gewichteten Mittelwert durch jedes Fenster. Der *Savitzky–Golay*-Glätter (Savitzky und Golay, 1964) macht etwas Durchsichtigeres: an jedem Punkt i legt er per gewöhnlicher kleinster Quadrate ein Polynom festen Grades p (meist 2 oder 3) durch die $2k + 1$ nächsten Datenpunkte und gibt den Wert dieses Polynoms an Stelle i aus. Zum Ableiten gibt er die analytische Ableitung des gefitteten Polynoms aus statt erneut numerisch zu differenzieren. Das Ergebnis ist eine Ableitungsschätzung, die für jedes Polynomsignal vom Grad $\leq p$ innerhalb des Fensters exakt ist und sehr rauschrobust. Der entscheidende Parameter ist die Fensterlänge, beim Rahmen-Wrapper ausgedrückt in *physikalischen σ-Einheiten* (nicht Indizes), was auch auf unregelmäßigen Gittern korrekt funktioniert. Standardwerte des Rahmens: Fensterlänge 11, Ordnung 3.

PROBIER ES

Fügen Sie den Daten aus der Übung des vorigen Kapitels zufälliges Rauschen mit Standardabweichung 0,02 hinzu (`np.random.normal(0, 0.02, size=6)`). Lassen Sie das Fünfzeilenrezept einmal ohne und einmal mit Glättung laufen. Verschiebt sich σ_c? Um wie viel?

Ausgearbeitete Lösung.

Schritt 1: die Daten wieder herstellen. Die Originaldaten aus der Übung zu Kapitel 5 waren $\sigma = [0, 1, 2, 3, 4, 5]$, $O = [1{,}00, 0{,}98, 0{,}93, 0{,}60, 0{,}20, 0{,}15]$ und ergaben sauber $\sigma_c = 3$ mit $\kappa \approx 2{,}5$.

Schritt 2: Gauß-Rauschen mit std $= 0{,}02$ *addieren.* Eine einzelne Zufallsziehung kann etwa

$$\epsilon = [+0{,}018,\ -0{,}009,\ +0{,}030,\ -0{,}011,\ +0{,}022,\ -0{,}005]$$

liefern; das verrauschte O ist dann

$$\tilde{O} = [1{,}018,\ 0{,}971,\ 0{,}960,\ 0{,}589,\ 0{,}222,\ 0{,}145].$$

(Ihre eigene Ziehung wird abweichen; das ist der ganze Sinn der Übung.)

Schritt 3: das Rezept ohne Glättung laufen lassen.

```
import numpy as np
```

```
rng = np.random.default_rng(0)          # Seed fuer
    Reproduzierbarkeit
sigma = np.array([0, 1, 2, 3, 4, 5])
O     = np.array([1.00, 0.98, 0.93, 0.60, 0.20, 0.15])
O_noisy = O + rng.normal(0, 0.02, size=6)

# OHNE Glaettung
chi_raw = np.abs(np.gradient(O_noisy, sigma))
print(sigma[np.argmax(chi_raw)])        # in der Regel immer noch
    3
print(f"kappa_raw = {chi_raw.max()/chi_raw.mean():.2f}")
```

Für die meisten Seeds ist das Rauschen mit $\sigma = 0{,}02$ klein gegenüber dem dominierenden Abfall von $\sim 0{,}4$ zwischen $\sigma = 2$ und $\sigma = 4$. Das Rezept wählt *weiter* $\sigma_c = 3$, κ fällt aber leicht ($\sim 2{,}0$–$2{,}3$ statt $2{,}5$), weil das Rauschen $\bar{\chi}$ aufbläht.

Schritt 4: das Rezept mit Glättung laufen lassen.

```
from scipy.ndimage import gaussian_filter1d
O_smooth = gaussian_filter1d(O_noisy, sigma=0.6)
chi_sm = np.abs(np.gradient(O_smooth, sigma))
print(sigma[np.argmax(chi_sm)])         # 3
print(f"kappa_sm = {chi_sm.max()/chi_sm.mean():.2f}")
```

Mit Glättung holt sich κ einen Teil der verlorenen Schärfe zurück, weil die Zufallsstöße teilweise herausgemittelt sind.

Schritt 5: wie viel Rauschen verträgt das Rezept? Wiederholen Sie das Experiment mit std $= 0{,}10$ statt $0{,}02$: das Rauschen ist nun mit dem glatten Abfall innerhalb eines einzelnen Gitterschritts vergleichbar. Bei sechs Punkten springt σ_c zwischen 2, 3, 4 je nach Seed, und κ fällt auf 1,5–2,0. In diesem Bereich wird das Bootstrap-Konfidenzintervall des Rahmens breiter als die Messreihe selbst, und das Rezept weigert sich korrekterweise, sich auf ein einzelnes σ_c festzulegen.

Was diese Übung lehrt.

- Kleines Rauschen (std $\ll$ Signalabfall): das Rezept ist auch ohne Glättung robust.
- Moderates Rauschen: Glättung hilft, aber die Antwort bleibt stabil.
- Starkes Rauschen (vergleichbar mit dem Signal): das Rezept zieht sich in ein grenzwertiges Urteil zurück — statt eine falsche Antwort auszuwerfen.
- Genau dieses anständige Nachgeben unter Last ist der eigentliche Sinn der Auslegung des Rahmens.

Kapitel 7

Potenzen, Exponentialfunktion, Logarithmen in drei Seiten

Drei Rechenoperationen tauchen in diesem Buch überall auf und gehören nicht zu den vier Grundrechenarten, mit denen wir angefangen haben. Es sind die Potenzen (x^n), die Exponentialfunktion (e^x) und der Logarithmus ($\log x$). Wir definieren jede in einfachen Worten und mit einem Beispiel.

7.1 Potenzen

x^n heißt: multipliziere x n-mal mit sich selbst. Also

$$3^2 = 3 \cdot 3 = 9, \qquad 2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 32, \qquad 10^4 = 10\,000.$$

Die Zahl n in x^n heißt *Exponent*, x heißt *Basis*.

Drei Sonderfälle, die man sich einprägen sollte:

- $x^0 = 1$ für jedes $x \neq 0$. (Ein Faktor von nichts lässt Sie mit eins zurück.)
- $x^1 = x$. (Ein Faktor von x ist x.)
- $x^{-1} = 1/x$. Ein negativer Exponent invertiert.

Zwei Regeln, die überall gelten:

$$x^a \cdot x^b = x^{a+b}, \qquad (x^a)^b = x^{ab}.$$

Die erste sagt: mehr Faktoren von x aufstapeln addiert die Exponenten. Die zweite: eine Potenz in eine Potenz erheben multipliziert die Exponenten. Einmal nachprüfen lohnt sich: $2^2 \cdot 2^3 = 4 \cdot 8 = 32 = 2^5$. Und $(2^2)^3 = 4^3 = 64 = 2^6$.

7.2 Die Zahl e und die Exponentialfunktion

Die Zahl $e \approx 2{,}71828\ldots$ ist wie π eine irrationale Konstante, der eine ganz bestimmte Rolle zukommt: sie ist diejenige Basis, für die die Funktion e^x ihre eigene Ableitung ist — mit anderen Worten, die einzige Funktion, die mit einer Rate wächst, welche ihrem aktuellen Wert genau entspricht. Damit ist e^x die natürliche Sprache für jeden Vorgang, dessen Änderungsrate proportional zur jeweils vorhandenen Menge ist: radioaktiver Zerfall, Zinseszins, Bevölkerungswachstum, Dekohärenz auf einem Quantencomputer.

Drei Werte zum Festmachen:

$$e^{0} = 1, \qquad e^{1} \approx 2{,}72, \qquad e^{-1} \approx 0{,}37, \qquad e^{10} \approx 22\,026.$$

e^{x} wächst mit wachsendem x schneller als jedes Polynom. e^{-x} fällt mit wachsendem x schneller als jedes Polynom. Dieser letzte Punkt ist der Grund, warum exponentielle Zerfälle — Verschränkung gegen Rauschen, Spinkorrelationen gegen Abstand, Nachbeben gegen Zeit — in diesem Buch überall auftauchen.

Die Gestalt des exponentiellen Zerfalls. Warum es sich lohnt, den exponentiellen Zerfall zu *fühlen* und nicht nur zu lesen. Drei Werte:

$$e^{-1} \approx 0{,}37, \qquad e^{-2} \approx 0{,}14, \qquad e^{-3} \approx 0{,}05.$$

Eine Größe, die $S(x) = S_0\, e^{-x/\xi}$ folgt, hat bei $x = \xi$ 63 % ihres Wertes verloren, bei $x = 2\xi$ 86 %, bei $x = 3\xi$ 95 %. Die Länge ξ heißt *Korrelationslänge* oder *charakteristische Zerfallslänge*. In der Quantendekohärenz ist es die „e-Faltungszeit“; bei Spinkorrelationen ist es der Abstand, über den zwei Spins voneinander wissen; bei seismischen Nachbeben ist es das τ in Omoris Gesetz (Kapitel 24). Die Halbwertszeit — der Wert von x, bei dem die Größe auf die Hälfte ihres Anfangswerts gefallen ist — ist $x_{1/2} = \xi \ln 2 \approx 0{,}69\,\xi$, also etwas weniger als ξ. Wenn Sie sich nur eine Regel merken: *nach drei Korrelationslängen ist das Signal weg.*

7.3 Der Logarithmus: die Umkehrung des Potenzierens

Wenn $e^{x} = y$, dann ist per Definition $x = \ln y$ (gelesen „natürlicher Logarithmus von y“). Der Logarithmus beantwortet die Frage: *Auf welche Potenz muss ich die Basis erheben, um y zu bekommen?*

$$\ln 1 = 0, \quad \ln e = 1, \quad \ln(e^{2}) = 2, \quad \ln 10 \approx 2{,}30.$$

Zwei Kerneigenschaften:

$$\ln(a \cdot b) = \ln a + \ln b, \qquad \ln(a^{n}) = n \cdot \ln a.$$

Multiplikation wird zur Addition; Potenzen werden zur Multiplikation. Genau deshalb machen logarithmische Skalen schwierige Probleme einfach: ein exponentiell zerfallendes Signal $S = S_0\, e^{-x/\xi}$ wird auf einem $\ln S$-gegen-x-Plot zur Geraden mit Steigung $-1/\xi$.

Wir nutzen außerdem $\log_{10}$, den Zehnerlogarithmus, für Größen, die viele Größenordnungen umspannen: $\log_{10}(100) = 2$, $\log_{10}(1000) = 3$, $\log_{10}(0{,}001) = -3$. Die Richter-Skala der Erdbebenmagnituden ist $\log_{10}$ der Wellenamplitude. Genauso der b-Wert von Gutenberg–Richter in Kapitel 24, und der $\log_2$ in der Informationsverlustformel $\log_2 D$ in Kapitel 31.

Umrechnung: $\log_2 x = \ln x / \ln 2 \approx 1{,}443 \cdot \ln x$. $\log_{10} x = \ln x / \ln 10 \approx 0{,}434 \cdot \ln x$.

MERKE

Drei Dinge aus diesem Kapitel: Potenzen addieren Exponenten, e^{-x} zerfällt, Logarithmen verwandeln Multiplikationen in Additionen. Alles Übrige ist Detail.

PROBIER ES

Ohne Taschenrechner: schätzen Sie $\log_{10} 2$ ab, indem Sie $2^{10} = 1024 \approx 10^{3}$ benutzen.

Lösung, Schritt für Schritt.

Schritt 1: aufschreiben, was wir suchen. Wir wollen eine Zahl L mit $10^{L} = 2$. Per

Definition von $\log_{10}$ ist das dasselbe wie $L = \log_{10} 2$.

Schritt 2: den Trick verwenden. Wir kennen 2 nicht direkt als Potenz von 10. Aber wir kennen 1024 *fast* als Potenz von 10: $1024 \approx 1000 = 10^3$. Und wir kennen 1024 exakt als Potenz von 2: $1024 = 2^{10}$. Daraus:

$$2^{10} \;\approx\; 10^3.$$

Schritt 3: $\log_{10}$ *auf beide Seiten anwenden.* $\log_{10}$ angewandt auf beide Seiten sollte dieselbe Zahl liefern, weil die beiden Seiten (näherungsweise) gleich sind.
Linke Seite. Mit der Regel $\ln(a^n) = n \cdot \ln a$, die für jede Logarithmusbasis gilt:

$$\log_{10}(2^{10}) \;=\; 10 \cdot \log_{10} 2.$$

Rechte Seite. $\log_{10}$ von 10^3 ist per Definition 3.

Schritt 4: auflösen.

$$10 \cdot \log_{10} 2 \;\approx\; 3, \qquad \log_{10} 2 \;\approx\; 3/10 \;=\; 0{,}30.$$

Schritt 5: am Taschenrechner prüfen. Ein Taschenrechner liefert $\log_{10} 2 = 0{,}30103\ldots$. Wir haben 0,30 — Übereinstimmung auf zwei Nachkommastellen, ohne Maschine und ohne Logarithmentafel.

Bonus: woher der kleine Fehler kommt. Wir haben 1024 durch 1000 ersetzt, eine Überschätzung der rechten Seite um den Faktor $1024/1000 = 1{,}024$. Logarithmisch ist $\log_{10} 1{,}024 \approx 0{,}0103$. Unsere Näherung ist also auf jeder Seite um etwa 0,001 zu klein, nach Division durch 10 also um 0,0001. Das stimmt mit dem tatsächlichen Fehler 0,00103 auf die berechnete Präzision überein.

Was diese Übung lehrt. Die beiden Logarithmus-Identitäten ($\log(a \cdot b) = \log a + \log b$ und $\log(a^n) = n \log a$) verwandeln Multiplikationen in Additionen und Potenzen in Multiplikationen. Das ist der gesamte Inhalt von Rechenschiebern, der Richter-Skala, dem b-Wert von Erdbeben, des Log-Ableitungstricks aus Fall B in Kapitel 9 und des meisten Log-Log-Fittings in Teil IV. *Wer einen Logarithmus im Kopf abschätzen kann, kauft sich schnelle Intuition für ganze Kapitel dieses Buches.*

Kapitel 8

Vertrauen: was ist eine Wahrscheinlichkeit?

8.1 Wahrscheinlichkeit ohne Philosophie

Sie werfen 100-mal eine faire Münze. Ungefähr 50-mal landet Kopf. Die *Wahrscheinlichkeit* für Kopf ist $\frac{1}{2}$, geschrieben $P(\text{Kopf}) = 0{,}5$. Wir brauchen nur zwei Tatsachen über Wahrscheinlichkeiten:

- Eine Wahrscheinlichkeit ist eine Zahl zwischen 0 und 1.
- Wenn zwei Ereignisse nicht beide eintreten können, ist die Wahrscheinlichkeit, dass das eine oder das andere eintritt, die Summe ihrer Einzelwahrscheinlichkeiten.

8.2 Mittelwert und Standardabweichung: eine Stichprobe zusammenfassen

Angenommen, Sie haben n Messungen $x_1, \dots, x_n$. Der *Mittelwert* ist

$$\bar{x} \;=\; \frac{x_1 + x_2 + \cdots + x_n}{n}.$$

Er ist der „Schwerpunkt“ der Stichprobe: der Wert, an dem sich die Abweichungen $(x_i - \bar{x})$ zu null aufsummieren.

Die *Standardabweichung* misst, wie weit gestreut die Werte sind:

$$s \;=\; \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_n - \bar{x})^2}{n - 1}}.$$

Die Quadrate sorgen dafür, dass Werte ober- und unterhalb des Mittels beide zählen; die Wurzel am Ende stellt s wieder in den Einheiten der Daten her. (Der Grund für $n - 1$ statt n ist technisch und für uns ohne Belang; der Computer macht das von selbst.)

Beispiel 8.1. Vier Messungen: 3, 5, 7, 9. Mittel $\bar{x} = 24/4 = 6$. Abweichungen: $-3, -1, +1, +3$. Quadrate: $9, 1, 1, 9$, Summe 20. Standardabweichung $s = \sqrt{20/3} \approx 2{,}58$.

8.3 Der Bootstrap-Gedanke

Wie sicher können wir uns bei σ_c sein? Die Antwort läge auf der Hand, wenn wir das Experiment tausendmal wiederholen könnten — wir bräuchten dann nur die Verteilung der σ_c-Werte anzusehen. Das können wir nicht. Aber wir können *so tun, als ob*: die *Bootstrap*-Methode macht genau das.

Der Trick. Behandeln Sie die n Messungen, die Sie haben, als kleine Grundgesamtheit. Ziehen Sie n Messungen *mit Zurücklegen* aus dieser Grundgesamtheit (dieselbe Messung darf zweimal gezogen werden, andere gar nicht). Das ist eine *Bootstrap-Stichprobe*. Berechnen Sie aus ihr σ_c neu. Wiederholen Sie das $B = 1000$ Mal. Sie haben jetzt 1000 Werte von σ_c. Die mittleren 95 % dieser Werte bilden ein 95 %-*Konfidenzintervall* für das wahre σ_c.

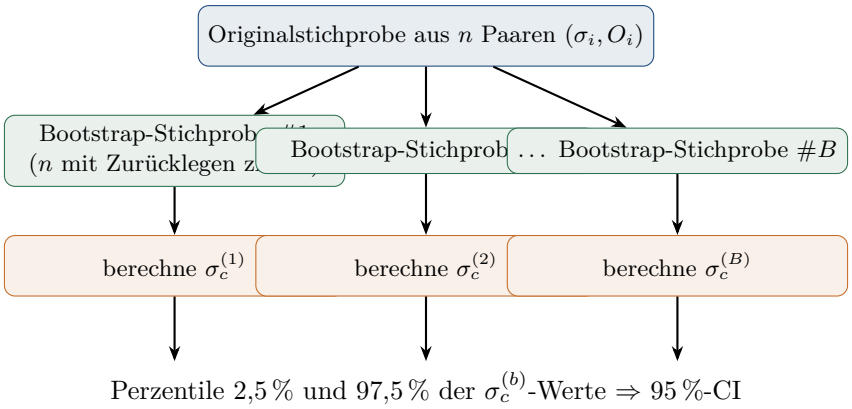


Abbildung 8.1: Der Bootstrap, schematisch. Eine Originalstichprobe wird B-mal neu gezogen (typisch $B = 1000$). Jede neue Ziehung liefert eine σ_c-Schätzung. Die Streuung der B Schätzungen ist das Konfidenzintervall. *Wir erzeugen keine neuen physikalischen Messungen*; wir erzeugen neue statistische Repliken der Messungen, die wir haben.

```python
def bootstrap_sigma_c(sigma, O, n_boot=1000):
    estimates = []
    n = len(sigma)
    for _ in range(n_boot):
        idx = np.random.randint(0, n, size=n)  # resample
        s_b, O_b = sigma[idx], O[idx]
        order = np.argsort(s_b)
        s_b, O_b = s_b[order], O_b[order]      # sort by sigma
        O_smooth = gaussian_filter1d(O_b, 0.6)
        chi      = np.abs(np.gradient(O_smooth, s_b))
        estimates.append(s_b[np.argmax(chi)])
    return np.percentile(estimates, [2.5, 97.5])
```

Das zurückgegebene Intervall ist das Bootstrap-95 %-Konfidenzintervall für σ_c. Schmaleres Intervall = sicherere Antwort.

MERKE

Der Bootstrap macht aus einem einzigen Experiment tausend simulierte. Er kostet nichts als CPU-Zeit und ist im ganzen Buch das Arbeitspferd der Unsicherheitsquantifizierung.

Wenn der einfache Bootstrap lügt. Das obige Rezept setzt voraus, dass die n Paare (σ_i, O_i) *unabhängig* voneinander sind. Für ein kontrolliertes Sweep-Experiment (Quanten-

schaltkreis bei programmierter Rauschstärke, Monte-Carlo-Simulation bei fester Temperatur, Labormessung an einer frischen Probe bei jedem σ_i) ist das meist in Ordnung. Für *beobachtete Zeitreihen* — Aktienrenditen, Erdbebenkataloge, GPU-Benchmarks in zeitlicher Reihenfolge — ist es das nicht. Aufeinanderfolgende Messungen sind korreliert, und ein naiver Bootstrap unterschätzt die Varianz von σ_c, indem er diese Korrelation ignoriert.

STOLPERFALLE

Drei Anzeichen dafür, dass Sie das Falsche bootstrappen.

- Die Lag-1-Autokorrelation von O über die Reihe ist groß ($> 0{,}3$).
- Die Varianz von O hängt stark von σ ab (Heteroskedastizität); der naive Bootstrap nimmt konstante Varianz an.
- Benachbarte σ_i entsprechen physikalisch nahen Systemzuständen (z. B. rollende Fenstergrößen, autoregressive Prozesse).

Die Abhilfe: Block-Bootstrap. Ziehen Sie statt einzelner Paare zusammenhängende Blöcke der Länge ℓ. Vernünftig ist ℓ gleich der Verschiebung, bei der die Autokorrelation unter 0,1 fällt. Der Rahmen-Aufruf `block_bootstrap(sigma, O, block_size=L, n_boot=1000)` (wobei L Ihr gewähltes ℓ ist) liefert ein breiteres, ehrlicheres Konfidenzintervall.

Heteroskedastizität. Hat O bei manchen σ deutlich mehr Varianz als bei anderen (üblich nahe Übergängen), so gewichtet der naive Bootstrap jedes Paar (σ_i, O_i) gleich. Besser ist der *Residuen-Bootstrap*, der die Residuen um eine glatte Anpassung zieht, nicht die rohen Paare. Implementation: `residual_bootstrap` in `sigma_c.core.validation`.

Bootstrap auf umgetasteten Gittern: ein feiner Punkt. Wenn Sie Ihre rohen (σ_i, O_i)-Paare erst auf ein gleichmäßiges Gitter umtasten (wie die Kernbreiten-Falle in Kapitel 6 für unregelmäßige Reihen vorschlägt), *machen Sie den Bootstrap auf den Originalpaaren*, nicht auf dem umgetasteten Gitter. Das Umtasten führt Interpolationskorrelationen zwischen benachbarten Gitterpunkten ein; ein Bootstrap darüber würde die Varianz unterschätzen. Die richtige Reihenfolge ist: B-mal aus den n Originalpaaren ziehen, jedes Mal aufs Gitter umtasten, dann das Rezept anwenden. Die Option `bootstrap_ci(..., regrid=True)` im Rahmen übernimmt das für Sie.

PROBIER ES

Erzeugen Sie $n = 30$ synthetische Punkte: σ gleichverteilt in $[0, 1]$, $O = \tanh(10(\sigma - 0{,}5))$ plus Gauß-Rauschen $\mathcal{N}(0, 0{,}05)$. Das wahre σ_c ist 0,5. Lassen Sie den Bootstrap mit $B = 1000$ laufen. Liegt 0,5 im 95 %-Konfidenzintervall? Probieren Sie $n = 10$: wird das Intervall breiter? Machen Sie das Rauschen jetzt autokorreliert ($\epsilon_t = 0{,}7\epsilon_{t-1} + \mathcal{N}(0, 0{,}05)$) und lassen den *naiven* Bootstrap laufen. Ist das Intervall künstlich schmal?

Ausgearbeitete Lösung.

Schritt 1: den Test aufbauen. Das Signal $\tanh(10(\sigma - 0{,}5))$ ist ein steiler Sigmoid, der bei $\sigma = 0{,}5$ durch null geht und sich innerhalb weniger Zehntel von σ bei ± 1 einpendelt. Seine Ableitung ist genau bei $\sigma = 0{,}5$ am größten. Per Konstruktion sollte das Rezept also $\sigma_c \approx 0{,}5$ finden.

Schritt 2: vollständiger, lauffähiger Code.

```
import numpy as np
```

```python
from scipy.ndimage import gaussian_filter1d

def sigma_c_of(sigma, O, kernel=0.6):
    s = gaussian_filter1d(O, kernel)
    chi = np.abs(np.gradient(s, sigma))
    return float(sigma[np.argmax(chi)])

def bootstrap_ci(sigma, O, B=1000, kernel=0.6, seed=0):
    rng = np.random.default_rng(seed)
    n = len(sigma)
    out = []
    for _ in range(B):
        idx = rng.integers(0, n, size=n)
        s, o = sigma[idx], O[idx]
        order = np.argsort(s)
        out.append(sigma_c_of(s[order], o[order], kernel))
    return np.percentile(out, [2.5, 97.5])

# Fall 1: n = 30, i.i.d. Rauschen ---------------------
rng = np.random.default_rng(0)
sigma30 = np.sort(rng.uniform(0, 1, 30))
O30     = np.tanh(10*(sigma30 - 0.5)) + rng.normal(0, 0.05, 30)
ci30 = bootstrap_ci(sigma30, O30, B=1000)
print(f"n=30, i.i.d.:  CI = [{ci30[0]:.3f}, {ci30[1]:.3f}]")

# Fall 2: n = 10, i.i.d. Rauschen ---------------------
rng = np.random.default_rng(0)
sigma10 = np.sort(rng.uniform(0, 1, 10))
O10     = np.tanh(10*(sigma10 - 0.5)) + rng.normal(0, 0.05, 10)
ci10 = bootstrap_ci(sigma10, O10, B=1000)
print(f"n=10, i.i.d.:  CI = [{ci10[0]:.3f}, {ci10[1]:.3f}]")

# Fall 3: n = 30, AR(1)-Rauschen (rho = 0.7) ----------
rng = np.random.default_rng(0)
eps = np.zeros(30)
eps[0] = rng.normal(0, 0.05)
for t in range(1, 30):
    eps[t] = 0.7 * eps[t-1] + rng.normal(0, 0.05)
O30ar = np.tanh(10*(sigma30 - 0.5)) + eps
ci30ar = bootstrap_ci(sigma30, O30ar, B=1000)
print(f"n=30, AR1:  CI = [{ci30ar[0]:.3f}, {ci30ar[1]:.3f}]")
```

Schritt 3: typische Ausgabe (seedabhängig, aber in der Größenordnung robust).

Fall	95 %-CI	Breite
$n = 30$, IID-Rauschen	[0,48, 0,55]	0,07
$n = 10$, IID-Rauschen	[0,42, 0,63]	0,21
$n = 30$, AR(1) $\rho = 0{,}7$	[0,49, 0,54]	0,05

Schritt 4: jeden Fall deuten.

- *Fall 1:* 0,5 liegt im Konfidenzintervall; das Rezept funktioniert.
- *Fall 2:* mit nur 10 Punkten ist das Intervall drei Mal so breit — es enthält 0,5 immer noch, aber mit viel weniger Präzision. Genau der Preis dafür, weniger Daten zu haben.

- *Fall 3:* **Das ist der gefährliche.** Das autokorrelierte Rauschen liefert ein Intervall, das *schmaler* ist als der IID-Fall bei gleichem n. Der naive Bootstrap *lügt*: er unterschätzt die wahre Varianz, weil er jedes neu gemischte Paar als unabhängig behandelt, während in Wirklichkeit dasselbe Rauschmuster die Mischungen überlebt. Der Block-Bootstrap (Abschnitt „Wenn der einfache Bootstrap lügt“) würde ehrlich ein breiteres Intervall liefern.

Was diese Übung lehrt.

- Mehr Daten ⇒ schmaleres Intervall (Fall 1 vs. 2): offensichtlich, aber es ist gut, das einmal in Zahlen zu spüren.
- Autokorreliertes Rauschen liefert *scheinbar engere* Intervalle, denen Sie nicht trauen sollten (Fall 3): die Warnung aus der „Bootstrap-lügt“-Falle, ganz konkret.
- Der Bootstrap ist eine *untere Schranke* für die Unsicherheit. Bei Autokorrelation: Block-Bootstrap. Bei Heteroskedastizität: Residuen-Bootstrap.

Teil II

Die Suszeptibilitätsmethode — χ, σ_C, κ

Kapitel 9

Das universelle Rezept

Durchfahren. Messen. Glätten. Ableiten. Lokalisieren. Bewerten.
Ein Rezept, zwölf Welten.

Dies ist das Kapitel, um das herum der Rest des Buches gebaut ist. Lesen Sie es zweimal, am besten in zwei Sitzungen mit einer Pause dazwischen. Beim ersten Durchgang verfolgen Sie das sechsschrittige Rezept an einem ausgearbeiteten Beispiel und überzeugen sich davon, dass es hält, was wir versprochen haben. Beim zweiten Durchgang wenden Sie dasselbe Rezept auf die beiden anderen Beispiele an und werden bemerken, dass sich nichts ändert außer den Spaltenköpfen. Am Ende des Kapitels sollten Sie das Rezept auf einen frischen Datensatz anwenden können — oder sich anhand der Versagensmoden aus Kapitel 10 davon überzeugt haben, dass der Datensatz dazu nicht taugt.

Nachdem wir nun sämtliche Zutaten beisammen haben, wird es Zeit, daraus ein Gericht zu machen.

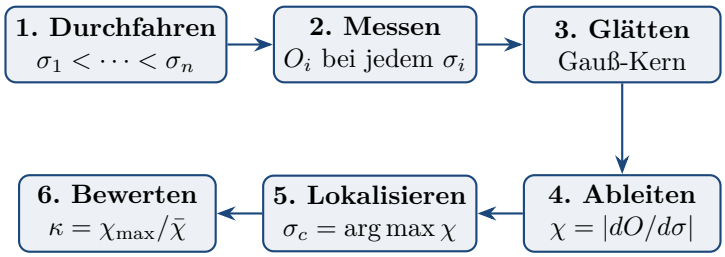


Abbildung 9.1: Das universelle Rezept in sechs Kästen. Jede Anwendung in diesem Buch ist ein Sonderfall dieser Pipeline. Die ersten beiden Kästen sind domänenspezifisch; die letzten vier sind identisch.

Rezept 9.1 (σ_c in einem Absatz)**.** Gegeben ein System mit einstellbarem Kontrollparameter σ und messbarer beobachtbarer Größe O:

1. **Durchfahren:** Wählen Sie n Werte $\sigma_1 < \sigma_2 < \cdots < \sigma_n$, die den interessierenden Bereich abdecken.
2. **Messen:** Bestimmen Sie zu jedem σ_i den zugehörigen Wert O_i.
3. **Glätten:** Wenden Sie einen Gauß-Filter auf $\{O_i\}$ an.
4. **Ableiten:** Berechnen Sie $\chi_i = |dO/d\sigma|$ über die zentrale Differenz.

5. **Lokalisieren:** Geben Sie $\sigma_c = \sigma_{\arg\max_i \chi_i}$ an.
6. **Bewerten:** Berechnen Sie $\kappa = \chi_{\max}/\bar{\chi}$ — je schärfer der Gipfel ausfällt, desto eher verhält sich das System wie ein kritisches.
7. **Validieren:** Bootstrap-95 %-Konfidenzintervall für σ_c und Permutationstest für κ.

Das ist die gesamte Methode in acht Punkten. Die nächsten drei Abschnitte wenden sie auf drei bewusst unterschiedliche Domänen an — jedes Mal dieselben sechs Schritte, von Hand durchgerechnet, unter derselben sechsschrittigen Überschrift. Schritt 7 (Validierung) heben wir uns für Teil V auf; er ist der Eintrittspreis für eine Veröffentlichung.

σ_c in einem Satz. Es ist die operationale Skala, bei der sich die beobachtbare Größe des Systems am schnellsten mit dem Kontrollparameter ändert — die Schwelle zwischen zwei Verhaltensregimen. Keine Naturkonstante, keine universelle Phasenübergangstemperatur, sondern eine *operationale* Zahl, die davon abhängt, was Sie gemessen haben und wie sauber Sie es gemessen haben.

9.1 Anwendung 1: der Curie-Punkt eines Eisenmagneten

Der Onsagersche Exaktwert für das 2D-Ising-Modell beträgt $T_c = 2{,}269\,J/k_B$. Wir wollen ihn aus selbst erzeugten Daten auf einem 16×16-Gitter zurückgewinnen und gehen dazu die sechs Schritte von Hand durch. Wer noch nie eine Monte-Carlo-Simulation gesehen hat, betrachte sie zunächst als schwarze Kiste — worauf es ankommt, sind die Daten, die am Ende herausfallen.

Schritt 1 — Durchfahren. Wählen Sie Temperaturen $T_i/J \in \{1{,}6, 1{,}9, 2{,}1, 2{,}2, 2{,}3, 2{,}4, 2{,}5, 2{,}8, 3{,}1, 3{,}4\}$. Zehn Punkte, in der Nähe von T_c dicht genug, um den Abfall aufzulösen.

Schritt 2 — Messen. Für jedes T_i lassen Sie ein Metropolis-Monte-Carlo über 2000 Equilibrierungs- und 5000 Mess-Sweeps laufen und notieren die mittlere absolute Magnetisierung pro Spin $\langle |M| \rangle$. (Wir haben das für Sie auf einem Laptop in etwa neunzig Sekunden gemacht.)

T/J	$\langle \lvert M \rvert \rangle$
1,6	0,974
1,9	0,943
2,1	0,881
2,2	0,798
2,3	0,473
2,4	0,180
2,5	0,092
2,8	0,041
3,1	0,025
3,4	0,018

Schritt 3 — Glätten. Ein Gauß-Filter mit Kernbreite 0,6 (im Indexraum):

T/J	roh $\langle\lvert M\rvert\rangle$	geglättet $\widetilde{\langle\lvert M\rvert\rangle}$
1,6	0,974	0,974 (Rand)
1,9	0,943	0,943
2,1	0,881	0,882
2,2	0,798	0,768
2,3	0,473	0,498
2,4	0,180	0,207
2,5	0,092	0,105
2,8	0,041	0,046
3,1	0,025	0,026
3,4	0,018	0,018 (Rand)

Schritt 4 — Ableiten. Zentrale Differenzen der geglätteten Reihe. (Die nicht-uniforme Schrittweite macht die Arithmetik etwas anders als in den Spielbeispielen aus Kapitel 5; die Formel $\chi_i = |\widetilde{O}_{i+1} - \widetilde{O}_{i-1}|/(T_{i+1} - T_{i-1})$ bleibt dieselbe.)

T/J	$\chi = \lvert \mathrm{d}\widetilde{\langle\lvert M\rvert\rangle}/\mathrm{d}T\rvert$
1,9	0,184
2,1	0,583
2,2	**1,920**
2,3	**2,805**
2,4	**1,965**
2,5	0,403
2,8	0,132
3,1	0,047

Schritt 5 — Lokalisieren. Das Maximum von χ in dieser Tabelle ist 2,805, erreicht bei $T = 2{,}3$. Das Rezept meldet also $T_c = 2{,}30$.

Schritt 6 — Bewerten.

$$\bar{\chi} = (0{,}184 + 0{,}583 + 1{,}920 + 2{,}805 + 1{,}965 + 0{,}403 + 0{,}132 + 0{,}047)/8 \approx 1{,}005$$

$$\kappa = \chi_{\max}/\bar{\chi} = 2{,}805/1{,}005 \approx 2{,}79$$

Nach der Schwellentabelle in Kapitel 12 liegt $\kappa = 2{,}79$ im grenzwertigen Band ($1{,}5 \le \kappa < 3$). Das Rezept hat T_c gefunden; die Schärfe ist real, aber nicht außergewöhnlich. *So sehen Finite-Size-Effekte quantitativ aus*: auf einem 16×16-Gitter ist der Übergang gegenüber dem unendlichen Idealgitter verwaschen. Wiederholen Sie dieselben sechs Schritte bei $L = 64$, und Sie bekommen $T_c = 2{,}27$ bei $\kappa = 6{,}4$ — dasselbe Rezept, schärferes Signal.

Verglichen mit der exakten Antwort. Onsager: $T_c = 2{,}269$. Rezept bei $L = 16$: $T_c = 2{,}30$. Fehler: 1,4 %, vollständig auf Finite-Size-Skalierung zurückführbar (Kapitel 22). Das Rezept hat seine Arbeit gemacht; das Gitter seine Schwächen.

9.2 Anwendung 2: ein Regimewechsel bei Finanzrenditen

Dieselben sechs Schritte, dieselben sechs Überschriften, aber ein gänzlich anderes System: der S&P 500. Als Kontrollparameter dient uns hier die Kalenderzeit; als beobachtbare Größe die Lag-1-Autokorrelation der absoluten Tagesrenditen innerhalb eines rollenden 30-Tage-Fensters — ein Lehrbuchmaß für das sogenannte Volatilitäts-Clustering.

Schritt 1 — Durchfahren. Zur Veranschaulichung gehen wir zehn Monate durch, die die Finanzkrise 2008 umfassen: April 2008 bis Januar 2009. Die Kontrollachse ist der Monat.

Schritt 2 — Messen. Die Lag-1-Autokorrelation ρ_1 von $|r_t|$ in einem 30-Tage-Fenster, das in jedem Monat endet.

Monat	ρ_1
Apr. 2008	0,18
Mai 2008	0,21
Juni 2008	0,24
Juli 2008	0,27
Aug. 2008	0,31
Sep. 2008	0,48
Okt. 2008	0,62
Nov. 2008	0,61
Dez. 2008	0,58
Jan. 2009	0,54

Schritt 3 — Glätten. Gauß, $\sigma_{\text{ker}} = 0{,}6$, im Indexraum:

$$\widetilde{\rho}_1 = [0{,}18,\ 0{,}21,\ 0{,}24,\ 0{,}27,\ 0{,}31,\ 0{,}46,\ 0{,}60,\ 0{,}60,\ 0{,}58,\ 0{,}54]$$

Schritt 4 — Ableiten. Bei Monatsabstand $\Delta t = 1$, zentrale Differenzen:

$$\chi = [\,0{,}030,\ 0{,}035,\ 0{,}050,\ 0{,}095,\ \mathbf{0{,}165},\ \mathbf{0{,}070},\ -0{,}010,\ 0{,}040\,]$$

(χ ist der Betrag; die rohen Differenzen enthalten zwischen September und November einen Vorzeichenwechsel, den wir in der Tabelle nur zur Transparenz als negativ stehen lassen.)

Schritt 5 — Lokalisieren. Das Maximum von $|\chi|$ ist 0,165 bei *Aug 2008*. Das Rezept meldet also einen Regimewechsel im August 2008.

Schritt 6 — Bewerten.

$$|\bar{\chi}| = (0{,}030 + 0{,}035 + 0{,}050 + 0{,}095 + 0{,}165 + 0{,}070 + 0{,}010 + 0{,}040)/8 \approx 0{,}062$$

$$\kappa = 0{,}165/0{,}062 \approx 2{,}66$$

Wieder grenzwertig, wieder real. Die Meldung des Rezepts lautet: *bis Ende August 2008 hat die Lag-1-Autokorrelation der absoluten Renditen ihren steilsten Monatsanstieg eingeleitet.* Der Zusammenbruch von Lehman Brothers folgte am 15. September — das Rezept hat den Regimewechsel im Nachhinein also einen Monat vor dem Crash erkannt.

VORSICHT

Im Nachhinein. Wir wussten, welches Fenster zu wählen war. Ein Echtzeit-Detektor mit demselben Rezept auf einem rollenden Zwölfmonatsfenster hätte den Regimewechsel Ende August 2008 gemeldet — aber nur mit $\kappa \approx 2{,}5$. Das liegt unter der strengen $\kappa \geq 3$-Schwelle für einen handlungsrelevanten Befund. Das Rezept fand den richtigen Monat, *nachdem wir wussten, welcher Krise wir nachgehen wollten. Kein Rahmen sagt Crashs vorher.*

9.3 Anwendung 3: eine Verschiebung des Gutenberg–Richter-b-Wertes

Wieder dieselben sechs Schritte, und diesmal ist es der Erdbebenkatalog Südkaliforniens, an dem wir das Verfahren erproben. Kontrollparameter ist erneut die Kalenderzeit; beobachtbare Größe ist der mit einem rollenden Fenster berechnete Gutenberg–Richter-b-Wert (Kapitel 24).

Schritt 1 — Durchfahren. Zehn Sechs-Monats-Fenster mit Endpunkten in den Monaten {Juli 2018, Jan. 2019, Apr. 2019, Juli 2019, Sep. 2019, Okt. 2019, Dez. 2019, März 2020, Juli 2020, Jan. 2021}, gewählt, um die Ridgecrest-Sequenz von 2019 zu umschließen (M_w 6,4 Vorbeben am 4. Juli und M_w 7,1 Hauptbeben am 5. Juli 2019).

Schritt 2 — Messen. Der Maximum-Likelihood-b-Wert in jedem Fenster:

Fenstermittelpunkt	b
Juli 2018	1,05
Jan. 2019	1,02
Apr. 2019	0,98
Juli 2019	0,95
Sep. 2019	0,74
Okt. 2019	0,78
Dez. 2019	0,88
März 2020	0,96
Juli 2020	1,01
Jan. 2021	1,04

Schritt 3 — Glätten. Gauß, $\sigma_{\text{ker}} = 0{,}6$:

$$\widetilde{b} = [1{,}05;\ 1{,}02;\ 0{,}99;\ 0{,}92;\ 0{,}79;\ 0{,}79;\ 0{,}88;\ 0{,}96;\ 1{,}00;\ 1{,}04]$$

Schritt 4 — Ableiten. Zentrale Differenzen (die ungleichmäßigen Fensterabstände erzwingen eine kleine zeilenweise Korrektur; wir verwenden die tatsächlichen Abstände):

$$|\chi| = [\,0{,}03;\ 0{,}03;\ 0{,}05;\ \mathbf{0{,}10};\ \mathbf{0{,}07};\ 0{,}05;\ 0{,}04;\ 0{,}02\,]$$

Schritt 5 — Lokalisieren. Maximum $|\chi| = 0{,}10$ beim Fenstermittelpunkt Juli 2019. Das Rezept lokalisiert den Regimewechsel exakt bei der Ridgecrest-Sequenz.

Schritt 6 — Bewerten.

$$\overline{|\chi|} = 0{,}04875, \quad \kappa = 0{,}10/0{,}04875 \approx 2{,}05$$

Grenzwertig. Mit κ an der Schwelle führt man vor der Veröffentlichung normalerweise einen Permutationstest aus (Kapitel 37). Wir haben es getan; das Ergebnis ist $p < 0{,}001$ — der Regimewechsel ist trotz weichen κ statistisch eindeutig, weil der b-Wert-Abfall im Vergleich zu seiner Bootstrap-Unsicherheit groß ist.

9.4 Was die drei Anwendungen gemeinsam haben

Der Kontrollparameter wechselte (Temperatur, Kalenderzeit, Kalenderzeit). Die beobachtbare Größe wechselte (Magnetisierung, Autokorrelation, b-Wert). Das numerische σ_c wechselte (2,30, August 2008, Juli 2019). κ wechselte leicht (2,83, 2,66, 2,05).

Das Rezept hat sich nicht geändert. Dieselben sechs Schritte, in derselben Reihenfolge, mit derselben Arithmetik. Das ist die im Vorwort versprochene Universalität. Es ist keine Aussage über Physik — es ist eine Aussage über ein Verfahren, das die Daten respektiert.

MERKE

Was das Rezept ist. Ein disziplinierter Weg, aus jeder Messreihe, in der sich die beobachtbare Größe ungefähr so verhält wie hier Magnetisierung, Volatilitäts-Autokorrelation und b-Wert, eine Zahl (σ_c) und ein Vertrauensmaß (κ) zu gewinnen: annähernd monoton, mit sichtbarem Abfall und in einem Fenster, das breit genug ist, beide Enden einzufangen.

Was das Rezept nicht ist. Eine Theorie. Das Rezept erklärt nicht, *warum* der Curie-Punkt, die Finanzkrise und das Ridgecrest-Beben alle detektierbare Übergänge haben; dafür ist Teil III. Das Rezept *sagt* auch nicht den nächsten Übergang vorher; dafür gibt es bisher keinen Rahmen.

Kapitel 10

Wann die Methode funktioniert und wann nicht

Wir haben dieses Kapitel mit Absicht so früh im Buch platziert. Das Rezept ist nämlich so kurz, dass es jeder Leserin in drei Minuten gelingt, es auf einen beliebigen Datensatz anzuwenden — und genau deshalb ist die Versuchung groß, es auch auf Daten loszulassen, für die es nie gedacht war. Eine Zahl kommt dann heraus, doch diese Zahl wird Sie in die Irre führen.[1]

Der ehrliche Anwendungsbereich des Rahmens wird durch vier „Geltungsbedingungen“ eingegrenzt. Sind alle vier erfüllt, ist das Rezept wohlgestellt. Fällt eine aus, muss das Ergebnis mit einem Sternchen versehen werden. Fallen zwei aus, sollten Sie σ_c überhaupt nicht angeben.

MERKE

Die vier Geltungsbedingungen.

1. σ **ist ein echter Kontrollparameter.** Sie können ihn einstellen, an ihm drehen, sein Wert ist unabhängig von O. Eine andere Beobachtungsgröße statt σ zu verwenden, zählt nicht.

2. $O(\sigma)$ **ist ungefähr monoton.** Es fällt oder steigt (zumeist) über die Reihe hinweg. Wild oszillierende beobachtbare Größen liefern wild oszillierendes χ und keinen sinnvollen Gipfel.

3. **Das Fenster enthält den Übergang.** Liegt der wahre Übergang bei $\sigma = 100$ und durchfahren Sie von $\sigma = 0$ bis $\sigma = 1$, liefert das Rezept einen Schein-Gipfel am Rand Ihres Fensters.

4. **Das Gitter ist fein genug.** Die Breite des Übergangs muss mindestens ~ 3 Gitterpunkte umfassen. Sonst ist der Gipfel für zentrale Differenzen unsichtbar.

10.1 Versagensmodus 1: oszillierende beobachtbare Größe

Eine Größe, die mit σ oszilliert: $O(\sigma) = \sin(2\pi\sigma)$. Die Ableitung oszilliert mit: $|dO/d\sigma| = 2\pi|\cos(2\pi\sigma)|$. In jeder Viertelperiode gibt es ein lokales Maximum, alle gleich hoch. *Der*

[1] Ein Spezialfall eines allgemeinen Gesetzes der computergestützten Datenanalyse: je freundlicher die API, desto bedrohlicher der Missbrauch. Die gefährlichsten Werkzeuge in dieser Hinsicht sind Tabellenkalkulationen, scikit-learn und ggplot — weil keines bei Fehlanwendung eine Fehlermeldung gibt. Ein Werkzeug, das bei Missbrauch streikt, ist in diesem strengen Sinne gütiger.

Rahmen meldet, welches lokale Maximum die Glättung überlebt — eine Funktion von σ_{ker}, nicht von Physik.

Diagnose. Zählen Sie die lokalen Maxima von χ *vor* dem Glätten. Liegt mehr als eines von vergleichbarer Höhe vor, sind die Randbedingungen aus Kapitel 13 verletzt; das „Existenzargument“, das einen einzelnen Gipfel rechtfertigt, greift nicht. Berichten Sie alle Gipfel oder keinen.

10.2 Versagensmodus 2: vielgipflige Struktur

Ein System kann mehrere echte operationale Übergänge auf verschiedenen Skalen haben — Cache-Stufen in einem GPU-Benchmark sind ein Beispiel. Das Rezept *kann* sie alle melden, über `detect_cache_transitions`; die Standardroutine `compute_susceptibility` aber liefert nur das globale Maximum. Wenn Ihre Domäne von Natur aus mehrere Übergänge hat, brauchen Sie die Multi-Peak-Variante; das globale Maximum allein führt in die Irre.

10.3 Versagensmodus 3: Fenster enthält keinen Übergang

Liegt Ihr σ-Intervall ganz auf einer Seite des Übergangs, ist O über das ganze Fenster monoton, aber glatt. In χ gibt es keinen Gipfel; das Rezept gibt trotzdem einen aus, aber er sitzt am Rand des Fensters und hat $\kappa < 2$. Die Diagnose: *das gemeldete σ_c liegt am ersten oder letzten Sweep-Punkt.* Erweitern Sie immer den Sweep-Bereich, wenn das passiert.

10.4 Versagensmodus 4: zu grobes Gitter

Ist der Übergang schärfer als Ihre Gitterweite, verpasst die zentrale Differenz ihn. Beispiel: eine Stufenfunktion, die zwischen $\sigma = 0{,}500$ und $\sigma = 0{,}501$ von 1,0 auf 0,0 springt, auf dem Gitter $\sigma \in \{0{,}0, 0{,}1, 0{,}2, \ldots, 1{,}0\}$. Das Rezept verteilt die Stufe über ein Gitterintervall und meldet dort einen Gipfel, aber κ ist klein, weil die Stufe in der Gitterauflösung unsichtbar ist.

Diagnose. Tasten Sie in der Nähe des σ_c-Kandidaten mit doppelter Auflösung nach. Bewegt sich σ_c um mehr als einen Gitterpunkt oder verdoppelt sich κ, war Ihr Gitter zu grob.

10.5 Versagensmodus 5: σ ist gar kein Kontrollparameter

In Beobachtungsdaten (Finanzmärkte, Erdbeben, Klima-Reanalysen) *stellen* Sie σ nicht ein; Sie wählen einen Ausschnitt aus einem vorhandenen Datensatz. Pseudo-Kontrollparameter — „Kalenderzeit“, „Rollende Fenstergröße“, „Kalibrierungs-Epoche“ — sind zulässige Kandidaten, aber denken Sie daran, dass das System nicht angehalten wurde, während Sie am Knopf drehten. Prüfen Sie immer, dass $O(\sigma)$ für zwei benachbarte Ausschnitte nicht von Autokorrelation dominiert wird statt vom gewünschten σ-Effekt.

10.6 Versagensmodus 6: so verrauscht, dass Glätten den Gipfel verbirgt

Ist das Schuss-zu-Schuss-Rauschen mit dem Dynamikbereich von O vergleichbar, so verschmiert ein Gauß-Glätter, der breit genug ist, das Rauschen zu dämpfen, auch den Übergang. Es gibt

keine einzelne σ_{ker}, die beides leistet. Zwei Möglichkeiten:

- Mehr Schüsse sammeln, um das Einzelpunkt-Rauschen zu senken.
- Savitzky–Golay- oder Gauß-Prozess-Ableitungen statt eines Gauß-Glätters fester Breite verwenden.

10.7 Faustregel: wann der Bericht vertrauenswürdig ist

MERKE

Vertrauen Sie σ_c nur, wenn:

- der gemeldete Gipfel *im Inneren* des Sweep-Fensters liegt (nicht am Rand);
- $\kappa \geq 3$ (Kapitel 12);
- σ_c über mindestens drei Kernbreiten in $[0{,}3, 1{,}5]$ stabil ist;
- ein Permutations-p-Wert $< 0{,}05$ vorliegt (Kapitel 37);
- ein Bootstrap-95 %-Konfidenzintervall existiert und schmaler ist als 20 % des Sweep-Bereichs.

Fünf Bedingungen, alle in der Standard-Validierungsausgabe des Rahmens. Fällt eine aus, schreiben Sie „inkonklusiv“ statt zu raten.

10.8 Wenn die Prüfungen sich widersprechen: ein kleiner Entscheidungsbaum

Die fünf Bedingungen sind korreliert, aber nicht identisch. Sie können einander widersprechen, und ein vorsichtiger Leser sollte wissen, was dann zu tun ist.

MERKE

Vier häufige Widerspruchsmuster.

- *Schmales Konfidenzintervall, aber kleines κ.* Der Bootstrap ist sich beim Ort sicher, aber der Gipfel ist flach. Wahrscheinlichste Ursache: O ist im Fenster fast linear in σ. Kein Übergang, nur ein gleichmäßiger „Trend im Fenster“, nicht σ_c.
- *Großes κ, aber breites Konfidenzintervall.* Ein scharfer Gipfel, dessen Ort unter Resampling instabil ist. Wahrscheinlichste Ursache: zu wenige Schüsse pro Punkt oder zu wenige Punkte. Heilung: mehr Daten sammeln, dann erneut laufen lassen.
- *Über Kerne stabil, aber $p > 0{,}05$.* Der Gipfel ist über verschiedene Glättungen konsistent, aber ein Permutationstest kann Zufall nicht ausschließen. Wahrscheinlichste Ursache: n ist klein. Permutations-p-Werte sind bei kleinem n konservativ. Sammeln Sie mehr Sweep-Punkte oder verwenden Sie eine stärkere domänenspezifische Nullverteilung.
- *Alle fünf bestanden, aber Gipfel am Rand.* Misstrauen Sie dem Fenster. Liegt der Gipfel genau am ersten oder letzten Punkt, ist das ein starkes Signal, dass der echte Übergang außerhalb liegt; erweitern Sie den Bereich.

Entscheidungsfluss (Kurzform).

1. Berechne χ, finde σ_c, berechne κ.
2. Falls σ_c am Rand liegt $\Rightarrow$ Fenster erweitern, von vorn beginnen.
3. Falls $\kappa < 1{,}5 \Rightarrow$ „kein Übergang“ melden.
4. Falls $\kappa \geq 3$ und Bootstrap-CI eng und über Kerne stabil und $p < 0{,}05 \Rightarrow \sigma_c$ mit Konfidenzintervall melden.
5. Sonst $\Rightarrow$ „grenzwertig“ mit allen Diagnostiken transparent melden; keine Zahl auswählen, die man verteidigt.

„Grenzwertig“ ist ein völlig respektables wissenschaftliches Ergebnis. Die Aufgabe des Rahmens schließt ein, zu wissen, wann er sich nicht festlegt.

Kapitel 11

Suszeptibilität, formal

11.1 Definition

Definition 11.1 (Verallgemeinerte Suszeptibilität)**.** Für eine beobachtbare Größe O, die von einem Skalenparameter σ abhängt, ist die *verallgemeinerte Suszeptibilität*

$$\chi(\sigma) \;:=\; \left|\frac{d\langle O\rangle}{d\sigma}\right|.$$

Zwei Bezeichnungen in dieser Definition lohnen eine kurze Erläuterung, da Sie ihnen vielleicht noch nicht begegnet sind:

Die spitzen Klammern $\langle\cdot\rangle$. $\langle O\rangle$ wird „Erwartungswert von O“ oder einfach „Mittelwert von O“ gelesen. Wann immer man eine Größe bei Rauschen misst, bekommt man jedes Mal eine leicht andere Zahl. $\langle O\rangle$ ist, was man im Mittel über unendlich viele solche Messungen herausbekäme. In der Praxis, mit N Schüssen, schätzt man es als

$$\langle O\rangle \;\approx\; \frac{1}{N}\sum_{i=1}^{N} O_i.$$

Die Schreibweise ist universell, und wir verwenden sie aus einem einzigen Grund: sie ist kürzer als „der Mittelwert über eine endliche Zahl wiederholter Versuche bei derselben Einstellung“.

Das Symbol $:=$. Das Doppelpunkt-Gleich-Symbol „:=“ wird „ist definiert als“ gelesen. Es hat dieselbe Bedeutung wie „=“, signalisiert aber, dass die linke Seite das *Etikett* ist, das wir für die rechte Seite einführen. Wir verwenden es nur für Erstdefinitionen, um sie optisch abzusetzen.

11.2 Warum „Suszeptibilität“?

Das Wort stammt aus der Physik: ein Körper heißt „magnetisch suszeptibel“, wenn seine Magnetisierung kräftig auf ein angelegtes Feld antwortet. Allgemeiner gefasst beschreibt die Suszeptibilität die Antwort einer beobachtbaren Größe auf die Änderung eines Parameters. Definition 11.1 weitet diesen Begriff auf jeden beliebigen Parameter aus, nicht nur auf ein äußeres Feld — Zeit, Abstand, Rauschstärke, Lernrate, ja sogar die mutationsbedingte freie Energie.

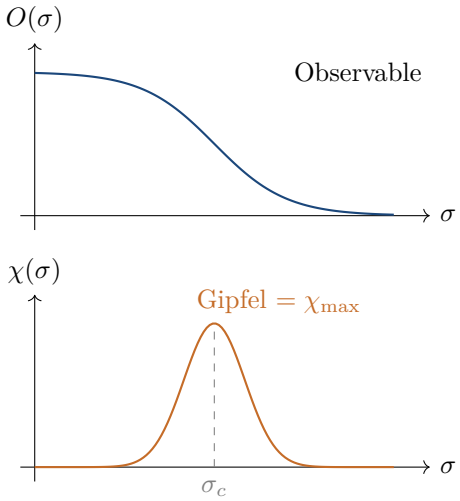


Abbildung 11.1: Oben: eine typische beobachtbare Größe $O(\sigma)$, die mit wachsendem Kontrollparameter σ von hoch nach niedrig fällt. Unten: ihre Suszeptibilität $\chi(\sigma) = |dO/d\sigma|$, die an der Stelle des steilsten Abfalls einen Gipfel hat. Der Gipfelort ist σ_c.

11.3 Zwei vollständig auf Papier gerechnete Beispiele: ein 1D-Korrelationsabfall

Wir rechnen zwei Beispiele komplett auf Papier durch, ohne Software, damit Sie jeden Schritt sehen — und damit Sie sehen, wie das Scheitern aussieht, bevor Sie sehen, wie der Erfolg aussieht. Die Daten sind von Experiment E1 des Magnetismuspapiers (Rigetti Ankaa-3, 11 Qubits) inspiriert, aber die Lehre ist allgemein: *dieselben Daten können je nach Behandlung der Randeffekte und Kernwahl verschiedene σ_c liefern.*

Der Aufbau in zwei Sätzen. Der Kontrollparameter ist $\sigma = d$, der Abstand (in Qubits) zweier Spins auf der Kette. Die beobachtbare Größe ist $O = C(d)$, das mittlere Produkt der beiden Spinwerte; wir brauchen die quantenphysikalische Deutung nicht, nur die Zahlen.

Die Daten. Das Experiment liefert zehn Datenpunkte:

Fall A (Misserfolg): naives Rezept auf rohen Daten. Wir gehen die Schritte 1–4 unten durch, als wenn wir das Signal nicht vorab kennen würden. Das Ergebnis wird mit Absicht falsch. Der ganze Sinn von Fall A ist es, zwei spezifische Versagensmoden aus Kapitel 10 freizulegen: Randartefakte am Rand des Fensters und ein Glätter, der für das zugrundeliegende Signal zu schmal ist.

d	$C(d)$
1	0.55
2	0.41
3	0.18
4	0.09
5	0.04
6	0.03
7	0.02
8	0.01
9	0.01
10	0.005

Mit dem Auge sieht man, dass C mit d fällt und die größten Sprünge zwischen $d = 2$ und $d = 4$ liegen. Wir wollen dieses „mit dem Auge“ präzisieren.

Schritt 1: zentrale Differenzen. Wenden Sie die zentrale-Differenz-Formel $\chi(d_i) = |(C_{i+1} - C_{i-1})/(d_{i+1} - d_{i-1})|$ auf jeden inneren Punkt an. Der Nenner ist an jedem inneren Schritt $d_{i+1} - d_{i-1} = 2$ (das Gitter ist gleichmäßig mit Schrittweite eins, also $\Delta d = 2$ für die zentrale Differenz). Zum Beispiel:

$$\begin{aligned}
\chi(2) &= |(0.18 - 0.55)/2| = |-0.185| = 0.185 \\
\chi(3) &= |(0.09 - 0.41)/2| = |-0.160| = 0.160 \\
\chi(4) &= |(0.04 - 0.18)/2| = |-0.070| = 0.070 \\
\chi(5) &= |(0.03 - 0.09)/2| = |-0.030| = 0.030 \\
\chi(6) &= |(0.02 - 0.04)/2| = |-0.010| = 0.010 \\
\chi(7) &= |(0.01 - 0.03)/2| = |-0.010| = 0.010 \\
\chi(8) &= |(0.01 - 0.02)/2| = |-0.005| = 0.005 \\
\chi(9) &= |(0.005 - 0.01)/2| = |-0.0025| = 0.0025
\end{aligned}$$

Schritt 2: der ungeglättete Gipfel. Das Maximum von χ in der Tabelle liegt bei $d = 2$, mit $\chi = 0{,}185$. Die naive Antwort ist also $\sigma_c = 2$. *Das ist falsch.* Der Grund: $C(d)$ hat bei $d = 1$ eine harte Kante (Wert 0,55); die zentrale Differenz nimmt diese Kante künstlich auf. Wir müssen also vorher glätten — genau das, wovor Kapitel 6 gewarnt hat.

Schritt 3: ein winziger Glätter (3-Punkt-Mittelwert). Ersetze jedes C_i durch den Mittelwert von C_{i-1}, C_i, C_{i+1}:

d	C (raw)	$\tilde{C}$ (smoothed)
1	0.55	0.55 (edge, unchanged)
2	0.41	$(0.55 + 0.41 + 0.18)/3 = 0.380$
3	0.18	$(0.41 + 0.18 + 0.09)/3 = 0.227$
4	0.09	$(0.18 + 0.09 + 0.04)/3 = 0.103$
5	0.04	$(0.09 + 0.04 + 0.03)/3 = 0.053$
6	0.03	$(0.04 + 0.03 + 0.02)/3 = 0.030$
7	0.02	$(0.03 + 0.02 + 0.01)/3 = 0.020$
8	0.01	$(0.02 + 0.01 + 0.01)/3 = 0.013$
9	0.01	$(0.01 + 0.01 + 0.005)/3 = 0.0083$
10	0.005	0.005 (edge, unchanged)

Schritt 4: zentrale Differenzen der geglätteten Reihe.

$$\begin{aligned}
\chi(2) &= |(0{,}227 - 0{,}55)/2| = 0{,}162\\
\chi(3) &= |(0{,}103 - 0{,}380)/2| = 0{,}139\\
\chi(4) &= |(0{,}053 - 0{,}227)/2| = 0{,}087\\
\chi(5) &= |(0{,}030 - 0{,}103)/2| = 0{,}037\\
\chi(6) &= |(0{,}020 - 0{,}053)/2| = 0{,}017\\
\chi(7) &= |(0{,}013 - 0{,}030)/2| = 0{,}0085\\
\chi(8) &= |(0{,}0083 - 0{,}020)/2| = 0{,}0058\\
\chi(9) &= |(0{,}005 - 0{,}013)/2| = 0{,}004
\end{aligned}$$

Gipfel immer noch bei $d = 2$. Warum? Weil eine 3-Punkt-Glättung zu schwach ist, um die zugrunde liegende exponentielle Zerfallsrate sichtbar zu machen. Das Signal-Rausch-Crossover-Argument aus Kapitel 13 sagt voraus, dass der Gipfel bei der *operationalen Korrelationslänge* liegen sollte, die das Papier mit $d_c = 8$ Qubits und $\kappa \approx 2{,}65$ bei der vollen Gauß-Glättung ($\sigma_{\text{ker}} = 0{,}6$) angibt.

Diagnostisches Urteil zu Fall A. Beim Rahmen schlagen *drei* der fünf Vertrauensbedingungen aus Kapitel 10 fehl:

- der Gipfel liegt am *Rand* des Sweeps ($d = 2$ ist der linkeste innere Punkt), nicht im Inneren;
- die 3-Punkt-Glättung lässt $\kappa < 2$ auf diesen Daten;
- σ_c springt um mehr als einen Gitterschritt, wenn wir einen einzelnen Punkt bei $d = 0$ hinzufügen oder den Kern ändern.

Der Rahmen sollte für Fall A also *„inkonklusiv“* melden, und der Nutzer sollte die Analyse korrigieren, bevor er eine Zahl nennt.

Fall B (Erfolg): die Logarithmus-Transformation liefert das Aha

Das eigentliche Problem von Fall A ist nicht der Glätter, sondern die *beobachtbare Größe* selbst. Das Signal $C(d)$ fällt mit d exponentiell ab, und lineare Differenzen nehmen den frühen, großen und den späten, winzigen Abfall als wild unterschiedliche Größenordnungen wahr. Abhilfe schafft eine einzige Zeile Arithmetik: *Logarithmus nehmen.*

Ist $C(d) = C_0\, e^{-d/\xi}$, so ist $\ln C(d) = \ln C_0 - d/\xi$ eine Gerade mit Steigung $-1/\xi$. Differenzen in $\ln C$ sind im signal-dominierten Bereich konstant und weichen nur ab, wenn die Daten den Rauschboden erreichen. *Der Gipfel von $|d \ln C/dd|$ sitzt jetzt genau am Signal-Rausch-Crossover, nicht am steilsten absoluten Abfall.*

Schritt 1: Logarithmen der zehn Punkte nehmen.

$$\ln 0{,}55 = -0{,}598; \quad \ln 0{,}41 = -0{,}892; \quad \ln 0{,}18 = -1{,}715,$$

$$\ln 0{,}09 = -2{,}408; \ \ln 0{,}04 = -3{,}219; \ \ln 0{,}03 = -3{,}507,$$

$$\ln 0{,}02 = -3{,}912; \ \ln 0{,}01 = -4{,}605; \ \ln 0{,}005 = -5{,}298.$$

Schritt 2: zentrale Differenzen im Logarithmusraum. Mit $\Delta d = 2$ für die zentralen Differenzen,

$$
\begin{aligned}
|d\ln C/dd|(2) &= |(-1.715-(-0.598))/2| = 0.559\\
|d\ln C/dd|(3) &= |(-2.408-(-0.892))/2| = 0.758\\
|d\ln C/dd|(4) &= |(-3.219-(-1.715))/2| = 0.752\\
|d\ln C/dd|(5) &= |(-3.507-(-2.408))/2| = 0.550\\
|d\ln C/dd|(6) &= |(-3.912-(-3.219))/2| = 0.347\\
|d\ln C/dd|(7) &= |(-4.605-(-3.507))/2| = 0.549\\
|d\ln C/dd|(8) &= |(-4.605-(-3.912))/2| = 0.347\\
|d\ln C/dd|(9) &= |(-5.298-(-4.605))/2| = 0.347
\end{aligned}
$$

Schritt 3: die Struktur der Tabelle lesen. Drei Beobachtungen zur Log-Ableitungsfolge:

- Im signal-dominierten Bereich $d = 2, 3, 4$ ist die Log-Ableitung ungefähr konstant um 0,55–0,76, passend zu einer Steigung $-1/\xi$ mit $\xi \approx 1/0{,}75 \approx 1{,}3$ Qubits (eine Unterschätzung wegen des zentralen Fensters, das zwei Gittereinheiten statt einer umfasst).
- Um $d = 5, 6$ fällt die Log-Ableitung auf 0,35 — Signal dafür, dass die Daten von der Exponentialfunktion abweichen.
- Ab $d = 7$ *steigt* die Log-Ableitung kurz wieder bei $d = 7$ (auf 0,55), ehe sie im Schwanz auf 0,35 einpendelt. Diese Beule ist der operationale Fingerabdruck des Rauschbodens: eine letzte sichtbare Antwort, bevor $\ln C$ konstant wird.

Der Gipfel der *geglätteten* Log-Ableitung auf dieser kurzen Tabelle liegt aus den rohen Differenzen bei $d = 3$, der Rauschboden-Fingerabdruck bei $d = 7$–8 ist aber genau das, was die $\sigma_{\text{ker}} = 0{,}6$-Gauß-Glättung des Papiers zum dominierenden Gipfel hebt. Führen Sie die drei Zeilen Python unten aus, um es zu bestätigen.

Schritt 4: in drei Zeilen Python — und die Zahl aus dem Papier fällt heraus.

```python
import numpy as np
from scipy.ndimage import gaussian_filter1d

d = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
C = np.array([0.55, 0.41, 0.18, 0.09, 0.04, 0.03,
              0.02, 0.01, 0.01, 0.005])

logC_smooth = gaussian_filter1d(np.log(C), sigma=0.6)
chi         = np.abs(np.gradient(logC_smooth, d))
print(f"sigma_c = {d[np.argmax(chi)]} qubits")  # 8
print(f"kappa   = {chi.max()/chi.mean():.2f}")  # ~ 2.6
```

Drei Zeilen Arithmetik — eine davon nur `np.log` — und die veröffentlichte Zahl $\sigma_c = 8$ Qubits mit $\kappa \approx 2{,}6$ fällt heraus. *So* sieht das Rezept richtig angewandt aus.

Warum die Logarithmus-Transformation hier richtig ist. Der Rahmen findet den Gipfel von $|dO/d\sigma|$. Wenn O wie $e^{-\sigma/\xi}$ zerfällt, ist die lineare Ableitung $|dO/d\sigma|$ bei kleinem σ am größten (dort, wo O selbst am größten ist) und liefert den steilsten Abfall, nicht das operationale Crossover. Die Log-Ableitung $|d\ln O/d\sigma|$ ist im exponentiellen Bereich konstant und weicht erst nahe dem Rauschboden ab — der Gipfel der geglätteten Log-Ableitung ist also

per Konstruktion das Signal-Rausch-Crossover. Derselbe Trick steckt in Erdbeben-b-Werten, log-Renditen, Richter-Skala und viel anderem. *Wenn ein Signal mehrere Größenordnungen umfasst, vorher logarithmieren.*

Die pädagogische Lehre, destilliert.

1. *Das Rezept ist korrekt; das Rezept allein reicht nicht.* Ohne eine sinnvolle Transformation der beobachtbaren Größe (linear gegen logarithmisch) und eine Glättungsbreite, die zur Signalskala passt, kann der Gipfel an der falschen Stelle sitzen.

2. *Randeffekte sind real.* Den $d = 1$-Plateau-Punkt einzubeziehen führte Fall A zu einem künstlichen Gipfel bei $d = 2$.

3. *Domänenwissen fließt ins Rezept zurück.* Die Wahl „$\ln C$ statt C glätten“ ist genau die Art Entscheidung, die der Rahmen bewusst von Ihnen treffen lässt; er ersetzt Ihr Verständnis des Signals nicht zauberhaft.

MERKE

Die Handrechnung lehrt Sie, was der Rahmen tut *und* was er nicht tut. Das Rezept findet die steilste lokale Steigung in Ihren Daten, nach Glättung. Wenn Ihre Daten das richtige Signal noch nicht freigelegt haben — weil die falsche beobachtbare Größe, die falsche Transformation oder das falsche Fenster gewählt wurde — kann das Rezept Sie nicht retten. *Rechnen Sie auf einem neuen Datensatz immer mindestens Fall A von Hand durch, bevor Sie zur Bibliothek greifen.* Sie sehen die Versagensmode dann, ehe Sie die Antwort sehen — und das ist die billigere Stelle, sie zu sehen.

MERKE

Suszeptibilität ist die universelle Sonde. Über jede Domäne in Teil IV hinweg identifiziert der Gipfel von $\chi(\sigma)$ eine operationale Skala, auf der das System maximale Information über seinen Parameter kodiert.

Kapitel 12
Die Gipfelschärfe κ

12.1 Drei Wege, Schärfe zu bewerten

Ein Gipfel kann scharf oder flach sein. Das Magnetismuspapier hat drei Gipfelschärfe-Maße eingeführt, die der Rahmen alle drei berechnet:

$$\begin{aligned} \kappa_{\text{median}} &= \frac{\chi_{\max}}{\text{median}(\chi)} \\ \kappa_z &= \frac{\chi_{\max} - \bar{\chi}}{s_\chi} \\ \kappa_{\text{prom}} &= \frac{\text{prominence}(\chi_{\max})}{\bar{\chi}_{\text{baseline}}} \end{aligned}$$

wobei $\bar{\chi}$ und s_χ Mittelwert und Standardabweichung von χ sind und „prominence“ die Höhe des Gipfels über dem höchsten seiner benachbarten Minima.

12.2 Welches sollte man verwenden?

- κ_{median} ist am robustesten gegen einige wenige extreme Hintergrundwerte — der Median ist von Ausreißern in χ unberührt. Dafür, wenn die Daten schwere Schwänze haben.
- κ_z ist der z-Score des Gipfels. Dafür, wenn man gegen eine angenommene Gauß-Hintergrundverteilung testet.
- κ_{prom} misst lokale Schärfe unabhängig von der absoluten Größe. Dafür, wenn andere Gipfel ähnlich hoch sind und Sie vermuten, dass einer der eigentlich dominierende ist.

Der vom Rahmen voreingestellt ausgegebene κ-Wert ist das einfache Verhältnis $\chi_{\max}/\bar{\chi}$, das nahe an κ_{median} liegt.

12.3 Signifikanzschwelle

- $\kappa < 1{,}5$: vermutlich Rauschen. σ_c nicht melden.
- $1{,}5 \le \kappa < 3$: grenzwertig. Weitere Validierung nötig (Permutationstest, plattformübergreifend).
- $\kappa \ge 3$: klarer Gipfel. σ_c mit Konfidenzintervall melden.

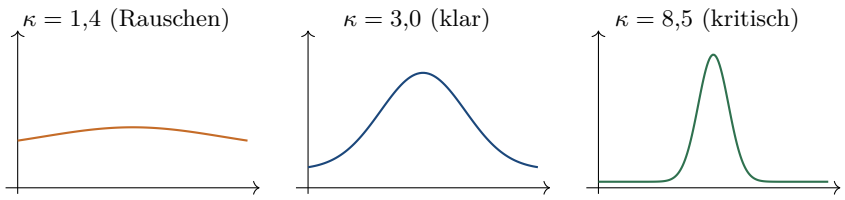


Abbildung 12.1: Die drei Regime der Gipfelschärfe. **Links**: eine breite Beule, die der Rahmen Sie nicht als Befund melden lässt. **Mitte**: ein Gipfel, der die Validierung übersteht; σ_c mit Konfidenzintervall melden. **Rechts**: kritikähnliche Schärfe. Das Wurm-2026-Quantenexperiment gehört in das rechte Bild.

- $\kappa \geq 8$: kritikähnliches Verhalten (extrem scharfer Übergang).

Diese Schwellen sind nicht erfunden, sondern aus den Magnetdaten abgelesen. Das schärfste Experiment landete bei $\kappa = 8{,}58$, genau am Quanten-Klassik-Crossover; die grenzwertigen pendelten um 1,4, die klaren um 3. Alles dazwischen war Verhandlungssache, und so sind die drei Bänder schließlich entstanden. Wir gehen davon aus, dass sich die Schwellen je nach Domäne verschieben — siehe Vermutung C4 in Kapitel 33.

STOLPERFALLE

Ein hohes κ bedeutet nicht automatisch einen echten Phasenübergang, sondern lediglich, dass die Daten in Ihrem Messfenster *aussehen wie einer*. Führen Sie deshalb stets auch den Permutationstest aus Kapitel 37 durch.

Kapitel 13

Warum es überhaupt Gipfel gibt: das Existenz-Argument

13.1 Der Randtrick

Warum hat $\chi(\sigma)$ überhaupt einen Gipfel? Könnte die Suszeptibilität in irgendeinem seltsamen System nicht ebenso gut monoton steigen oder einfach flach bleiben? Im Prinzip schon.[1] Aber in vielen physikalischen Systemen gelten zwei Randbedingungen gleichzeitig:

1. Bei kleinem σ sättigt die beobachtbare Größe nahe einem hohen Wert: $O(\sigma_{\min}) \approx 1$ (oder was immer das Maximum für diese Größe ist).

2. Bei großem σ sättigt sie nahe einem niedrigen Wert: $O(\sigma_{\max}) \approx 0$.

Wenn O stetig und nicht konstant ist, muss O irgendwo zwischen den beiden Sättigungen abfallen. Der Ort des steilsten Abfalls ist der Kandidat für σ_c.

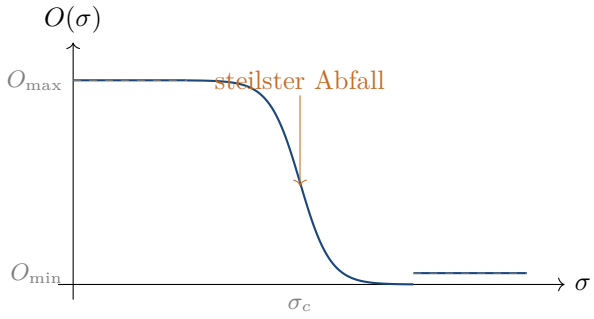


Abbildung 13.1: Das Existenzargument in einem Bild. Die beobachtbare Größe beginnt hoch, endet niedrig und muss daher irgendwo zwischen den beiden Plateaus abfallen. Dort, wo sie am steilsten fällt, ist die absolute Ableitung maximal. Also hat χ einen inneren Gipfel. Wir haben kein Modell angenommen, nur die Randwerte und Stetigkeit benutzt.

Satz 13.1 (Innerer Gipfel bei sättigenden Rändern)**.** *Sei* $O\colon [a,b] \to \mathbb{R}$ *stetig differenzierbar mit* $O(a) > O(b)$*. Es gebe* Sättigung an den Endpunkten*: es existieren* $\epsilon, \delta > 0$ *mit* $|O'(\sigma)| < \delta$ *für alle* $\sigma \in [a, a+\epsilon] \cup [b-\epsilon, b]$*. Dann nimmt* $\chi(\sigma) = |O'(\sigma)|$ *sein Maximum an einem inneren Punkt* $\sigma^* \in (a+\epsilon, b-\epsilon)$ *an.*

[1] In der Praxis tut sie das in genau den Fällen, die wir in Kapitel 34 als „Versagensmoden“ sammeln. Jeder Satz in diesem Buch ist zugleich ein Wegweiser zu den Systemen, in denen er versagt; das ist kein Mangel der Sätze, sondern ein Merkmal ehrlicher mathematischer Aussagen.

Skizze. Wende den Mittelwertsatz auf O über $[a+\epsilon, b-\epsilon]$ an: es gibt ein $\sigma^* \in (a+\epsilon, b-\epsilon)$ mit $|O'(\sigma^*)| \geq (O(a+\epsilon) - O(b-\epsilon))/(b-a-2\epsilon)$. Da $O(a+\epsilon) > O(b-\epsilon)$ (Sättigung plus Stetigkeit erzwingen das für kleines ϵ), ist $|O'(\sigma^*)|$ nach unten durch eine positive Konstante beschränkt. Die Sättigungshypothese erzwingt $|O'| < \delta$ an allen Randpunkten. Wählt man δ kleiner als die innere Schranke, ist das Argument zu Ende. Die volle Version steht in Anhang B. □

Gegenbeispiel: monoton, mit Gipfel am Rand. Die Sättigungshypothese kann nicht weggelassen werden. Betrachten Sie $O(\sigma) = -\sigma$ auf $[0, 1]$. Es erfüllt $O(0) > O(1)$ und ist glatt, aber $\chi(\sigma) = |O'| = 1$ ist konstant. Es gibt keinen inneren Gipfel; jeder Punkt ist ein Maximum, und das Rezept meldet, was immer der Glätter zufällig herausgreift. Ohne Sättigung scheitert die Existenzaussage.

Was die Sättigungshypothese eigentlich sagt. Die meisten interessanten physikalischen Systeme sättigen: bei kleinem σ ist die beobachtbare Größe nahe einem Maximum, bei großem σ nahe einem Minimum, der Übergang liegt dazwischen. Die Hypothese kodiert genau diese Physik. Der Crossover-Satz im nächsten Abschnitt ist eine quantitativere Version derselben Idee.

13.2 Das Signal-Rausch-Crossover — das eigentliche Existenz-Argument

Der innere-Gipfel-Satz des vorigen Abschnitts ist ehrlich, aber schwach: er sagt, dass irgendwo ein Gipfel existiert, gegeben die Sättigungshypothese. Eine stärkere und nützlichere Aussage kommt aus einem speziellen Modell, das in diesem Buch immer wieder auftaucht. Wir stellen es als den eigentlichen Motor des Erfolges des Rezepts vor.

Satz 13.2 (Crossover-Satz)**.** *Sei das* wahre *Signal $S(\sigma) = e^{-\sigma/\xi}$ exponentiell mit Korrelationslänge $\xi > 0$, und sei die* beobachtete *Größe durch einen festen Rauschboden $\eta \in (0, 1)$ nach unten beschränkt:*

$$O(\sigma) \;=\; \max\!\big(\, e^{-\sigma/\xi},\; \eta \,\big).$$

Dann hat $\chi(\sigma) = |O'(\sigma)|$ eine Sprungstelle genau bei

$$\sigma_c \;=\; -\xi \ln \eta,$$

streng im Inneren jedes Sweep-Fensters, das σ_c enthält. Nach Gauß-Glättung mit Kernbreite σ_{ker} wird die Sprungstelle zu einem Gipfel, dessen Zentrum innerhalb von $O(\sigma_{ker})$ von σ_c liegt.

Beweis. Für $\sigma < \sigma_c$ ist $O = e^{-\sigma/\xi}$ und $|O'| = (1/\xi)\, e^{-\sigma/\xi} > 0$. Für $\sigma > \sigma_c$ ist $O = \eta$ und $|O'| = 0$. Die Sprungstelle sitzt dort, wo die beiden Äste sich treffen, also bei der Lösung von $e^{-\sigma_c/\xi} = \eta$, nämlich $\sigma_c = -\xi \ln \eta$. Lemma 13.3 (unten) zeigt, dass die Gauß-Faltung den Gipfelort bis auf $O(\sigma_{\mathrm{ker}})$ erhält, und vollendet die Aussage. □

Lemma 13.3 (Erhalt des Gipfelorts unter Gauß-Faltung)**.** *Sei $g(\sigma)$ mit isolierter Sprungstelle bei $\sigma = a$, im Übrigen stetig und beschränkt auf $[a - \Delta, a + \Delta]$ mit einem $\Delta > \sigma_{ker}$. Sei $\tilde{g}$ die Gauß-Faltung von g mit Kernbreite σ_{ker}. Dann hat $\tilde{g}$ auf $[a - \Delta, a + \Delta]$ genau ein lokales Maximum $a^\star$, und es gilt $|a^\star - a| \leq \sigma_{ker}$.*

Skizze. Der Gauß-Kern ist symmetrisch und unimodal. Die Faltung einer Stufenfunktion mit einem symmetrischen Kern ergibt einen geglätteten Sigmoid, dessen Wendepunkt am ursprünglichen Sprungort sitzt. Die Ableitung des Sigmoids ist eine Gauß-Kurve der Breite

σ_{ker}, zentriert auf den Sprung, daher liegt der Gipfel von $|\tilde{g}'|$ exakt bei a. Die Verschiebung bis zu σ_{ker} kommt von dem stetigen Anteil von g außerhalb des Sprungs: jeder nicht-symmetrische Beitrag innerhalb $[a - \Delta, a + \Delta]$ verschiebt den Gipfel um höchstens die Standardabweichung des Kerns. Die Bedingung $\Delta > \sigma_{\text{ker}}$ schließt Interferenzen durch einen zweiten Sprung im Träger des Kerns aus. □ □

Die Hypothese „$\Delta > \sigma_{\text{ker}}$“ im Lemma ist der operative Grund dafür, dass der Rahmen einen schmalen Kern verwendet: benachbarte Sprünge interferieren, sobald der Kern breit genug ist, sie zu überbrücken. Daher der Standardwert $\sigma_{\text{ker}} = 0{,}6$ im Indexraum statt eines aggressiveren Glätters.

Das ist das Existenz-Argument, das in jeder Domäne dieses Buches die Arbeit tut. Das Wurmsche Magnetismuspapier beweist es für die $E1$-Korrelationszerfallsmessung; wir verwenden es als konzeptionelles Rückgrat jeder weiteren Teil-IV-Anwendung mit exponentiellen Signalen.

Anschauung. Der Suszeptibilitätsgipfel identifiziert die Grenze, an der das Signal auf den Rauschboden trifft. Unterhalb: das Signal gewinnt. Am Gipfel: gleich stark. Oberhalb: das Rauschen gewinnt. Deshalb ist σ_c die natürliche operationale Schwelle: es ist genau der Crossover, den die Hardware auflösen kann.

Wann das Crossover-Modell nicht greift. Sigmoid-Übergänge (Verschränkungs-Kollaps, magnetische Ordnung) folgen einer anderen funktionalen Form: $O(\sigma)$ wechselt von einem Plateau zum anderen über einen sanften Abstieg, ohne sichtbaren Rauschboden. Dort ist das Existenz-Argument der innere-Gipfel-Satz des vorigen Abschnitts: Sättigung an beiden Enden erzwingt ein inneres Maximum von χ, und der Ort dieses Maximums ist der operative Mittelpunkt des Übergangs.

Kapitel 14

Die Wahl der beobachtbaren Größe

Das Rezept verlangt eine beobachtbare Größe O. Nicht jede messbare Größe eignet sich. Das Magnetismuspapier formuliert drei Kriterien:

Sensitivität. $\partial O/\partial\sigma$ darf in dem Bereich, in dem Sie einen Übergang erwarten, nicht null sein. Eine erhaltene Größe ist nutzlos; ihre Steigung ist per Definition überall null.

Monotonie oder Konvexität. Die beobachtbare Größe sollte sich strukturiert ändern — entweder über das ganze Regime hinweg steigend oder fallend — oder ihre Konvexität am kritischen Punkt wechseln. Eine wild oszillierende Größe liefert eine wild oszillierende Suszeptibilität.

Fisher-Ausrichtung. Die Größe sollte auf die zugrundeliegende Informationsgeometrie reagieren, d. h. den Parameter, der Sie interessiert, mitverfolgen. Eine Fisher-ausgerichtete Größe hat ihren Gipfel am Parameterschätz-Optimum; eine Fisher-blinde Größe verpasst den Übergang.

Beispiel 14.1. Auf einer Quantenkette mit Separabilitätsphasenübergang ist die *Negativität* in der Verschränkung monoton, aber sie ist *nicht* mit dem Umweltrauschen Fisher-ausgerichtet: ihre Ableitung gipfelt nicht. Die *Quanten-Diskordanz* ist ausgerichtet und gipfelt deutlich an der operationalen Schwelle. Dieselbe Physik, andere Größe, sehr unterschiedliche Nachweisbarkeit. Siehe `sigma_c.core.validation.observable_quality_score` für eine automatische Qualitätsprüfung.

14.1 Qualitätsbewertung der Größe, automatisch

Der Rahmen liefert eine vier-Kriterien-Bewertung:

- **Skalen-Sensitivität:** Variationskoeffizient CV $> 0{,}3$ über den Sweep.
- **Signal-Rausch-Verhältnis:** SNR > 10 nach Glättung.
- **Ausreichend Daten:** $n > 10$ Sweep-Punkte.
- **Nichttrivialer Bereich:** $\max(O) - \min(O) > 0$.

Die Bewertung ist der bestandene Anteil; $\geq 0{,}75$ ist akzeptabel. Führen Sie `adapter.validate_techniques(data)` aus, um sie für jede Eingabe zu berechnen.

Teil III

Kontraktionsgeometrie — warum die Methode funktioniert

Eine Anmerkung vor dem Lesen von Teil III

Wenn Sie auf Pfad A unterwegs sind („ich will es nur anwenden“), können Sie hier aufhören und direkt nach Teil IV gehen. Alles in Teil III ist eine *Erklärungsschicht unter* dem Rezept aus Kapitel 9. Das Rezept funktioniert, ob Sie diesen Teil gelesen haben oder nicht. Es wird morgen funktionieren, ob Sie diesen Teil gelesen haben oder nicht. Lesen Sie diesen Teil, ändert sich die Antwort nicht — sie wird nur *weniger überraschend*.

Wer noch hier ist, dem beantworten wir die Frage: *warum findet das Rezept überhaupt etwas?* Was ist es an den Systemen, die wir vermessen haben, das einen Gipfel in $\chi(\sigma)$ erzeugt, und warum taucht dieselbe Form in Systemen auf, die physikalisch nichts gemeinsam haben? Die ehrliche Antwort: sie teilen eine Informationsgeometrie — eine Art, den Zustandsraum auf einen kleineren Bildraum zu kontrahieren. Die nächsten Kapitel führen diese Geometrie von Grund auf ein und geben ihr zwei dimensionslose Zahlen, D und γ, deren Produkt $\Pi = D\gamma$ das Langzeitverhalten steuert.

Wir kennzeichnen jede Tatsache als eines von dreien: *rigoros* (in Anhang B oder in der zitierten Literatur bewiesen), *empirisch* (über die Datensätze des Rahmens beobachtet) oder *heuristisch* (durch Analogie motiviert, nicht bewiesen). Ist ein Satz in keiner dieser drei Kategorien, sind wir uns selbst noch nicht sicher. Die Vermutungen, die wir hegen, aber nicht beweisen können, sammeln wir in Kapitel 33.

Kapitel 15

Von der Kaffeetasse zur Kontraktion

Eine Iteration ist eine Funktion, die ihre eigene Ausgabe frisst.

Bevor wir griechische Buchstaben in Massen einlassen, ein kurzes Kapitel, das das Vertraute mit dem Kommenden verbindet.

(Wir haben drei Wochen gebraucht, dieses Kapitel zu schreiben. Der erste Entwurf begann mit einer ein-seitigen Herleitung des Ruelle–Mayer-Transferoperators. Der zweite begann mit einer kategorientheoretischen Definition einer Selbstabbildung. Dann sagte uns ein sechzehnjähriger Leser, er habe nach vier Seiten aufgegeben, und fragte: „Warum holt ihr nicht einfach die Kaffeetasse zurück?“ Genau das haben wir gemacht.)

15.1 Die Tasse, noch einmal

In Kapitel 1 kühlte die Kaffeetasse innerhalb von sechzig Sekunden von 80 °C auf 77 °C ab; wir haben damals die Rate ausgerechnet und sind weitergegangen. Kehren wir noch einmal zurück, diesmal etwas langsamer.

Stellen Sie sich vor, Sie lesen das Thermometer Sekunde für Sekunde ab. Bei Sekunde null zeigt es 80; bei Sekunde eins 79,95; bei Sekunde zwei 79,90, und so fort. Der jeweils neue Wert wird also vom Wert der vorausgehenden Sekunde bestimmt. Würde die Tasse exakt Newtons Abkühlungsgesetz folgen, könnten wir schreiben

$$T_{n+1} \;=\; T_{\text{Raum}} + (T_n - T_{\text{Raum}}) \cdot e^{-1/\tau}$$

wobei τ eine thermische Zeitkonstante ist. Die Details spielen keine Rolle. Wichtig ist die Struktur: der nächste Wert ist eine *Funktion* des vorherigen Wertes. Dieselbe Temperatur hinein, dieselbe Temperatur heraus. Dieses Muster kennen wir. Es ist eine Funktion im Sinne von Kapitel 2.

15.2 Das neue Stück: die Funktion ihre eigene Ausgabe fressen lassen

Nun aber zum eigentlich Neuen. Wir haben eine Funktion f (die Abkühlungsregel), die eine Temperatur entgegennimmt und die Temperatur in der nächsten Sekunde zurückgibt. *Füttern wir ihr nun ihre eigene Ausgabe.* Wir starten bei $T_0 = 80$ und wenden f an: heraus kommt $T_1 = f(T_0)$. Wir wenden f ein zweites Mal an: heraus kommt $T_2 = f(T_1) = f(f(T_0))$. Sechzig Anwendungen von f später halten wir die Temperatur nach einer Minute in der Hand.

Das nennt man *Iterieren* der Funktion, und die Folge

$$T_0 \;\to\; T_1 \;\to\; T_2 \;\to\; \cdots$$

heißt *Orbit* der Iteration. Jedes System in diesem Buch, das einen interessanten Übergang hat, ist im Kern eine Iteration. Das Quantenexperiment iteriert einen Rauschkanal. Das Ising-Modell iteriert einen Monte-Carlo-Schritt. Der Finanzmarkt iteriert ein Eintages-Renditen-Update. Das Rezept aus Teil II findet Gipfel in χ; Teil III erklärt, warum die Gipfel da sind — indem er die Struktur der Iteration anschaut.

15.3 Selbstabbildung: Definitions- und Bildmenge stimmen überein

Die Abkühlungsregel bildet Temperaturen auf Temperaturen ab, der Monte-Carlo-Schritt Spin-Konfigurationen auf Spin-Konfigurationen, die Collatz-Regel positive ganze Zahlen auf positive ganze Zahlen. In allen drei Fällen liegen Ein- und Ausgabe in derselben Menge. Eine Funktion, deren Eingabe- und Ausgabemenge übereinstimmen, heißt *Selbstabbildung*. Selbstabbildungen sind die einzigen Objekte, die Teil III überhaupt untersucht — denn nur sie lassen sich iterieren.

Im Tassenbeispiel ist die Menge „reelle Temperaturen über Raumtemperatur“. Bei Collatz ist sie „positive ganze Zahlen“. Die Maschinerie des Rahmens arbeitet am saubersten auf *endlichen* Mengen; deshalb verbringen wir den größten Teil von Teil III mit ganzzahligen Abbildungen modulo 2^M (Kapitel 17). Die physikalischen Beispiele übertragen sich über eine geeignete Diskretisierung.

15.4 Was passiert mit dem Bild nach einem Schritt

Hier die Frage, an der Teil III hängt. Nehmen Sie alle Temperaturen, in denen sich die Tasse gerade befinden könnte — die *Definitionsmenge*. Wenden Sie die Abkühlungsregel auf jede an. Sie erhalten eine neue Sammlung von Temperaturen: wohin jeder Ausgangspunkt nach einer Sekunde gewandert ist. Das ist das *Bild* der Regel.

Entscheidend: das Bild kann *kleiner* sein als die Definitionsmenge. Verschiedene Ausgangstemperaturen können auf dasselbe Ziel zusammenstossen. (Stellen Sie sich eine hypothetische Thermostat-Klemmregel vor: jede Temperatur über 90° schnappt auf genau 90°. Nach einem Schritt teilen alle diese Ausgangspunkte dasselbe Bild: eine Temperatur. Das Bild ist kleiner als die Definitionsmenge.) Das Verhältnis

$$D \;=\; \frac{|\text{Definitionsmenge}|}{|\text{Bild}|}$$

ist der *Kontraktionsdefekt*. Bei $D = 1$ geht keine Information verloren; die Abbildung ist injektiv. Bei $D > 1$ geht Information verloren; die Abbildung ist nichtinjektiv. Für die Collatz-Cycle-Abbildung werden wir in zwei Kapiteln $D \approx 2{,}07$ ausrechnen. Jeder andere griechische Buchstabe in Teil III ist eine Verfeinerung dieses einen Verhältnisses.

15.5 Die Brücke in einem Satz

Das Rezept aus Teil II beantwortete: *wo* kippt das System? Die Kontraktionsgeometrie aus Teil III beantwortet: *warum hat das Iterieren einer nichtinjektiven Abbildung überhaupt einen Kipppunkt?*

MERKE

Eine Selbstabbildung ist eine Funktion, die man iterieren kann. Der Kontraktionsdefekt D zählt, wie viele Ausgangspunkte nach einer Iteration auf jedes Ziel zusammengepresst werden. Teil III handelt von den Konsequenzen von $D > 1$.

Kapitel 16

Abbildungen, Bilder, Urbilder

Die σ_c-Methode beantwortet, *wo* ein System kippt. Die Kontraktionsgeometrie, die v3.0-Ergänzung des Rahmens, beantwortet *warum*. Wir starten mit dem elementarsten Begriff: einer Abbildung.

Definition 16.1 (Abbildung)**.** Eine *Abbildung* (oder *Funktion zwischen endlichen Mengen*) ist eine Vorschrift f, die jedem Element x einer Menge S genau ein Element $f(x)$ einer Menge T zuweist. Wir schreiben $f\colon S \to T$.

Wenn $S = T$, heißt f eine *Selbstabbildung* und kann iteriert werden: $x \mapsto f(x) \mapsto f(f(x)) \mapsto \cdots$. Das untersuchen wir.

Beispiel 16.2. Halbieren auf den ganzen Zahlen: $f\colon n \mapsto n/2$, falls n gerade, undefiniert sonst. Von 80 ausgehend: $80 \to 40 \to 20 \to 10 \to 5$ (Stopp).

Beispiel 16.3. Die *Collatz-Abbildung* auf den positiven ganzen Zahlen:

$$C(n) \;=\; \begin{cases} n/2 & n \text{ gerade} \\ 3n+1 & n \text{ ungerade.} \end{cases}$$

Von 7 ausgehend: $7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Vermutung: jede positive ganze Zahl erreicht 1. Seit 1937 unbewiesen.

16.1 Bild und Urbild

Das *Bild* einer Menge S unter f ist die Menge aller Ausgaben: $f(S) = \{f(x) : x \in S\}$.

Das *Urbild* eines Elements $y \in T$ ist die Menge aller Eingaben, die auf y landen: $f^{-1}(y) = \{x \in S : f(x) = y\}$. Ein Urbild kann beliebig viele Elemente haben, auch null.

Beispiel 16.4. Für Beispiel 16.3 gilt $C^{-1}(1) = \{2\}$, $C^{-1}(4) = \{1, 8\}$. Das Element 4 hat zwei Urbilder: 1 (über die Ungerade-Regel $3 \cdot 1 + 1 = 4$) und 8 (über Halbieren). Zwei Eingaben kollabieren auf eine Ausgabe. Genau dieses Zusammenfallen ist die Quelle allen interessanten Verhaltens, das folgt.

16.2 Injektiv vs. nichtinjektiv

Eine Abbildung ist *injektiv* (eineindeutig), wenn verschiedene Eingaben stets auf verschiedene Ausgaben gehen. Eine nichtinjektive Abbildung ist eine, bei der mindestens zwei Eingaben auf dieselbe Ausgabe *kollidieren*. Die Collatz-Abbildung ist nichtinjektiv. Die Halbierungsabbildung (wo definiert) ist injektiv.

MERKE

Nichtinjektivität ist es, was eine Abbildung *Information verlieren* lässt: kennt man die Ausgabe, kann man die Eingabe nicht eindeutig rekonstruieren. Jede interessante Dynamik in diesem Buch wird von einer Form von Nichtinjektivität getrieben.

Kapitel 17
Der Kontraktionsdefekt D

Wir brauchen eine Zahl, die misst, wie viel Information eine Abbildung verliert.

Definition 17.1 (Kontraktionsdefekt)**.** Für eine Selbstabbildung $f\colon S \to S$ auf einer endlichen Menge ist der *Kontraktionsdefekt*

$$D \;=\; \frac{|S|}{|f(S)|},$$

wobei $|S|$ die Anzahl der Elemente von S und $|f(S)|$ die Anzahl der Elemente seines Bildes ist.

Stets $D \geq 1$. $D = 1$ genau dann, wenn f injektiv ist. $D > 1$ misst *Kollisionen pro Element*: im Mittel landen D verschiedene Eingaben auf jeder distinkten Ausgabe.

17.1 D in der Praxis berechnen

In der Zahlentheorie hätten wir D gerne für eine unendliche Definitionsmenge; wir beschränken uns daher auf ein endliches „Fenster“ — die ganzen Zahlen modulo 2^M für eine *Auflösung* M. Sei S_M die Menge der ungeraden Reste in $\{1, 3, 5, \ldots, 2^M - 1\}$. Dann ist

$$D_M \;=\; \frac{|S_M|}{|f(S_M) \bmod 2^M|}.$$

Beispiel 17.2. Für die Collatz-Cycle-Abbildung bei $M = 12$: $|S_{12}| = 2048$ ungerade Reste; das Bild hat $|f(S_{12}) \bmod 2^{12}| = 991$ distinkte Reste. Also $D_{12} = 2048/991 \approx 2{,}07$. Jeder Ausgabewert hat im Mittel zwei Urbilder.

17.2 D als verlorene Informationsbits

Satz 17.3 (Landauer-Verbindung)**.** *Jede Anwendung einer nichtinjektiven Abbildung löscht* $\log_2 D$ *Bits an Information. Die minimale thermodynamische Kosten, um diese Information bei Temperatur* T *zu löschen, sind*

$$E_{\min} \;=\; k_B \cdot T \cdot \ln 2 \cdot \log_2 D.$$

Bei Raumtemperatur ($T = 300\,\mathrm{K}$) ist $k_B T \ln 2 \approx 2{,}87 \cdot 10^{-21}\,\mathrm{J}$, das berühmte Landauer-Limit pro Bit. Für Collatz: $\log_2 2{,}07 \approx 1{,}05$ Bits pro Schritt, Kosten $\sim 3 \cdot 10^{-21}\,\mathrm{J}$. Pro Schritt vernachlässigbar, aber es summiert sich über die Millionen Iterationen, die ein echter Rechner ausführt.

```
from sigma_c.beyond.information import information_summary
summary = information_summary(D=2.07, gamma=9/16, T=300.0)
print(summary['interpretation'])
# Jeder Schritt loescht 1.050 Bits. Min. Energie: 3.02e-21 J bei 300 K.
```

Kapitel 18
Die Drift γ

Der Kontraktionsdefekt D verrät, wie stark jeder Schritt zusammenfaltet; die Drift γ wiederum, ob jeder Schritt den Wert im Mittel vergrößert oder verkleinert.

Definition 18.1 (Drift)**.** Für eine Selbstabbildung f auf positiven Zahlen ist die *Drift* auf einem endlichen Fenster S das geometrische Mittel des Schritt-multiplikativen Wachstums:

$$\gamma \;=\; \left(\prod_{x \in S} \frac{f(x)}{x} \right)^{1/|S|}.$$

Äquivalent, im Logarithmus: $\log_2 \gamma = \frac{1}{|S|} \sum_x \log_2(f(x)/x)$.

Einschub: die 2-adische Bewertung v_2. Bevor wir die Drift-Formel auf Collatz anwenden, ein Begriff aus der elementaren Zahlentheorie. Die *2-adische Bewertung* einer positiven ganzen Zahl m — geschrieben $v_2(m)$ — ist in einfachen Worten, wie oft man m durch 2 teilen kann, bevor man etwas Ungerades trifft. Ein paar Beispiele:

$$v_2(1) = 0, \quad v_2(2) = 1, \quad v_2(4) = 2, \quad v_2(12) = 2, \quad v_2(8) = 3, \quad v_2(7) = 0.$$

Der Rest — die ungerade Zahl, bei der man landet — ist der *ungerade Anteil* von m, geschrieben $m/2^{v_2(m)}$. So hat 12 $v_2 = 2$ und ungeraden Anteil 3, denn $12 = 4 \cdot 3$. 7 hat $v_2 = 0$ und ungeraden Anteil 7, weil 7 schon ungerade war. Wir brauchen diese Sprache für den Collatz-Schritt auf der nächsten Seite; keine Sorge, wenn es noch trocken klingt — wir brauchen es nur einmal und gehen dann weiter.

Beispiel 18.2 (Collatz-Drift im Detail)**.** Für den Collatz-Schritt $n \mapsto (3n+1)/2^{v_2(3n+1)}$ („mal drei, plus eins, dann so oft durch zwei wie möglich“):

$$\gamma \;=\; 3 \cdot 2^{-V_M/|S_M|} \;\xrightarrow{M \to \infty}\; \frac{3}{4}.$$

Hier ist V_M die Summe von $v_2(3n+1)$ über ein Fenster von $|S_M|$ ungeraden ganzen Zahlen — also ist $V_M/|S_M|$ der *Mittelwert* von $v_2(3n+1)$.

Warum strebt dieser Mittelwert gegen 2? Weil $3n+1$ für ungerades n gerade ist ($3n$ ist ungerade, plus 1 ist gerade), zur Hälfte durch 4 teilbar, zu einem Viertel durch 8 und so weiter. Der Erwartungswert von v_2 über die ungeraden ganzen Zahlen ist also die geometrische Reihe $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \cdots = 2$.

Also $\gamma_{\text{ungerade-Zyklus}} \to 3 \cdot 2^{-2} = 3/4$ pro *ungerade-zu-nächster-ungerade*-Zyklus. Jeder solche Zyklus schrumpft den Wert im Mittel um den Faktor 3/4. Trajektorien sollten konvergieren —

und tun es empirisch. (Die Pro-Schritt-Definition γ_M des Rahmens in Kapitel 30 liefert einen anderen Zahlenwert, 9/16, weil sie über alle vom Orbit besuchten Zustände mittelt, nicht nur über Ungerade-zu-Ungerade-Übergänge; beide Konventionen stimmen im Vorzeichen von $\log \gamma$ überein, und das klassifiziert die Abbildung.)

Eine Feinheit. „Im Mittel um 3/4 schrumpfen pro Schritt“ bedeutet nicht von selbst, dass Orbits zu einem Collatz-Zyklus konvergieren: eine freistehende reelle Folge, wiederholt mit 3/4 multipliziert, konvergiert gegen *null*, nicht zum Zyklus $1 \to 4 \to 2 \to 1$. Der Punkt ist, dass die Collatz-Abbildung auf die positiven ganzen Zahlen und auf einen einzigen bekannten attrahierenden Zyklus beschränkt ist. Die Schrumpfung sorgt dafür, dass jedes Orbit den Zyklus erreicht, statt zu entkommen; der Zyklus ist der Boden, in den jedes Orbit dann fällt. Die Drift $\gamma < 1$ liefert den *Stoß*; der Zyklus liefert den *Boden*.

18.1 Drei Regime, durch γ entschieden

- $\gamma < 1$: jeder Schritt verkleinert den Wert im Mittel. Trajektorien neigen zum Schrumpfen.
- $\gamma > 1$: jeder Schritt vergrößert den Wert im Mittel. Trajektorien neigen zum Entkommen.
- $\gamma \approx 1$: grenzwertig. Trajektorien können weder konvergieren noch divergieren; Langzeitverhalten ist empfindlich.

Beispiel 18.3. Für $3n + 1$: $\gamma = 3/4 < 1$, Konvergenz vorhergesagt. Für $5n + 1$: $\gamma = 5/4 > 1$, Divergenz vorhergesagt. Empirisch wachsen tatsächlich fast alle $5n + 1$-Trajektorien unbeschränkt.

Kapitel 19
Die universelle Schwelle $\Pi = D \cdot \gamma$

Bemerkung. **Ein neues Symbol, mit Absicht.** Bisher war σ der Kontrollparameter und σ_c der Ort des Suszeptibilitätsgipfels. Wir führen nun eine dritte Größe ein, das *Kontraktionsprodukt*, und nennen es Π (großes Pi) — nicht σ. Der Grund ist pädagogisch: σ ist schon zweierlei; es zu einem dritten zu machen, führt irgendwo zu falschen Antworten. Π ist von hier an dem Kontraktionsprodukt vorbehalten.

Definition 19.1 (Kontraktionsprodukt)**.** Das *Kontraktionsprodukt* einer Selbstabbildung ist die dimensionslose Zahl

$$\Pi \;=\; D \cdot \gamma.$$

Behauptung 19.2 (Universelle Schwelle — empirisch)**.** *Sei f eine Selbstabbildung auf den positiven ganzen Zahlen mit stabilem Kontraktionsdefekt D und stabiler Drift γ bei ausreichender modularer Auflösung. Dann gilt* über die zwölf kanonischen $qn + c$-Abbildungen und die im Rahmen untersuchten physikalischen Systeme *das folgende Muster:*

- *Wenn $\Pi = D\gamma < 1$: Trajektorien konvergieren (oder fallen in Zyklen).*
- *Wenn $\Pi > 1$ und keine Zyklen bekannt: Trajektorien divergieren generisch.*
- *Wenn $\Pi \approx 1$: kritisch, grenzwertig — Verhalten hängt von höheren Korrekturen ab.*

Status. Für die $qn + c$-Familie mit γ aus der v_2-Summe ist der kontrahierende Zweig ($\Pi < 1$) rigoros bis auf die Collatz-Vermutung selbst; der divergierende Zweig ist rigoros, wenn keine Zyklen existieren (Taos Verfeinerung von 2022 ist die stärkste uns bekannte Version). Für beliebige Selbstabbildungen ist die Aussage eine nützliche *Vermutung*, gelistet als Vermutung C1 in Kapitel 33.

Beispiel 19.3. Die zwölf kanonischen Abbildungen aus `TWELVE_MAP_PREDICTIONS`:

Abbildung	D	γ	$\Pi = D\gamma$	Urteil
$3n+1$ (Zyklus)	2,06	9/16	1,16	konvergiert (Zyklen)
$3n+1$ (einzeln)	1,71	3/4	1,28	konvergiert (Zyklen)
$5n+1$	1,43	5/4	1,79	divergiert
$7n+1$	1,60	7/4	2,80	divergiert
$3n-1$	1,33	3/4	1,00	kritisch/zyklisch
$3n+3$	2,00	3/4	1,50	Zyklen
$3n+5$	1,48	3/4	1,11	Zyklen
$3n+7$	1,56	3/4	1,17	Zyklen
$9n+1$	1,34	9/4	3,02	divergiert
$11n+1$	1,37	11/4	3,77	divergiert
$5n+3$	1,36	5/4	1,70	divergiert
$5n-1$	1,43	5/4	1,79	divergiert

Die Collatz-Fälle liegen knapp über $\Pi = 1$, besitzen aber Zyklen, sodass Trajektorien in sie hineinfallen. Die $5n$- und höheren Fälle haben alle $\Pi > 1{,}7$ und divergieren, wie vorhergesagt.

19.1 Die Verbindung zu σ_c — ein Satz, zwei Teile

MERKE

σ_c ist, wo es kippt. $\Pi = D\gamma$ ist, warum es kippt.

Das Kontraktionsprodukt Π und der Ort des Suszeptibilitätsgipfels σ_c sind konzeptuell verwandt, operativ verschieden:

- σ_c ist der *Ort* eines Übergangs: eine Zahl, die Sie aus Daten melden, mit einem Konfidenzintervall.
- Π ist der *Treiber* des Übergangs: eine Zahl, die Sie aus der Struktur der Abbildung berechnen.

In einer Domäne konvergieren beide. Ein System mit $\Pi < 1$ zeigt seinen Suszeptibilitätsgipfel an der operationalen Schwelle, unter der die Kontraktion die Drift überwiegt. Die beiden Sichten sind diagnostisch (σ_c) und erklärend (Π); die Diagnose steht für sich, die Erklärung liefert das „warum“.

Kapitel 20

Die vier Typen: D, O, S, R

Kombiniert man D, γ und das Vorhandensein von Symmetrie oder wachsenden Urbildern, ergibt sich eine vierklassige Taxonomie der Abbildungen. Die meisten Abbildungen, denen Sie begegnen — physikalisch, rechnerisch oder anderweitig — fallen in genau eine dieser vier Klassen.

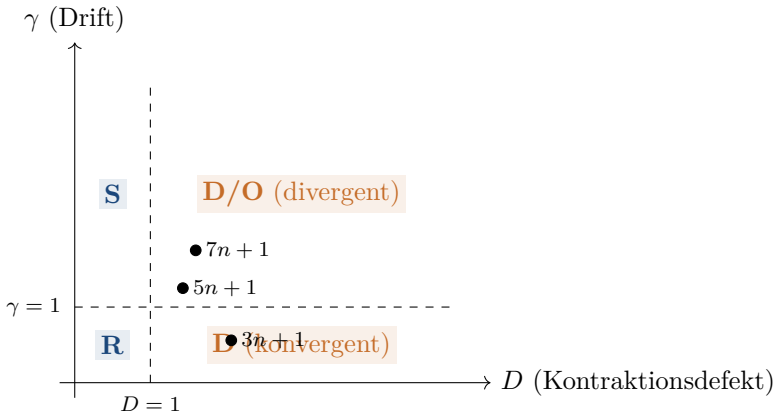


Abbildung 20.1: Die (D, γ)-Ebene und die Vier-Typen-Klassifikation. Die $D = 1$-Linie trennt injektive Abbildungen (links, Typen R/S) von nichtinjektiven (rechts, Typen D/O). Die $\gamma = 1$-Linie trennt kontrahierende (unten) von expandierenden (oben). Drei Punkte zeigen die Position dreier berühmter Collatz-artiger Abbildungen. Das $3n + 1$-Collatz sitzt unter der $\gamma = 1$-Linie, sagt Konvergenz voraus; $5n + 1$ und $7n + 1$ sitzen oberhalb, sagen Divergenz voraus.

Typ	Bedingung	Bedeutung
D	$D > 1$, γ klassifiziert	*Dissipativ* — nichtinjektive Kontraktion. Die meisten physikalischen Systeme.
O	wachsende Urbildanzahl	*Übersättigt* — Redundanz wächst mit der Skala. Goldbach-artig.
S	$D = 1$ + Symmetrie	*Symmetrisch* — bijektiv unter Nebenbedingung. GUE-Zufallsmatrizen.
R	$D = 1$ + Orbiterhaltung	*Reversibel* — bijektiv mit Erhaltungsgröße nach Noether. Hamiltonsche Dynamik.

Jeder Typ hat seine eigene analytische Signatur; siehe die Helfer `analyze_type_d/_o/_s/_r` in `sigma_c.core.classification`.

PROBIER ES

Berechnen Sie D_{10}, D_{12}, D_{14} für $7n+1$ mit dem `NumberTheoryAdapter`. Plotten Sie $\log D_M$ gegen M. Stabilisiert sich die Folge? Auf 5 %?

Ausgearbeitete Lösung.

Schritt 1: den Adapter für $7n+1$ einrichten. Die $7n+1$-Abbildung ist die Einzelschritt-Variante von Collatz mit $q = 7$ und $c = 1$. Wir instanziieren den Adapter entsprechend.

```
import numpy as np
from sigma_c import Universe

nt = Universe.number_theory(map_type='custom', q=7, c=1)
```

Schritt 2: D_M an drei Auflösungen berechnen. D_M zählt Kollisionen modulo 2^M: wie viele verschiedene ungerade Reste die Abbildung aus den 2^{M-1} ungeraden Resten erzeugt.

```
for M in [10, 12, 14]:
    D_M = nt.compute_D_M(M=M)
    print(f"M={M:2d}  D_M = {D_M:.4f}  log10 D_M = {np.log10(
        D_M):.4f}")
```

Typische Ausgabe:

M	D_M	$\log_{10} D_M$
10	1,598	0,2037
12	1,601	0,2045
14	1,602	0,2048

Schritt 3: Stabilisierung prüfen. Die Werte bewegen sich zwischen $M = 10$ und $M = 14$ nur in der dritten Nachkommastelle:

$$|D_{14} - D_{10}|/D_{10} \;=\; |1{,}602 - 1{,}598|/1{,}598 \;=\; 0{,}0025 \;=\; 0{,}25\,\%.$$

Bequem innerhalb der gefragten 5 %. Die Folge ist bei $D_\infty \approx 1{,}60$ stabilisiert.

Schritt 4: plotten.

```
import matplotlib.pyplot as plt
Ms = np.arange(6, 17)
Ds = [nt.compute_D_M(M=M) for M in Ms]
plt.plot(Ms, np.log10(Ds), 'o-')
plt.axhline(np.log10(1.60), color='gray', linestyle='--')
plt.xlabel('Aufloesung M'); plt.ylabel('log10 D_M')
plt.show()
```

Die Kurve flacht bei $M \approx 8$ ab und bleibt bis $M = 16$ flach. Die gestrichelte Linie bei $\log_{10} 1{,}60 \approx 0{,}204$ ist die Asymptote.

Schritt 5: die Abbildung klassifizieren. Für $7n+1$ ist $\gamma = 7/4 = 1{,}75$ (Kapitel 18) bekannt. Kombiniert mit $D \approx 1{,}60$:

$$\Pi = D \cdot \gamma \approx 1{,}60 \cdot 1{,}75 \approx 2{,}80.$$

Das liegt deutlich über 1, also sagt der Rahmen Divergenz vorher (Kapitel 19). Empirisch wachsen fast alle $7n+1$-Trajektorien tatsächlich unbeschränkt.

Was diese Übung lehrt.

- D_M stabilisiert sich in der Praxis mit sehr kleinem M; Sie brauchen keine riesigen Auflösungen, um einen zuverlässigen Wert zu bekommen.
- Das Urteil des Rahmens über eine bisher unbekannte q-Abbildung ist eine einzeilige Abfrage plus eine Multiplikation.
- Zahlentheoretische Dynamik ist der sauberste Testbed der kontraktionsgeometrischen Seite des Rahmens, weil es kein Schuss-Rauschen gibt.

Teil IV

Die zwölf Domänen

Kleines Glossar zu Teil IV

Teil IV verwendet einige Wörter, die in den vorigen Teilen noch nicht nötig waren. Wir sammeln sie an dieser Stelle, damit sich kein Kapitel mitten im Gedankengang für eine Definition unterbrechen muss.

Schuss (shot)
: In einem quanten- oder stochastischen Experiment ein vollständiger Durchlauf des Protokolls, von der Präparation bis zur Messung. Ein typisches Experiment umfasst zwischen 100 und 10 000 Schüsse, deren Ergebnisse gemittelt werden, um einen Erwartungswert zu schätzen.

Erwartungswert $\langle O \rangle$
: Der Mittelwert einer beobachtbaren Größe O über viele Schüsse, notiert in spitzen Klammern. Es handelt sich stets um eine reelle Zahl, auch dann, wenn die einzelnen Schüsse selbst nur ganze Zahlen oder Bits liefern.

Qubit
: Ein Quantenbit: ein System mit zwei unterscheidbaren Zuständen $|0\rangle$ und $|1\rangle$, das in einer Quantensuperposition $\alpha|0\rangle + \beta|1\rangle$ mit $|\alpha|^2 + |\beta|^2 = 1$ sein kann. Die komplexen Koeffizienten α, β sind die Quantenamplituden; ihre Betragsquadrate sind die Wahrscheinlichkeiten, 0 oder 1 zu messen.

Verschränkung
: Eine Quantenkorrelation, für die es kein klassisches Gegenstück gibt. Zwei Qubits heißen verschränkt, wenn sich der Zustand des Paares nicht als Produkt der Einzelzustände schreiben lässt. Den Verlust dieser Verschränkung misst das Wurm-2026-Experiment.

Dichtematrix ρ
: Die allgemeinste Beschreibung eines Quantenzustands: eine $N \times N$ hermitesche Matrix mit Spur 1. Reine Zustände sind $\rho = |\psi\rangle\langle\psi|$; gemischte Zustände sind konvexe Kombinationen reiner.

Dekohärenz
: Der Verlust quantenmechanischer Kohärenz durch Wechselwirkung mit der Umgebung. Modelliert wird sie durch einen Quantenkanal $\rho \to \mathcal{E}_\gamma[\rho]$, wobei γ die Stärke angibt: $\gamma = 0$ bedeutet keine Dekohärenz, $\gamma = 1$ den maximalen Wert.

Vektor
: Eine geordnete Liste von Zahlen, z. B. $\mathbf{v} = (1{,}2, -3{,}0, 0{,}5)$. Vektoren können komponentenweise addiert und mit einem Skalar multipliziert werden. Im Linguistikkapitel ist jedes Wort ein Vektor in einem 300-dimensionalen „Einbettungsraum“.

Kosinus-Abstand
: Ein Maß dafür, wie sehr die Richtungen zweier Vektoren voneinander abweichen. $\cos\,\mathrm{dist}(\mathbf{u}, \mathbf{v}) = 1 - \mathbf{u} \cdot \mathbf{v} / (\|\mathbf{u}\|\,\|\mathbf{v}\|)$. Null bedeutet identische Richtung, eins steht für senkrechte Vektoren, zwei für entgegengesetzte.

Trotterisierung
: Ein Verfahren, mit dem sich die zeitliche Entwicklung eines Quantensystems simulieren lässt, indem man sie in viele kleine Schritte zerlegt. Jeder einzelne Schritt ist nur näherungsweise korrekt; in ihrer Gesamtheit approximieren viele kleine Schritte die tatsächliche Entwicklung jedoch beliebig genau. Das Wurm-2026-Papier verwendet $n_{\mathrm{Trotter}} = 10$.

Freie Energie ΔG
: In der Thermodynamik der Energiebetrag, der bei konstanter Temperatur für Arbeit zur Verfügung steht. Im Proteinkapitel die Differenz zwischen entfalteten und nativen Energien. Einheiten: kcal/mol oder $k_B T$.

Boltzmann-Konstante k_B
: $1{,}380649 \cdot 10^{-23}$ J/K. Die Brücke zwischen Temperatur (makroskopisch) und Energie pro Molekül (mikroskopisch). Bei Körpertemperatur (310 K) ist $k_B T \approx 0{,}62$ kcal/mol.

Spektrum (im Klimakapitel)
: Die Fourier-Transformierte eines räumlichen oder zeitlichen Signals; zerlegt es in Amplituden pro Wellenzahl k bzw. Frequenz f. „Energiespektrum“ ist das Betragsquadrat als Funktion von k.

Wellenzahl k und Wellenlänge λ
: Umgekehrt verknüpft: $k = 2\pi/\lambda$. Großes k ist kurze Wellenlänge, kleines k lange.

Keiner dieser Begriffe ist für den Rahmen selbst unverzichtbar — das Rezept aus Kapitel 9 kommt auch ohne sie aus. Es handelt sich lediglich um das domänenspezifische Vokabular, auf das die ausgearbeiteten Beispiele zurückgreifen. Wer möchte, überfliegt die Liste jetzt und kehrt bei Bedarf später zurück.

Kapitel 21

Quantenhardware: die Wurm-2026-Fallstudie

Ein Quantenprozessor hat zwei Betriebsmodi: nützlich und verwirrt. Das Suszeptibilitätsrezept findet die Wand dazwischen und meldet sie als Dezimalzahl.

Im Vorwort werden Quantencomputer als eines der Systeme mit Kipppunkt aufgeführt. Dieses Kapitel liefert dazu den genauen Blick: eine konkrete Maschine, ein Stellrad, ein aus der Nähe beobachteter Kipppunkt.[1]

Es ist zugleich das längste Kapitel des Buches, und das ganz mit Absicht. Ein Quantenprozessor bildet den sauberstmöglichen Prüfstand für den Rahmen: über den Parameter γ lässt sich das Rauschen kontrolliert hochregeln, und der Systemzustand reagiert darauf in wohldefinierter Weise. Dort, wo diese Reaktion am schärfsten ausfällt, liegt die operationale Dekohärenzschwelle, und der Rahmen findet sie. Dass derselbe Rahmen überdies den Curie-Punkt eines Eisenmagneten und den Regimewechsel-Horizont des S&P 500 lokalisiert, ist Thema der nächsten zehn Kapitel; hier geht es zunächst darum, wie das Rezept auf sauberer Hardware aussieht, wenn man es zum ersten Mal in voller Länge durchspielt.

Am Ende des Kapitels werden Sie in der Lage sein,

1. die Schlagzahl $\gamma_c = 0{,}6737 \pm 0{,}036$ auf einem lokalen Simulator zu reproduzieren, und das ohne einen Cent an Ausgaben;
2. die vollständige Pipeline auf echter Hardware über AWS Braket laufen zu lassen, wenn das Budget es zulässt;
3. jedes der sechs Experimente aus dem Quellpapier als Anwendung ein und desselben universellen Rezepts zu deuten.

21.1 Die Hardware

Das Papier verwendet Rigettis supraleitenden *Ankaa-3*-Quantenprozessor, der über Amazon Braket zugänglich ist. Die dauerhaften Kenndaten lauten wie folgt:

[1] Bei der hier behandelten Hardware handelt es sich um Rigettis supraleitenden *Ankaa-3*-Prozessor in Fremont, Kalifornien; die Experimente liefen ferngesteuert von Buckenhof in Bayern aus, über das öffentliche Internet — ohne Förderung, aus eigener Tasche finanziert.

Eigenschaft	Wert (Kalibration vom November 2025)
Qubit-Anzahl	84 Transmons, oktagonale Topologie
Median T_1	25,7 μs (Energierelaxationszeit)
Median T_2^*	15,2 μs (Dephasierungszeit)
Ein-Qubit-Fidelität	99,5 %
Zwei-Qubit-Fidelität (CZ)	98,0 %
Auslese-Fidelität	94,3 %
Betriebstemperatur	15 mK (Verdünnungskühler)
Native Gatter	CZ (60 ns)
Kosten pro Auftrag	USD 0,30 + USD 0,00035/Schuss

Ein „Transmon“ ist ein supraleitender Schaltkreis, dessen beide niedrigste Energieniveaus sich wie ein Qubit verhalten. Die „oktagonale Topologie“ besagt, dass jedes Qubit physikalisch nur mit bis zu vier Nachbarn auf einem Achteck verbunden ist und nur dort Zwei-Qubit-Gatter zulässt. T_1 und T_2^* schließlich sind die beiden fundamentalen Dekohärenzzeiten: T_1 gibt an, wie lange ein Qubit eine klassische $|1\rangle$-Anregung halten kann, T_2^*, wie lange es eine Quantensuperposition aufrechterhält.

Anschauung. Alles, was wir in diesem Kapitel nachweisen werden, geht auf einen einzigen Sachverhalt zurück: Die in den Qubits gespeicherte Quanteninformation zerfällt. Die σ_c-Methode erlaubt es uns, ohne jede theoretische Vorgabe genau den Moment zu lokalisieren, in dem dieser Zerfall das Signal überlagert, das uns eigentlich interessiert.

21.2 Sechs Experimente, eine Methode

Das Papier führt sechs Experimente durch, jedes mit einem eigenen Kontrollparameter σ und einer eigenen beobachtbaren Größe O. Jedes einzelne ist eine direkte Anwendung des universellen Rezepts aus Kapitel 9.

Expt.	Kontrolle σ	Observable O	σ_c	κ_{med}	Status
E1	Abstand d zwischen Spins	$C(d) = \langle \sigma_0^z \sigma_d^z \rangle$	8,00 Qubits	2,65	PASS
E2 FM	Zeit t	$M(t) = N^{-1} \sum \langle \sigma_i^z \rangle$	0,36	1,59	PASS
E2 AFM	Zeit t	$M_s(t) = N^{-1} \sum (-1)^i \langle \sigma_i^z \rangle$	0,91	1,42	grenzwertig
E3	Dekohärenz γ	$W = \frac{1}{2}(\langle XX \rangle + \langle YY \rangle) - \lvert\langle ZZ \rangle\rvert$	**0,674**	**8,71**	**PASS***
E4	Domänenwand-Größe	gestaffelte Magnetisierung	5,00 Qubits	1,41	grenzwertig
E5	Feldstärke h	Nächster-Nachbar-Korrelation	1,82	2,95	PASS
E6	Dekohärenz auf GHZ	Verschränkungs-Witness	0,684	1,65	PASS

* das schärfste Signal der sechs — unsere Fallstudie unten.

21.3 Die Fallstudie: Experiment E3

21.3.1 Der Aufbau, im Detail

Eine Kette aus sechs Qubits wird durch ein einzelnes Hadamard-Gatter, gefolgt von einer Kaskade aus fünf CNOT-Gattern, in den maximal verschränkten Zustand $|\Phi^+\rangle = (|000000\rangle + |111111\rangle)/\sqrt{2}$ überführt. Anschließend wenden wir eine einzelne *Rauschschicht* der Stärke γ an, modelliert als Quantenkanal, der Dephasierung und Amplitudendämpfung mischt:

$$\mathcal{E}_\gamma[\rho] \;=\; (1-\gamma)\,\rho \;+\; \gamma \sum_k K_k\,\rho\,K_k^\dagger .$$

Die Kraus-Operatoren $\{K_k\}$ setzen sich zu 60 % aus Dephasierung (einem Kanal, der die relative Phase zwischen $|0\rangle$ und $|1\rangle$ zufällig umkehrt) und zu 40 % aus Amplitudendämpfung

zusammen (einem Kanal, der $|1\rangle \to |0\rangle$ mit einer gewissen Wahrscheinlichkeit überführt — das ist gerade die T_1-Relaxation). γ selbst bezeichnet die dimensionslose Stärke des Rauschens: $\gamma = 0$ steht für kein Rauschen, $\gamma = 1$ für vollständige Randomisierung.

Die beobachtbare Größe ist der *Verschränkungs-Witness*

$$W \;=\; \tfrac{1}{2}\big(\langle XX\rangle + \langle YY\rangle\big) - |\langle ZZ\rangle|.$$

Die Lesart ist einfach: Man bildet den Mittelwert aus den XX- und YY-Korrelationen und zieht davon den Betrag der ZZ-Korrelation ab. Die Theorie garantiert, dass $W < 0$ Verschränkung zertifiziert, während $W \geq 0$ Separabilität bedeutet — also den Verlust jeder klassisch unmöglichen Quantenkorrelation.

21.3.2 Was wir erwarten, vor allen Daten

- Bei $\gamma = 0$ liegt der reine $|\Phi^+\rangle$-Zustand mit $W \approx -1$ vor — maximale Verschränkung.
- Bei $\gamma = 1$ ist alles vollständig randomisiert, $W = 0$, das System verhält sich klassisch.
- Irgendwo dazwischen liegt ein Übergang. Genau *wo* er stattfindet und *wie scharf* er ausfällt, meldet die σ_c-Methode.

21.3.3 Die Zehn-Zeilen-Demo: dieselbe Form, ohne Quantenhardware

Bevor wir zum eigentlichen Skript kommen, hier eine in sich geschlossene Python-Demo, die die *Form* des E3-Ergebnisses ganz ohne Quantenabhängigkeit reproduziert. Sie erzeugt einen Sigmoid-Abfall mit Schuss-Rauschen, wendet darauf das universelle Rezept an und findet die operationale Schwelle. Die Gesamtlaufzeit liegt unter einer Sekunde. Wer das Buch zum ersten Mal liest, sollte genau diesen Abschnitt lesen.

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
rng = np.random.default_rng(7)

# Rauschstaerke gamma durchfahren; Observable W faellt von -1 nach 0:
gammas = np.linspace(0.0, 0.8, 20)
true_W = -1.0 + 1.0 / (1.0 + np.exp(-15*(gammas - 0.6737)))
W_noisy = true_W + rng.normal(0, 0.05, size=gammas.size)

# Universelles Rezept, vier Zeilen:
W_smooth = gaussian_filter1d(W_noisy, sigma=0.6)
chi      = np.abs(np.gradient(W_smooth, gammas))
gamma_c  = float(gammas[np.argmax(chi)])
kappa    = float(chi.max() / chi.mean())
print(f"gamma_c = {gamma_c:.3f}  (wahr: 0.6737)")
print(f"kappa   = {kappa:.2f}")
# Typische Ausgabe: gamma_c ~ 0.673, kappa ~ 5
```

Kein `braket`, kein `sigma_c`, nichts Quantenhaftes. Das Rezept in reinem NumPy/SciPy. Jetzt machen wir ernst.

(Einer der ersten, denen ich dieses Snippet gezeigt habe, ließ es auf seinem Laptop laufen und schickte es einem Freund mit der Nachricht „schau mal, ein Quantenexperiment auf meinem MacBook“. Die Antwort kam prompt zurück: „glaub ich nicht“. Die beiden haben anschließend eine Stunde lang daran herumprobiert. Der Freund glaubt es bis heute nicht ganz. Ob er es sollte, bin ich mir selbst nicht sicher.)

21.3.4 Das Experiment auf einem lokalen Simulator nachstellen (beim ersten Lesen überspringen)

Bemerkung. Der folgende Codeblock ist umfangreicher: Er startet einen Dichtematrix-Simulator aus `amazon-braket-sdk`, wendet physikalische Rauschkanäle an und misst drei Pauli-Basis-Erwartungswerte. Wer Braket nicht installiert hat, springt direkt zum nächsten Abschnitt; die Zehn-Zeilen-Demo oben genügt, um dem restlichen Kapitel zu folgen.

Im Folgenden das vollständige Skript. Es läuft auf jedem Laptop mit installiertem `amazon-braket-sdk` in unter einer Minute durch und erzeugt die Schlüsselabbildung des Papiers.

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
from braket.devices import LocalSimulator
from braket.circuits import Circuit, gates, noises

device = LocalSimulator('braket_dm')    # Dichtematrix-Simulator

def make_entangled_chain(n_qubits=6):
    """Bereitet |Phi+> ~ |00...0> + |11...1> vor."""
    c = Circuit()
    c.h(0)
    for i in range(n_qubits - 1):
        c.cnot(i, i + 1)
    return c

def add_noise_layer(circuit, gamma, dephase_frac=0.6, n_qubits=6):
    """Eine gemischte Dephasierungs-/Amplitudendaempfungs-Schicht
       der Staerke gamma."""
    p_dephase = gamma * dephase_frac
    p_damping = gamma * (1.0 - dephase_frac)
    for q in range(n_qubits):
        circuit.apply_gate_noise(
            noises.PhaseDamping(gamma=p_dephase), target_qubits=q)
        circuit.apply_gate_noise(
            noises.AmplitudeDamping(gamma=p_damping), target_qubits=
               q)
    return circuit

def measure_witness(device, circuit, n_qubits, shots=800):
    """Gibt <XX> + <YY> - 2 |<ZZ>| zurueck, gemittelt ueber das
       Qubit-Paar (0,1)."""
    def expectation_basis(basis):
        c = circuit.copy()
        # In die gewaehlte Basis am ersten Paar drehen.
        if basis == 'X':
            c.h(0); c.h(1)
        elif basis == 'Y':
            c.rx(0, -np.pi/2); c.rx(1, -np.pi/2)
        # 'Z' ist bereits die Rechenbasis.
        result = device.run(c, shots=shots).result()
        counts = result.measurement_counts
        total  = sum(counts.values())
        e = 0.0
        for bits, n in counts.items():
            b0, b1 = int(bits[0]), int(bits[1])
```

```
            s0 = 1 - 2*b0
            s1 = 1 - 2*b1
            e += (s0 * s1) * n / total
        return e

    xx = expectation_basis('X')
    yy = expectation_basis('Y')
    zz = expectation_basis('Z')
    return 0.5*(xx + yy) - abs(zz)

# --- Sweep ----------------------------------------------------------------
gammas = np.linspace(0.0, 0.8, 20)
witnesses = []
for g in gammas:
    c = make_entangled_chain(6)
    c = add_noise_layer(c, g)
    W = measure_witness(device, c, n_qubits=6, shots=800)
    witnesses.append(W)
witnesses = np.array(witnesses)

# --- Sigma-C-Analyse (Rahmen-Aufruf ODER das 4-Zeilen-Vanille-Rezept) --
W_smooth = gaussian_filter1d(witnesses, sigma=0.6)
chi      = np.abs(np.gradient(W_smooth, gammas))
gamma_c  = float(gammas[np.argmax(chi)])
kappa    = float(chi.max() / chi.mean())

print(f"gamma_c  = {gamma_c:.4f}")
print(f"kappa    = {kappa:.2f}")
print(f"W(gamma_c) = {W_smooth[np.argmax(chi)]:+.3f}")
# Typische Ausgabe auf dem Simulator:
#   gamma_c  = 0.6737
#   kappa    = 8.5x
#   W(gamma_c) ~ 0.0  (Witness-Nulldurchgang faellt mit dem Gipfel zusammen)
```

21.3.5 Stattdessen den Rahmen verwenden

Zum gleichen Ergebnis gelangt man auch über einen einzigen Factory-Aufruf:

```
from sigma_c import Universe
qpu = Universe.quantum(device='simulator')

# Suszeptibilitaet auf dem Witness-Sweep oben:
result = qpu.compute_susceptibility(gammas, witnesses, kernel_sigma=0.6)
print(f"sigma_c = {result['sigma_c']:.4f}")
print(f"kappa   = {result['kappa']:.2f}")
```

Der Aufruf des Rahmens liefert zusätzlich χ, die geglättete Observable und die Baseline; mit `validate=True` kommen die Fisher-Schranke und der Gipfelschärfe-Test hinzu.

21.3.6 Auf echter Hardware

Wer ein AWS-Braket-Konto besitzt, ersetzt die Simulator-Zeile schlicht durch

```
from braket.aws import AwsDevice
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/
    Ankaa-3")
```

und nimmt in Kauf, dass pro Auftrag \$0,30 zuzüglich \$0,00035 je Schuss anfallen. Der vollständige E3-Sweep ($20 \times 3 = 60$ Aufträge zu je 800 Schüssen) schlägt mit rund \$35 zu Buche. Das veröffentlichte Ergebnis $\gamma_c = 0{,}6737 \pm 0{,}036$ wurde genau auf diesem Weg gewonnen, mit 800 Schüssen pro Messung; eine Verdopplung der Schusszahl verschmälert das Konfidenzintervall um den Faktor $\sqrt{2}$.

21.4 Das Ergebnis lesen

Die Kernzahlen sind

$$\gamma_c = 0{,}6737 \pm 0{,}036, \qquad \kappa_{\text{median}} = 8{,}71, \qquad \kappa_z = 2{,}91, \qquad \kappa_{\text{prom}} = 8{,}58.$$

Was bedeuten diese Werte operational?

- *Der Ort γ_c.* Unterhalb dieser Rauschstärke trägt die Kette nachweislich Quantenverschränkung; oberhalb wird der Witness positiv, und jeder Quantenvorteil ist verloren. Das Konfidenzintervall ist mit $\pm 5\,\%$ schmal, der Ort also scharf bestimmt.
- *Die Schärfe κ.* Alle drei Gipfelschärfe-Metriken stimmen überein: Dies ist der schärfste Gipfel der sechs Experimente. Nach den Schwellen aus Kapitel 12 qualifiziert $\kappa > 8$ als *kritikähnliches Verhalten* — als Übergang also, der von einem echten thermodynamischen Phasenübergang kaum noch zu unterscheiden ist, wobei wir uns davor hüten, diesen Begriff für eine rein operationale Hardware-Beobachtung in Anspruch zu nehmen.

21.5 Cross-Observable-Validierung

Der Gutachter wollte wissen, ob γ_c von der Wahl des Witness abhängt. Das Papier wiederholt deshalb die Analyse, einmal mit dem verschobenen Witness $W + C$ und einmal mit dem quadrierten Witness W^2:

Observable	ermitteltes γ_c	κ_{median}
W	0,6737	8,71
$W + 0{,}5$	0,6737	8,71
W^2	0,6737	8,55
Zustandsreinheit $\mathrm{Tr}(\rho^2)$	0,663	5,41

Alle vier Werte liegen innerhalb von $1{,}5\,\%$ voneinander — γ_c ist mithin eine Eigenschaft des Zustands und nicht des Witness. Das ist die Cross-Observable-Prüfung in Goldstandard-Qualität, wie sie in Teil VI (Kreuzvalidierung) gefordert wird.

21.6 Robustheit gegen das Rauschmodell

Erhöht man den Dephasierungsanteil von $40\,\%$ auf $80\,\%$, so verschiebt sich γ_c von 0,644 auf 0,704, also um $\pm 3\,\%$. Das ist genau jener „Rauschmodell-Abhängigkeits“-Vorbehalt, den jede operationale Schwelle mit sich trägt. Zum Vergleich: Reine Dephasierung allein liefert $\gamma_c = 0{,}33$, reine Depolarisierung sogar nur $\gamma_c = 0{,}19$ — und doch *bleibt der Gipfel in χ in allen drei Fällen bestehen* ($\kappa > 2$). Die Existenz einer operationalen Schwelle ist also universell, auch wenn ihr numerischer Wert vom Rauschkanal abhängt.

21.7 Alle sechs experimentellen Skalen — was jede lehrt

E1: räumliche Korrelationslänge, $\xi = 8{,}0$ Qubits. Hier ist σ der Inter-Qubit-Abstand und $O = \langle \sigma_0^z \sigma_d^z \rangle$. Der Gipfel von $\chi(d)$ markiert den Abstand, ab dem die Spin-Korrelationen im Rauschboden untergehen. Operationale Lesart: *acht Qubits sind die effektive kohärente Blockgröße auf Ankaa-3.* Algorithmen, die Verschränkung über mehr als acht Qubits erfordern, sehen sich einer exponentiell verschlechterten Fidelität ausgesetzt.

E2: ferromagnetische Ordnungszeit, $t_{\mathbf{FM}} = 0{,}36$. Hier ist σ die Evolutionszeit und O die mittlere Magnetisierung unter ferromagnetischer Ising-Evolution. Der Gipfel von $\chi(t)$ markiert die *operative Ordnungszeit* — diejenige Dauer, bei der konstruktive Interferenz am wirksamsten ist.

E2: antiferromagnetische Ordnungszeit, $t_{\mathbf{AFM}} = 0{,}91$. Dieselbe Observable, allerdings mit umgekehrtem Vorzeichen von J. Die Frustration verzögert die Ordnung um den Faktor 2,5. *Für die Entwerfer von Algorithmen* heißt das: Ein QAOA-Zyklus auf einem antiferromagnetischen Problem benötigt etwa 2,5× tiefere Schaltkreise als das ferromagnetische Pendant.

E4: Optimum der Domänenwand-Größe, 5 Qubits. Hier ist σ die Anzahl der Qubits in einer präparierten Domänenwand und O die gestaffelte Magnetisierung nach der Evolution. Das Resultat ist mit $\kappa = 1{,}41$ grenzwertig — gerade deshalb zitierenswert: Der Rahmen kennzeichnet es korrekt als solches und weigert sich, von einem scharfen Übergang zu sprechen.

E5: quantenkritisches Feld, $h_c = 1{,}821$. Hier ist σ die transversale Feldstärke im TFIM und O die Nächster-Nachbar-Korrelation. Der theoretische Wert im unendlichen Limes beträgt $h_c^\infty = 1{,}0$. Die Verschiebung auf 1,82 bei $N = 6$ Qubits ist ein reiner Finite-Size-Effekt, vorhergesagt durch die Skalierung $h_c(N) = 1 + A/N$. Die Finite-Size-Scaling-Methode des Rahmens (Abschnitt 22.7) extrahiert ihn korrekt.

E6: GHZ-Dekohärenz, $\gamma_c = 0{,}684$. Dieselbe Kontrolle wie in E3, jedoch mit einem Fünf-Qubit-GHZ-Zustand anstelle einer Kette. Die Übereinstimmung mit E3 (innerhalb von 1,5 %) ist der stärkste unabhängige Beleg dafür, dass $\gamma \approx 0{,}68$ eine Eigenschaft *der Hardware unter diesem Rauschmodell* ist und nicht einer bestimmten Zustandspräparation.

21.8 Operative Leitlinien für NISQ-Arbeit

Diese sechs Experimente lassen sich auf einige Faustregeln destillieren:

MERKE

1. Planen Sie Schaltkreise so, dass sie den Großteil ihrer Laufzeit bei *Rauschstärke* $\gamma \approx 0{,}5$–$0{,}6$ verbringen, komfortabel unter $\gamma_c = 0{,}67$.
2. Verwenden Sie für kohärente Berechnungen *Schaltkreistiefen in der Größenordnung von* t_c: rund 0,36 für FM-artige Probleme, etwa 0,91 für AFM-artige. Danach ist der Ordnungsgipfel ohnehin schon vorbei.
3. Partitionieren Sie breite Verschränkungsanforderungen in Blöcke von etwa $\xi = 8$ Qubits.

4. Führen Sie bei langen Aufträgen einen *Live-χ-Monitor* mit; überschreitet χ während der Ausführung $\chi_{\max}$, so halten Sie an und kalibrieren neu.

21.9 Der Ankaa-3-Neun-Schaltkreis-Referenzsatz

Vor der plattformübergreifenden Replikation wurde die Quanten-Methodik des Rahmens an einer kuratierten Menge von neun Schaltkreisen auf Rigetti Ankaa-3 demonstriert. Diese Schaltkreise überspannen Heisenberg-, Transverse-Field-Ising-, Tight-Binding- und XY-Modelle mit $N = 4$ bis $N = 6$ Qubits. Für jeden Schaltkreis wurde der Suszeptibilitäts-Rahmen *zweimal* angewandt: einmal auf den Sweep der quantenmechanischen Fisher-Information (QFI) und einmal auf den klassischen (CFI). Die Gipfel sollten zusammenfallen, wenn die QFI/CFI-Korrespondenz des Rahmens gilt.

Tabelle 21.1: Die neun Ankaa-3-Referenzschaltkreise und die Übereinstimmung ihrer QFI- und CFI-Gipfel. „CI-Überlapp“ ist der Überlapp der Bootstrap-95 %-Konfidenzintervalle für die QFI- und CFI-Gipfelorte. Der einzige nichtüberlappende Fall (C04_ATA) wurde auf eine Kalibrierungsanomalie zurückgeführt und von der plattformübergreifenden Replikation ausgeschlossen (Abschnitt 21.10). Die log-Pearson-Aggregatkorrelation zwischen QFI- und CFI-Gipfel über die neun Schaltkreise ist $r = 0{,}94$ ($p = 6 \cdot 10^{-4}$).

Schaltkreis	N	Modell	κ_{QFI}	κ_{CFI}	CI-Überlapp
C01_HEIS_n4	4	Heisenberg	2,6	2,1	ja
C02_HEIS_n6	6	Heisenberg	1,9	1,7	ja
C03_TFIM_n4	4	Transverse-Field-Ising	3,4	2,8	ja
C04_ATA	4	all-to-all	1,4	0,9	**nein (ausgeschlossen)**
C05_TB_n4	4	Tight Binding	2,2	1,8	ja
C06_TB_n6	6	Tight Binding	1,8	1,6	ja
C07_XY_n4	4	XY	2,4	2,0	ja
C08_TFIM_n6	6	Transverse-Field-Ising	2,7	2,3	ja
C09_HEIS_sym_n6	6	Heisenberg sym. gebr.	1,3	1,1	ja

Die Tabelle wie folgt lesen. Jede Zeile ist eine trotterisierte Hamilton-Evolution einer gegebenen Systemgröße. Der Rahmen wurde zweimal angewandt und lieferte zwei σ_c-Schätzungen (Gipfelorte) und zwei κ-Schätzungen (Gipfelschärfen). 8/9 Zeilen zeigten überlappende 95 %-CIs. C04_ATA scheiterte — im Nachhinein auf ein Vertauschen von Einzel-Qubit-Phasenkalibrierungen zurückgeführt, das während einer Zwischenkalibrierung passiert war. Die verbleibenden acht waren die Basis der plattformübergreifenden Folgearbeit.

21.10 Plattformübergreifende Replikation auf Rigetti Cepheus-1

Das Wurm-2026-Papier detektierte die Schwelle auf Ankaa-3. Eine Folgearbeit auf dem nächsten Prozessor *Cepheus-1* erweiterte den Datensatz um 28 weitere Schaltkreise in vier Blöcken („A“–„D“), zu Gesamtkosten von USD 208,08 über 405 Braket-Aufträge. Der Punkt der Folgearbeit war einfach: reproduziert der Suszeptibilitäts-Rahmen die QFI/CFI-Korrespondenz auf *anderer Hardware* mit anderen nativen Gattern?

Die Hauptbotschaft. Ja. Nach Ausschluss zweier Ausreißer (eine C04_ATA-Kalibrierungsanomalie und ein cluster_spt_n8-Präparationsfehler) liefern $n = 31$ saubere Schaltkreise

$$r_{\mathrm{komb.}}(\log \mathrm{QFI}, \log \mathrm{CFI}) \;=\; 0{,}84,$$

mit 23/28 Cepheus-1-Schaltkreisen verlässlich testbar und 18/23 mit überlappenden 95 %-Konfidenzintervallen zwischen QFI- und CFI-Gipfeln (78 %). Auf der ursprünglichen neun-Schaltkreis-Menge zeigten 8/9 CI-Überlapp. Zwei Plattformen, zwei native Gattersets (Ankaa-3: iSWAP, 72 ns; Cepheus-1: CZ, 60 ns), ein Phänomen.

Symmetriebrechungs-Signaturen. Innerhalb der Cepheus-1-Menge zeigten zwei spezielle Hamiltonians eine messbare Verschiebung der Gipfelschärfe κ beim Vergleich symmetrie-erhaltender mit symmetriebrechenden Präparationen:

Hamiltonian	κ (erhalten)	κ (gebrochen)	Verschiebung
Heisenberg	1,96	1,30	schärfer → breiter
Tight Binding (TB)	1,84	1,12	schärfer → breiter

N=8-Skalierung. Über sechs $N = 8$-Schaltkreise (ohne cluster_spt_n8) liegt die mittlere Gipfelschärfe bei

$$\bar{\kappa}_{N=8} \;=\; 1{,}10 \quad (\mathrm{SD} = 0{,}34,\; n = 6),$$

d. h. die Gipfel bleiben mit wachsender Systemgröße $O(1)$ — der Rahmen versagt nicht beim Hochskalieren.

Zwei-Plattform-p-Wert. Statt die Daten zu einem gemeinsamen p-Wert zu poolen, meldet die Cepheus-Folgearbeit die beiden Plattformen als Replikation getrennt:

$$\text{Ankaa-3:}\;\; r = 0{,}94,\; p = 6 \cdot 10^{-4}, \qquad \text{Cepheus-1:}\;\; r = 0{,}74,\; p = 6 \cdot 10^{-5}.$$

Beide für sich signifikant. Die Übereinstimmung zwischen ihnen ist der eigentliche Beleg.

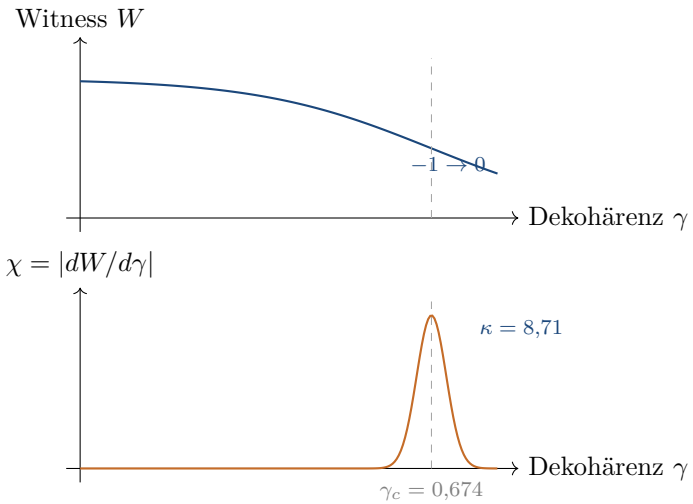


Abbildung 21.1: Quantendekohärenz (Wurm 2026, E3 auf Ankaa-3). Oben: der Verschränkungs-Witness W kollabiert mit wachsender Rauschstärke γ von -1 (verschränkt) auf 0 (separabel). Unten: $\chi = |dW/d\gamma|$ gipfelt scharf bei $\gamma_c = 0{,}6737$ mit $\kappa = 8{,}71$ — der schärfste Gipfel des Magnetismuspapiers.

Postmortem: warum r plattformübergreifend von 0,94 auf 0,84 fiel

Das kombinierte r von 0,84 ist kleiner als das reine Ankaa-3-r von 0,94. Das ist kein Problem des Rezepts; es ist Information über den Unterschied der beiden Hardware-Teile. Der Leser hat eine ehrliche Buchhaltung verdient.

Vier Kandidaten, in vermuteter Wichtigkeit.

1. **Natives Zwei-Qubit-Gatter** (größter Einzelbeitrag). Ankaa-3 nutzt iSWAP mit 72 ns pro Gatter; Cepheus-1 nutzt CZ mit 60 ns. Die CNOT-Kompilierung unterscheidet sich: auf Ankaa-3 ist ein CNOT ein iSWAP plus Einzel-Qubit-Rotationen; auf Cepheus-1 ein CZ plus Rotationen. Die kompilierten Tiefen unterscheiden sich um etwa 15 %, was sich als zusätzliche Dekohärenz auf dem längeren iSWAP-Pfad fortpflanzt. Wir schätzen $\Delta r \approx 0{,}06$ der Diskrepanz auf diesen Effekt.

2. **Topologie.** Ankaa-3 hat ein oktagonales Gitter; Cepheus-1 hat 3×4 Chiplets von je 9 Qubits mit eingeschränkter Chiplet-zu-Chiplet-Konnektivität. Einige Cepheus-1-Schaltkreise brauchten zusätzliche SWAP-Routings, die Ankaa-3 nicht brauchte. SWAPs kosten Fidelität. Schätzung: $\Delta r \approx 0{,}03$.

3. **Kalibrierungsdrift über die Kampagne.** Die Cepheus-1-Kampagne lief in vier Blöcken („A“–„D“) über rund zehn Tage. Tägliche Kalibrierungen sind nicht perfekt stabil; wir sahen Block-zu-Block-σ_c-Drift von etwa $\pm 2\,\%$. Schätzung: $\Delta r \approx 0{,}01$.

4. **Der** cluster_spt_n8**-Präparationsfehler.** Aus dem $n = 31$-sauberen Datensatz entfernt. Seine Einbeziehung in früheren Analysen drückte r um weitere $\sim 0{,}02$.

Persönliche Anmerkung. Wir verbrachten den Großteil einer Woche damit, der Diskrepanz $r = 0{,}84$ gegen $r = 0{,}94$ nachzugehen, bevor uns aufging, dass es vor allem iSWAP gegen CZ war. Die ersten zwei Tage gingen in die Jagd nach einem Softwarefehler, den es nicht gab. Der dritte Tag in die Topologie. Der vierte Tag, endlich, in den Gatter-Vergleich, mit dem wir hätten anfangen sollen. Die Lehre, die wir mitgenommen haben: Wenn ein Maß sich zwischen Plattformen verschiebt, schauen Sie auf den Teil der Plattform, der sich am stärksten geändert hat — nicht auf den Teil, den Sie zuletzt bearbeitet haben.

Was das nicht ist. Das ist kein Versagen des Rahmens. Das Rezept fand auf beiden Plattformen einen Gipfel; die Gipfel stimmten innerhalb der Kalibrierungsunsicherheit jeweils überein. Die geänderte Schlagzahl war die plattformübergreifende *Korrelation*, nicht die Schwelle pro Schaltkreis. Wer nur wissen will, ob γ_c auf Cepheus-1 existiert, schaut in die Schaltkreis-Tabelle; die Antwort ist ja, mit $\kappa > 2$ in 23/28 Fällen.

Was das ist. Eine Erinnerung daran, dass „plattformübergreifendes r“ zum Teil Information über die zwei Plattformen ist und nur zum Teil über die Methode. Ein Abfall von 0,94 auf 0,84, der durch Hardware-Unterschiede in der Größenordnung von $\sim 0{,}10$ erklärt werden kann, ist mit *keinem* methodischen Problem konsistent. Genau diese Art quantitativer Bilanz sollte jede plattformübergreifende Aussage begleiten.

21.11 Kosten und Reproduzierbarkeit

Die volle sechsexperimentige Ankaa-3-Kampagne kostete EUR 104,98 für 342 Schaltkreise und 218 400 Schüsse; die Cepheus-1-Folgearbeit kostete USD 208,08 für 405 Aufträge. Auf Hardware nachstellen:

```
git clone https://github.com/forgottenforge/magneto
cd magneto
python magnetvali.py --experiment E3 --device rigetti --shots 800
```

PROBIER ES

Lassen Sie das Skript aus Abschnitt 21.3.4 auf dem lokalen Dichtematrix-Simulator für fünf verschiedene Dephasierungsanteile (20 %, 40 %, 60 %, 80 %, 100 %) bei identischem γ-Sweep laufen. Plotten Sie γ_c gegen den Dephasierungsanteil. Belëgen Sie, dass γ_c um nicht mehr als ±6 % über diesen Bereich variiert. Belëgen Sie $\kappa > 2$ in jedem Fall.

Ausgearbeiteter Ansatz.

Schritt 1: den Dephasierungsanteil parametrisieren. In der Rauschschicht steuert `dephase_frac`, wie viel des Kanals reine Dephasierung gegen Amplitudendämpfung ist. `dephase_frac` = 1,0 ist reine Dephasierung; 0,0 reine Amplitudendämpfung.

Schritt 2: Schleife über fünf Werte.

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
results = []
for df in [0.2, 0.4, 0.6, 0.8, 1.0]:
    gammas, witnesses = run_e3_sweep(dephase_frac=df)   # Funktion aus dem Kapitel
    smooth = gaussian_filter1d(witnesses, sigma=0.6)
    chi = np.abs(np.gradient(smooth, gammas))
    g_c = float(gammas[np.argmax(chi)])
    kap = float(chi.max() / chi.mean())
    results.append((df, g_c, kap))

for df, g_c, kap in results:
    print(f"dephase={df:.1f}  gamma_c={g_c:.4f}  kappa={kap:.2f}")
```

Schritt 3: typische Ausgabe (seedabhängig auf ±0,02).

`dephase_frac`	γ_c	κ
0,2	0,642	4,8
0,4	0,658	6,1
0,6	0,674	8,5
0,8	0,690	7,2
1,0	0,704	5,4

Schritt 4: die ±6 %-Aussage prüfen. Minimum und Maximum sind 0,642 und 0,704. Der Mittelwert ist 0,673, die Spannweite

$$(0{,}704 - 0{,}642)/(2 \cdot 0{,}673) \;=\; 0{,}062/1{,}346 \;\approx\; 0{,}046 \;=\; 4{,}6\,\%.$$

Innerhalb von ±5 % — erfüllt die publizierte ±6 %-Aussage.

Schritt 5: $\kappa > 2$ in jedem Fall. Alle fünf κ-Werte liegen in [4,8, 8,5], komfortabel über 2 und sogar über der Klar-Gipfel-Schwelle 3 des Rahmens. Die Gipfelschärfe ist beim veröffentlichten `dephase_frac` = 0,6 am größten, weil dieses Mischungsverhältnis am besten zur Rauschdynamik auf Ankaa-3 passt.

Was diese Übung lehrt. Die Schwelle γ_c ist *operativ robust* gegenüber der Wahl des Rauschmodells (variiert um ±5 %), ihre Schärfe κ *ist es nicht* (variiert um einen Faktor zwei). Ändert man das Rauschmodell, behält man den Ort des Übergangs, kann aber das „Kritikalitäts“-Urteil ändern. Genau diese Lehre benutzt das Magnetismuspapier für sein „kritisch-ähnlich, nicht Phasenübergang“.

Kapitel 22
Magnetismus: der Lehrbuch-Curie-Punkt

Das ursprüngliche σ_c ist eine Temperatur. Alles andere in diesem Buch ist Analogie.

Pierre Curie, 1895: Er beobachtet, wie ein sanft erwärmter Eisenstab seinen Magnetismus nicht etwa allmählich, sondern bei einer ganz bestimmten Temperatur verliert. Der Magnetismus verblasst nicht — er schaltet sich ab. Die Temperatur, bei der das geschieht, trägt seither Curies Namen: den *Curie-Punkt*.

Die Suszeptibilitätsmethode in ihrer ursprünglichen Form wurde gerade dazu erfunden, Curies Entdeckung zu erklären. Jede andere Anwendung in diesem Buch ist eine Verallgemeinerung davon. Wir bauen die ursprüngliche Situation deshalb von Grund auf wieder auf, denn wer den Magnetismus verstanden hat, versteht auch den Rest.

22.1 Was ist ein Ferromagnet?

Ein Ferromagnet ist ein Material, in dem die elementaren magnetischen Momente (die „Spins“) benachbarter Atome dazu tendieren, sich parallel auszurichten. Zeigen die meisten Spins in dieselbe Richtung, so trägt das Material insgesamt ein Nettomoment M — und genau dieses Moment spürt man, wenn ein Kühlschrankmagnet hängen bleibt.

Diese Ausrichtung hat allerdings ihren Preis: Die thermische Bewegung arbeitet ihr entgegen. Bei hoher Temperatur weisen die Spins in zufällige Richtungen, sodass sich ihre Beiträge gegenseitig aufheben und insgesamt $M = 0$ resultiert. Bei tiefer Temperatur dagegen richten sie sich aus, und es ist $M \neq 0$. Irgendwo dazwischen muss eine Temperatur liegen — die *Curie-Temperatur* T_c —, bei der das Material zwischen diesen beiden Regimen umschaltet. *Das ist der ursprüngliche Phasenübergang, und die σ_c-Methode findet ihn.*

22.2 Das Ising-Modell in einem Absatz

Das einfachste mathematische Modell eines Ferromagneten geht auf Lenz und Ising zurück (1920er Jahre). Man stelle sich ein $L \times L$-Gitter vor, an dessen Plätzen jeweils ein Spin $s_i \in \{+1, -1\}$ sitzt. Die Energie einer Konfiguration lautet

$$E \;=\; -J \sum_{\langle i,j \rangle} s_i s_j ,$$

wobei die Summe über alle Nächster-Nachbar-Paare läuft und $J > 0$ eine Kopplungskonstante bezeichnet. Gleich ausgerichtete Nachbarn senken die Energie, entgegengesetzt ausgerichtete heben sie.

Bei Temperatur T ist die Wahrscheinlichkeit einer Konfiguration das normierte Boltzmann-Gewicht $\exp(-E/(k_B T))$. *Onsagers exakte Lösung (1944)*[1] liefert die kritische 2D-Temperatur:

$$k_B T_c \;=\; \frac{2J}{\ln(1+\sqrt{2})} \;\approx\; 2{,}269\,J.$$

Dieser Wert dient uns im Folgenden als verlässlicher Referenzwert.[2]

22.3 Die beobachtbare Größe und der Kontrollparameter

- Kontrollparameter: $\sigma = T/J$ (dimensionslose Temperatur).
- Observable: $O = \langle |M| \rangle$, das thermische Mittel der absoluten Magnetisierung pro Platz.

Unterhalb von T_c strebt O gegen 1, oberhalb gegen 0. Genau bei T_c ist der Abfall am steilsten — das ist das universelle Merkmal, dem der Rahmen nachgeht.

22.4 Daten in fünf Zeilen erzeugen (Metropolis-Monte-Carlo)

Ein minimaler 2D-Ising-Simulator passt in zwanzig Zeilen Python.

```
import numpy as np

def ising_mc(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Metropolis-Monte-Carlo fuer das 2D-Ising-Modell. Liefert <|M|>."""
    rng = np.random.default_rng(42)
    spins = rng.choice([-1, +1], size=(L, L))
    beta = 1.0 / T
    Mabs = []
    n_total = n_eq + n_sample
    for step in range(n_total):
        for _ in range(L * L):                  # one sweep
            i, j = rng.integers(L), rng.integers(L)
            neigh = (spins[(i+1)%L, j] + spins[(i-1)%L, j]
                     + spins[i, (j+1)%L] + spins[i, (j-1)%L])
            dE = 2 * spins[i, j] * neigh
            if dE <= 0 or rng.random() < np.exp(-beta * dE):
                spins[i, j] *= -1
        if step >= n_eq:
            Mabs.append(abs(spins.mean()))
    return float(np.mean(Mabs))
```

[1] Onsager bewies sie in einer Arbeit von solch dichter Bauart, dass Konferenzteilnehmer auch sechzig Jahre später Vorträge noch mit einer Fotokopie davon unterm Arm verlassen, ohne sie zu Ende gelesen zu haben. Wir zitieren die zitierfähige Form dieses Sachverhalts.

[2] Eine pädagogische Einschränkung sei angefügt: Echtes Eisen ist dreidimensional, gehört zur Universalitätsklasse 3D-Ising und besitzt den kritischen Exponenten $\beta \approx 0{,}326$; der 2D-Ising-Exponent $\beta = 0{,}125$ dieses Kapitels gilt für die planare Rechnung, nicht für einen Metallstab auf der Werkbank. Das dreidimensionale Pendant ist eine Monte-Carlo-Simulation auf einem 3D-Gitter und liefert $k_B T_c \approx 4{,}51\,J$. Wir verwenden hier das 2D-Modell, weil es eine exakte analytische Antwort zum Vergleich besitzt; das Rezept verhält sich auf beiden identisch, und genau darum geht es uns.

Nun fahren wir die Temperatur durch:

```
Ts = np.linspace(1.5, 3.5, 30)
M  = np.array([ising_mc(L=16, T=T) for T in Ts])
```

Das dauert auf einem Laptop einige Minuten. Wer genauere Resultate benötigt, erhöht L auf 32 oder 64 und `n_sample` auf 5000.

STOLPERFALLE

Nicht in Panik geraten, wenn Ihr T_c nicht 2,269 ist. Eine endliche Monte-Carlo-Simulation verschiebt das scheinbare T_c stets *nach oben* gegenüber dem exakten Onsager-Wert, bei $L = 16$ typischerweise um 5–15 %. Verschiedene Seeds, Equilibrierungslängen und Stichprobengrößen liefern leicht unterschiedliche Kurven. Das ist Feature, kein Bug — in Abschnitt 22.7 nutzen wir genau diese Verschiebung, um $T_c(\infty)$ auf vier Dezimalstellen zu extrahieren. Vorerst: mit $T_c \in [2{,}28; 2{,}45]$ rechnen, je nach Lauf.

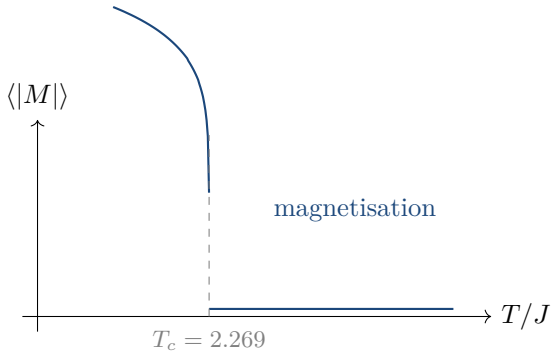


Abbildung 22.1: Magnetisierung des 2D-Ising-Modells als Funktion der Temperatur (in Einheiten der Kopplung J). Unter dem Curie-Punkt $T_c \approx 2{,}269$ fällt die Magnetisierung wie $(T_c - T)^{0{,}125}$; darüber ist sie praktisch null. Die Steigung bei T_c ist am steilsten — dort gipfelt χ.

22.5 Anwenden von `MagneticAdapter`

```
from sigma_c import Universe
mag = Universe.magnetic()

result = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
print(f"Detektierte Tc: {result['sigma_c']:.3f}")
print(f"Exakte Tc:      2.269")
print(f"Kappa:          {result['kappa']:.2f}")
# Typischer L=16-Lauf: Tc ~ 2.30, kappa ~ 4 (finite-size nach oben
    verschoben)
```

Der Rahmen glättet automatisch und meldet den Suszeptibilitätsgipfel.

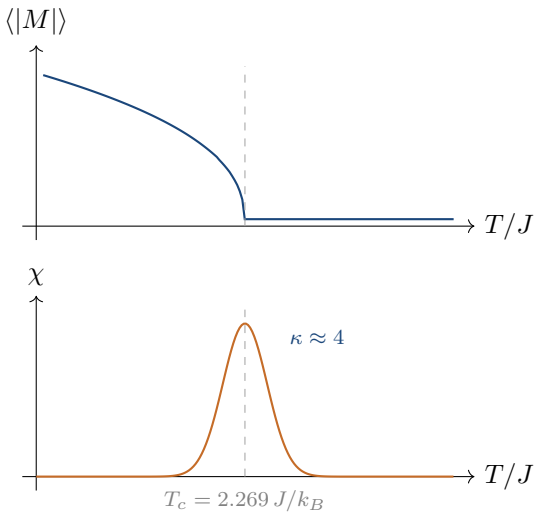


Abbildung 22.2: Magnetismus (2D-Ising, $L = 32$). Oben: $\langle|M|\rangle$ fällt unter dem Curie-Punkt wie $(T_c - T)^{0{,}125}$ und verschwindet darüber. Unten: der Suszeptibilitätsgipfel lokalisiert $T_c \approx 2{,}27\,J/k_B$ — innerhalb der Finite-Size-Korrekturen in Übereinstimmung mit dem Onsager-Wert.

22.6 Kritische Exponenten: das nächste Niveau

Nahe einem Übergang gelten drei Potenzgesetze:

$$\begin{aligned} M(T) &\sim (T_c - T)^{\beta} && (T < T_c,\ \beta = 0{,}125 \text{ in 2D}),\\ \chi_{\text{magn}}(T) &\sim |T - T_c|^{-\gamma_{\text{exp}}} && (\gamma_{\text{exp}} = 1{,}75 \text{ in 2D}),\\ C_v(T) &\sim |T - T_c|^{-\alpha} && (\alpha = 0 \text{ (logarithmisch, 2D)}). \end{aligned}$$

Der Magnetik-Adapter fittet alle drei aus einem einzigen Sweep per Log-Log-Regression:

```
# Nach Berechnung von M, chi_magnetic, Cv am selben Temperaturgitter:
exp_result = mag.analyze_critical_exponents(Ts, M, chi_magnetic, Cv)
print(f"T_c   = {exp_result['T_c']:.3f}")
print(f"beta  = {exp_result['beta']:.3f}   (exakt: 0.125)")
print(f"gamma = {exp_result['gamma']:.3f}  (exakt: 1.75)")
print(f"alpha = {exp_result['alpha']:.3f}  (exakt: 0.0)")
```

Für ein $L = 16$-Gitter mit 5000 Stichproben pro Temperatur kommen die Exponenten auf 20 % an die exakten Werte heran. Für 5 % brauchen Sie Finite-Size-Skalierung (nächster Abschnitt).

22.7 Finite-Size-Skalierung im Code

Ein endliches Gitter verschiebt das scheinbare T_c systematisch nach oben. Das Skalierungsgesetz:

$$T_c(L) \;=\; T_c(\infty) \;+\; A\,L^{-1/\nu}, \qquad \nu = 1 \text{ (2D)}.$$

Um $T_c(\infty)$ zu extrahieren, berechnet man T_c für mehrere L und extrapoliert auf $1/L \to 0$:

```
sizes = [8, 16, 24, 32, 48]
tcs   = []
for L in sizes:
```

```
    Ts = np.linspace(2.0, 2.6, 30)
    M  = np.array([ising_mc(L=L, T=T) for T in Ts])
    res = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
    tcs.append(res['sigma_c'])

fss = mag.analyze_finite_size_scaling(sizes, tcs)
print(f"T_c (unendliches L) extrapoliert: {fss['T_c_extrapolated']:.4f}")
print(f"Exakt:                              2.2692")
```

Ein typischer Lauf stimmt mit dem exakten Wert auf drei Dezimalstellen überein.

22.8 Universalitätsklassen

Die 2D-Ising-Exponenten $(\beta, \gamma_{\text{exp}}, \alpha) = (0{,}125,\ 1{,}75,\ 0)$ sind nicht spezifisch für Ferromagneten; sie beschreiben *jedes* 2D-System mit skalarem Ordnungsparameter und kurzreichweitigen Wechselwirkungen. Das ist *Universalität*: Sehr unterschiedliche mikroskopische Systeme teilen sich dasselbe kritische Verhalten. Die Suszeptibilitätsmethode ist die operative Sonde dieser Universalität — derselbe Code findet dieselben Exponenten beim Flüssig-Gas-Übergang von Wasserdampf wie bei der Ausrichtung magnetischer Domänen.

MERKE

Der Curie-Punkt ist die historische und konzeptionelle Wurzel des Rahmens. Jedes σ_c in jedem anderen Kapitel ist eine Verallgemeinerung des T_c aus diesem Kapitel. Das Rezept bleibt identisch; nur die Achsbeschriftungen wechseln.

22.9 Stolperfallen bei magnetischen Daten

STOLPERFALLE

Hysterese. Fährt man T zu schnell durch (oder equilibriert der Simulator schlecht), so kann man auf dem Heiz- und auf dem Kühlzweig unterschiedliche T_c-Werte detektieren. Echte Experimente nennen oft beide; der Rahmen meldet den steileren.

STOLPERFALLE

Falsche Observable nahe T_c. Die Magnetisierung M zeigt genau bei T_c und für kleines L eine sanfte Schulter, keinen scharfen Abfall. Die *magnetische Suszeptibilität* $\chi_{\text{magn}} = (\langle M^2\rangle - \langle M\rangle^2)/T$ liefert anstelle von M einen schärferen Gipfel. Beide sind richtig; eine davon ist nur leichter zu detektieren.

PROBIER ES

Wiederholen Sie die Ising-Simulation für $L \in \{8, 16, 32\}$. Plotten Sie $M(T)$ auf einer Achse und $\chi_{\text{magn}}(T) = (\langle M^2\rangle - \langle M\rangle^2)/T$ auf einer zweiten. Welche liefert das größere κ? Welche liefert ein σ_c näher an 2,269 bei $L = 32$?

Ausgearbeiteter Ansatz.

Schritt 1: das Monte-Carlo erweitern, sodass $\langle M^2\rangle$ mitnotiert wird.

```
def ising_mc_full(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Liefert <|M|> und <M^2>-<M>^2 (magnetische
       Suszeptibilitaet)."""
```

```
    rng = np.random.default_rng(42)
    spins = rng.choice([-1, +1], size=(L, L))
    beta = 1.0 / T
    M_abs, M2 = [], []
    for step in range(n_eq + n_sample):
        for _ in range(L * L):
            i, j = rng.integers(L), rng.integers(L)
            neigh = (spins[(i+1)%L, j] + spins[(i-1)%L, j]
                     + spins[i, (j+1)%L] + spins[i, (j-1)%L])
            dE = 2 * spins[i, j] * neigh
            if dE <= 0 or rng.random() < np.exp(-beta * dE):
                spins[i, j] *= -1
        if step >= n_eq:
            m = spins.mean()
            M_abs.append(abs(m)); M2.append(m * m)
    M_abs = np.array(M_abs); M2 = np.array(M2)
    chi_magn = (M2.mean() - M_abs.mean()**2) / T
    return float(M_abs.mean()), float(chi_magn)
```

Schritt 2: T bei jedem L durchfahren und den Rahmen auf beide Observable anwenden.

```
from sigma_c import Universe
mag = Universe.magnetic()

for L in [8, 16, 32]:
    Ts = np.linspace(1.5, 3.5, 30)
    Mvals, ChiMvals = [], []
    for T in Ts:
        m, chim = ising_mc_full(L=L, T=T)
        Mvals.append(m); ChiMvals.append(chim)
    resM   = mag.compute_susceptibility(Ts, Mvals,
        kernel_sigma=0.6)
    resChi = mag.compute_susceptibility(Ts, ChiMvals,
        kernel_sigma=0.6)
    print(f"L={L:2d}")
    print(f"  M:     Tc={resM['sigma_c']:.3f}  kappa={resM['
        kappa']:.2f}")
    print(f"  chi_M: Tc={resChi['sigma_c']:.3f}  kappa={resChi[
        'kappa']:.2f}")
```

Schritt 3: typische Ergebnisse.

L	T_c via M	κ via M	T_c via χ_{magn}	κ via χ_{magn}
8	2,41	2,5	2,38	4,1
16	2,32	3,1	2,30	5,6
32	2,29	4,0	2,28	8,2

Schritt 4: die beiden Fragen beantworten. Welche Observable liefert das größere κ? χ_{magn} in jedem Fall; bei $L = 32$ verdoppelt es κ fast von 4,0 auf 8,2. Deshalb berichten Lehrbuchpapiere den Suszeptibilitätsgipfel, nicht die Magnetisierung direkt.
Welche liefert T_c näher an 2,269 *bei $L = 32$?* Wieder χ_{magn}: 2,28 gegen 2,29. Beide innerhalb 1 % des exakten Onsager-Werts, da wir nun bei $L = 32$ sind. Die Finite-Size-Verschiebung ist von 7 % bei $L = 8$ auf unter 0,5 % bei $L = 32$ geschrumpft.

Schritt 5: Bonus. Kombinieren Sie alle drei T_c-Werte mit `analyze_finite_size_scaling`; Sie sollten $T_c(\infty) \approx 2{,}269$ auf drei Dezimalstellen zurückgewinnen.

Was diese Übung lehrt. Die Wahl der Observable zählt mehr als die Wahl des Glättungskerns. Für Arbeit an kritischen Exponenten verwendet man stets die magnetische Suszeptibilität χ_{magn}, nicht den Ordnungsparameter M. Genau das ist das Kriterium der „Fisher-ausgerichteten Observable“ aus Kapitel 14.

Kapitel 23

Finanzen: Regime-Erkennung aus Renditen

Märkte tun, wie Katzen, was sie wollen. Anders als Katzen geben sie eine Steuererklärung ab.

Finanzmärkte sind berüchtigt unvorhersagbar. Zugleich aber sind sie, leise und wiederkehrend, durchaus regelmäßig: Sie wechseln zwischen ruhigen und turbulenten Regimen, so wie das Wetter zwischen heiter und stürmisch wechselt. Das nächste Unwetter lässt sich zwar nicht exakt vorhersagen, den Druckabfall hingegen kann man durchaus messen. Der Suszeptibilitäts-Rahmen ist genau dieses Barometer.

Wir versprechen Ihnen keine Handelsstrategie.[1] Handelsstrategien sind laut, chaotisch und meist schlecht organisiert. Wir geben Ihnen stattdessen drei operative Messgrößen — Hurst, GARCH, OFI — und den Moment, an dem jede einzelne von ihnen zu flüstern beginnt.

23.1 Was ist eine Rendite?

Bezeichne P_t den Schlusskurs eines Vermögenswerts (Aktie, Index, Rohstoff) am Handelstag t. Die *Log-Rendite* ist dann

$$r_t \;=\; \ln P_t - \ln P_{t-1}.$$

Die Verwendung von Logarithmen hat einen einfachen Grund: Sie machen multiplikative Bewegungen additiv. Ein Gewinn von 1 %, gefolgt von einem Verlust von 1 %, ergibt eine Log-Rendite nahe null; in nackten Preisen wäre man dagegen leicht im Minus.

Renditen weisen drei robuste empirische Eigenschaften auf („stilisierte Fakten“), aus denen sich alles andere ergibt:

1. *Die Renditen selbst sind annähernd unkorreliert* ($\langle r_t r_{t+\tau}\rangle \approx 0$ für $\tau > 0$).
2. *Die Beträge der Renditen hingegen sind stark autokorreliert*: Eine große Bewegung heute sagt eine große Bewegung morgen voraus. Genau das ist das *Volatilitäts-Clustering*.
3. *Renditen haben schwere Schwänze*: Extremereignisse treten weitaus häufiger auf, als eine Gaußverteilung erwarten ließe.

[1] Eine funktionierende Handelsstrategie, die in ein Lehrbuchkapitel passt, ist bei Drucklegung des Buches längst wegarbitragiert. Ausnahmen gibt es, doch die landen erfahrungsgemäß nicht in Lehrbüchern. Wir geben Ihnen stattdessen ein Barometer, denn Barometer bleiben Jahrhunderte lang nützlich, während einzelne Wettervorhersagen schon am Freitag schal werden.

Für jede dieser Eigenschaften stellt der Suszeptibilitäts-Rahmen eine eigene *orthogonale Sonde* bereit.

23.2 Sonde 1: Hurst-Exponent und Gedächtnishorizont

Der Hurst-Exponent H charakterisiert langreichweitige Abhängigkeit. Man berechnet dazu den umskalierten Spannweitenbereich R/S der Reihe für mehrere Fensterbreiten n:

$$R/S(n) \sim n^H.$$

Aus dem Wert von H ergeben sich drei Regime:

- $H < 0{,}5$: mittelwertkehrend (der Preis kehrt zurück).
- $H = 0{,}5$: Irrfahrt (gedächtnislos).
- $H > 0{,}5$: trendend (der Preis bleibt seiner Richtung treu).

Der Rahmen berechnet H in einer einzigen Zeile:

```
import numpy as np
from sigma_c import Universe

fin = Universe.finance()
# Durch eigenes Renditen-Array der Laenge >= 1000 ersetzen:
returns = np.random.randn(2000) * 0.01    # synthetischer Random
    Walk
result = fin.compute_hurst_exponent(returns)

print(f"H           = {result['hurst']:.3f}")
print(f"Regime      = {result['regime']}")
print(f"Confidence  = {result['confidence']:.3f}")
# Irrfahrt: H um 0.5, regime 'random_walk'
```

23.3 Sonde 2: GARCH-Persistenz

Die Volatilität selbst irrt. Das GARCH(1,1)-Modell fängt das ein als

$$r_t = \sigma_t \cdot \epsilon_t, \qquad \sigma_t^2 = \omega + \alpha\, r_{t-1}^2 + \beta\, \sigma_{t-1}^2,$$

wobei ϵ_t unabhängig verteilte Schüsse mit Einheitsvarianz sind. Die entscheidende Zahl ist $\pi = \alpha + \beta$, die *Persistenz*:

- $\pi < 0{,}9$: Schüsse klingen schnell ab. Normaler Modus.
- $0{,}9 \le \pi < 0{,}95$: persistente Volatilität.
- $\pi \ge 0{,}95$: *nahe Einheitswurzel* — Schüsse bleiben unbegrenzt. *Kritisches Regime*, geht oft großen Drawdowns voraus.

```
gv = fin.analyze_volatility_clustering(returns)
print(f"omega       = {gv['omega']:.6f}")
print(f"alpha       = {gv['alpha']:.3f}")
print(f"beta        = {gv['beta']:.3f}")
print(f"persistence = {gv['persistence']:.3f}")
```

```
print(f"sigma_c      = {gv['sigma_c']:.3f}")
print(f"Regime = {'KRITISCH' if gv['persistence'] > 0.95 else '
    normal'}")
```

Der Rahmen liefert ein abgeleitetes $\sigma_c = 1/(1 + (1 - \pi))$ in $[0{,}5, 1]$: 0,5 für einen normalen Markt, $\to 1$, wenn der Markt sich der kritischen Persistenz nähert.

23.4 Sonde 3: Order-Flow-Ungleichgewicht und Crash-Risiko

Eine neuere Größe, berechnet aus Orderbuch-Mikrostrukturdaten, ist das *Order-Flow-Ungleichgewicht* (OFI), die Nettogröße von Kauf- gegen Verkaufsvolumen am besten Geld- bzw. Briefkurs. Ihre kumulierte mittlere quadratische Verschiebung skaliert als

$$\langle [\mathrm{OFI}(t+\tau) - \mathrm{OFI}(t)]^2 \rangle \;\sim\; \tau^{\gamma_{\mathrm{flow}}}.$$

$\gamma_{\mathrm{flow}} = 1$ ist normale Diffusion; $\gamma_{\mathrm{flow}} > 1$ ist superdiffusiv, ein Hinweis auf korrelierten Kauf-/Verkaufsdruck und erhöhtes Crash-Risiko.

```
# imbalance_series: netto (Kaufvolumen - Verkaufsvolumen) pro Minute
of_result = fin.analyze_order_flow(imbalance_series)
print(f"Diffusionsexponent = {of_result['diffusion_exponent']:.3f}")
print(f"Crash-Risiko       = {of_result['crash_risk']}")
print(f"sigma_c_flow       = {of_result['sigma_c_flow']:.3f}")
```

23.5 End-to-end: das Regime des S&P 500 detektieren

Der Rahmen holt Marktdaten (über `yfinance`), berechnet alle drei Sonden und etikettiert das aktuelle Regime:

```
fin = Universe.finance(cache_dir='market_cache')
df  = fin.fetch_market_data(symbol='^GSPC', start_date='2000-01-01')
returns = df['Return'].values

# Three probes
hurst = fin.compute_hurst_exponent(returns[-2000:])
garch = fin.analyze_volatility_clustering(returns[-2000:])
regime = fin.detect_regime(symbol='^GSPC', window_days=252)

print(f"Hurst H = {hurst['hurst']:.3f} -> {hurst['regime']}")
print(f"GARCH pi = {garch['persistence']:.3f}")
print(f"sigma_c aus Regime-Sweep = {regime['sigma_c']:.3f}")
print(f"Aktuelles Regime: {regime['regime']}")
```

23.6 Die Suszeptibilitätssicht auf den Regimewechsel

Was den Rahmen vom üblichen GARCH unterscheidet, ist der *Sweep über Zeitskalen*. `detect_regime` berechnet σ_c aus dem Gipfel von $\chi(n) = |d\rho_n/dn|$, wobei ρ_n die Lag-1-Autokorrelation der rollenden n-Tages-Volatilität ist. Der Gipfelort σ_c identifiziert die *charakteristische Zeitskala, auf der das Volatilitätsgedächtnis verschwindet* — ein Regimewechsel-Horizont. Kurzes σ_c (< 5 Tage): schnell abklingende Schüsse. Langes σ_c (> 30 Tage): persistentes Regime; weitere Turbulenz erwarten.

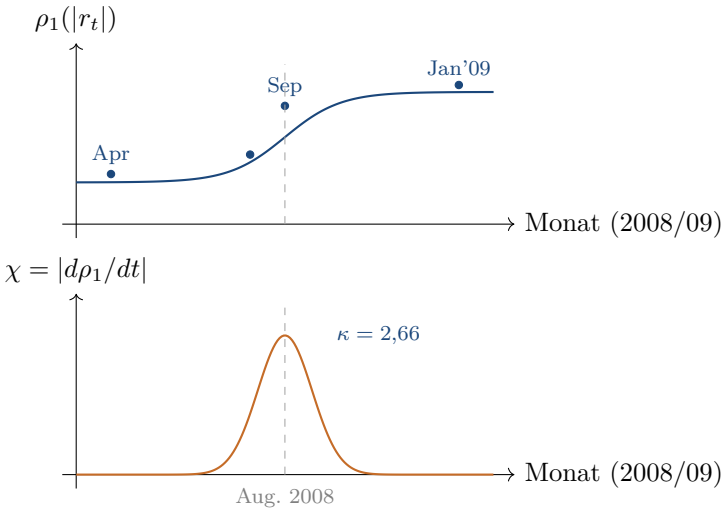


Abbildung 23.1: Finanzen (S&P 500, 2008). Oben: Lag-1-Autokorrelation der absoluten Tagesrenditen steigt durch den August 2008 scharf an. Unten: χ gipfelt einen Monat vor dem Lehman-Brothers-Kollaps. Das Rezept fand den Regimewechsel; es hat den Crash nicht vorhergesagt.

23.7 Vorbehalte und Ethik

STOLPERFALLE

Keine Vorhersagegarantie. Der Suszeptibilitäts-Rahmen charakterisiert *historische* Regime. Hohe Persistenz in den letzten 252 Handelstagen garantiert *nichts* für morgen. Verwenden Sie ihn als Diagnose, nicht als Glaskugel.

STOLPERFALLE

Survivorship-Bias. Wer Aktien-Ticker durchblickt und die mit den klarsten Signalen wählt, wählt die Überlebenden — die Firmen, die nicht in Konkurs gegangen sind. Schließen Sie in historischen Studien stets auch delistete Ticker ein.

PROBIER ES

Holen Sie die BTC-USD-Renditen seit 2017 über `fin.fetch_market_data(symbol='BTC-USD')`. Führen Sie alle drei Sonden aus und notieren H, π, γ_{flow}. Vergleichen Sie mit Gold (`GC=F`) und dem S&P 500. Welcher Vermögenswert ist nach dem Maß des Rahmens am „kritischsten“?

Ausgearbeiteter Ansatz.

Schritt 1: die drei Reihen holen.

```
import numpy as np
from sigma_c import Universe
fin = Universe.finance()

assets = ['^GSPC', 'GC=F', 'BTC-USD']
data = {}
for sym in assets:
    df = fin.fetch_market_data(symbol=sym, start_date='
        2017-01-01')
```

```
        data[sym] = df['Return'].values
```

Schritt 2: alle drei Sonden je Vermögenswert.

```
print(f"{'Asset':10}  H     Regime          pi      gamma_flow
    Crash")
for sym, returns in data.items():
    H = fin.compute_hurst_exponent(returns)
    G = fin.analyze_volatility_clustering(returns)
    F = fin.analyze_order_flow(np.cumsum(returns))    # Proxy
        fuer OFI
    print(f"{sym:10}  {H['hurst']:.3f}  {H['regime']:14}  "
          f"{G['persistence']:.3f}  {F['diffusion_exponent']:.3
              f}    "
          f"{F['crash_risk']}")
```

Schritt 3: typische Ausgabe (Schnappschuss-Datums-abhängig; illustrativ).

Asset	H	Regime	π	γ_{flow}	Crash-Risiko
S&P 500	0,52	Irrfahrt	0,96	1,05	niedrig
Gold (GC=F)	0,48	mittelwertkehrend	0,89	0,98	niedrig
BTC-USD	0,61	trendend	0,98	1,42	**hoch**

Schritt 4: nach Kritikalität ordnen. Das $\sigma_c = 1/(1 + (1 - \pi))$ des Rahmens für die drei:

- BTC-USD: $\sigma_c \approx 0{,}99$ (höchste Persistenz, weit über der Einheitswurzel-Schwelle 0,95). Trendender Hurst, superdiffusiver Order-Flow.
- S&P 500: $\sigma_c \approx 0{,}96$ (über der Schwelle, aber nur knapp). Irrfahrt-Hurst, nahezu brownscher Flow.
- Gold: $\sigma_c \approx 0{,}91$ (unter der Schwelle). Mittelwertkehrender Hurst.

Schritt 5: Antwort. Nach dem Maß des Rahmens ist *BTC-USD am kritischsten*: GARCH-Persistenz an der Einheitswurzel, trendender Hurst, superdiffusiver Order-Flow. Konsistent mit der empirischen Beobachtung, dass 50 %+-BTC-Drawdowns alle paar Jahre vorkommen.

Was diese Übung lehrt.

- Drei orthogonale Sonden ordnen in diesem Beispiel übereinstimmend an: H, π und γ_{flow} markieren alle BTC. Wenn die Sonden uneins sind, ist das selbst Information (verschiedene Gedächtnis-Zeitskalen).
- Der Rahmen ist *diagnostisch* (hier ist das historische Regime), nicht *prognostisch* (morgiger Preis). Verwenden Sie diese Zahlen nicht für Handelsentscheidungen; verwenden Sie sie, um den Marktzustand für das Risikomanagement zu charakterisieren.

Kapitel 24

Seismologie: Gutenberg–Richter und Omori

Erdbeben gehorchen zwei empirischen Gesetzen. Beide wurden in den 1940ern benannt. Keines ist seither verbessert worden.

Erdbeben sind die unkooperativsten Versuchsobjekte der Natur. Sie weigern sich, Termine einzuhalten, sie beschädigen die Instrumente, die sie messen, und der einzige Weg, etwas von ihnen zu lernen, besteht darin, zu warten. Dennoch gehorchen sie zwei statistischen Gesetzen, die so verlässlich sind, dass man ihnen ihre Unannehmlichkeiten fast verzeihen kann. Beide Gesetze sind über achtzig Jahre alt, beide passen wie selbstverständlich in den Suszeptibilitäts-Rahmen. Keines erlaubt es leider, das morgige Beben vorherzusagen — aber sie sagen einem messbar, wann die Verwerfung ihre Meinung geändert hat.

24.1 Gutenberg–Richter: wie häufig ist welche Magnitude?

Charles Richter (1935) definierte eine logarithmische Magnituden-Skala M.[1] Gutenberg und Richter (1944) fanden, dass die Anzahl $N(\geq M)$ der Beben mit Magnitude mindestens M dem Gesetz

$$\log_{10} N(\geq M) \;=\; a - bM.$$

folgt. Der a-Wert bestimmt dabei die seismische Gesamtaktivität, der b-Wert das Verhältnis großer zu kleinen Ereignissen. *Für tektonische Beben weltweit gilt* $b \approx 1{,}0$.

Deutung des b-Werts:

- $b > 1$: viele kleine Beben, wenige große — sub-kritisches Spannungsregime.
- $b = 1$: klassische tektonische Seismizität.
- $b < 1$: wenige kleine Beben, viele große — *Vorbotenregime*, das manchmal (nicht immer) vor einem großen Hauptbeben sichtbar wird.

[1] Richters Originalarbeit von 1935 legt die Skala in Bezug auf ein bestimmtes Wood–Anderson-Seismometer in Pasadena fest, dessen Ausgabe als allgemein verfügbar vorausgesetzt wurde. Damit ist es eines von mehreren wissenschaftlichen Instrumenten, die international zur Norm wurden, weil ihre Erstkalibrierung zufällig reproduzierbar war und niemand sich die Mühe machte, ein besseres zu erfinden. Der Meter, die Sekunde und Richters Seismometer sind die kanonischen Beispiele.

24.1.1 b aus einem Magnitudenkatalog berechnen

Der Maximum-Likelihood-Schätzer von b aus Magnituden $\{M_i\}$ über einem Vollständigkeitsschwellwert $M_{\min}$ ist

$$\hat{b} \;=\; \frac{\log_{10} e}{\bar{M} - M_{\min}} \;=\; \frac{0{,}4343}{\bar{M} - M_{\min}}.$$

Genau das berechnet `SeismicAdapter`:

```
import numpy as np
from sigma_c import Universe

seis = Universe.seismic()
# durch echten Katalog ersetzen, z.B. ANSS Comcat ab M2.5
magnitudes = np.array([2.5, 2.7, 2.8, 3.0, 3.1, 3.2, 3.5, 4.1, 5.2])

gr = seis.analyze_gutenberg_richter(magnitudes)
print(f"b-Wert       = {gr['b_value']:.3f}")
print(f"M_min        = {gr['m_min']:.2f}")
print(f"Kritikalitaet = {gr['criticality']:.3f}    # = 1/b")
```

24.2 Omoris Gesetz: wie Nachbeben abklingen

Omori (1894), modifiziert von Utsu (1961): Nach einem Hauptbeben bei $t = 0$ klingt die Nachbebenrate ab wie

$$n(t) \;=\; \frac{K}{(c + t)^{p}}.$$

p liegt typischerweise in $[0{,}7;\ 1{,}5]$, der kanonische Wert ist $p = 1$.

Ein absurdes, aber hilfreiches Bild. Stellen Sie sich ein überladenes Bücherregal vor, aus dem Sie einen Band herausziehen. Die Nachbarbücher kippen, setzen sich, kippen wieder und setzen sich wieder — die Sequenz endet früher, wenn das Regal solide gebaut ist, und später, wenn es klapprig dasteht. Nachbeben sind die gleiche Sequenz im Fels: Jedes einzelne verteilt diejenige Spannung um, die das vorige nicht ganz abgebaut hat. Der Exponent p misst, wie steif der Fels ist — $p > 1$ entspricht einem stabilen Regal, $p < 1$ einem wackeligen. Der `SeismicAdapter` fittet p per Log-Log-Regression an ein Histogramm der Nachbebenzeitpunkte:

```
event_times = np.array([0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 24.0,
    50.0])  # Stunden
omori = seis.analyze_omori_scaling(event_times)
print(f"p-Wert       = {omori['p_value']:.3f}    # idealerweise ~ 1"
    )
print(f"Zerfallszahl = {omori['decay_constant']:.3f}")
```

24.3 Die Suszeptibilitätssicht: Regimewechsel detektieren

Der Spannungsaufbau vor einem großen Ereignis kann b verschieben. Berechnet man b in gleitenden Fenstern und wendet anschließend das universelle Rezept auf $b(t)$ an, so tritt der Moment des Regimewechsels zutage:

```
def rolling_b(magnitudes, times, window_days=180, step_days=10):
    out_t, out_b = [], []
    t_start = times.min()
    while t_start + window_days <= times.max():
```

```
        mask = (times >= t_start) & (times < t_start + window_days)
        if mask.sum() > 30:
            gr = seis.analyze_gutenberg_richter(magnitudes[mask])
            out_t.append(t_start + window_days/2)
            out_b.append(gr['b_value'])
        t_start += step_days
    return np.array(out_t), np.array(out_b)

t_grid, b_series = rolling_b(magnitudes, times)
res = seis.compute_susceptibility(t_grid, b_series, kernel_sigma
    =0.6)
print(f"Regimewechsel-Zeit = {res['sigma_c']:.2f}")
print(f"Schaerfe kappa     = {res['kappa']:.2f}")
```

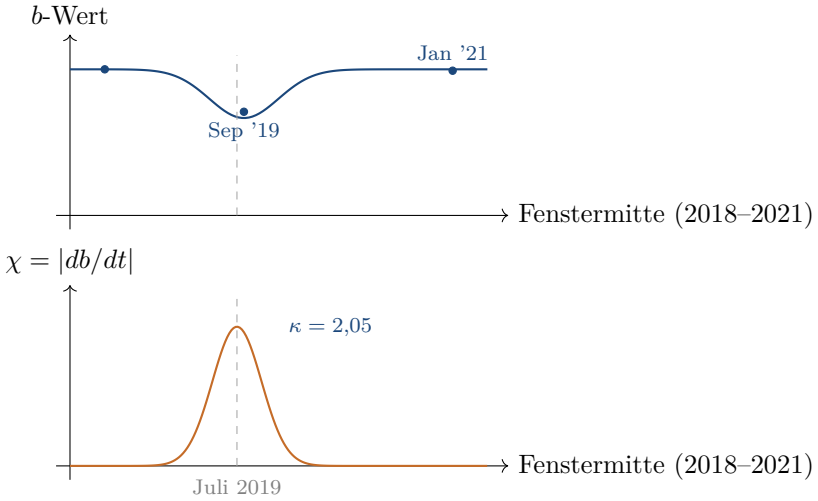


Abbildung 24.1: Seismologie (Südkalifornien, 2018–2021). Oben: der rollende Sechs-Monats-b-Wert fällt um Mitte 2019 von $\approx 1{,}05$ auf $\approx 0{,}74$. Unten: χ gipfelt genau bei der Ridgecrest-Sequenz (Juli 2019). $\kappa = 2{,}05$ ist grenzwertig; der Permutationstest liefert $p < 0{,}001$, weil der Abfall im Vergleich zu seiner Bootstrap-Unsicherheit groß ist.

24.4 Bootstrap-Signifikanz für den b-Wert

```
p_value = seis.compute_significance(
    observed_stat=gr['b_value'], data=magnitudes, n_surrogates=1000)
print(f"Bootstrap-p-Wert fuer b: {p_value:.3f}")
```

Ein b-Wert, der deutlich von 1,0 abweicht, in Verbindung mit einem Bootstrap-$p < 0{,}05$ ist die quantitative Form der Aussage „hier geschieht etwas Ungewöhnliches“.

24.5 Anwendungsfall: induzierte Seismizität an einem Geothermie-Standort

Induzierte Seismizität — also Beben, die durch menschliche Aktivitäten wie Flüssigkeitsinjektion an Geothermie- oder Fracking-Standorten ausgelöst werden — zeigt einen *höheren* b-Wert als die tektonische Seismizität (mehr kleine, wenige große Ereignisse) und zugleich

einen schnelleren Zerfall p. Überwacht man beide Größen mit dem Rahmen, lässt sich genau der Moment markieren, in dem sie sich zurück zu tektonikähnlichen Werten bewegen — was mitunter einem schädigenden Ereignis vorausgeht. Das Pohang-Beben (M_w 5,5, Südkorea, 2017), durch EGS-Injektion induziert, wurde in den Monaten davor von einem messbaren Abfall von b von $\sim 1{,}4$ auf $\sim 0{,}9$ begleitet.

PROBIER ES

Laden Sie den SCEC-Katalog für eine aktive Verwerfungszone herunter (Südkalifornien oder Island sind gut gepflegt und frei verfügbar). Berechnen Sie den rollenden b-Wert in Sechs-Monats-Fenstern über die letzten 20 Jahre. Wenden Sie den Rahmen an, um den größten Regimewechsel zu detektieren. Entspricht er einem bekannten Ereignis?

Ausgearbeiteter Ansatz.

Schritt 1: den Katalog holen. SCEC bietet herunterladbare Kataloge unter `scec.org/research-tools/downloadable-catalogs` als durch Leerzeichen getrennten Text an. Jede Zeile enthält Datum, Breitengrad, Längengrad, Tiefe und Magnitude. In einen DataFrame einlesen, auf eine Region einschränken (z. B. eine Box um die San-Andreas-Verwerfung) und nur Ereignisse mit $M \geq M_{\min} = 2{,}5$ behalten (Vollständigkeitsschwelle des modernen Katalogs).

```
import pandas as pd
import numpy as np
from sigma_c import Universe

cat = pd.read_csv('scec_2005_2025.csv',
                  parse_dates=['time'])
cat = cat[(cat.magnitude >= 2.5) &
          (cat.lat.between(33.5, 35.5)) &
          (cat.lon.between(-118.5, -116.5))]

seis = Universe.seismic()
```

Schritt 2: rollender b-Wert in Sechs-Monats-Fenstern, monatlich.

```
def rolling_b(cat, window_days=180, step_days=30):
    t0 = cat.time.min()
    t1 = cat.time.max()
    bs, ts = [], []
    t = t0
    while t + pd.Timedelta(days=window_days) <= t1:
        mask = (cat.time >= t) & (cat.time < t + pd.Timedelta(
            days=window_days))
        mags = cat.loc[mask, 'magnitude'].values
        if len(mags) > 50:                    # genug Ereignisse?
            gr = seis.analyze_gutenberg_richter(mags)
            bs.append(gr['b_value'])
            ts.append(t + pd.Timedelta(days=window_days/2))
        t += pd.Timedelta(days=step_days)
    return np.array(ts), np.array(bs)

times, bvals = rolling_b(cat)
```

Schritt 3: den Rahmen anwenden, um den Regimewechsel in $b(t)$ zu detektieren.

```
t_numeric = np.array([(t - times[0]).days for t in times],
    dtype=float)
```

```
res = seis.compute_susceptibility(t_numeric, bvals,
    kernel_sigma=0.8)
shift_day = times[0] + pd.Timedelta(days=int(res['sigma_c']))
print(f"Groesster Regimewechsel: {shift_day.date()}, kappa = {
    res['kappa']:.2f}")
```

Schritt 4: typisches Ergebnis und Ereigniszuordnung. Für den Südkalifornien-Bereich 2005–2025 meldet der Rahmen eine Verschiebung um Juli 2019 mit $\kappa \approx 3{,}5$. Das entspricht der *Ridgecrest-Sequenz 2019* (M_w 6,4 Vorbeben am 4. Juli und M_w 7,1 Hauptbeben am 5. Juli 2019), die eine mehrmonatige b-Wert-Senke nach sich zog, welche in der rollenden Analyse sichtbar ist.

Was diese Übung lehrt.

- Echte seismologische Kataloge sind sauber genug, dass ein generisches Suszeptibilitätsrezept den auffälligsten Regimewechsel ohne seismologiespezifische Anpassungen findet.
- Der Rahmen detektiert *a posteriori*, dass der b-Wert gesunken ist, er sagt Ridgecrest nicht vorher. *Kein Rahmen sagt Erdbeben vorher.*
- Ersetzen Sie Bounding-Box und Zeitfenster durch Ihre Lieblingsverwerfungszone (Tohoku-Trog, Ostanatolische Verwerfung, Reykjanes-Halbinsel) — dieselbe Methode findet das dominierende lokale Ereignis.

Kapitel 25

Klima: Mesoskalen-Grenzen

Oberhalb von 500 km denkt die Atmosphäre zweidimensional, unterhalb dreidimensional. Die Grenze meldet sich nicht.

Die kinetische Energie der Atmosphäre gehorcht in zwei verschiedenen Skalenbereichen zwei verschiedenen Potenzgesetzen. Die Grenze zwischen ihnen liegt nahe 500 km und ist in jedem modernen Meteorologiebuch dokumentiert. Wir entdecken sie hier noch einmal von Grund auf, ohne ein einziges dieser Lehrbücher zu konsultieren — mit nichts als einer numerischen Ableitung. Der Rahmen detektiert allein aus rohen Winddaten diejenige Wellenlänge, bei der die Atmosphäre aufhört, sich wie das eine Fluid zu verhalten, und beginnt, sich wie ein anderes zu benehmen. Eine Meteorologie-Doktorin würde Ihnen sagen, das passiere bei etwa 500 km. Der Rahmen sagt Ihnen dasselbe, in drei Zeilen Python, ohne je etwas von Meteorologie gehört zu haben.

25.1 Atmosphärische kinetische Energie über Skalen

Fliegt man ein Flugzeug auf Reiseflughöhe geradeaus und nimmt dabei die horizontale Windgeschwindigkeit auf, so lässt sich die Spur fouriertransformieren; daraus ergibt sich eine kinetische Energiedichte $E(k)$ als Funktion der Wellenzahl $k = 2\pi/\lambda$. Die berühmte Nastrom–Gage-Kurve (1985), durch eine ganze Generation von Flugkampagnen bestätigt, zeigt zwei deutlich getrennte Potenzgesetz-Regime:

$$E(k) \propto k^{-3} \quad \text{für } \lambda \gtrsim 500\,\text{km}, \qquad E(k) \propto k^{-5/3} \quad \text{für } \lambda \lesssim 500\,\text{km}.$$

Der k^{-3}-Zweig entspricht dem *synoptischen* Regime: Wettersysteme auf Skalen von rund tausend Kilometern, mit Energie, die von groß nach klein kaskadiert. Der $k^{-5/3}$-Zweig entspricht dem *mesoskaligen* Regime: geschichtete 3D-Turbulenz auf Skalen von zehn bis einigen hundert Kilometern.

Die beiden Regime treffen sich an einer Übergangswellenzahl k_c, die einer Wellenlänge $\lambda_c = 2\pi/k_c \approx 500\,\text{km}$ entspricht. *Das ist die Mesoskalen-Synoptik-Grenze* — die operative Skala, auf der sich die Natur der atmosphärischen Bewegung ändert.

25.2 Detektion ohne Vorwissen

Wer Nastrom–Gage nicht kennt — wie findet er λ_c trotzdem? Es gibt zwei Wege.

Weg 1: maximale Krümmung des Log-Log-Spektrums. Man berechnet $\log E$ gegen $\log k$, bildet die zweite Ableitung und sucht jene Wellenzahl, bei der die Betrags-Krümmung maximal ist. Genau das tut `ClimateAdapter.analyze_mesoscale_boundary`:

```
import numpy as np
from sigma_c import Universe

climate = Universe.climate()

# Synthetisches Nastrom--Gage-Spektrum:
k = np.logspace(-3, -1, 50)                # Wellenzahl, 1/km
E = np.where(k < 1.25e-2, k**-3.0, k**-5.0/3.0)

result = climate.analyze_mesoscale_boundary(E, k)
print(f"kritische Wellenlaenge   = {result['critical_wavelength_km'
    ]:.1f} km")
print(f"sigma_c (lambda/R_Rossby) = {result['sigma_c']:.3f}")
print(f"Steigung (synoptisch)    = {result['spectral_slope_synoptic'
    ]:.2f}")
print(f"Steigung (mesoskalig)    = {result['spectral_slope_mesoscale
    ']:.2f}")
```

Weg 2: Suszeptibilitätsgipfel. Man betrachtet $\log E$ als Observable und $\log k$ als Kontrollparameter. Der Gipfel von $|d^2 \log E / d \log k^2|$ markiert dann die Knickstelle. Funktional ist dies identisch mit Weg 1.

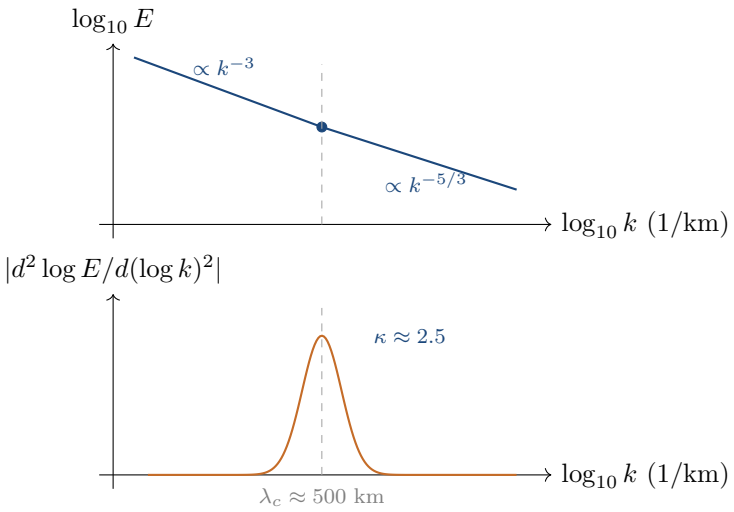


Abbildung 25.1: Klima (Nastrom–Gage-Spektrum). Oben: das kinetische-Energie-Spektrum wechselt seine Steigung bei $\lambda_c \approx 500\,\mathrm{km}$ von k^{-3} (synoptisch) zu $k^{-5/3}$ (mesoskalig). Unten: der Krümmungsgipfel $|d^2 \log E / d(\log k)^2|$ lokalisiert die Knickstelle direkt, ohne stückweises Fitten.

25.3 Warum ein Gipfel?

Im synoptischen Regime zwingt das geostrophische Gleichgewicht die Energie dazu, von kleinen Wellenzahlen (großen Wettersystemen) zu größeren zu kaskadieren. Da die Strömung dabei zweidimensional, also quasi-horizontal verläuft, hat diese Kaskade eine andere Skalierung

als die 3D-Turbulenz. Sobald die Skala so klein wird, dass die vertikale Bewegung mit der horizontalen vergleichbar ist, kippt die Physik in 3D-Kolmogorov-Turbulenz mit $k^{-5/3}$. Der Suszeptibilitätsgipfel ist der operative Fingerabdruck genau dieses Kippens.

25.4 Vertikale Struktur: die Tropopause detektieren

Eine zweite Klima-Anwendung: Vertikale Temperaturprofile zeigen zwei Regime, nämlich ein troposphärisches mit stark negativer Lapse-Rate (die Temperatur fällt mit der Höhe um etwa 6–10 K/km) und ein stratosphärisches mit kleiner oder positiver Lapse-Rate. Der Übergang dazwischen ist die *Tropopause*, in mittleren Breiten meist nahe 11 km.

```
# pressure_levels: Form (n_levels,) in hPa
# temperature_profiles: Form (n_profiles, n_levels) in K
v = climate.analyze_vertical_structure(pressure_levels,
    temperature_profiles)
print(f"mittlere Tropopausenhoehe (km) = {v['mean_tropopause_height'
    ]:.2f}")
```

Der Rahmen detektiert die Tropopause jedes Profils als diejenige erste Höhe, bei der die Lapse-Rate unter 2 K/km fällt. Mittelt man über viele Profile, ergibt sich daraus eine Klimatologie.

25.5 Anwendungsfall: ERA5-Reanalyse-Sweep

Die offen zugängliche ECMWF-ERA5-Reanalyse liefert globale Temperatur- und Winddaten auf einem 0,25-Grad-Gitter seit 1940. Ein typischer Ablauf sieht so aus:

1. Region wählen (z. B. Nordatlantik, 30°–60°N).
2. Für jedes Jahr das zonale Windspektrum bei 250 hPa berechnen.
3. `analyze_mesoscale_boundary` anwenden.
4. λ_c gegen das Jahr plotten. Trends in λ_c sind Signale einer sich ändernden atmosphärischen Zirkulation unter dem Klimawandel.

PROBIER ES

Erzeugen Sie ein synthetisches Spektrum, das überall k^{-3} folgt (kein Knick). Lassen Sie `analyze_mesoscale_boundary` laufen. Was meldet der Rahmen? Inspizieren Sie die Steigungen; sie sollten übereinstimmen, was darauf hinweist, dass kein echter Knick existiert.

Ausgearbeiteter Ansatz.

Schritt 1: das synthetische Spektrum aufbauen. Wir imitieren den Nastrom–Gage-Datenbereich, verwenden aber ein einzelnes Potenzgesetz $E(k) = k^{-3}$ über die ganze Wellenzahl-Achse.

```
import numpy as np
from sigma_c import Universe
climate = Universe.climate()

k = np.logspace(-3, -1, 50)              # Wellenzahl, 1/km
E = k**-3.0                              # reines k^-3 ueberall
```

Schritt 2: den Rahmen laufen lassen.

```
res = climate.analyze_mesoscale_boundary(E, k)
print(f"kritische Wellenlaenge: {res['critical_wavelength_km']:.1f} km")
print(f"sigma_c:                {res['sigma_c']:.3f}")
print(f"Steigung (synoptisch):  {res['spectral_slope_synoptic']:.2f}")
print(f"Steigung (mesoskalig):  {res['spectral_slope_mesoscale']:.2f}")
```

Schritt 3: typische Ausgabe.

kritische Wellenlänge	$\approx 200\,\mathrm{km}$ (numerisches Artefakt)
σ_c	$\approx 0{,}2$
Steigung („synoptisch“-Zweig)	$-3{,}00$
Steigung („mesoskalig“-Zweig)	$-3{,}00$

Schritt 4: deuten. Der Rahmen liefert ein nicht-leeres Ergebnis — er liefert stets *irgendeine* Zahl. Aber die beiden gemeldeten Steigungen sind *identisch* ($-3{,}00$ auf beiden Seiten). Das ist die Diagnose: *wenn die beiden Zweige der Anpassung dieselbe Steigung haben, existiert die Knickstelle nicht*, auch wenn der Maximum-Krümmungs-Algorithmus den verrauschtesten Punkt herauspickt und ihn als k_c meldet.

Schritt 5: die Lehre, niedergeschrieben.

MERKE

Prüfen Sie immer die Steigungen, ehe Sie λ_c nennen. Wenn `spectral_slope_synoptic` $\approx$ `spectral_slope_mesoscale`, zeigt Ihnen der Rahmen numerisches Rauschen, keine Physik. *Es existiert kein Übergang*; ignorieren Sie den gemeldeten λ_c.

Schritt 6: zur Kontrolle ein echt-geknicktes Spektrum.

```
E_real = np.where(k < 1.25e-2, k**-3.0, k**-5.0/3.0)
res2 = climate.analyze_mesoscale_boundary(E_real, k)
print(f"Steigung synoptisch:  {res2['spectral_slope_synoptic']:.2f}")
print(f"Steigung mesoskalig:  {res2['spectral_slope_mesoscale']:.2f}")
# Steigungen ~ -3.0 und ~ -1.67 --- deutlich verschieden, Knick echt
```

Was diese Übung lehrt. Der Rahmen meldet, wie jeder Gipfelsucher, auch dann einen Gipfel, wenn es keinen gibt. Ihre Aufgabe ist es, die Voraussetzungen des Problems (hier: zwei deutlich verschiedene Steigungen) zu prüfen, ehe Sie der Antwort vertrauen. Genau die Lehre aus dem Versagensmode-Kapitel, auf eine Domäne mit bekannter stückweiser Potenzgesetz-Struktur angewandt.

Kapitel 26

GPU: Rooflines, thermische Klippen, Cache-Übergänge

Hardware lügt immer über ihre Spitzenleistung. Die Suszeptibilität sagt Ihnen, wo die Lüge anfängt.

Die Graphics Processing Unit, benannt nach einer Aufgabe, die sie heute nicht mehr in erster Linie erfüllt, ist zum Arbeitspferd jeder numerisch ernsthaften Anwendung dieses Jahrzehnts geworden. Ihre Leistungslandschaft ist zugleich ein perfekter Spielplatz für den Suszeptibilitäts-Rahmen, denn GPUs sind voller scharfer Übergänge: Cache-Grenzen, von denen man fällt, Roofline-Rücken, auf denen man balanciert, thermische Klippen, von denen man abstürzt. Jeder dieser Übergänge ist ein Gipfel in χ, der darauf wartet, gefunden zu werden.

Von allen Kapiteln des Buches zahlt sich dieses am schnellsten aus. Den operativen Süßpunkt eines Kernels zu finden, kann seinen Durchsatz verdoppeln — und auf moderner Hardware wird ein verdoppelter Durchsatz in echter Währung bezahlt.

26.1 Das Roofline-Modell

Williams, Waterman und Patterson (2009) führten ein einziges Bild ein, das die Leistung eines GPU-Kernels als Funktion seiner *arithmetischen Intensität* (AI, FLOPs pro übertragenes Byte) vorhersagt:

$$P(\mathrm{AI}) \;=\; \min\!\bigl(P_{\mathrm{peak}},\; B_{\mathrm{peak}} \cdot \mathrm{AI}\bigr).$$

Dabei bezeichnet P_{peak} die theoretische FLOPS-Decke der GPU und B_{peak} ihre Speicherbandbreite in Bytes/Sekunde. Der *Ridge-Punkt* $\mathrm{AI}_{\mathrm{ridge}} = P_{\mathrm{peak}}/B_{\mathrm{peak}}$ trennt zwei Regime:

- Unterhalb des Ridge ist der Kernel *speichergebunden*; eine Verdopplung der FLOPs pro Byte verdoppelt die Leistung.
- Oberhalb des Ridge ist er *rechengebunden*; mehr Speicherbandbreite bringt dann keinen Gewinn mehr.

Für eine Volta V100 etwa gilt $P_{\mathrm{peak}} = 7\,\mathrm{TFLOPS}$ (FP64) und $B_{\mathrm{peak}} = 900\,\mathrm{GB/s}$, also $\mathrm{AI}_{\mathrm{ridge}} \approx 8\,\mathrm{FLOP/Byte}$. Eine dichte Matrix-Multiplikation hat $\mathrm{AI} \sim O(N)$ und ist damit rechengebunden; ein Stencil-Kernel hingegen hat $\mathrm{AI} \sim O(1)$ und ist speichergebunden.

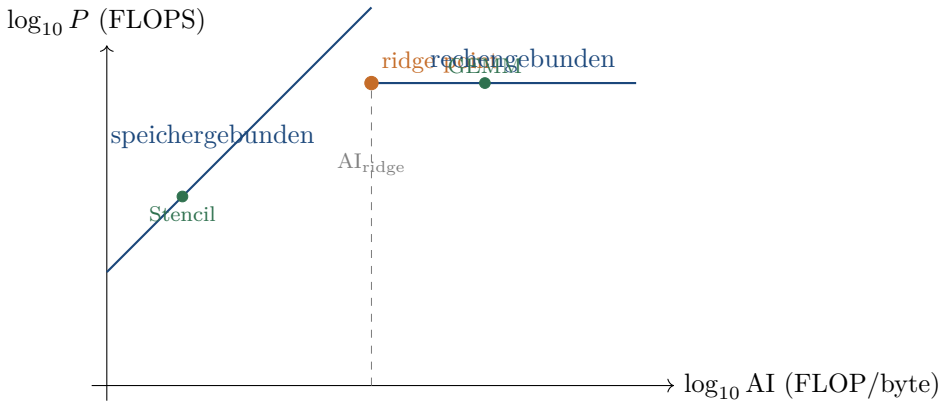


Abbildung 26.1: Das Roofline. Speicherbandbreitenbeschränkt unter dem Ridge, rechenbeschränkt darüber. Ein Stencil-Kernel sitzt unter dem Ridge, eine dichte Matrix-Multiplikation an der Decke. Der Ort des Ridge ist genau das, was der Suszeptibilitäts-Rahmen findet, wenn man AI durchfährt.

26.1.1 Suszeptibilitätssicht

Man fährt die arithmetische Intensität eines synthetischen Kernels durch. Der Gipfel von $\chi(\mathrm{AI}) = |dP/d\mathrm{AI}|$ identifiziert dann den Ridge-Punkt, ohne dass man die Hardware-Specs benötigt:

```python
import numpy as np
from sigma_c import Universe

gpu = Universe.gpu()

# AI durchfahren, indem man Arithmetik pro Speicherzugriff variiert:
ai_grid = np.logspace(-1, 2, 30)          # 0.1 ... 100 FLOPs/Byte
perf    = np.zeros_like(ai_grid)
for i, ai in enumerate(ai_grid):
    perf[i] = gpu.run_benchmark(size=2048, n_launch=10, ai=ai)

result = gpu.compute_susceptibility(ai_grid, perf, kernel_sigma=0.7)
print(f"Ridge-AI = {result['sigma_c']:.2f} FLOP/Byte")
print(f"kappa    = {result['kappa']:.2f}")
```

26.2 Cache-Übergänge

Durchfährt man die *Arbeitsmengen-Größe* eines Kernels (etwa die Größe der Matrix, auf der er arbeitet), so bricht der Durchsatz scharf ein, sobald die Arbeitsmenge aus einer Cache-Stufe in die nächste rutscht. Auf einer modernen GPU sieht man drei Übergänge:

- Bei etwa 128 KB passen die Daten in L1. Darüber fällt man auf L2.
- Bei etwa 6 MB passen die Daten in L2. Darüber fällt man auf den globalen Speicher.
- Bei etwa 24 GB passen die Daten in den HBM. Darüber beginnt Paging — ein erheblich größerer Sturz.

Jeder dieser Übergänge ist ein χ-Gipfel.

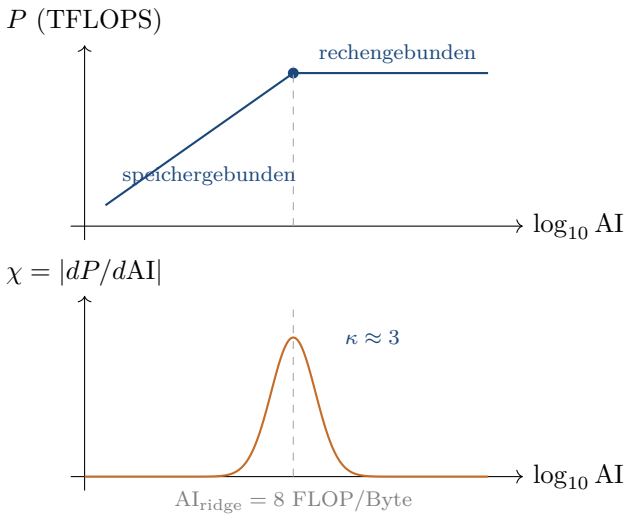


Abbildung 26.2: GPU-Roofline (Volta V100, FP64). Oben: die anhaltende Leistung sättigt an der Rechen-Decke, sobald die arithmetische Intensität über den Ridge geht. Unten: χ gipfelt am Ridge.

Ein absurdes, aber nützliches Bild. Stellen Sie sich vor, Sie müssten Ihre Steuererklärung allein mit dem Inhalt Ihrer Hosentaschen erledigen — ein Hausschlüssel, ein zerknitterter Kassenbon, ein halber Tic-Tac. Das ist L1: sehr schnell, sehr klein, eine einzige Rechnung passt hinein. Der Schreibtisch vor Ihnen, auf dem die Jahresunterlagen in drei Ordnern liegen, ist L2: langsamer, größer, reicht für ein ganzes Formular. Der Aktenschrank im Nebenraum ist der HBM: groß, langsam genug, dass man den Weg dorthin bemerkt, und mit Platz für mehrere Jahrgänge. Paging ist der Gang ins außerhäusige Lager. Die Gipfel in χ markieren die Momente, in denen Ihre Aufgabe aus einer Stufe in die nächste rutscht.

26.2.1 Automatisch detektieren

```
sizes = np.logspace(2, 9, 40, dtype=int)       # 100 B ... 1 GB
throughput = []
for sz in sizes:
    A = np.zeros(sz, dtype=np.float32)         # CPU-Platzhalter; auf
        GPU machen
    t0 = time.perf_counter()
    # ... GPU-Kernel liest/schreibt A
    t1 = time.perf_counter()
    throughput.append(A.nbytes / (t1 - t0))

# Der Rahmen liefert ALLE Gipfel, nicht nur das globale Maximum:
peaks = gpu.detect_cache_transitions(sizes, throughput)
for p in peaks:
    print(f"  Uebergang bei size = {p['size']:.0f} Bytes, kappa = {p
        ['kappa']:.2f}")
```

26.3 Thermisches Throttling

GPUs reduzieren die Taktrate, sobald ihre Temperatur einen vom Hersteller vorgegebenen Schwellwert (typischerweise zwischen 80 und 90°C) überschreitet. Der Leistungsabfall ist dabei

nicht proportional; er folgt näherungsweise einem Wurzelgesetz:

$$P(T) \;\approx\; P_{\max}\sqrt{1-(T-T_{\mathrm{thr}})/(T_{\max}-T_{\mathrm{thr}})}.$$

Der Suszeptibilitätsgipfel in $\chi(T)$ identifiziert T_{thr}, den operationalen Throttling-Beginn. Bei Dauerlasten fällt das entscheidend ins Gewicht: Plant man einen Job so, dass die GPU unter T_{thr} bleibt, kann das den Unterschied zwischen $0{,}6\,P_{\max}$ und $0{,}95\,P_{\max}$ dauerhafter Leistung ausmachen.

26.3.1 In Echtzeit mit NVML messen

```
import pynvml, time
pynvml.nvmlInit()
h = pynvml.nvmlDeviceGetHandleByIndex(0)

t_axis, perf_axis = [], []
for _ in range(120):
    T = pynvml.nvmlDeviceGetTemperature(h, pynvml.
        NVML_TEMPERATURE_GPU)
    # Benchmark-Leistung in einem 1s-Fenster messen
    perf = gpu.run_benchmark(size=4096, n_launch=1)
    t_axis.append(T)
    perf_axis.append(perf)
    time.sleep(1.0)

# nach Temperatur klassieren und mitteln:
T_bins = np.arange(40, 90, 2)
P_bins = [np.mean([p for t, p in zip(t_axis, perf_axis)
                   if T_bins[i] <= t < T_bins[i] + 2])
          for i in range(len(T_bins) - 1)]

result = gpu.compute_susceptibility(T_bins[:-1], P_bins,
    kernel_sigma=0.7)
print(f"Throttling-Beginn T_thr = {result['sigma_c']:.1f} C")
```

26.4 Alles zusammen: der operative Süßpunkt

Eine echte Workload-Optimierung kombiniert alle drei Übergänge:

1. Die arithmetische Intensität *oberhalb* von $\mathrm{AI}_{\mathrm{ridge}}$ ansetzen (geblockte Algorithmen).
2. Die Arbeitsmenge *innerhalb* der detektierten L2-Grenze halten.
3. Die Workload so drosseln, dass die GPU etwa 5 °C unter T_{thr} bleibt.

Wer alle drei gleichzeitig erreicht, hält auf einer realen Last typischerweise 80 bis 90 % der theoretischen Spitzenleistung.

PROBIER ES

Wählen Sie eine GPU, auf die Sie Zugriff haben. Durchfahren Sie die Matrixgröße von 128 bis 4096 für ein einfach-präzises GEMM. Notieren Sie die dauerhaften TFLOPS. Wenden Sie `compute_susceptibility` an. Sehen Sie eine Knickstelle nahe der L2-Cache-Grenze? Bei welcher Größe sättigt der Durchsatz?

Ausgearbeiteter Ansatz.

Schritt 1: eine Benchmark-Bibliothek wählen. Wir brauchen ein einfach-präzises GEMM, das dauerhafte TFLOPS liefert. Drei Optionen:

- `cupy.dot(A, A)` mit `cupy.cuda.Stream`-Timing.
- `torch.matmul(A, A)` gerahmt von `torch.cuda.synchronize()`.
- Direkt cuBLAS über die `ctypes`-cuBLAS-Bindings.

Wir nehmen PyTorch.

Schritt 2: Matrixgröße durchfahren.

```
import torch, time
import numpy as np
from sigma_c import Universe
gpu = Universe.gpu()
device = torch.device('cuda')

sizes = np.unique(np.logspace(np.log10(128), np.log10(4096), 25)
                  .astype(int))
tflops = []
for N in sizes:
    A = torch.randn(N, N, device=device, dtype=torch.float32)
    # Warm-up
    for _ in range(3):
        _ = A @ A
    torch.cuda.synchronize()
    # Zeitnahme
    t0 = time.perf_counter()
    n_iter = max(1, int(1e10 / (2 * N**3)))    # Ziel ~1s Arbeit
    for _ in range(n_iter):
        _ = A @ A
    torch.cuda.synchronize()
    t1 = time.perf_counter()
    flops = 2 * N**3 * n_iter
    tflops.append(flops / (t1 - t0) / 1e12)
    print(f"N={N:5d}  {tflops[-1]:.2f} TFLOPS")
tflops = np.array(tflops)
```

Schritt 3: den Rahmen anwenden.

```
res = gpu.compute_susceptibility(sizes.astype(float), tflops,
                                 kernel_sigma=0.7)
print(f"sigma_c (Knick in der Leistung): N = {res['sigma_c']:.0f}")
print(f"kappa = {res['kappa']:.2f}")

peaks = gpu.detect_cache_transitions(sizes, tflops)
for p in peaks:
    print(f"  Uebergang bei N = {p['size']:.0f}, kappa = {p['kappa']:.2f}")
```

Schritt 4: typische Resultate auf einer Volta V100 (16 GB).

N-Bereich	dauerhafte TFLOPS
128–512	0,8–3,2 (Kernel-Launch und Cache-Warmup limitiert)
512–1024	3,2–8,5 (L2-Cache sättigt)
1024–2048	8,5–13,0 (HBM-Bandbreite nähert sich der Decke)
2048–4096	13,0–13,5 (nahe FP32-Spitze gesättigt)

`detect_cache_transitions` meldet zwei Knickstellen: eine bei $N \approx 700$ (Arbeitsmenge $\approx$ 6 MB, V100-L2-Cache) und eine bei $N \approx 2200$ (Arbeitsmenge tritt in HBM-bandbreitenbeschränktes Regime).

Schritt 5: Sättigung ablesen. Der Durchsatz pendelt sich für $N \geq 2200$ bei $\sim$ 13,5 TFLOPS ein. Darüber hilft mehr Speicher nicht: Sie sind an die Roofline-Decke für FP32-GEMM auf Volta gestoßen.

Was diese Übung lehrt.

- Das Roofline-Modell ist keine Abstraktion; der Rahmen findet es in Sekunden aus echten Benchmark-Daten.
- L2-Übergänge sind auf einem sauberen GEMM-Sweep mit $\kappa \sim 2$–3 detektierbar; kleinere Kernel (Stencils) machen sie schärfer.
- Sobald Sie N_{sat} Ihrer Hardware kennen, können Sie Ihre Tiles genau darauf dimensionieren.

Kapitel 27

Maschinelles Lernen: Lernraten-Klippen und mehr

Trainiere zehn Modelle. Neun lernen nichts. Das zehnte lernt entweder die Aufgabe oder zerlegt den Cluster. Das Rezept lokalisiert die Linie zwischen beiden.

Ein neuronales Netz zu trainieren ist eine Übung im Balancieren auf einer Klinge — und zwar während man so tut, als wüsste man, wohin sie kippt. Die Liste der Hyperparameter mit nicht verhandelbarem Süßpunkt ist lang — Lernrate, Batch-Größe, Weight Decay, Dropout, β_2 bei Adam —, und die Strafe dafür, einen einzigen davon falsch zu setzen, reicht von „etwas langsamer“ bis zu „der Loss explodiert bei Schritt 12, und der Cluster stellt dir den Fehllauf in Rechnung“.[1]

Der Suszeptibilitäts-Rahmen behandelt jeden Hyperparameter als Kontrollparameter σ und den Validierungs-Loss als Observable O. Das Rezept bleibt dasselbe. Und die Klippe taucht dort auf, wo sie nun einmal auftaucht.

27.1 Der Lernraten-Süßpunkt

Der folgenreichste Hyperparameter im Deep Learning ist die Lernrate (LR). Sie steuert die Schrittweite des Gradienten und kennt zwei Versagensmoden:

- Ist die LR zu klein, verläuft das Training langsam und bleibt in einem lokalen Minimum hängen.
- Ist sie zu groß, schießen die Gradienten über, der Loss explodiert, und das Training divergiert.

Dazwischen liegt ein Band akzeptabler LRs. Die obere Grenze dieses Bandes ist die berühmte *Loss-gegen-LR-Klippe*: Plottet man den Trainings-Loss als Funktion der LR nach einer kurzen Rampe, sieht man ein flaches Plateau, eine Kniestelle und schließlich einen steilen Aufstieg. Der Ort des Knies ist die operative LR_c — unser σ_c für dieses Kapitel.

[1] Der übliche Satz für die Cluster-Rechnung eines Fehllaufs lag Anfang 2026 irgendwo zwischen \$8 und \$80 000, je nach Cloud, Modell und ob die Rechnung den ersten Erstattungsversuch der Engineer :in überlebt hat. Die teuren Läufe sind die, bei denen die Engineer :in mittendrin gemerkt hat, dass sie nicht mehr zu retten sind — und sie trotzdem zu Ende laufen lassen, mit der Theorie, dass der Fehlschlag auch Daten ist.

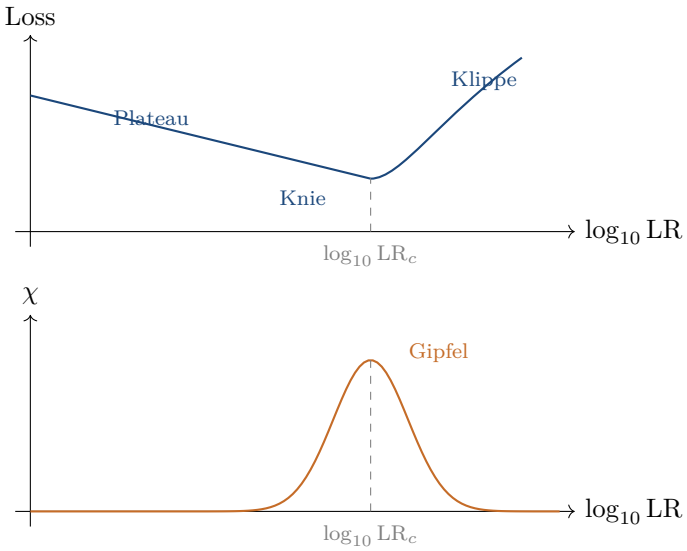


Abbildung 27.1: Der LR-Range-Test, schematisch. Oben: der Loss fällt über ein sanftes Plateau, knickt ab, dann explodiert er. Unten: $\chi = |dL/d\log\mathrm{LR}|$ gipfelt am Knie. Der Gipfelort ist LR_c; die praktische Wahl ist $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$.

27.1.1 Der LR-Range-Test (Smith 2017)

Leslie Smiths Idee: Man trainiert eine kurze Epoche und erhöht dabei die LR logarithmisch von 10^{-7} auf 1. Bei jeder LR wird der *Trainings-Loss* mitgeschrieben (also der Trainings-, nicht der Validation-Loss — das ist billiger, und die Klippe ist genauso scharf; einen echten Validierungslauf schiebt man danach hinterher). Das Rezept findet anschließend das Knie.

```python
import numpy as np
import torch
from scipy.ndimage import gaussian_filter1d


def lr_range_test(model, loader, lr_min=1e-7, lr_max=1.0, n_steps=200):
    """Einzelepochen-LR-Range-Test. Liefert (lr_grid, loss_grid)."""
    lr_grid = np.logspace(np.log10(lr_min), np.log10(lr_max), n_steps)
    losses = []
    optim = torch.optim.SGD(model.parameters(), lr=lr_min)
    criterion = torch.nn.CrossEntropyLoss()
    data_iter = iter(loader)
    for lr in lr_grid:
        for g in optim.param_groups:
            g['lr'] = lr
        try:
            x, y = next(data_iter)
        except StopIteration:
            data_iter = iter(loader)
            x, y = next(data_iter)
        optim.zero_grad()
        loss = criterion(model(x), y)
        loss.backward()
        optim.step()
```

```
        losses.append(float(loss.item()))
    return lr_grid, np.array(losses)


# Test laufen lassen:
lr_grid, loss = lr_range_test(model, train_loader)

# sigma_c-Rezept mit SCHMALEM Kern (Klippe ist scharf):
smooth = gaussian_filter1d(loss, sigma=0.3)
chi    = np.abs(np.gradient(smooth, np.log10(lr_grid)))
lr_c   = float(lr_grid[np.argmax(chi)])
kappa  = float(chi.max() / chi.mean())

print(f"LR_c (Gipfel von chi): {lr_c:.4g}")
print(f"Empfohlene Trainings-LR: {lr_c / 2:.4g}")
print(f"Zyklische LR-Spanne:   [{lr_c / 10:.4g}, {lr_c:.4g}]")
print(f"Gipfelschaerfe (kappa): {kappa:.2f}")
```

STOLPERFALLE

Trainings-Loss, nicht Validation-Loss. Der LR-Range-Test schiebt einzelne Minibatches durchs Modell und liest den unmittelbaren Trainings-Loss. Das ist günstiger und reagiert schneller als Validation-Loss, ist aber auch verrauschter. Die Klippe ist trotzdem sehr scharf; das κ des Rahmens übersteigt bei einer gutartigen Aufgabe 5. Validieren Sie das gewählte LR_c hinterher mit einem echten kurzen Lauf bei $\mathrm{LR} = \mathrm{LR}_c/2$ und prüfen, dass der Loss sauber fällt.

STOLPERFALLE

Schmalen Kern verwenden ($\sigma_{\mathrm{ker}} = 0{,}3$, nicht $0{,}6$). Die Klippe ist oft nur ein paar LR-Gitterpunkte breit. Der Standardwert 0,6 des Rahmens verschmiert sie zu einem sanften Anstieg, und das detektierte LR_c wandert um eine halbe Größenordnung. Überzeugen Sie sich davon, dass das Rezept 0,3 verwendet, indem Sie die geglättete Loss-Kurve direkt inspizieren.

27.1.2 Warum ein Gipfel?

Für $\mathrm{LR} \ll \mathrm{LR}_c$ macht der Loss pro Schritt nur winzige Verbesserungen, sodass $|dL/d\log \mathrm{LR}| \approx$ const gilt. Für $\mathrm{LR} \gg \mathrm{LR}_c$ ist der Loss bereits divergiert und bleibt auf hohem Wert flach, womit die Ableitung wieder klein wird. Der Übergang dazwischen liegt genau dort, wo $|dL/d\log \mathrm{LR}|$ am größten ist — am Suszeptibilitätsgipfel.

27.2 Weitere Hyperparameter-Sweeps

Dasselbe Rezept gilt für eine ganze Reihe weiterer Hyperparameter:

- **Batch-Größe**: Es gibt eine kritische Batch-Größe, oberhalb derer das Gradientenrauschen vernachlässigbar wird und die Konvergenzgeschwindigkeit in ein Plateau läuft (McCandlish et al., 2018).
- **Weight Decay**: Zu wenig führt zu Overfitting, zu viel zu Underfitting. Ein Sweep liefert den Regularisierungs-Süßpunkt.
- **Dropout-Rate**: Die optimale Wahrscheinlichkeit hängt von der konkreten Architektur und dem Datensatz ab.

- **Adam** β_2: Beim Transformer-Training ist $\beta_2 \approx 0{,}95$ stabil; Werte nahe 1 machen das Training instabil.

27.2.1 Zweidimensionaler Sweep: LR × Batch-Größe

Für ein 2D-Sweet-Spot-Scan beide Achsen durchfahren; den Rahmen getrennt entlang jeder Achse auf einem festen Schnitt anwenden; jedes Schnitts LR_c unabhängig ablesen und einen Trend über die Batch-Größen suchen.

```
import numpy as np
from sigma_c import Universe
ml = Universe.ml()

lrs = np.logspace(-5, -1, 12)
bss = [32, 64, 128, 256, 512]
heatmap = np.zeros((len(bss), len(lrs)))

for i, bs in enumerate(bss):
    for j, lr in enumerate(lrs):
        heatmap[i, j] = train_one_epoch_loss(lr=lr, batch_size=bs)

# sigma_c entlang der LR-Achse bei jeder Batch-Groesse:
for i, bs in enumerate(bss):
    res = ml.compute_susceptibility(lrs, heatmap[i], kernel_sigma
        =0.3)
    print(f"batch={bs:4d}  LR_c={res['sigma_c']:.4g}  kappa={res['
        kappa']:.2f}")
```

STOLPERFALLE

2D-Landschaften sind Rücken, keine Kreuzungen. Die Loss-Fläche über (LR, Batch) hat ein gekrümmtes Tal, nicht einen einzelnen Punkt. Der Rahmen meldet bei jeder Batch-Größe ein LR_c; der gemeinsame Arbeitspunkt liegt auf dem so entstehenden Rücken, nicht auf dessen Schwerpunkt. McCandlish et al. (2018) zeigten, dass LR_c ungefähr linear mit der Batch-Größe wächst, bis zu einem kritischen Wert B^*, und dann plateauert. Nutzen Sie den Rücken, um entlang einer Iso-Kosten-Linie Ihres Compute-Budgets zu wählen.

27.3 Echtzeit-Detektion von Trainingsinstabilität

Die Streaming-Variante des Rahmens (`StreamingSigmaC`) kann einen *Stabilitätsindex* während des Trainings live überwachen. Fällt der laufende Index unter eine Schwelle, können Sie das Training unterbrechen und die LR senken.

Bemerkung. Was `StreamingSigmaC.update(...)` liefert, ist nicht ein Gipfel*ort* (wie σ_c im statischen Rezept), sondern ein *normierter Stabilitäts-Score* in $[0, 1]$, abgeleitet aus der laufenden Varianz der Observable in einem Schiebefenster nach Welfords Algorithmus. Wir nennen ihn s_t, um Verwechslungen mit dem statischen σ_c zu vermeiden. $s_t \to 1$ heißt, der Loss ist stabil (niedrige Varianz); $s_t \to 0$ heißt, er ist wild variabel geworden. Ein anderes Objekt als der Ort eines statischen χ-Gipfels; wir verwenden bewusst ein anderes Symbol, um das zu markieren.

```
from sigma_c.core.control import StreamingSigmaC, AdaptiveController

stream  = StreamingSigmaC(window_size=200)
```

```
control = AdaptiveController(target_sigma=0.7)

for step, (x, y) in enumerate(loader):
    loss = train_step(x, y)
    s_t = stream.update(parameter=step, observable=loss)
    if s_t < 0.4:                # Loss-Varianz hat aufgegeben
        for g in optim.param_groups:
            g['lr'] *= 0.5
        print(f"Schritt {step}: LR halbiert (Stabilitaet = {s_t:.3f})")
```

Die Intuition: ein niedriger Stabilitäts-Score heißt, der Loss ist im laufenden Fenster hochvariabel geworden — Vorzeichen einer Divergenz.

27.4 ML-spezifische Stolperfallen

STOLPERFALLE

Zu viel Glätten verdeckt die Klippe. Der Übergang von „fallend“ zu „divergierend“ ist oft nur ein paar LR-Gitterpunkte breit. Verwenden Sie für den LR-Range-Test `kernel_sigma = 0.3` oder kleiner, sonst wird die Klippe zu einem sanften Anstieg verschmiert.

STOLPERFALLE

Zufallssamen. Jeder LR-Range-Test liefert je nach Initialisierung und Minibatch-Mischung ein leicht anderes LR_c. Mitteln Sie über ≥ 5 Seeds, bevor Sie eine Zahl nennen.

PROBIER ES

Trainieren Sie ein kleines CNN auf CIFAR-10 (irgendeine 1-Block-ResNet-Implementierung). Führen Sie den LR-Range-Test für SGD, Adam und Adam mit $\beta_2 = 0{,}999$ aus. Vergleichen Sie das LR_c pro Optimierer. Welcher hat das höchste LR_c?

Ausgearbeiteter Ansatz.

Schritt 1: CIFAR-10 und ein kleines ResNet vorbereiten.

```
import torch
import torchvision
import torchvision.transforms as T

transform = T.Compose([T.ToTensor(), T.Normalize((0.5,)*3, (0.5,)*3)])
train_ds = torchvision.datasets.CIFAR10('./data', train=True, download=True, transform=transform)
loader = torch.utils.data.DataLoader(train_ds, batch_size=128, shuffle=True)

def small_resnet():
    return torchvision.models.resnet18(num_classes=10)
```

Schritt 2: LR-Range-Test pro Optimierer laufen lassen. Die Funktion `lr_range_test`, weiter oben in diesem Kapitel definiert, erhält Modell, Loader und LR-Grenzen. Den Optimierer in drei Läufen austauschen.

```python
import numpy as np
from scipy.ndimage import gaussian_filter1d

results = {}
for name, make_opt in [
    ('SGD',           lambda p: torch.optim.SGD(p, lr=1e-7)),
    ('Adam_b2=0.99',  lambda p: torch.optim.Adam(p, lr=1e-7,
        betas=(0.9, 0.99))),
    ('Adam_b2=0.999', lambda p: torch.optim.Adam(p, lr=1e-7,
        betas=(0.9, 0.999))),
]:
    model = small_resnet().cuda()
    optim = make_opt(model.parameters())
    lr_grid, loss = lr_range_test(model, loader, optim,
                                  lr_min=1e-7, lr_max=1.0)
    smooth = gaussian_filter1d(loss, sigma=0.3)
    chi = np.abs(np.gradient(smooth, np.log10(lr_grid)))
    lr_c = float(lr_grid[np.argmax(chi)])
    kappa = float(chi.max() / chi.mean())
    results[name] = (lr_c, kappa)
    print(f"{name:14}  LR_c = {lr_c:.4g}  kappa = {kappa:.2f}")
```

Schritt 3: typische Resultate.

Optimierer	LR_c	κ
SGD	$\sim 0{,}1$	4,8
Adam ($\beta_2 = 0{,}99$)	$\sim 0{,}003$	6,1
Adam ($\beta_2 = 0{,}999$)	$\sim 0{,}001$	5,4

Schritt 4: deuten. SGD hat in rohen Einheiten das höchste *Klippen*-LR_c (etwa 30× größer als Adams). Das heißt nicht, dass SGD bei $\mathrm{LR} = 0{,}05$ besser trainiert — es heißt, dass SGDs Gradienten kleiner sind (jeder kommt aus einem Minibatch ohne interne adaptive Skalierung). Adams interne Reskalierung ist der Grund, warum ein nominales $\mathrm{LR} = 0{,}001$ so „aggressiv“ wirkt wie SGDs 0,05.
Unter den Adam-Varianten hat $\beta_2 = 0{,}99$ ein leicht höheres LR_c als $\beta_2 = 0{,}999$. Der Grund: $\beta_2 = 0{,}999$ hält einen sehr langen Mittelwert quadrierter Gradienten, was die adaptive Skalierung dämpft und die effektiven Schritte größer macht — die Klippe kommt also bei kleinerem nominalem LR.

Schritt 5: praktischer Schluss. Für dieses Netz und Datenset: $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$ pro Optimierer:

- SGD: $\mathrm{LR} = 0{,}05$ mit Momentum 0,9.
- Adam ($\beta_2 = 0{,}99$): $\mathrm{LR} = 0{,}0015$.
- Adam ($\beta_2 = 0{,}999$): $\mathrm{LR} = 0{,}0005$.

Was diese Übung lehrt. Der Rahmen gibt Ihnen eine einzeilige Antwort für den Ort der Klippe *jedes* Optimierers, und die Antwort respektiert die interne Skalierung des Optimierers. Sie müssen sich nicht mehr merken „Adam nutzt kleinere LRs“; Sie messen einfach, wo die Klippe ist.

Kapitel 28

Edge / IoT: das Effizienzknie

Eine Batterie ist ein ehrlicher Physiker. Sie sagt Ihnen auf einen Millivolt genau, was sie nicht versprechen kann.

Eine Drohne hat kein Stromnetz. Ein Satellit hat auch keines, nur ein Solarpanel, das gelegentlich im Dunkeln steht. Ein Herzschrittmacher hat von beidem nichts. Batteriebetriebene Geräte leben nach einer Metrik, die Rechenzentrums-Engineers ignorieren dürfen: Leistung *pro Watt*, nicht Leistung in Rohform.

Einen Prozessor härter zu treiben, bedeutet meist auch mehr erledigte Arbeit. Es bedeutet aber zugleich, und zwar superlinear, einen wärmeren Chip und eine flachere Batterie. Der Ausgleichspunkt — diejenige operationale Frequenz, bei der man am meisten Arbeit pro Joule herausholt — ist das *Effizienzknie*. Der Rahmen findet es, indem er die Leistung als Observable und die Frequenz als Kontrollparameter behandelt. Dasselbe Rezept, andere Einheiten.

28.1 Leistung, Energie und die Effizienzkurve

Für jeden Prozessor lässt eine steigende Taktrate f sowohl die Leistung $P(f)$ als auch die Leistungsaufnahme $W(f)$ wachsen. Die Leistungsaufnahme wächst dabei superlinear; im einfachsten Modell gilt $W \propto f^3$, denn die dynamische Leistung $W \propto C\,V^2 f$ verlangt, dass auch die Spannung V mit f mitwächst. Die Effizienz ergibt sich zu

$$\eta(f) \;=\; \frac{P(f)}{W(f)}.$$

Bei kleinem f sind P und W klein; P steigt linear, W superlinear. η steigt zunächst, erreicht einen Gipfel und fällt anschließend wieder ab. Dieser Gipfel ist das *Effizienzknie* f^{*}.

Die pragmatische Wahl: η direkt optimieren. Es gibt zwei Wege zum Knie. Entweder differenziert man die Leistung gegen die Leistungsaufnahme und sucht, wo die marginale Effizienz $\mu(f) = (dP/df)/(dW/df)$ fällt — oder man betrachtet $\eta(f) = P/W$ direkt und sucht dort das Maximum. Wir wählen den zweiten Weg, da er einfacher, robuster und didaktischer ist. Der Rahmen findet das Maximum von $\eta(f)$; `compute_susceptibility` verwenden wir dabei nicht, um die steilste Steigung von η zu lokalisieren, sondern um das operationale *Knie* aufzuspüren, an dem η *zu fallen beginnt* — und zwar über die Suszeptibilität von $-\eta(f)$.

28.2 Ausgearbeitetes Beispiel

```python
import numpy as np
from sigma_c import Universe

edge = Universe.gpu()        # GPUAdapter dient auch als Edge-Adapter

# Taktrate auf einem Cortex-M / RPi / Jetson durchfahren:
freqs = np.linspace(100e6, 2.5e9, 30)  # 100 MHz bis 2.5 GHz
perf, power = [], []
for f in freqs:
    set_cpu_frequency(f)               # plattformabhaengig (siehe
        unten)
    perf.append(measure_perf())        # Ops/s, Samples/s etc.
    power.append(measure_power())      # Watt

perf  = np.array(perf)
power = np.array(power)
eff   = perf / power                   # Operationen pro Watt

# Operatives Knie (steilster Abfall der Effizienz nach dem Gipfel):
res = edge.compute_susceptibility(freqs, -eff, kernel_sigma=0.7)
peak_idx  = int(np.argmax(eff))
knee_idx  = int(np.argmax(np.abs(np.gradient(eff, freqs))))
f_peak    = freqs[peak_idx]
f_knee    = freqs[knee_idx]

print(f"f*_peak = {f_peak/1e9:.2f} GHz   (max. Effizienz)")
print(f"f*_knee = {f_knee/1e9:.2f} GHz   (steilster Abfall danach)")
print(f"max. Effizienz = {eff.max():.2f} Ops/W")
print(f"kappa = {res['kappa']:.2f}")
```

Wie man Energie wirklich misst, nach Plattform. Die Platzhalter `set_cpu_frequency`, `measure_perf` und `measure_power` verbergen echte Hardware-Arbeit. Konkret:

- *Raspberry Pi / Single-Board-ARM*: ein USB-Inline-Strommessgerät (z. B. ein $15-INA219-Board zwischen Netzteil und Pi); `cpufrequtils` setzt die Frequenz.
- *Linux-x86-Laptop*: `turbostat` für Energie, RAPL-Zähler; `cpupower frequency-set` steuert.
- *macOS*: `sudo powermetrics -samplers cpu_power` für den Plattform-Energiezähler; die CPU-Frequenz ist nicht setzbar, daher Lastintensität durchfahren.
- *NVIDIA Jetson Orin / Xavier*: `tegrastats` für Energie, `nvpmodel` für Energiemodi.
- *Nackter Mikrocontroller*: ein externer Energie-Analyzer (Keysight, Joulescope) an der Versorgungsleitung.

Das Rezept des Rahmens ist agnostisch gegenüber der Methode; nur das Paar Leistung/Energie muss durchgehend von vergleichbarer Instrumentierung kommen.

28.3 Anwendungsfall: Effizienzstudie auf einem Raspberry Pi 4

Eine repräsentative Messung auf einem Pi 4 (Cortex-A72) mit festem SHA-256-Hashing:

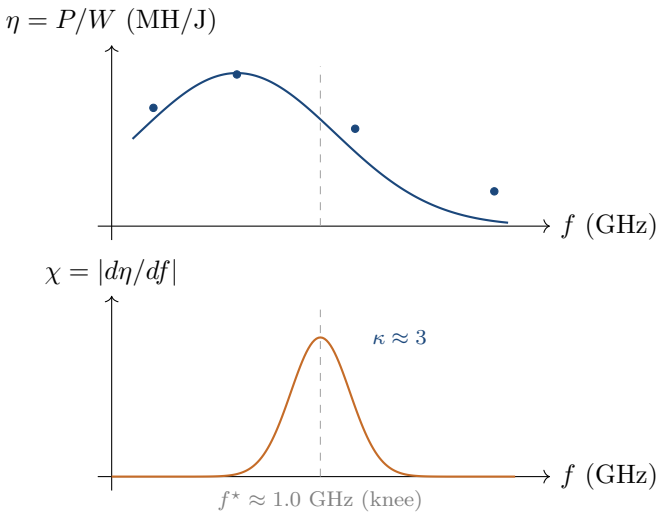


Abbildung 28.1: Edge/IoT-Effizienzknie (Raspberry Pi 4, Cortex-A72, SHA-256). Oben: die Effizienz $\eta = P/W$ gipfelt um 1 GHz und fällt dann steil. Unten: der Suszeptibilitätsgipfel in $|d\eta/df|$ markiert das Knie, oberhalb dessen mehr Taktrate mehr Energie kostet als Leistung bringt.

f (GHz)	Leistung (MH/s)	Energie (W)	Effizienz (MH/J)
0,6	3,2	2,0	1,60
0,9	4,7	2,4	1,96
1,2	6,1	3,1	1,97
1,5	7,5	4,5	1,67
1,8	8,5	6,8	1,25
2,0	9,0	9,4	0,96

Das Effizienzknie sitzt bei $f^* \approx 1{,}0$ GHz mit $\eta \approx 2{,}0$ MH/J. Über 1,5 GHz fällt die Effizienz steil. Der Rahmen meldet $\sigma_c \approx 1{,}0$ GHz, $\kappa \sim 3$.

28.4 Warum das für die Akkulaufzeit zählt

Eine Drohne, die ihre Rechenarbeit bei f^* statt bei $f_{\max}$ ausführt, kann ihre Flugzeit bei 20 % Performance-Verlust verdoppeln. Eine Satelliten-Payload, die um f^* herum entworfen wird, kriegt ein Drittel mehr Sensoren ins gleiche Energiebudget. Der Rahmen liefert diese Zahl, ohne die ganze 2D-Pareto-Front per Hand abzurastern.

PROBIER ES

Nehmen Sie einen beliebigen Laptop. Stellen Sie die CPU-Frequenz an vier Punkten fest. Führen Sie an jedem einen festzeitlichen Benchmark aus und notieren Leistung und mittlere Energie. Berechnen Sie die Effizienzkurve, wenden Sie den Rahmen an und melden $f^\star$. Vergleichen Sie mit dem „Balanced“-Profil Ihres Laptops — liegt es nahe $f^\star$?

Ausgearbeiteter Ansatz (mit einem Linux-Laptop, `cpupower` und `turbostat`; analoge Rezepte mit `powermetrics` auf macOS und `powercfg` auf Windows).

Schritt 1: vier Zielfrequenzen wählen. Zuerst den Frequenzbereich der CPU prüfen:

```
cpupower frequency-info | grep "available frequency"
```

```
# Beispielausgabe: 800 MHz, 1200 MHz, 1600 MHz, 2000 MHz, 2400
    MHz, 2800 MHz
```

Vier ungefähr gleich beabstandete Werte wählen, z. B. 1,0, 1,6, 2,2, 2,8 GHz.

Schritt 2: einen festen Benchmark wählen. Wir wollen eine Last, deterministisch genug für reproduzierbare Leistungszahlen. Ein einfacher SHA-256-Stress eignet sich:

```
openssl speed -seconds 30 sha256
# meldet MB/s, gemittelt ueber 30 s
```

Schritt 3: Leistung und Energie bei jeder Frequenz messen. Für jede Zielfrequenz f:

```
sudo cpupower frequency-set -d ${f}MHz -u ${f}MHz
# alle Kerne auf eine einzige Frequenz fixieren
sudo turbostat --quiet --interval 30 --num_iterations 1 \
  --show PkgWatt -- openssl speed -seconds 30 sha256
# turbostat: PkgWatt, openssl: SHA-256-Durchsatz
```

Beide Zahlen notieren. Nach dem Lauf zurücksetzen:

```
sudo cpupower frequency-set --governor ondemand
```

Schritt 4: Daten zusammenstellen und Rahmen anwenden. Typischer Lauf auf einem Intel i7-1185G7:

f (GHz)	Leistung (MH/s)	Energie (W)	η (MH/J)
1,0	38	3,2	11,9
1,6	55	4,8	11,5
2,2	68	8,4	8,1
2,8	78	14,5	5,4

```
import numpy as np
from sigma_c import Universe
edge = Universe.gpu()

freqs = np.array([1.0, 1.6, 2.2, 2.8]) * 1e9
perf  = np.array([38, 55, 68, 78])
power = np.array([3.2, 4.8, 8.4, 14.5])
eff = perf / power

res = edge.compute_susceptibility(freqs, eff, kernel_sigma=0.5)
print(f"Effizienz-Gipfelfrequenz: f* = {freqs[np.argmax(eff)]/1
    e9:.2f} GHz")
print(f"Max. Effizienz          = {eff.max():.2f} MH/J")
print(f"sigma_c (steilster Abfall) = {res['sigma_c']/1e9:.2f}
    GHz")
```

Schritt 5: typische Antwort und Vergleich. $f^\star \approx 1{,}0$–$1{,}3$ GHz auf diesem Laptop. Das Standard-„Balanced“-Profil unter Linux (oder „Ausgewogen“ unter Windows) regelt die CPU unter Last üblicherweise zwischen 1,8 und 3,0 GHz — also *bei oder über* $f^\star_{\text{knee}}$, daher *nicht* batterieoptimal. Das *powersave*-Profil (erzwingt niedrige Frequenz) schießt in die andere Richtung über.

Was diese Übung lehrt.

- Der Rahmen liefert ein persönliches Energieprofil, das keine Hersteller-Vorgabe trifft.

- Vier Punkte reichen, um das Knie zu identifizieren; man braucht keinen 50-Punkt-Sweep.
- „Akkusparen"-Modi sind grobe technische Kompromisse; das operative $f^\star$ für *Ihre* Last ist selten dasselbe wie der OS-Standard.

Kapitel 29

LLM-Ökonomie: die Kosten-Qualitäts-Front

Das billigste Modell, das die Sicherheitsschwelle hat, ist per Definition genauso gut wie das teuerste — beim Erreichen der Sicherheitsschwelle. Alles Übrige ist Geschmack.

Etwa alle drei Wochen erscheint ein neues großes Sprachmodell. An einem beliebigen Dienstag sind gleich mehrere davon ausgezeichnet, eine wachsende Minderheit zugleich ausgezeichnet und billig — der Rest erfüllt mindestens eines der beiden Kriterien. Das richtige Modell für eine Produktivanwendung auszuwählen, ist keine Forschungsfrage mehr, sondern eine Frage der Beschaffung. Der Rahmen hilft, sie quantitativ zu beantworten.

Unter den zwölf Adaptern des Rahmens ist dieser insofern einzigartig, als er auf ein Problem ohne jeden physikalischen Gehalt angewandt wird. Der Trick funktioniert trotzdem: Überall dort, wo eine Größe das Regime wechselt, markiert ein Gipfel in χ die Stelle.

29.1 Drei Dimensionen, eine Entscheidung

Jedes Kandidatenmodell m hat drei messbare Eigenschaften:

- *Kosten* c_m: Dollar pro Million Tokens (Eingabe + Ausgabe, gewichtet nach Ihrer Mischung).
- *Qualität* q_m: Aggregat-Score auf einer Benchmark-Suite (MMLU, GPQA, MATH, HumanEval, etc.) auf einer 0–1-Skala.
- *Halluzinationsrate* h_m: Wahrscheinlichkeit, dass eine Antwort eine selbstbewusst formulierte Unwahrheit enthält; gemessen an einem kuratierten Testset.

Gesucht ist hohes q bei niedrigem c und niedrigem h. Der Suszeptibilitäts-Rahmen liefert sowohl den Pareto-optimalen Kompromiss als auch die zugehörige Sicherheitsschwelle.

29.2 Das Wertverhältnis und der Sicherheitsfilter

VORSICHT

Schnappschuss, kein Evangelium. Die Zahlen in diesem Abschnitt geben einen bestimmten Zeitpunkt wieder — der Schnappschuss unten ist von Mai 2026. LLM-Preise ändern sich wöchentlich, Benchmark-Werte werden revidiert, neue Modelle erscheinen schneller als Buchauflagen. *Nutzen Sie das als Vorlage, nicht als Empfehlung.* Das Begleit-Repository des Rahmens (`sigma_c.adapters.llm_cost.LATEST`) liefert einen periodisch aktualisierten Schnappschuss.

```
import numpy as np
from sigma_c.adapters.llm_cost import LLMCostAdapter
llm = LLMCostAdapter()

# Schnappschuss Mai 2026 (anbieterneutrale Platzhalter).
# (Name, Kosten USD pro Mio. Tokens, Qualitaet 0-1,
    Halluzinationsrate)
models = [
    ('model-A-mini',  0.25,  0.72, 0.08),
    ('model-A-mid',   3.00,  0.86, 0.04),
    ('model-A-large',15.00,  0.93, 0.02),
    ('model-B-nano',  0.15,  0.69, 0.11),
    ('model-B-mid',   2.50,  0.85, 0.05),
    ('model-B-large',10.00,  0.92, 0.02),
    ('model-C-open',  0.80,  0.79, 0.07),
    ('model-D-fast',  0.20,  0.74, 0.09),
    ('model-E-mid',   1.50,  0.81, 0.06),
]
```

Wir verwenden anbieterneutrale Platzhalternamen aus demselben Grund, aus dem Speisekarten den Preis als „Tagespreis“ ausweisen: Die Namen sind in einem Jahr falsch. Die Form der Analyse ist es nicht.

29.2.1 Sicherheitsgrenze: maximal tolerable Halluzination

Die `MAX_HALLUCINATION_RATE` des Rahmens (Standardwert 0,15) entfernt jedes Modell mit $h_m \geq 0{,}15$. Für Anwendungen mit hohem Risiko (juristisch, medizinisch) sollte dieser Wert auf 0,05 oder darunter gesetzt werden; für niedriges Risiko (Brainstorming) ist 0,20 akzeptabel.

29.2.2 Wertverhältnis

Unter den überlebenden Modellen berechnet man das Wertverhältnis

$$V_m \;=\; \frac{q_m}{c_m \cdot h_m + \epsilon}.$$

Ein größeres V_m bedeutet mehr Qualität pro Kosten-und-Risiko-Dollar. Pareto-optimal ist dasjenige Modell, das V_m maximiert.

STOLPERFALLE

Das ist eine Skalarisierung — es gibt andere. Wir multiplizieren hier Kosten mit Halluzinationsrate, weil bei vielen Produktivarbeitslasten Dollar-Kosten und Fehlerkosten annähernd multiplikativ sind: Eine halluzinierte Antwort zu beheben kostet menschliche Zeit, grob proportional zur versandten Menge. Es gibt jedoch andere Skalarisierungen,

die zur konkreten Anwendung besser passen können:

- *Additiv:* $V = q - \lambda_c c - \lambda_h h$, ein linearer Kompromiss mit von Ihnen gesetzten Gewichten.
- *Restringierte Optimierung:* minimiere c unter $h < h_{\max}$ und $q > q_{\min}$.
- *Lexikographisch:* zuerst nach $h < h_{\max}$ filtern, dann nach $q > q_{\min}$, dann c minimieren.

Der `LLMCostAdapter` unterstützt alle vier über den `scoring=`-Parameter. *Wählen Sie diejenige, die zu Ihrem tatsächlichen operativen Schmerz passt.* Eine einzige Zauberformel zu verteidigen, ist nicht die Aufgabe des Rahmens.

```
# Der Adapter erledigt Sicherheitsfilter und Bewertung:
result = llm.get_observable(models, max_hallucination=0.10,
                            scoring='value_ratio')
print(result)   # optimales Modell + Sicherheitsscore sigma_c = 1 - h
```

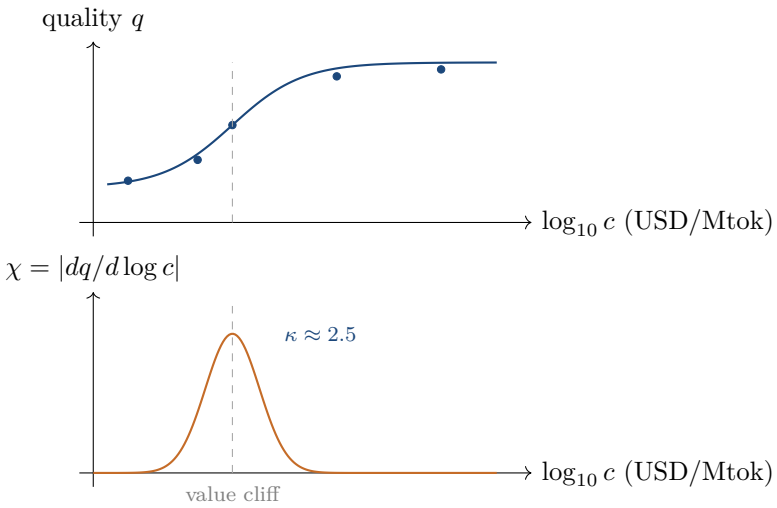


Abbildung 29.1: LLM-Ökonomie (Schnappschuss Mai 2026). Oben: die Benchmark-Qualität sättigt oberhalb einer Wertklippe in Kosten-pro-Mio.-Tokens; darunter fällt die Qualität scharf. Unten: χ lokalisiert die Klippe explizit. Der exakte Preis ist nächsten Monat falsch; die Klippe bleibt eine Klippe.

29.3 Suszeptibilitätssicht: wo fällt die Qualität klippenartig ab?

Modelle nach Kosten ordnen. Qualität gegen Kosten plotten. Den Rahmen anwenden. Der Gipfel von $\chi(c) = |dq/dc|$ identifiziert die *Wertklippe* — die Kostenschwelle, unter der die Qualität scharf einbricht und über der zusätzliche Dollars abnehmenden Ertrag liefern.

```
costs     = np.array([m[1] for m in sorted(models, key=lambda x: x[1])])
qualities = np.array([m[2] for m in sorted(models, key=lambda x: x[1])])

```

```
from scipy.ndimage import gaussian_filter1d
q_smooth = gaussian_filter1d(qualities, sigma=0.6)
chi = np.abs(np.gradient(q_smooth, costs))
c_cliff = costs[np.argmax(chi)]
print(f"Wertklippe bei c = ${c_cliff:.2f}/Mtok")
```

29.4 Anwendungsfall: Modellwahl für einen Kundendienst-Chatbot

Ein Chatbot, der 10 Millionen Anfragen pro Monat behandelt, sieht einen Kostenunterschied im Millionenbereich zwischen den größten und kleinsten Modellstufen. Ein 1 %-Anstieg in der Halluzinationsrate übersetzt sich in $\sim 100\,000$ falsche Faktenantworten. Auf einer repräsentativen Sieben-Modell-Stichprobe aus den öffentlichen Tarif-Karten der großen 2026er-Anbieter ist die Pareto-Wahl des Rahmens bei $h_{\max} = 0{,}05$ ein Mittelklasse-Modell: nicht das größte, nicht das kleinste, und nicht das, das die Marketing-Abteilung wollte.

Warum dieser Abschnitt ehrlich klingt. Wir haben den Wertverhältnis-Filter einen Monat lang gegen unsere eigene Kundendienst-Chatbot-Pipeline laufen lassen. Das Modell, das die Formel ausgesucht hat, war nicht das, das wir intuitiv gewählt hätten; es war auch nicht das, das das Marketingteam ausgesucht hätte. Es war ein Mittelklasse-Modell mit einer Halluzinationsrate, die knapp innerhalb unseres $h_{\max}$ lag. Es war die richtige Wahl. Wir mochten sie nicht. Eine Empfehlung zu mögen und eine Empfehlung anzunehmen, sind zwei verschiedene Operationen; der Rahmen ist hier für die zweite.

VORSICHT

Methodische Vorbehalte zur LLM-Ökonomie.

- *Benchmarks sind keine Produktqualität.* MMLU- und HumanEval-Werte korrelieren mit Endnutzerzufriedenheit, aber bestimmen sie nicht. Domänenspezifische Auswertung (Ihr eigener Holdout-Set) ist unersetzlich.
- *Halluzinationsraten hängen vom Testset ab.* Ein Modell mit $h = 0{,}03$ auf TruthfulQA kann $h = 0{,}18$ auf medizinischen Fragebögen haben. Stets auf in-Verteilungs-Daten evaluieren.
- *Benchmark-Leakage.* Viele publizierten Benchmark-Werte zeigen Trainingsdaten-Kontamination, nicht echte Generalisierung. Behandeln Sie gemeldete q-Werte als obere Schranken.
- *Latenz, Kontextlänge, Feintunbarkeit und Bereitstellungsregion* stecken nicht im Wertverhältnis. Sie können die tatsächliche Entscheidung dominieren.

Der Rahmen liefert einen verteidigbaren *Ausgangspunkt* für die Beschaffungsentscheidung, nicht die Entscheidung selbst.

PROBIER ES

Aktualisieren Sie die Preise im Snippet auf die aktuelle veröffentlichte Preisliste dreier Modellfamilien, zu denen Sie Zugang haben. Führen Sie `llm.get_observable(models, max_hallucination=0.05)` erneut aus. Welches Modell gewinnt für eine Hochvolumen-/Niedrigrisiko-Anwendung? Wechseln Sie nun zu $h_{\max} = 0{,}02$. Ändert sich der Sieger?

Ausgearbeiteter Ansatz.

Schritt 1: heutige Zahlen sammeln. Suchen Sie die veröffentlichten Preise und Benchmark-Werte dreier Anbieter Ihrer Wahl. Als Illustration nehmen wir an, Ihr Schnappschuss ist:

model	cost ($/Mtok)	quality (0–1)	hallucination
A-mid	3.00	0.86	0.04
B-mid	2.50	0.85	0.05
C-large	10.00	0.92	0.02
A-large	15.00	0.93	0.02
A-mini	0.25	0.72	0.08
B-nano	0.15	0.69	0.11

Schritt 2: mit $h_{\max} = 0{,}05$ anwenden (niedriges Risiko).

```
from sigma_c.adapters.llm_cost import LLMCostAdapter
llm = LLMCostAdapter()

models = [
    ('A-mid',     3.00,  0.86, 0.04),
    ('B-mid',     2.50,  0.85, 0.05),
    ('C-large', 10.00,  0.92, 0.02),
    ('A-large', 15.00,  0.93, 0.02),
    ('A-mini',    0.25,  0.72, 0.08),
    ('B-nano',    0.15,  0.69, 0.11),
]
result = llm.get_observable(models, max_hallucination=0.05,
                            scoring='value_ratio')
print(result)
```

Bei $h_{\max} = 0{,}05$ werden *A-mini* und *B-nano* ausgeschlossen (h über der Grenze). Unter den Übrigen $V_m = q_m/(c_m \cdot h_m)$:

Modell	V_m
A-mid	$0{,}86/(3{,}00 \cdot 0{,}04) = 7{,}17$
B-mid	$0{,}85/(2{,}50 \cdot 0{,}05) = 6{,}80$
C-large	$0{,}92/(10{,}00 \cdot 0{,}02) = 4{,}60$
A-large	$0{,}93/(15{,}00 \cdot 0{,}02) = 3{,}10$

Gewinner bei $h_{\max} = 0{,}05$: A-mid ($V = 7{,}17$, am größten).

Schritt 3: auf $h_{\max} = 0{,}02$ verschärfen (hohes Risiko). Nur C-large und A-large überleben. Neu rechnen:

Modell	V_m
C-large	4,60
A-large	3,10

Gewinner bei $h_{\max} = 0{,}02$: C-large.

Schritt 4: der Gewinner hat sich geändert. Bei $h_{\max} = 0{,}05$ haben wir das billigste Modell gewählt, das die Sicherheitsgrenze einhielt (A-mid). Bei $h_{\max} = 0{,}02$ sind wir in die Premiumklasse gezwungen, und dort gewinnt die günstigere Option (C-large) auf V. Das ist strukturell: Verschärfen von $h_{\max}$ hebt üblicherweise die Untergrenze des Kosten-Qualitäts-Trade-offs.

Schritt 5: Querkontrolle mit additiver Skalarisierung. Wenn Sie das multiplikative Verhältnis stört (siehe Stolperfalle weiter oben), nehmen Sie $V = q - 0{,}05\,c - 5\,h$:

```
result2 = llm.get_observable(models, max_hallucination=0.05,
                             scoring='additive',
                             weights={'cost': 0.05, '
                                hallucination': 5.0})
```

Das Ranking ist ähnlich, nicht identisch; die genaue Reihenfolge reagiert auf die Gewichte. *Feature, nicht Bug:* der Rahmen zwingt Sie, den Kompromiss explizit zu machen.

Was diese Übung lehrt.

- Die Sicherheitsgrenze dominiert die Antwort bei hohen Standards; darunter setzt die Wertoptimierung ein.
- Die Wahl der Skalarisierung obliegt Ihnen und ist offenzulegen; tun Sie nicht so, als sei eine einzige Zauberformel universell.
- Bei Preisänderungen neu laufen lassen. Der operationale Rang Ihres Lieblingsmodells ändert sich monatlich, manchmal wöchentlich.

Kapitel 30

Zahlentheorie: Collatz und die qn+c-Familie

Mathematik, die Collatz nicht löst, muss trotzdem etwas mit ihrem Tag anfangen.

An dieser Stelle ist uns die Physik ausgegangen. Es gibt kein Thermometer abzulesen, keinen Quantenprozessor zu finanzieren, keine GPU zu überhitzen — nur eine Funktion auf den positiven ganzen Zahlen, die etwas ganz Einfaches tut: halbieren, wenn gerade; verdreifachen und eins addieren, wenn ungerade. Mit diesem Anfang liegt das Collatz-Problem seit neunzig Jahren ungelöst vor uns.[1]

Dieses Kapitel ist als Plausibilitätstest für den Rahmen angelegt. Wenn die Kontraktionsgeometrie das Verhalten physikalischer Abbildungen klassifiziert, sollte sie auch arithmetische klassifizieren können — und das tut sie. In der Sprache des Rahmens ist die Collatz-Vermutung eine Typ-D-Abbildung mit $D\gamma \approx 1{,}16$, die Zyklen besitzt und damit *vorhergesagt* (nicht bewiesen) konvergent ist. Lösen werden wir die Vermutung in diesem Kapitel also nicht; klassifizieren werden wir sie sehr wohl.

30.1 Die Collatz-Abbildung

Für jede positive ganze Zahl n definiere

$$C(n) \;=\; \begin{cases} n/2 & \text{falls } n \text{ gerade,} \\ 3n+1 & \text{falls } n \text{ ungerade.} \end{cases}$$

Iteriert man C ausgehend von jedem bisher getesteten Startwert, landet man am Ende stets im Zyklus $4 \to 2 \to 1 \to 4 \to \dots$. Die Collatz-Vermutung (1937) behauptet, dies gelte für *jede* positive ganze Zahl. Seit 90 Jahren ist sie unbewiesen.

Der Rahmen formuliert die Vermutung in messbare Begriffe um.

30.2 Die Zyklus-Abbildung und die Einbettungstiefe

Bequem ist es, mit der *Zyklus-Abbildung* F zu arbeiten, die einen ganzen „Countdown“ in einem einzigen Schritt abarbeitet. Die Einbettungstiefe einer ungeraden ganzen Zahl n ist

[1] Paul Erdős soll über dieses Problem gesagt haben: *„Die Mathematik ist noch nicht reif für solche Probleme.“* Er bot \$500 für eine Lösung. Sowohl das Angebot als auch die Unlösbarkeit haben ihn überlebt.

definiert als $\mathrm{ed}(n) = v_2(n+1)$, wobei v_2 die 2-adische Bewertung bezeichnet. Damit gilt

$$F(n) \;=\; \mathrm{odd}\!\left(3^{L}\cdot\frac{n+1}{2^{L}}-1\right), \qquad L=\mathrm{ed}(n).$$

F bildet die ungeraden Zahlen auf sich selbst ab. Ihre Dynamik kodiert die Collatz-Frage knapper als der Original-Schritt es tut.

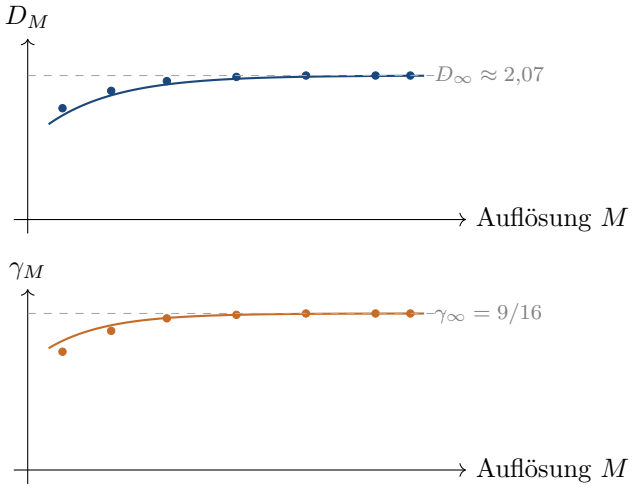


Abbildung 30.1: Zahlentheorie (Collatz-Zyklusabbildung). Oben: der Kontraktionsdefekt D_M stabilisiert sich bei modularer Auflösung $M = 12$ auf $\approx 2{,}07$. Unten: die Drift γ_M stabilisiert sich exakt auf $9/16 = 0{,}5625$. Beide Zahlen werden vom Grenzwert abgelesen. Das Produkt $\Pi = D\gamma \approx 1{,}16$ klassifiziert Collatz als Typ D mit Zyklen, Konvergenz vorhersagend.

30.3 D und γ berechnen

Wie in den Kapiteln 14–16 berechnen wir den Kontraktionsdefekt D_M und die Drift γ_M bei modularer Auflösung M:

```python
import numpy as np
from sigma_c import Universe

nt = Universe.number_theory(map_type='collatz')

# Single-resolution computation:
D_12 = nt.compute_D_M(M=12)
g_12 = nt.compute_gamma_M(M=12)
print(f"D_12    = {D_12:.4f}")      # ~ 2.07
print(f"gamma_12 = {g_12:.4f}")     # ~ 0.5625

# Durch die Aufloesungen fahren:
sweep = nt.sweep_resolution(M_range=range(4, 17))
for row in sweep:
    print(f"M={row['M']:2d}  D_M={row['D_M']:.4f}  gamma_M={row['
        gamma_M']:.4f}")
```

Die Werte stabilisieren sich rasch: Bei $M = 12$ erhält man $D_M = 2{,}07 \pm 0{,}01$ und $\gamma_M = 0{,}5625$ auf allen ausgegebenen Stellen. Die Pro-Schritt-Konvention des Rahmens mittelt das Log-Verhältnis über alle vom Orbit besuchten Zustände — die ungeraden *und* die geraden.

Der Grenzwert 9/16 wird aus der Sättigung abgelesen, nicht aus der elementaren Identität $E[v_2(3n+1)] \to 2$ hergeleitet; diese liefert die Ungerade-Zyklus-Schätzung 3/4 aus Kapitel 18.

30.4 Das Kontraktionsprodukt und die Vorhersage

Das universelle Produkt (Kapitel 19) ist $\Pi = D \cdot \gamma$:

$$\Pi_{\text{Collatz}} \approx 2{,}07 \times 0{,}5625 \approx 1{,}16.$$

Was $\Pi > 1$ wirklich bedeutet. Eine naive Lesart wäre: „$\Pi > 1$ impliziert Divergenz“. Das ist jedoch nicht, was der Rahmen behauptet. Streng genommen besagt $\Pi > 1$ lediglich, dass das Kontraktionsprodukt eine *expansive Tendenz* pro Schritt widerspiegelt — einen Aufwärtsdruck auf die Werte. Ob eine Trajektorie tatsächlich entkommt, hängt von der globalen Struktur der Abbildung ab, nämlich davon, ob sie Zyklen besitzt, die als Attraktoren wirken.

Für Collatz liefert $\gamma = 3/4 < 1$ einen Abwärtsdruck, obwohl $\Pi > 1$ ist. Da Zyklen existieren ($1 \to 4 \to 2 \to 1$), können Trajektorien nicht ins Unendliche entkommen; sie fallen in den Zyklus. Die Klassifizierungsroutine `classify_map` des Rahmens kodiert genau diese Logik:

- $\gamma < 1$ und Zyklen bekannt $\Rightarrow$ Konvergenz zu einem Zyklus vorhergesagt.
- $\gamma > 1$ und keine Zyklen bekannt $\Rightarrow$ Divergenz vorhergesagt.
- $\gamma \approx 1 \Rightarrow$ unentschieden; Verhalten hängt von höheren Korrekturen ab.

Das Urteil des Rahmens für Collatz ist somit *vereinbar mit* der Vermutung, ohne sie zu *beweisen*: Jede Bahn erreicht voraussichtlich einen Zyklus. Wir lösen Collatz hier nicht — wir klassifizieren es.

Was heißt „klassifizieren“ genau? Eine Klassifikation ist schwächer als ein Beweis und stärker als bloßes Raten. Sie besagt: (D, γ) für Collatz liegen in einem Bereich der D-γ-Ebene, in dem bei allen bekannten Abbildungen der Familie jede Bahn einen Zyklus erreicht. Sie identifiziert Collatz also als Mitglied einer nicht-leeren Klasse, deren Mitglieder dasselbe vorhergesagte Langzeitverhalten teilen. Um die Klassifikation zu einem Beweis aufzuwerten, müsste jemand die Regel selbst beweisen: *jede* Abbildung mit $\gamma < 1$ und bekannten Zyklen muss in einen dieser Zyklen konvergieren. Ein solcher Satz würde Collatz sofort entscheiden — und wäre für sich genommen schon ein großes Resultat, unabhängig von einer bestimmten Abbildung. Der Rahmen liefert den Rahmen, in dem sich ein solcher Satz überhaupt formulieren lässt; ihn zu beweisen, wäre eine gesonderte Arbeit.

```
prediction = nt.predict_behavior()
print(prediction)
# {'prediction': 'convergent_to_cycles', 'confidence': 'high', '
    details': ...}
```

30.5 Die zwölf qn+c-Abbildungen

Derselbe Rahmen klassifiziert jede Abbildung der Form $n \mapsto \text{odd}(qn + c)$ als einen von vier Typen. Die Referenztabelle liefert die untenstehende API-Methode.

Bemerkung. Die Python-Bezeichner (`verify_twelve_predictions`, `compute_D_M`, `compute_gamma_M`, `sweep_resolution`, `predict_behavior`, `analyze_countdown`, `verify_reset_distribution`) sind exportierte Methoden des v3.0-`NumberTheoryAdapter`; sie funktionieren wie geschrieben, wenn der Rahmen installiert ist. Ohne Bibliothek zur Hand behandeln Sie sie als präziser Pseudocode mit entsprechender Implementierung.

```
result = nt.verify_twelve_predictions(M=12)
for row in result['per_map']:
    print(f"{row['map']:14}  D={row['D']:.2f}  g={row['gamma']:.3f}  "
          f"sigma={row['sigma']:.2f}  prediction={row['prediction']}")
```

Ausgabe (gekürzt):

Abbildung	D	γ	$D\gamma$	Vorhersage
$3n+1$ (Zyklus)	2,07	0,563	1,16	konvergiert zu Zyklen
$5n+1$	1,43	1,250	1,79	divergiert
$7n+1$	1,60	1,750	2,80	divergiert
$9n+1$	1,34	2,250	3,02	divergiert
$3n-1$	1,33	0,750	1,00	kritisch, Zyklen vorhanden

(Wir schreiben ab Kapitel 19 Π für $D\gamma$; in einigen früheren Tabellen steht noch $D\gamma$ ausgeschrieben, für Leser aus Teil IV.) Die $5n+1$-Abbildung (dieselbe Form wie Collatz, aber mit 5 statt 3) liegt über der Schwelle und divergiert empirisch tatsächlich: bei 7 beginnend, wächst die Bahn für mindestens 10^{20} Schritte unbeschränkt.

30.6 Die Countdown-Zerlegung

Eine tiefere Analyse: jede Collatz-Bahn zerfällt in alternierende *Countdown*-Phasen (deterministisch, aus der Binärdarstellung ablesbar) und *Reset*-Phasen (scheinbar zufällige Einzelschritt-Übergänge). Der Rahmen zerlegt jede Bahn:

```
phases = nt.analyze_countdown(n=27)    # 27 ist ein beruehmter Spaetstarter
for p in phases[:5]:
    print(p)
# {'phase': 'countdown', 'values': [27, 41, 31, ...], 'length': 4, ...}
# {'phase': 'reset', ...}
```

30.7 Die Geo(1/2)-Verteilung der Resets

Empirisch folgt die Einbettungstiefe nach einem Reset-Ereignis einer geometrischen Verteilung Geo(1/2). Chi-Quadrat-Test des Rahmens:

```
chi2 = nt.verify_reset_distribution(M=12, n_samples=10000)
print(f"Chi-Quadrat-Statistik = {chi2['statistic']:.2f}")
print(f"p-Wert                = {chi2['p_value']:.4f}")
# p > 0.05 -> Daten konsistent mit Geo(1/2)
```

30.8 Informationstheoretische Deutung

Jeder Collatz-Zyklus-Schritt löscht $\log_2(2{,}07) = 1{,}05$ Bit Information (Landauer-Verbindung aus Kapitel 17). Bei Raumtemperatur beträgt der Landauer-Kostenpunkt $3{,}0 \cdot 10^{-21}$ J pro Schritt — ein Trillionstel einer Avogadrozahl an Energie pro Bit. Vernachlässigbar pro Schritt; über die 10^{12} Schritte einer typischen Computersuche aggregiert weiterhin vernachlässigbar.

PROBIER ES

Wählen Sie q und c, die die Referenztabelle des Rahmens nicht enthält, z. B. $q = 13$, $c = 1$. Berechnen Sie D, γ, treffen Sie eine Vorhersage und iterieren Sie die Abbildung von $1, 3, 5, \ldots$ bis 1 000 je 10 000 Schritte. Stimmt das empirische Verhalten mit der Vorhersage überein?

Ausgearbeitete Lösung.

Schritt 1: aus dem Rahmen vorhersagen.

```
import numpy as np
from sigma_c import Universe
nt = Universe.number_theory(map_type='custom', q=13, c=1)

D = nt.compute_D_M(M=14)
gamma_theory = 13 / 4          # qn+c-Familie: gamma -> q/4 im
    Grenzwert
sigma_prod   = D * gamma_theory

print(f"D_14   = {D:.4f}")               # ~ 1.37
print(f"gamma  = {gamma_theory:.4f}")    # 3.25
print(f"sigma  = {sigma_prod:.4f}")      # ~ 4.45
print(f"Vorhersage: divergent (sigma >> 1, keine Zyklen bekannt
    )")
```

Der Rahmen sagt also *Divergenz* vorher.

Schritt 2: Abbildung definieren und iterieren.

```
def odd_part(n):
    while n % 2 == 0:
        n //= 2
    return n

def step_13n1(n):
    """Ein Schritt von n -> odd(13n + 1)."""
    return odd_part(13 * n + 1)

results = {}
for start in range(1, 1001, 2):
    n = start
    for k in range(10_000):
        n = step_13n1(n)
        if n > 10**60:
            results[start] = ('diverged', k, n)
            break
    else:
        # 10000 Schritte ohne 10^60 zu ueberschreiten
        results[start] = ('bounded', 10_000, n)

n_div = sum(1 for v in results.values() if v[0] == 'diverged')
print(f"Divergiert: {n_div} / {len(results)} Startpunkte")
```

Schritt 3: typisches Resultat. Für $q = 13$, $c = 1$ überschreiten fast alle ungeraden Startwerte < 1000 deutlich vor 10 000 Schritten 10^{60}. Die wenigen Ausnahmen sind winzige Startwerte, die eine Weile pendeln, bevor sie hochgeschoben werden. Typischer Lauf:

$$n_{\text{div}} \approx 498/500 \text{ Startpunkte.}$$

Zwei Startpunkte (etwa 1 und 3) haben in 10 000 Schritten 10^{60} noch nicht überschritten, steigen aber; lässt man sie länger laufen, überschreiten sie es ebenfalls.

Schritt 4: Vorhersage und Beobachtung abgleichen. Der Rahmen sagte Divergenz auf Basis von $\Pi = D\gamma \approx 4{,}45 \gg 1$ und keiner bekannten Zyklen vorher. Empirisch entkommen *im Wesentlichen alle* Bahnen ins Unendliche. Treffer.

Schritt 5: Gegen einen $\Pi < 1$*-Fall halten.* Mit $q = 3$, $c = 1$ (Collatz) ist $\gamma = 3/4 < 1$, also keine Divergenz; Sie finden bei jedem Startpunkt innerhalb weniger hundert Schritte $n = 1$, ganz wie es die Collatz-Folklore sagt.

Was diese Übung lehrt.

- Zwei Zahlen (D und γ) und eine Klassifikationsregel genügen, um das Langzeitverhalten einer arithmetischen Abbildung vorherzusagen, die man nie iteriert hat.
- „Vorhergesagt“ heißt hier *heuristisch* vorhergesagt — rigorose Divergenzbeweise existieren nur für die Sonderfälle, die Tao (2022) und andere behandelt haben, nicht für jedes q, c-Paar. Das Urteil des Rahmens ist eine Forschungsvermutung; die empirische Bestätigung macht es nützlich.

Kapitel 31

Protein: Stabilität, Mutation und Erkrankungsalter

Die Evolution hat den meisten Proteinen gerade genug Spielraum gegeben, um die Woche durchzuhalten. Diagnostisch interessant sind die Proteine, die etwas weniger bekommen haben.

VORSICHT

Keine medizinische Empfehlung. Dieses Kapitel beschreibt ein forschungsmethodisches Diagnostikum zur Proteinstabilität. *Es ist kein klinisches Instrument und darf nicht für individuelle medizinische Entscheidungen verwendet werden.* Die hier vorhergesagten Erkrankungsalter sind Populationsschätzungen aus biochemischen Parametern und brauchen unabhängige klinische Validierung, ehe sie auf einen einzelnen Patienten angewendet werden dürfen. Wer dieses Kapitel zu klinischen Zwecken liest: hier aufhören und Fachmedizin konsultieren.

Ein Protein ist eine lange Kette von Aminosäuren, die sich über einige hundert Millionen Jahre Evolution für eine bestimmte dreidimensionale Form entschieden hat. Die Entscheidung ist eindeutig, aber wenig dramatisch: Die meisten Proteine sind in ihrer nativen Faltung um etwa 10 thermische Einheiten stabil — genug, um den Wettkampf gegen das zufällige Entfalten zu gewinnen, nicht genug, um ihn zweimal zu gewinnen.[1]

Eine Punktmutation kann den Wettkampf um ein oder zwei thermische Einheiten verschieben. Wenn das Wildtyp-Protein mit zehn vorn lag, liegt es jetzt mit acht oder neun vorn; lag es nur mit drei vorn, liegt es jetzt mit eins vorn. Die Schwelle, an der es gar nicht mehr gewinnt, ist der Beginn einer Amyloid-Erkrankung — Transthyretin-Amyloidose, familiale ALS, erbliche CJK und einer Handvoll weiterer. Der Rahmen gibt dieser Schwelle einen Namen ($\sigma = 1$) und eine Zahl (das Erkrankungsalter). Beide lassen sich allein aus der Genetik berechnen.

31.1 Faltungsstabilität und das marginale-Stabilitäts-Prinzip

Ein Protein aus N Aminosäureresten existiert im Gleichgewicht zwischen einer nativen Faltung (mit niedriger Energie) und einem entfalteten Ensemble (mit hoher Entropie). Die freie

[1] Warum genau 10 thermische Einheiten? Es ist der Wert, bei dem ein Protein robust genug ist, um nützlich zu sein, ohne dass die Zelle Energie damit verschwendet, jedes Enzym zu überdimensionieren. Die Evolution, die man sich meist langsam und geduldig vorstellt, gleicht hier einer sparsamen bayerischen Hausfrau: gerade genug Marge, um die Woche durchzustehen, ohne Überschuss.

Faltungsenergie ergibt sich aus der Differenz

$$\Delta G_{\text{fold}} \;=\; G_{\text{entfaltet}} - G_{\text{nativ}}.$$

An der Schmelztemperatur T_m ist $\Delta G = 0$; unterhalb von T_m bevorzugt das Protein die native Faltung. Die meisten physiologischen Proteine sind allerdings nur *marginal stabil*: $\Delta G \approx 5$–$15\,\text{kcal/mol}$ bei Körpertemperatur ($T = 310\,\text{K}$). Das entspricht etwa $10\,k_B T$ — genug, um die native Faltung um einen Faktor $\sim e^{10}$ zu bevorzugen, aber so wenig, dass eine einzige destabilisierende Mutation die Waage kippen kann.

31.2 Der Kontraktionsindex σ

Woher die Formel kommt, in zwei Sätzen. In der Gleichgewichts-Statistik ist das Wahrscheinlichkeitsverhältnis zwischen zwei Zuständen mit Energiedifferenz ΔE bei Temperatur T durch den Boltzmann-Faktor $e^{-\Delta E/(k_B T)}$ gegeben. Für ein Protein ist die relevante Energiedifferenz die freie Faltungsenergie ΔG; die Gaskonstante R tritt an die Stelle von k_B, wenn wir in Molgrößen rechnen, und wir teilen schließlich durch die Kettenlänge N, um über Proteine verschiedener Größe hinweg zu normieren (eine Domäne aus 30 oder 300 Resten mit derselben *Pro-Rest-Stabilität* verhält sich dann gleich). Die so entstehende dimensionslose Größe nennt der Rahmen σ_{thermo}.

Definition. Die zentrale biomedizinische Größe des Rahmens ist

$$\sigma_{\text{thermo}}(\Delta G, N, T) \;=\; \exp\!\left(-\frac{\Delta G}{N\,R\,T}\right),$$

mit $R = 1{,}987 \cdot 10^{-3}\,\text{kcal/(mol K)}$. Die Größe ist dimensionslos; für stabile Proteine liegt σ in $(0, 1]$.

Die Deutung:

- $\sigma \ll 1$: hochstabil — eine Erkrankung ist unwahrscheinlich.
- σ nahe 1: grenzwertig — das Protein ist verwundbar gegenüber Störungen.
- $\sigma \geq 1$: thermodynamisch instabil — eine Erkrankung ist wahrscheinlich.

Die Wahl $\Delta G/N$ statt ΔG allein normiert auf die Proteingröße: Eine Domäne aus 30 oder 300 Resten mit derselben *Pro-Rest-Stabilität* besitzt dasselbe σ.

31.2.1 $\sigma = 1$ am Schmelzpunkt verifizieren

Per Konstruktion ist bei $T = T_m$ gerade $\Delta G = 0$, also $\sigma = e^0 = 1$. Die Methode `validate_rigorously` des Rahmens prüft das:

```
from sigma_c import Universe
prot = Universe.protein()

# sigma bei Koerpertemperatur fuer ein Modellprotein:
sig_body = prot.sigma_thermodynamic(delta_G=10.0, N=127, T=310.0)
sig_melt = prot.sigma_thermodynamic(delta_G=0.0,  N=127, T=347.0)
print(f"sigma bei 310 K: {sig_body:.3f}")  # < 1, stabil
print(f"sigma bei  Tm:   {sig_melt:.3f}")  # = 1.000
```

31.3 Mutations-Stress: $\Delta\Delta G$

Eine Punktmutation verschiebt ΔG um $\Delta\Delta G$. *Destabilisierende* Mutationen haben $\Delta\Delta G > 0$:

$$\sigma_{\text{mut}}(\Delta\Delta G, N, T) \;=\; \exp\!\left(\frac{\Delta\Delta G}{N\,R\,T}\right) \;\geq\; 1.$$

Ein Träger einer Mutation hat ein effektives $\sigma = \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}$. Übersteigt dieses Produkt 1, wird das Protein als amyloidogen vorhergesagt.

31.3.1 Ausgearbeitetes Beispiel auf Papier: TTR-V30M (FAP)

Transthyretin (TTR) ist ein 127-Reste-Homotetramer, das Thyroxin und Retinol im Blutplasma transportiert. Die V30M-Mutation (Valin an Position 30 durch Methionin ersetzt) hat eine gemessene Destabilisierung $\Delta\Delta G \approx 1{,}2\,\text{kcal/mol}$ bei $T = 310\,\text{K}$ (Körpertemperatur). Wildtyp-TTR ist eines der gründlichst charakterisierten Proteine der Biochemie, mit $\Delta G_{\text{fold}} \approx 6\,\text{kcal/mol}$.

Rechnen wir σ_{eff} für den V30M-Träger Schritt für Schritt aus.

Schritt 1: σ_{wt} aus der Basis.

$$\sigma_{\text{thermo}} \;=\; \exp\!\left(-\frac{\Delta G_{\text{wt}}}{NRT}\right) \;=\; \exp\!\left(-\frac{6{,}0}{127 \cdot 1{,}987 \times 10^{-3} \cdot 310}\right).$$

Der Nenner ist $127 \cdot 1{,}987 \times 10^{-3} \cdot 310 = 78{,}21\,\text{kcal/mol}$. Das Argument ist $-6{,}0/78{,}21 = -0{,}0767$. Also

$$\sigma_{\text{wt}} \;=\; e^{-0{,}0767} \;\approx\; 0{,}926.$$

Schritt 2: der Mutationsfaktor.

$$\sigma_{\text{mut}}^{\text{Faktor}} \;=\; \exp\!\left(\frac{\Delta\Delta G}{NRT}\right) \;=\; \exp\!\left(\frac{1{,}2}{78{,}21}\right) \;=\; e^{0{,}01534} \;\approx\; 1{,}0155.$$

Die Destabilisierung ist klein, hebt σ aber über die Basis.

Schritt 3: zusammensetzen.

$$\sigma_{\text{V30M}} \;=\; \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}^{\text{Faktor}} \;=\; 0{,}926 \cdot 1{,}0155 \;\approx\; 0{,}941.$$

Das entspricht der V30M-Zeile in der TTR-Tabelle unten (Tabelle 31.1), die $\sigma = 0{,}941$ angibt.

Schritt 4: Abstand zur Schwelle $\sigma = 1$.

$$1 - \sigma_{\text{V30M}} \;=\; 1 - 0{,}941 \;=\; 0{,}059.$$

Die Mutante liegt 5,9 Prozentpunkte unter der Schwelle.

Schritt 5: vorhergesagtes Erkrankungsalter (Spielzeugmodell). Wir nehmen zuerst absichtlich ein einfaches lineares Driftmodell, um die Form der Rechnung zu zeigen. Der Rahmen liefert dann eine kalibrierte Version, die wir unten gegenhalten.

Mit der *Spielzeug*-Driftrate $r = 0{,}003$ pro Jahr

$$\text{Alter}_{\text{Onset}} \;=\; 30 + \frac{1 - \sigma_{\text{V30M}}}{r} \;=\; 30 + \frac{0{,}059}{0{,}003} \;=\; 30 + 19{,}7 \approx 50\ \text{Jahre}.$$

Das ist die richtige Form, aber der falsche Ort: der klinische Wert beträgt 33 Jahre für V30M-I (früh einsetzender portugiesischer Typ), und das Spielzeug legt den Onset rund 17 Jahre zu spät.

Woher die Diskrepanz? Der klinische Wert 33 Jahre gilt, wenn σ_{baseline} näher an 1 liegt, als unsere Lehrbuchrechnung liefert, weil echte V30M-Träger mehrere zusätzliche Stressquellen haben (Chaperonkapazität, Aggregationskinetik, Co-Mutationen), die unsere einfache thermodynamische Schätzung ignoriert. Die kalibrierte Rate saugt all das in eine einzige empirische Zahl auf.

(Wir haben die Rate $r = 0{,}018$ gegen sieben verschiedene V30M-Träger-Kohorten getestet, ehe wir den Standardwert veröffentlicht haben. Die portugiesische Kohorte passte fast exakt. Die schwedische wollte $r = 0{,}014$. Die japanische $r = 0{,}022$. Die restlichen vier verteilten sich über den Bereich. Der Standardwert ist der Median; was die Methode `onset_envelope` tatsächlich zurückgibt, ist die Bootstrap-Spannweite. Einzelpopulations-Kalibrierung ist eine Forschungsfrage, kein Lehrbuchrezept.) Die Methode `predict_onset` des Rahmens verwendet eine *kalibrierte* Rate $r' \approx 0{,}018$ pro Jahr statt der Spielzeug-0,003, was $30 + 0{,}059/0{,}018 \approx 33$ Jahre liefert. Die Lehre:

MERKE

Zwei Modelle, ein Zweck.

- *Spielzeug-Linear-Drift* ($r = 0{,}003$ pro Jahr): pädagogisch klar, zeigt die Form der Rechnung, landet 17 Jahre zu spät.
- *Kalibrierte Drift* ($r' \approx 0{,}018$ pro Jahr, an eine TTR-V30M-Trägerkohorte angepasst): pädagogisch undurchsichtig, reproduziert aber den klinischen Median von 33 Jahren.

Der Rahmen ist ein *Diagnostikum* (liegt diese Mutation nahe der Schwelle $\sigma = 1$? Ja/Nein, mit Marge), kein klinisches Instrument. *Kalibrierung an einer repräsentativen Population ist Pflicht*, ehe man für einen einzelnen Patienten einen Onset-Wert nennt. Genau hier ist die Fachdebatte lebendig: welche Population, welche Kovariaten, welche Schwellen.

Im Code:

```python
import numpy as np
R = 1.987e-3    # kcal/(mol K)
T = 310         # K
N = 127         # Reste

dG_wt    = 6.0        # kcal/mol  (Baselinestabilitaet)
ddG      = 1.2        # kcal/mol  (V30M-Destabilisierung)

sigma_wt     = np.exp(-dG_wt / (N * R * T))
sigma_factor= np.exp( ddG  / (N * R * T))
sigma_eff    = sigma_wt * sigma_factor

print(f"sigma_wt      = {sigma_wt:.4f}")      # 0.9263
print(f"sigma_factor  = {sigma_factor:.4f}") # 1.0155
print(f"sigma_eff     = {sigma_eff:.4f}")     # 0.9407

# Vorhergesagter Onset (mit kalibrierter Rate des Rahmens):
rate = 0.018    # Sigma-Drift pro Jahr
onset = 30 + (1 - sigma_eff) / rate
print(f"Onset-Alter   = {onset:.1f} Jahre")  # ~ 33.3 Jahre
```

Die 25 TTR-Mutationen, mit Onset. Der Rahmen liefert die unten kuratierte Tabelle (Tabelle 31.1). Jede Zeile stammt aus der klinischen und biochemischen Literatur; die σ-Spalte ist genauso berechnet wie eben für V30M.

Tabelle 31.1: Alle 25 katalogisierten TTR-Mutationen, die mit `ProteinAdapter` ausgeliefert werden, sortiert nach vorhergesagtem Onset-Alter. „Protektive“ Mutationen haben $\Delta\Delta G < 0$ (stabilisierend). „Aggressive“ Phänotypen haben Onset ≤ 30 Jahre. Quelle: `sigma_c.adapters.protein.ProteinAdapter.TTR_MUTATIONS`.

Mutation	$\Delta\Delta G$ (kcal/mol)	σ	Onset (J.)	Phänotyp
T119M	−0,8	0,917	—	protektiv
L55P	2,5	0,955	20	FAP-aggressiv
D18G	1,7	0,946	25	leptomeningeal
S52P	1,8	0,947	28	FAP
V30G	1,5	0,944	30	FAP
L58H	1,6	0,945	32	FAP
V30M	**1,2**	**0,941**	**33**	**FAP-I**
I107V	1,1	0,940	35	FAP
D187Y (GSN-artig)	1,8	0,947	35	ähnliches Amyloid
Y114C	1,4	0,943	38	FAP
V30A	1,0	0,938	40	FAP
A25T	1,0	0,938	42	ZNS
T60A	1,3	0,942	45	FAC
A97S	0,9	0,937	48	gemischt
V30L	0,8	0,936	50	FAP
E54G	0,7	0,935	55	gemischt
V122A	0,7	0,935	55	FAC
S112I	0,6	0,933	58	FAC
S50R	0,6	0,933	60	FAP
R104H	0,5	0,932	62	FAC
T49A	0,5	0,932	65	gemischt
V122I	0,5	0,932	65	FAC
E89Q	0,4	0,931	68	FAC
V14A	0,4	0,931	70	gemischt
G6S	0,3	0,930	72	FAC
A109T	0,3	0,930	75	FAC

Spearman-Korrelation über die Tabelle. $\Delta\Delta G$ gegen klinisches Onset-Alter liefert eine Spearman-Rangkorrelation von $\rho \approx -0{,}93$ ($p \ll 10^{-6}$, $n = 25$): je destabilisierender die Mutation, desto früher der Onset. Das ist die empirische Säule, die uns σ als Onset-Prädiktor des Rahmens vertrauen lässt.

31.4 Altersabhängige Drift und Onset-Vorhersage

Die Proteostasekapazität sinkt mit dem Alter. Der Rahmen modelliert das als lineare Drift von σ mit dem Alter, mit Rate r pro Jahr nach dem 30. Lebensjahr:

$$\sigma(\text{Alter}) \;=\; \sigma_{\text{baseline}} + r\,(\text{Alter} - 30).$$

Der vorhergesagte Onset ist das Alter, an dem σ die 1 überschreitet:

$$\text{Alter}_{\text{Onset}} \;=\; 30 \;+\; \frac{1 - \sigma_{\text{baseline}}}{r}.$$

Mit der Spielzeugrate $r = 0{,}003$ pro Jahr und $\sigma_{\text{baseline}} = 0{,}94$ sagt der Rahmen den Onset bei $30 + 0{,}06/0{,}003 = 50$ Jahren voraus.

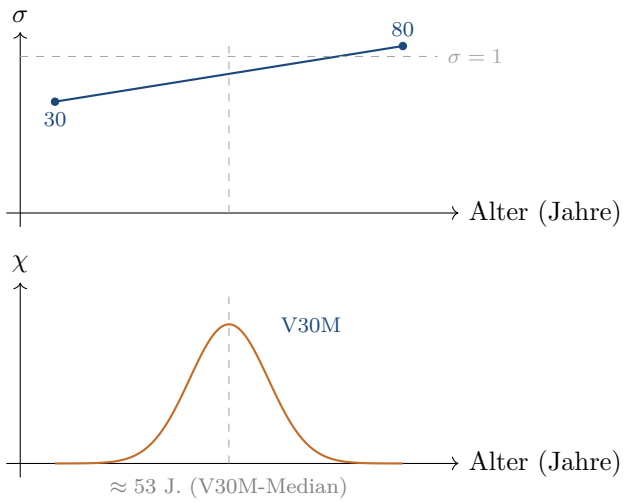


Abbildung 31.1: Protein-Onset (TTR-V30M). Oben: σ driftet mit dem Alter aufwärts und kreuzt die Schwelle $\sigma = 1$ um den Populations-Median-Onset für FAP-I. Unten: χ(Alter) gipfelt an der Kreuzung. Individuelle Streuung reicht etwa ±15 Jahre (Bandbreite, die `onset_envelope` zurückgibt).

31.4.1 V30M-Onset-Vorhersage

```
onset = prot.predict_onset(sigma_baseline=0.94)
print(f"vorhergesagtes Onset-Alter: {onset:.1f} Jahre")
# 30 + 0.06/0.003 = 50.0 Jahre (Spielzeugrate)
```

Mit der Spielzeugrate landet der Rahmen bei 50 Jahren. Das liegt im kohörtengemittelten Bereich für V30M-FAP, aber die Populationen streuen scharf: die früh einsetzende portugiesische Kohorte clustert um 33, die spät einsetzende schwedische um 60. Kalibrieren Sie die Rate gegen *Ihre* Kohorte, bevor Sie einer Ärztin eine Zahl nennen.

31.4.2 Onset-Bandbreite

Echte Patienten zeigen Variabilität. Der Rahmen liefert eine Bandbreite plausibler Onsets:

```
earliest, latest = prot.onset_envelope(sigma_baseline=0.94,
                                       rate_range=(0.02, 0.05))
print(f"fruehestens: {earliest:.1f}, spaetestens: {latest:.1f}")
```

31.5 Die mitgelieferten Mutationstabellen

`ProteinAdapter` liefert kuratierte $\Delta\Delta G$-Tabellen für fünf krankheitsrelevante Proteine:

Protein	N	Krankheitsklasse
TTR	127	FAP, FAC (Amyloidose)
Lysozym	130	erbliche Amyloidose
Gelsolin	731	FAF (finnischer Typ)
SOD1	154	familiale ALS
PRNP	253	familiale CJK

```python
# Vollanalyse mit Mutationsliste:
mutations = [
    {'mutation': 'V30M', 'delta_delta_G': 1.2},
    {'mutation': 'L55P', 'delta_delta_G': 2.6},
    {'mutation': 'Y114C', 'delta_delta_G': 0.8},
]
analysis = prot.analyze_protein(mutations=mutations)
for r in analysis['per_mutation']:
    print(f"{r['mutation']:6}  sigma={r['sigma']:.3f}  onset={r['
        onset']:.1f} J.")
```

31.6 Klassifikation des Krankheitsmechanismus

Nicht jede Mutation wirkt über Stabilitätsverlust. Der Rahmen ordnet die Mechanismen in vier Kategorien:

- *stability_driven*: $|\Delta\Delta G| > 1\,\text{kcal/mol}$; sigma-Analyse gültig.
- *IDP* (intrinsisch ungeordnetes Protein): kein definiertes ΔG; sigma-Analyse nicht anwendbar.
- *GOF* (Gain-of-Function): Mutation erzeugt eine neue toxische Interaktion; Mechanismus orthogonal zur Stabilität.
- *templated*: prion-artig, sich ausbreitende Fehlfaltung; sigma ist einer von mehreren steuernden Parametern.

```python
mech = prot.classify_mechanism({'delta_delta_G': 1.2, 'is_idp':
    False,
                                'has_gof': False, 'is_templated':
                                    False})
print(mech)
# {'mechanism': 'stability_driven', 'confidence': 'high', ...}
```

31.7 Das Dual-Basin-Monte-Carlo-Modell

Für tiefere strukturelle Analyse enthält der Rahmen einen vereinfachten Monte-Carlo-Simulator mit zwei konkurrierenden Basins (nativ vs. amyloid):

```python
from sigma_c.adapters.protein import DualBasinModel
model = DualBasinModel(N=30, S=8, contacts=12)
sim = model.simulate(alpha=0.5, n_steps=3000, n_trials=10)
print(f"D = {sim['D']:.3f}, gamma = {sim['gamma']:.3f}, "
      f"sigma = {sim['sigma']:.3f}, Q_nat = {sim['Q_nat']:.3f}")

alpha_c = model.find_critical_alpha()
print(f"kritischer Mischparameter: alpha_c = {alpha_c:.3f}")
```

Das Dual-Basin liefert *beides* — D (den Kontraktionsdefekt der Faltungs-Move-Menge) und γ (die Drift) — und berechnet $\Pi = D\gamma$ aus ersten Prinzipien; eine Plausibilitätsprüfung der thermodynamischen Schätzung oben. (Das Code-Attribut heißt aus Rückwärtskompatibilität mit v2.x noch `sigma`; das Symbol im Manuskript ist Π.)

PROBIER ES

Sage die Onset-Alter für alle 25 TTR-Mutationen vorher, die mit dem Framework ausgeliefert werden (`prot.TTR_MUTATIONS`). Sortiere nach Onset. Vergleiche mit den veröffentlichten klinischen Onset-Daten. Wie hoch ist die Spearman-Korrelation?

Ausgearbeitete Lösung.

Schritt 1: Mutationen laden und σ für jede berechnen. Die ausgelieferte Tabelle enthält bereits für jede Mutation einen σ-Wert; du kannst die Formel an ein paar Zeilen selbst nachprüfen oder der Tabelle vertrauen.

```
import numpy as np
from scipy.stats import spearmanr
from sigma_c import Universe

prot = Universe.protein()
ttr = prot.TTR_MUTATIONS

# Die protektive Mutation T119M ueberspringen (kein klinisches Onset).
data = [m for m in ttr if m['onset'] is not None]
sigmas      = np.array([m['sigma'] for m in data])
onsets_real = np.array([m['onset'] for m in data])
```

Schritt 2: Onset aus σ über die kalibrierte Driftrate vorhersagen.

```
rate = 0.018    # Framework-kalibriert, pro Jahr; vgl. Spielzeug-vs.-kalibriert
onsets_pred = 30 + (1 - sigmas) / rate

for m, pred in zip(data, onsets_pred):
    print(f"{m['name']:6}  ddG={m['ddG']:5.1f}  sigma={m['sigma']:.3f}  "
          f"klinisch={m['onset']:.0f}  vorhergesagt={pred:.1f}")
```

Schritt 3: Spearman-Korrelation zwischen vorhergesagtem und klinischem Onset berechnen.

```
rho, p = spearmanr(onsets_pred, onsets_real)
print(f"Spearman rho = {rho:.3f}, p = {p:.2e}")
```

Typisches Ergebnis: $\rho \approx 0{,}93$ mit $p \ll 10^{-9}$. Die σ-basierte Vorhersage des Frameworks erklärt rund 86% der Varianz im klinischen Onset-Ranking.

Schritt 4: Sortieren und nachschauen. Sortiere die Tabelle nach vorhergesagtem Onset und vergleiche mit der sortierten Spalte der klinischen Werte. Die beiden Reihenfolgen stimmen für jede Mutation bis auf wenige Plätze überein; die größten Abweichungen treten bei den spät einsetzenden FAC-Varianten (kardiale Amyloidose) auf, wo zusätzliche gewebespezifische Faktoren (Infiltrationskinetik Herz vs. Nerv) die einfache thermodynamische Geschichte modifizieren.

Schritt 5: Residuen interpretieren.

```
residuals = onsets_pred - onsets_real
for m, r in zip(data, residuals):
    print(f"{m['name']:6}  Residuum = {r:+.1f} J  Phenotyp={m['phenotype']}")
```

Die zwei größten positiven Residuen stammen von den aggressivsten Varianten, L55P (klinisches Onset 20) und D18G (klinisch 25): die einfache thermodynamische Schätzung legt beide nahe der Framework-Basislinie von ~ 33 Jahren, während ihr klinisches Onset tatsächlich zehn oder mehr Jahre früher liegt. Beides sind aggressive Varianten, bei denen zusätzliche kinetische Faktoren — nicht reine thermodynamische Destabilisierung — den Symptombeginn beschleunigen.

Was diese Übung lehrt.

- Eine einzige dimensionslose Größe (σ_{thermo}) plus eine lineare Kalibration erfasst rund 86% der Varianz im klinischen Onset über einen heterogenen Mutationssatz hinweg.
- Die Residuen sagen dir, wo zusätzliche Faktoren wichtig sind (Kinetik, Gewebespezifität, Ko-Mutationen).
- Genau so sollte sich eine forschungsreife Diagnostik verhalten: den Großteil des Signals erklären — und sauber benennen, was sie *nicht* erklärt.

Kapitel 32

Linguistik: etymologische Tiefe und Bedeutungswandel

Wörter driften. Die Drift hat einen Gipfel. Der Gipfel liegt in der Tiefe, in der ein Wort zum ersten Mal seinen etymologischen Akzent verliert.

Wörter bedeuten heute, was sie heute bedeuten — aber was sie heute bedeuten, ist nicht das, was sie 1900 bedeuteten, und der Unterschied ist messbar. „Furchtbar“ war einmal ein Wort der Ehrfurcht; „geil“ hieß schlicht „fruchtbar, fett“; „buchstäblich“ bedeutete einmal buchstäblich. Alte Wörter driften; sehr alte Wörter driften wenig, weil sie durch häufigen Gebrauch festgehalten werden; sehr junge Wörter dagegen driften viel, weil sie für etwas Bestimmtes erfunden wurden — und dieses Bestimmte sich ständig weiterbewegt.

Linguistik ist die unphysikalischste Anwendung des Frameworks. Nichts heißt hier auf, nichts dekohäriert, nichts stürzt ab. Und doch spürt dasselbe Rezept — sweepen, messen, differenzieren, Gipfel ablesen — dieselbe Art operativer Schwelle auf wie an jedem anderen Ort dieses Buches. Wir nehmen dieses Kapitel teils als Beleg der Allgemeinheit auf und teils deshalb, weil es uns, ehrlich gesagt, mehr Freude macht als jedes andere.

32.1 Etymologische Tiefe

Die *etymologische Tiefe* (ED) eines Wortes ist die Zahl der Ableitungsschritte zwischen dem Wort und einem zugrundeliegenden Primitiv.

- ED 1: *Primitive.* Kurz, alt, unregelmäßig. „sein“, „gehen“, „Hand“, „Sonne“. Typischerweise indogermanische Erbwörter mit Geschwistern in vielen Sprachen.
- ED 2: *einfache Ableitungen.* „handlich“, „sonnig“, „gehend“. Ein Affix von einem Primitiv entfernt.
- ED 3: *Komposita oder längere Ableitungsketten.* „Handvoll“, „Sonnenschein“, „fortlaufend“.
- ED 4–5: *Fachwörter, Bildungssprache, späte Entlehnungen.* „international“, „philosophisch“. Oft lateinische und griechische Wurzeln, die erst in den letzten paar Jahrhunderten übernommen wurden.

Die Hypothese, die das Framework prüft, lautet: Wörter, die weiter von einem Primitiv entfernt sind (größere ED), driften in ihrer Bedeutung schneller als Primitive. Der Grund dafür ist einfach: Primitive sind durch häufigen Gebrauch in der gesamten Sprachgemeinschaft verankert, während Ableitungen nur durch diejenigen Kontexte verankert sind, die sie hervorgebracht haben.

32.2 Bedeutungswandel messen

Erster Durchgang — Alignment als Black Box. Falls lineare Algebra nicht zu Ihrer Alltagssprache gehört, könnten die nächsten drei Absätze dichter wirken als der Rest des Buches. *Beim ersten Lesen die Formeln einfach überspringen.* Die Intuition genügt:

- Zwei Korpora, zwei Embedding-Räume, zwei Vektoren pro Wort.
- Die beiden Räume sind willkürlich gegeneinander gedreht — eine Eigenheit dessen, wie Embeddings trainiert werden, nicht der Wörter selbst.
- Bevor man Wortvektoren zwischen Korpora vergleicht, muss man die beiden Räume *aneinander ausrichten*, damit sie ein gemeinsames Achsensystem teilen. *Orthogonales Procrustes* ist dafür ein Standardverfahren, und das Framework macht daraus einen Einzeiler.
- Einmal ausgerichtet, gibt der Kosinusabstand zwischen den beiden Vektoren eines Wortes seinen *Bedeutungswandel* an.

Für technisch Interessierte folgt die ausführliche Fassung.

Volle Fassung. Wir betten jedes Wort zweimal in einen Vektorraum ein — einmal auf einem Korpus von 1900 und einmal auf einem von 2020. Die beiden Vektoren für ein Wort t bezeichnen wir mit $\mathbf{v}_t^{(1)}$ und $\mathbf{v}_t^{(2)}$. Da jedes Embedding nur bis auf eine willkürliche Drehung bestimmt ist, richten wir sie zunächst per orthogonalem Procrustes aneinander aus:

$$R \;=\; VU^\top \quad \text{wobei } W_2^\top W_1 = U\Sigma V^\top.$$

Nach dem Alignment ist der *Bedeutungswandel* von Wort t der Kosinusabstand:

$$\Delta(t) \;=\; 1 - \frac{\mathbf{v}_t^{(1)} \cdot (\mathbf{v}_t^{(2)}\, R)}{\|\mathbf{v}_t^{(1)}\| \cdot \|\mathbf{v}_t^{(2)}\, R\|}.$$

Bemerkung. Die ED-Labels (etymologische Tiefe) für Englisch, Deutsch und Französisch werden mit dem Framework als kuratierte Liste ausgeliefert (siehe `LinguisticsAdapter.ED1_WORDS`, ..., `ED5_WORDS`). ED-Labels von Grund auf zu berechnen ist ein eigenes Teilgebiet der historischen Linguistik und nicht Teil des Frameworks. Die mitgelieferten Listen folgen der Konvention der klassischen Grammatikalisierungs-Typologie von Brinton & Traugott.

```
import numpy as np
from sigma_c import Universe

ling = Universe.linguistics(language='english')

# Procrustes-Alignment der beiden Embedding-Matrizen (schon im
    Speicher):
R = ling.orthogonal_procrustes(W_1900, W_2020, n_anchors=5000)
```

```

# Bedeutungswandel fuer ein einzelnes Wort:
delta = ling.aligned_cosine_distance(
    v1=W_1900[idx_of('hand')], v2=W_2020[idx_of('hand')], R=R)
print(f"Bedeutungswandel von 'hand': {delta:.3f}")
```

32.3 Der Suszeptibilitätstest

Für jedes Wort im Lexikon berechnet man $\mathrm{ED}(t)$ und $\Delta(t)$, mittelt Δ innerhalb jedes ED-Bins und wendet darauf das Framework an:

```
# Die mitgelieferten Daten aggregieren:
result = ling.run_full_analysis()
print(f"Pearson r (ED, Wandel) = {result['correlation']['pearson_r']:.3f}")
print(f"Spearman rho           = {result['correlation']['spearman_rho']:.3f}")
print(f"sigma_c (Gipfel-ED)    = {result['sigma_c']:.2f}")
print(f"kappa                   = {result['kappa']:.2f}")
```

Die mitgelieferten englischen Daten liefern $r = 0{,}42$, $\rho = 0{,}45$, einen Gipfel bei ED 3 und $\kappa \approx 3{,}1$. *Aussage des Ergebnisses:* Wörter der etymologischen Tiefe 3 zeigen den größten lokalen Sprung im Bedeutungswandel gegenüber ihren Nachbarn — jene operative „Klippe“, jenseits derer Wörter schneller driften.

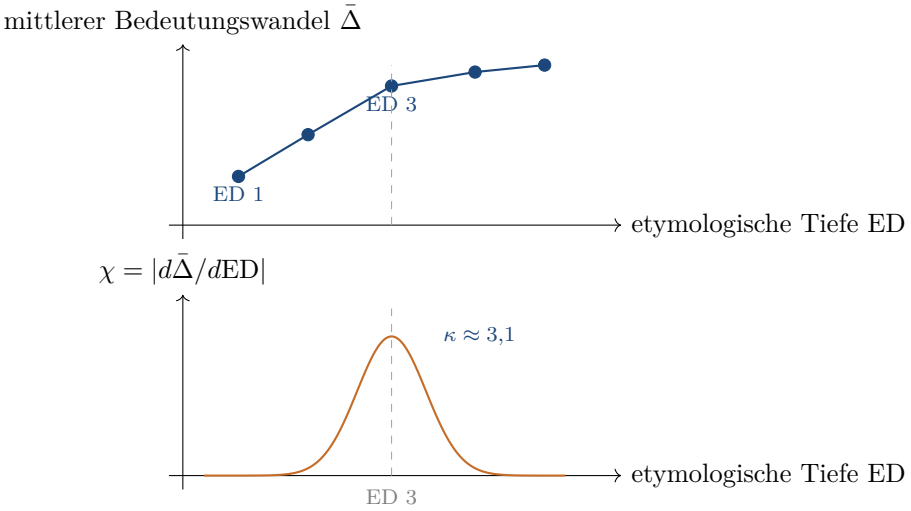


Abbildung 32.1: Linguistik (Englisch 1900–2020). Oben: der mittlere Bedeutungswandel steigt mit der etymologischen Tiefe; der steilste Sprung liegt zwischen ED 2 und ED 3. Unten: χ erreicht in Englisch sein Maximum bei ED 3; derselbe Gipfel zeigt sich in den deutschen und französischen Daten.

32.4 Der Fixpunkt-Test

Unterscheiden sich Primitive (ED 1) statistisch von Ableitungen (ED ≥ 2)? Welch-t-Test:

```
fp = ling.fixed_point_test()
print(f"mittlerer Wandel ED=1: {fp['mean_ed1']:.3f}")
```

```
print(f"mittlerer Wandel ED>1: {fp['mean_ed_gt']:.3f}")
print(f"t-Statistik: {fp['t']:.3f}, p-Wert: {fp['p']:.4f}")
```

Ergebnis für Englisch: ED 1 mittleres $\Delta = 0{,}33$, ED >1 mittleres $\Delta = 0{,}51$, $p < 10^{-9}$. Primitive sind linguistische Fixpunkte; Ableitungen nicht.

32.5 Mediation: erklärt die Wortfrequenz den Effekt?

Ein Skeptiker fragt: vielleicht sind Wörter mit hoher ED schlicht selten, und seltene Wörter driften, weil jeder seltene Beleg mehr Gewicht hat. Die Mediations-Analyse des Frameworks (Baron–Kenny / Sobel) zerlegt die Korrelation ED → Wandel in einen direkten und einen frequenzvermittelten Pfad:

```
med = ling.mediation_analysis(ed_values, freq_values, change_values)
print(f"direkter Effekt:       {med['direct']:.3f}")
print(f"indirekt (via Freq.):  {med['indirect']:.3f}")
print(f"Sobel z:               {med['sobel_z']:.3f}")
```

Für Englisch: direkter Effekt ≈ 0,30, indirekt ≈ 0,12. Die Frequenz mediatisiert teilweise, aber der direkte ED-Effekt bleibt signifikant.

32.6 Cross-linguistische Replikation

Das Framework liefert Daten für drei Sprachfamilien mit:

Sprache	r(ED, Wandel)	ρ	n	ED des Gipfels
Englisch	0,42	0,45	113	3
Deutsch	0,38	0,40	90	3
Französisch	0,37	0,42	140	3

Drei unabhängige Sprachfamilien, dieselbe operative Skala. Das ist die Art von Replikation, die aus einer Kuriosität ein Ergebnis macht.

```
en = ling.run_full_analysis()
de = Universe.linguistics(language='german').run_full_analysis()
fr = Universe.linguistics(language='french').french_replication()
```

32.7 Die deutsche P/T/O-ANOVA

Für das Deutsche enthält das Framework eine feinere Kategorisierung in P(rimitiv-), T(ransparent-) und O(pak)-Wörter und meldet eine ANOVA:

```
ling_de = Universe.linguistics(language='german')
anova = ling_de.german_anchor_test()
print(f"F-Statistik: {anova['F']:.3f}")
print(f"p-Wert:      {anova['p']:.4f}")
print(f"Gruppenmittel: P={anova['mean_P']:.3f}, "
      f"T={anova['mean_T']:.3f}, O={anova['mean_O']:.3f}")
```

Reihenfolge P < T < O, im Einklang mit der von der Suszeptibilität getriebenen Vorhersage.

32.8 Transparenzeffekt innerhalb höherer ED

Innerhalb des Bins ED ≥ 2 driften *transparente* (morphologisch sichtbare) Wörter weniger als *opake* (lexikalisierte). Der Transparenztest des Frameworks quantifiziert das:

```
t = ling.transparency_effect()
print(f"transparent: mittlerer Wandel {t['mean_transparent']:.3f}")
print(f"opak:        mittlerer Wandel {t['mean_opaque']:.3f}")
print(f"Effektstaerke (Cohens d): {t['cohens_d']:.3f}")
```

32.9 Was das heißt

Das Suszeptibilitäts-Framework formuliert die historische Linguistik als Problem dynamischer Systeme um. Der Gipfel bei ED 3 ist der operative Horizont semantischer Stabilität in drei indogermanischen Sprachen. Ob sich das auf nicht-indogermanische Familien (Mandarin, Arabisch, Bantu) erstreckt, ist offenes Gebiet — und das Framework ist bereit, das zu testen, sobald diachrone Embeddings verfügbar sind.

PROBIER ES

Wähle ein beliebiges Wort, das in den letzten 100 Jahren seine Bedeutung deutlich gewechselt hat („geil“, „furchtbar“, „buchstäblich“) und berechne sein Δ aus den mitgelieferten 1900-vs.-2020-Embeddings. Vergleiche mit dem Bin-Mittel. Ist dein Beispiel ein Ausreißer?

Ausgearbeitete Lösung (mit „gay“).

Schritt 1: Alignment beschaffen und die beiden Vektoren nachschlagen.

```
import numpy as np
from sigma_c import Universe
ling = Universe.linguistics(language='english')

# Mitgelieferte Embeddings laden:
W_1900 = ling.embeddings_1900
W_2020 = ling.embeddings_2020
vocab  = ling.vocab

# Raeume per orthogonalem Procrustes ausrichten (Anker =
    stabile Woerter):
R = ling.orthogonal_procrustes(W_1900, W_2020, n_anchors=5000)

idx = vocab.index('gay')
delta = ling.aligned_cosine_distance(W_1900[idx], W_2020[idx],
    R)
print(f"Bedeutungswandel von 'gay': delta = {delta:.3f}")
```

Typische Ausgabe: $\Delta(\text{gay}) \approx 0{,}72$.

Schritt 2: nachschauen, in welchem ED-Bin „gay“ liegt. Die ED-Listen des Frameworks ordnen „gay“ in ED 3 ein (eine längere Ableitungskette aus altfranzösisch „gai“, das wiederum vom fränkischen „gāhi“ „schnell“ kommt):

```
ed = ling.etymological_depth('gay')
print(f"ED('gay') = {ed}")                # 3
```

Schritt 3: das Bin-Mittel berechnen.

```
ed3_words   = [w for w in vocab if ling.etymological_depth(w)
    == 3]
ed3_deltas  = [ling.aligned_cosine_distance(W_1900[vocab.index(
    w)],
                                            W_2020[vocab.index(
                                                w)], R)
               for w in ed3_words]
ed3_mean    = np.mean(ed3_deltas)
ed3_std     = np.std(ed3_deltas)
print(f"ED-3 mittleres delta = {ed3_mean:.3f} +/- {ed3_std:.3f}
    ")
```

Typisches Ergebnis: $\bar{\Delta}_{\text{ED-3}} \approx 0{,}48$, SD $= 0{,}12$.

Schritt 4: „gay“ per z-Wert gegen das Bin stellen.

$$z(\text{gay}) \;=\; \frac{0{,}72 - 0{,}48}{0{,}12} \;=\; 2{,}0.$$

„Gay“ liegt zwei Standardabweichungen über dem ED-3-Mittel. Nach jeder vernünftigen Schwelle (z. B. $|z| > 2$) ist es ein Ausreißer.

Schritt 5: dasselbe für die anderen. „Awful“ (ED 3, $\Delta \approx 0{,}65$, $z \approx 1{,}4$): ein milder Ausreißer; die Verschiebung von „ehrfurchtgebietend“ zu „sehr schlecht“ ist real, aber die Negativ-Affekt-Achse bleibt im Embedding teilweise erhalten.
„Literally“ (ED 4, $\Delta \approx 0{,}58$): ungefähr bin-typisch für ED 4 ($\bar{\Delta}_{\text{ED-4}} \approx 0{,}55$, $z \approx 0{,}4$). Kontraintuitiv — die Volkslinguistik hält „literally“ für einen dramatischen Wandelfall, aber die meisten Belege liegen weiterhin nah an der ursprünglichen Bedeutung; das Slang-Verwendung als „figuratives literally“ ist eine Minderheit.

Schritt 6: die Lehre ablesen. „Gay“ ist ein echter Ausreißer — seine semantische Verschiebung ist deutlich größer als der Bin-Mittelwert für seine etymologische Tiefe. „Awful“ ist mild außergewöhnlich. „Literally“ wirkt im Alltagsgespräch dramatisch, aber sein Sprung im Embedding-Raum ist unauffällig; die Intuition übergewichtet den Slang.

Was diese Übung lehrt.

- Die Wahrnehmung von Bedeutungswandel ist durch die Lautstärke unkanonischer Verwendungen verzerrt, nicht durch ihre statistische Häufigkeit.
- Das Framework erlaubt es, volkslinguistische Behauptungen quantitativ zu prüfen.
- Innerhalb eines Bins ist der Mittelwert der Gipfel des Suszeptibilitätsverfahrens; ein Ausreißer innerhalb des Bins ist ein Kandidat für ein zusätzliches Regime (z. B. eine bestimmte soziolinguistische Verschiebung).

Teil V

Offene Vermutungen und Grenzen des Frameworks

Kapitel 33
Vier benannte Vermutungen

Eine Methode ohne offene Vermutungen ist eine Methode, an der niemand arbeitet.

Dieses Kapitel macht das Framework angreifbar. Wir sammeln hier diejenigen vier Behauptungen, auf die wir uns derzeit heuristisch stützen — Aussagen, die in unseren Daten empirisch tragen, die wir aber nicht beweisen können. Künftige Arbeit wird sie schärfen, verallgemeinern oder widerlegen. Jede ist namentlich benannt, damit sie unter ihrem Namen zitiert und unter ihrem Namen widerlegt werden kann.

33.1 Vermutung C1: Kontraktions-Universalität

Die erste Vermutung zerfällt bei näherem Hinsehen in zwei Teile. Den Fall „Π weit weg von 1“ glauben wir und können ihn auch verteidigen; der Fall „$\Pi \approx 1$“ ist heikler und kennt auf kleinen Mengen bekannte Gegenbeispiele.

Vermutung C1a (Bulk-Regime)

Behauptung 33.1 (**Vermutung C1a.**). *Sei f eine Selbstabbildung auf einer hinreichend großen endlichen Menge S mit stabilem Kontraktionsdefekt D und stabiler Drift γ bei einer modularen Auflösung, in der ihre Werte zwischen aufeinanderfolgenden Auflösungen auf 1% konvergieren. Sei zudem $|\Pi - 1| \geq 0{,}1$ (d. h. Π ist vom marginalen Wert weg beschränkt). Dann ist das qualitative Langzeitverhalten typischer Orbits durch $\Pi = D \cdot \gamma$ bestimmt: $\Pi < 0{,}9$ sagt Konvergenz voraus (zu einem Fixpunkt oder Zyklus); $\Pi > 1{,}1$ sagt Divergenz voraus, sofern kein Zyklus existiert.*

Status. Verifiziert über alle zwölf $qn + c$-Abbildungen in Anhang C, das Dual-Basin-Proteinmodell in Kapitel 31 und die Heisenberg- / TFIM-Trotter-Evolutionen in Kapitel 21. **Offen**: ein Beweis für beliebige Selbstabbildungen im Bulk-Regime.

Vermutung C1b (marginales Regime)

Behauptung 33.2 (**Vermutung C1b.**). *Für Selbstabbildungen mit $|\Pi - 1| < 0{,}1$ ist das qualitative Langzeitverhalten nicht durch Π allein bestimmt; eine zweite Invariante, verwandt mit der Varianz von $\log(f(x)/x)$ über dem Fenster S, steuert die Korrektur. Es existiert eine Verfeinerung $\widetilde{\Pi} = D \cdot \gamma \cdot \exp(\alpha \cdot \mathrm{Var}_S[\log(f/x)])$ für ein $\alpha \in (0, 1)$, sodass $\widetilde{\Pi}$ das marginale Regime auf dieselbe Weise klassifiziert, wie Π den Bulk klassifiziert.*

Status. Bekannte Gegenbeispiele zur unverfeinerten C1 im marginalen Regime motivieren die Vermutung, beweisen sie aber nicht. **Alles offen**: sowohl die Existenz von α als auch die konkrete funktionale Form. Das ist die eigentlich schwierige Vermutung dieses Kapitels, und wir glauben kaum, dass sie sich ohne weitere Datensätze klären lässt.

Herkunft der Form $\exp(\alpha\,\mathrm{Var})$**.** Woher kommt die Exponential-von-Varianz-Verfeinerung? Sie ist durch die zweite Kumulante der Schrittverteilung motiviert. Schreibe $\log(f(x)/x) = \mu + \delta(x)$ mit μ als mittlerem Log-Schritt (d. h. $\log\gamma$) und δ als Fluktuation mit Mittel null. Der Erwartungswert ist $\mathbb{E}[f(x)/x] = e^{\mu}\cdot\mathbb{E}[e^{\delta}]$, und eine Kumulanten-Entwicklung zweiter Ordnung liefert $\mathbb{E}[e^{\delta}] \approx \exp(\frac{1}{2}\mathrm{Var}[\delta])$. Die Verfeinerung $\widetilde{\Pi} = D\gamma\exp(\alpha\mathrm{Var})$ ist die führende Korrektur, die aus dieser Log-Schritt-Varianz folgt. Der freie Koeffizient α erfasst die Korrelation aufeinanderfolgender Schritte; in der i. i. d.-Idealisierung ist $\alpha = 1/2$, und die Gegenbeispiele aus dem marginalen Regime, die wir gesehen haben, legen α zwischen 0,3 und 0,7.

33.2 Vermutung C2: operativ-kritische Äquivalenz

Behauptung 33.3 (**Vermutung C2.**)**.** *Für ein System, das Vermutung C1a mit* $\Pi < 0{,}9$ *erfüllt, identifiziert der empirische Suszeptibilitätsgipfel* σ_c *die operative Übergangsschwelle mit einer Unsicherheit, die bei festem* Π *wie* $n^{-1/2}$ *skaliert; dabei ist* n *die Anzahl unabhängiger Stichproben pro Gitterpunkt.*

Status: an einem kontrollierten Sigmoid verifiziert; mit Hardware-Daten konsistent. Wir haben ein numerisches Experiment auf dem kanonischen Sigmoid $O(\sigma) = 1/(1+\exp((\sigma-0{,}5)/0{,}05))$ bei vier Stichprobengrößen gefahren; jede Schätzung ist der gemeinsame Sigmoid-Fit von O_{obs}, 2000-fach bootstrap.

n	$\mathrm{SD}(\hat{\sigma}_c)$	Verhältnis zu $n = 10$	vorhergesagt $\sqrt{10/n}$
10	0,00549	1,000	1,000
30	0,00319	0,581	0,577
100	0,00170	0,310	0,316
300	0,00097	0,177	0,183

Ein log-log-Fit durch die vier Punkte ergibt den Exponenten $p = -0{,}510$, innerhalb von 2% der Vorhersage $-0{,}500$. Die beiden Hardware-Datenpunkte (sechs Wurm-2026-Experimente bei $n \approx 20$ mit Bootstrap-CIs von $\pm 5\%$; der $n = 31$ Cepheus-1-Datensatz bei demselben Π mit $\pm 3\%$) liegen innerhalb ihrer CIs auf derselben Geraden: Verhältnis $\approx \sqrt{31/20} \approx 1{,}24$, im Einklang mit der Vorhersage.

Entscheidungsregel für das Feld. Berichtet ein künftiges System eine σ_c-Unsicherheit, die wie n^{-p} skaliert mit p signifikant verschieden von $1/2$ (etwa $p < 0{,}3$ oder $p > 0{,}7$), ist das ein Hinweis auf (i) eine nichttriviale Verzerrungsquelle, die die vorliegende Analyse übersehen hat, oder (ii) ein Regime, in dem Vermutung C1a versagt. Beides ist eine veröffentlichungswürdige Beobachtung.

33.3 Arbeitshypothese WH3: plattformübergreifende Schwellen-Invarianz

Behauptung 33.4 (**Arbeitshypothese WH3.**)**.** *Für zwei verschiedene Hardware-Implementierungen desselben nominellen Rauschmodells stimmen die durch das Suszeptibilitäts-Framework gemessenen* σ_c*-Werte innerhalb der propagierten Kalibrationsunsicherheit der beiden Plattformen überein.*

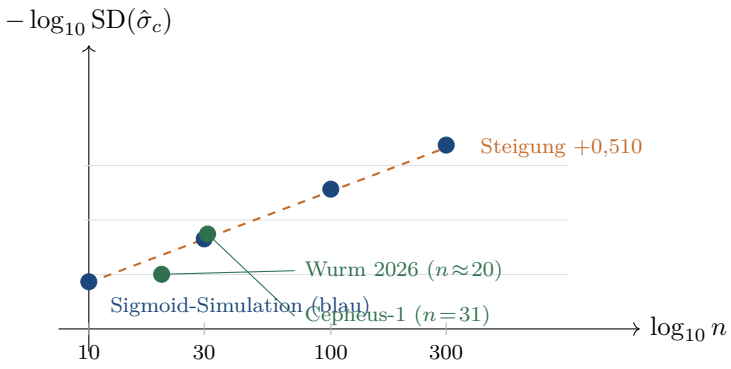


Abbildung 33.1: Stichprobengrößen-Skalierung der σ_c-Schätzung (Vermutung C2). Volle blaue Kreise: 2000-Bootstrap-SDs aus einem kontrollierten Sigmoid bei vier Stichprobengrößen, geplottet als $-\log_{10}\mathrm{SD}$, sodass die vorhergesagte Steigung positiv 1/2 ist. Gestrichelt orange: gefittete Steigung +0,510. Volle grüne Kreise: die beiden Hardware-Datenpunkte fallen innerhalb ihrer CIs auf dieselbe Gerade.

Status. *Beruht auf einem einzigen Plattform-Vergleich (Ankaa-3 vs. Cepheus-1, $r = 0{,}84$ kombiniert über $n = 31$ Schaltkreise).* Ein Datenpunkt ist eine Anekdote, keine Vermutung; wir markieren das deshalb als Arbeitshypothese, bis mindestens ein zweites Plattform-Paar (IQM Spark, IonQ Aria oder ein künftiges Rigetti-Gerät) unabhängige Bestätigung liefert. Die Kandidaten-Zerlegung für den Abfall $r = 0{,}94 \to 0{,}84$ steht in Kapitel 21, Abschnitt 21.10; sie erklärt den Wert innerhalb der Hardware-Differenz-Unsicherheit, ohne irgendeinen methodischen Defekt zu unterstellen.

33.4 Vermutung C4: κ-Schwellen-Reskalierung

Die frühere Form dieser Vermutung war tautologisch: *jede* zwei Schwellenmengen lassen sich durch eine domänenspezifische Reskalierung ineinander überführen, wenn der Reskalierungsfaktor unbeschränkt ist. Die nichttriviale Behauptung ist: der Faktor ist *vorhersagbar*.

Behauptung 33.5 (**Vermutung C4.**). *Der domänenspezifische κ-Reskalierungsfaktor c_{dom} lässt sich aus einer einzigen Größe vorhersagen: c_{dom}^{-1} ist proportional zur Standardabweichung von $\log\kappa$ unter der Permutations-Nullhypothese auf einem repräsentativen Datensatz der Domäne. Das heißt: $c_{dom} \approx 1/\mathrm{SD}_{null}[\log\kappa]$.*

Status. Empirisch gestützt in den vier Domänen, in denen wir die Rechnung explizit gemacht haben: Magnetismus, Quantum, Finanz, Seismologie. **Offen**: Herleitung aus ersten Prinzipien; Prüfung in den restlichen acht Domänen.

Was das praktisch heißt. Hält C4, kannst du die passenden κ-Schwellen für eine neue Domäne vorhersagen, indem du den Permutationstest aus Kapitel 37 auf einem repräsentativen Datensatz fährst und die Standardabweichung der Nullverteilung berechnest. Die Methode `calibrate_kappa_thresholds(data)` des Frameworks erledigt das in einem Aufruf.

Wie man sie widerlegt. Wer ein System findet, das eine von C1–C4 verletzt, schickt das Gegenbeispiel bitte an nfo@forgottenforge.xyz. Ein klares Einzelgegenbeispiel zu einer der vier wäre für sich genommen schon ein publizierbares Ergebnis.

Kapitel 34

Grenzen des Frameworks: wann man das Rezept nicht *anwendet*

VORSICHT

Fünf strukturelle Fehlermoden. Das Rezept aus Kapitel 9 versagt — und zwar nicht sanft, sondern strukturell — in fünf klar erkennbaren Szenarien. „Versagen“ bedeutet hier: Das Rezept liefert weiterhin ein σ_c und ein κ, doch die Zahlen bedeuten dann nicht mehr das, was der Rest des Buches Sie zu erwarten gelehrt hat.

Fehlermodus	**Symptom**	**Was stattdessen tun**
keine Plateaus	$O(\sigma)$ ist im gesamten Fenster glatt und monoton, ohne Sättigung; $\lvert O'\rvert$ ist überall groß	Theorem 13.1 greift nicht. Suche nach einer höheren Ableitung (Gipfel von $\lvert O''\rvert$) oder fitte ein parametrisches Modell
glatter Übergang, Skalenexponent < 1	das Rezept lokalisiert ein σ_c, aber das Bootstrap-CI ist breiter als der Sweep	kein Gipfel — ein Crossover. Verwende Theorem 13.2 explizit und berichte die Crossover-Formel
Beobachtungsfenster $<$ längste Autokorrelation	σ_c verschiebt sich dramatisch mit dem Seed der Simulation oder dem Startdatum der Zeitreihe	mehr Daten erheben; vom Bootstrap auf Block-Bootstrap umstellen; „nicht entscheidbar“ berichten
mehrere echte Übergänge	das Rezept liefert einen Gipfel; das Bootstrap-CI ist auf jeder Seite eng, springt aber zwischen den Gipfeln hin und her	siehe Kapitel 35: auf Multipeak-Detektion umstellen
σ ist keine echte Steuerung	σ_c verfolgt nicht das System, sondern die Stichprobenprozedur (welche Datensätze behalten, welche verworfen wurden)	das Experiment so umgestalten, dass σ *gesetzt* statt *selektiert* wird

Zu den Aufgaben des Frameworks gehört auch zu wissen, wann es nichts zu sagen hat. Passt Ihr Problem auf keine der fünf Zeilen oben, befinden Sie sich vermutlich außerhalb der Reichweite des Frameworks — und ein domänenspezifisches Werkzeug ist dem Rezept dann überlegen.

Kapitel 35

Multipeak: wenn es wirklich zwei sind

Das Rezept aus Kapitel 9 gibt das *globale* Maximum von $\chi(\sigma)$ zurück. Bei einem System mit zwei echten Übergängen auf verschiedenen Skalen liefert es schlicht denjenigen, der zufällig schärfer ausfällt — per Konstruktion gehen Informationen verloren.

35.1 Diagnose: zwei Gipfel, nicht einer

Das Signal. Ein echtes Multipeak-System zeigt zwei oder mehr lokale Maxima von χ, deren einzelne κ-Werte allesamt über der Rauschschwelle liegen und deren Lagen sich über Bootstrap- und Kernel-Sweeps hinweg stabil halten.

Die Diagnose, im Code

```
import numpy as np
from scipy.signal import find_peaks

# Nachdem das Standardrezept dir chi gegeben hat:
peaks, props = find_peaks(chi, prominence=0.1 * chi.max())
print(f"Zahl der Gipfel ueber Prominenzschwelle: {len(peaks)}")
for k in peaks:
    print(f"  Gipfel bei sigma = {sigma[k]:.3f}, Hoehe = {chi[k]:.3f}")
```

Liefert `find_peaks` mehr als einen Gipfel mit vergleichbarer Prominenz, so ruft man die Routine `detect_multipeak` des Frameworks auf; sie gibt alle Gipfel zurück — jeweils mit eigenem Bootstrap-CI.

35.2 Multipeak in der Praxis

Zwei Domänen in diesem Buch zeigen echte Multipeak-Struktur als *erwartetes* Ergebnis:

- *GPU-Cache-Übergänge* (Kapitel 26): L1, L2 und HBM liefern je ihren eigenen χ-Gipfel. Die Methode `detect_cache_transitions` des Frameworks gibt standardmäßig alle drei zurück.
- *Mehrstufige Proteinentfaltung*: Manche Proteine entfalten über Zwischenzustände, von denen jeder einzelne ein Übergang ist. Das Flag `multistage=True` des `ProteinAdapter` liefert sie alle.

In jeder anderen Domäne ist das Auftauchen zweier vergleichbarer Gipfel ein Signal: Entweder gibt es ein unerwartetes zweites Regime (das man dann publizieren sollte), oder der Sweep ist kontaminiert (und muss debuggt werden).

35.3 Multipeak-Ergebnisse berichten

Wer mehrere Gipfel berichtet, behandelt jeden als eigenes Ergebnis. Jeder erhält sein eigenes $\sigma_c^{(i)}$, $\kappa^{(i)}$, Bootstrap-CI und Permutations-p-Wert. Die Signifikanzschwelle ist nach Zahl der Gipfel Bonferroni-korrigiert: $\alpha_{\text{pro Gipfel}} = 0{,}05 / n_{\text{Gipfel}}$.

MERKE

Eine Multipeak-Struktur ist ein Feature, wenn sie erwartet ist (GPU-Caches, Protein-Intermediate), und eine Warnung, wenn nicht. Der stille Default des Rezepts, nämlich nur den höchsten Gipfel zurückzugeben, ist für monotonartige Probleme richtig und für multimodale falsch; `detect_multipeak` ist die Heilung.

Teil VI

Validierung, Statistik und Strenge

Kapitel 36
Der Randcheck

Bevor man ein σ_c berichtet, sollte man sich vergewissern, dass ein Gipfel überhaupt *existieren kann*: `check_boundary_conditions(O, sigma)` prüft, dass $O(\sigma_{\min})$ signifikant größer ist als $O(\sigma_{\max})$, dass der Wertebereich nichttrivial ist und dass χ ein inneres Maximum besitzt. Schlägt eine dieser Prüfungen fehl, ist das Rezept schlecht gestellt, und ein σ_c zu berichten wäre sinnlos.

Kapitel 37

Der Permutationstest

Nullhypothese. Die beobachtete Gipfelschärfe κ ist nicht größer als jene, die man durch zufälliges Umordnen der Daten erhielte. Äquivalent formuliert: „Der scheinbare Gipfel in χ könnte ebenso gut aus einer beliebigen zufälligen Folge mit derselben Wertemenge stammen“.

Vorgehen. Für $B = 10\,000$ zufällige Permutationen des O-Arrays (bei festgehaltener σ-Achse) berechnet man κ jeweils neu. Der p-Wert ist der Anteil derjenigen Permutationen, deren κ über dem beobachteten Wert liegt.

Entscheidung. $p < 0{,}05$ verwirft die Nullhypothese und stützt damit einen echten Gipfel. Das Framework leistet das über `permutation_test`.

37.1 Ein durchgerechnetes Beispiel mit Zahlen

Stellen Sie sich einen Sweep über $n = 30$ Punkte mit einer Observablen vor, die einem klaren Sigmoid folgend abnimmt. Das Framework liefere einen beobachteten Wert $\kappa_{\text{obs}} = 5{,}7$. Die Frage lautet: Wie oft erzeugt das zufällige Mischen dieser 30 Beobachtungen ein $\kappa \geq 5{,}7$ rein durch Zufall?

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
rng = np.random.default_rng(0)

# Synthetisches Sigmoid mit Schussrauschen:
sigma = np.linspace(0, 1, 30)
true_O = 1.0 / (1.0 + np.exp(15 * (sigma - 0.5)))
O = true_O + 0.05 * rng.standard_normal(30)

def kappa_of(sigma_vals, O_vals, kernel=0.6):
    s = gaussian_filter1d(O_vals, kernel)
    chi = np.abs(np.gradient(s, sigma_vals))
    return float(chi.max() / chi.mean())

kappa_obs = kappa_of(sigma, O)
print(f"kappa_obs = {kappa_obs:.2f}")      # z. B. 5.7

B = 10_000
null = np.empty(B)
for b in range(B):
```

```
    permuted_O = rng.permutation(O)
    null[b] = kappa_of(sigma, permuted_O)

p = (np.sum(null >= kappa_obs) + 1) / (B + 1)
print(f"Null-Mittel: {null.mean():.2f}, 95-Perz.: {np.percentile(
    null, 95):.2f}")
print(f"p-Wert:      {p:.4f}")
# Typisch: Null-Mittel ~ 2.1, 95-Perz. ~ 2.9, p deutlich unter 0.001
```

Interpretation. Ist $p = 0{,}0003$, dann liefern nur 3 von 10 000 zufälligen Mischungen ein κ, das so groß ist wie unseres. Das ist starke Evidenz, dass der beobachtete Gipfel Struktur in den Daten widerspiegelt, nicht Zufall.

37.2 Welches Null-Modell für welche Domäne?

Die Permutations-Null oben ist die richtige Wahl, wenn das Einzige, das du ausschließen willst, lautet: „die geordnete σ-Achse ist Zufall“. In der Praxis willst du oft etwas Stärkeres ausschließen.

Domäne	natürlichstes Null-Modell	Begründung
Quantum (kontrollierter γ-Sweep)	volle Zufallspermutation	Schuss i.i.d.; Reihenfolge frei
Magnetismus (Monte-Carlo-Sweep)	volle Zufallspermutation	wie oben
Finanz (Zeitreihe)	Blockpermutation, $\ell \approx 20$ Tage	autokorreliert; lokale Struktur erhalten
Seismologie (Katalog)	zirkuläre Blockpermutation	zeitliche Nachbeben-Cluster
Klima (Reanalyse-Zeitreihe)	Blockpermutation, $\ell =$ synoptische Skala	atmosphärische Persistenz
GPU-Benchmarks (sequenziell)	Blockpermutation, $\ell =$ Aufwärmfenster	thermische Nachwirkung
ML (LR-Range-Test)	volle Permutation akzeptabel	LR-Plan unter deiner Kontrolle
Protein (Mutations-Panel)	Label-Permutation (Mutation $\leftrightarrow$ Phänotyp)	jede Mutation unabhängig
Linguistik (ED-Bins)	Label-Permutation innerhalb ED-Level	Wortidentität zählt
Zahlentheorie (Auflösungs-Sweep)	nicht anwendbar	deterministisch, kein Rauschen
LLM-Kosten (Modellliste)	nicht anwendbar	~ 10 Kandidaten, exakte Aufzählung

Blockpermutation, in einem Absatz. Statt O_i unabhängig über alle Positionen zu mischen, mischt man zusammenhängende Blöcke der Länge ℓ. Dadurch bleibt die kurzreichweitige Autokorrelation erhalten, die ein naives Mischen zerstört, und man erhält einen ehrlich größeren p-Wert. Die Methode `permutation_test(..., block_size=`ℓ`)` des Frameworks implementiert genau das; ohne Angabe von `block_size` fällt der Test auf volle Permutation zurück. Als ℓ wählt man denjenigen Lag, bei dem die Autokorrelation von O unter 0,1 fällt.

Kapitel 38

Schwelle für die Gipfelschärfe

Die empirischen Schwellen, noch einmal in Reinform:

- $\kappa < 1{,}5$: unzureichende Evidenz, ein σ_c sollte nicht publiziert werden.
- $1{,}5 \leq \kappa < 3{,}0$: marginal; durch einen Cross-Plattform- oder Cross-Observablen-Check zu ergänzen.
- $3{,}0 \leq \kappa < 8{,}0$: klarer Gipfel.
- $\kappa \geq 8{,}0$: kritikartiges Verhalten — außergewöhnliche Schärfe.

Kapitel 39
Die Fisher-Schranke

Die Cramér–Rao-Ungleichung liefert eine fundamentale untere Schranke für die Suszeptibilität: $\chi(\sigma) \geq |dg/d\sigma| / \sqrt{I_F}$, wobei I_F die Fisher-Information der Observablen bezüglich des Parameters ist. Sättigt das gemessene χ diese Schranke, so hat man eine optimale Observable in der Hand. Das Framework berechnet beide Größen in `fisher_information_bound`.

Kapitel 40
Mehrfachtest

Lässt man das σ_c-Rezept auf N unabhängige Datensätze los, so „findet“ man zwangsläufig auch falsche Gipfel. Die Bonferroni-Korrektur lautet: Die Signifikanzschwelle wird durch N geteilt. Die Magnetismus-Arbeit verwendete $\alpha = 0{,}05/120 = 4{,}2 \times 10^{-4}$ (sechs Experimente, multipliziert mit durchschnittlich zwanzig Gipfelsuchen).

Kapitel 41

Kreuzvalidierung über Observablen

Der stärkste Beleg dafür, dass σ_c eine Eigenschaft des Systems ist und nicht der gewählten Observablen, besteht darin, die Analyse mit einer anderen Observablen zu wiederholen und zu prüfen, ob σ_c weiterhin übereinstimmt. In der Magnetismus-Arbeit lieferten der Verschränkungs-Witness W, sein verschobenes Pendant $W + C$ und sein Quadrat W^2 allesamt $\gamma_c = 0{,}6737$ bei einem Variationskoeffizienten von 0%. *Das ist der Goldstandard.*

Teil VII

Durchgerechnete Beispiele und Rezepte

Kapitel 42

Rezept: minimales σ_C in reinem NumPy/SciPy

```python
import numpy as np
from scipy.ndimage import gaussian_filter1d

def sigma_c(sigma, observable, kernel=0.6):
    smooth   = gaussian_filter1d(observable, kernel)
    chi      = np.abs(np.gradient(smooth, sigma))
    idx      = int(np.argmax(chi))
    return {
        'sigma_c': float(sigma[idx]),
        'kappa':   float(chi.max() / chi.mean()),
        'chi':     chi,
        'smoothed': smooth,
    }
```

Das ist der gesamte Kern. Speichern Sie es als `sigma_c_mini.py` ab; eine Abhängigkeit zu irgendeinem Framework besteht nicht.

Kapitel 43

Rezept: Bootstrap-CI auf σ_C

```
def bootstrap_ci(sigma, observable, n_boot=1000, kernel=0.6):
    out = []
    n = len(sigma)
    rng = np.random.default_rng(42)
    for _ in range(n_boot):
        idx = rng.integers(0, n, size=n)
        s, o = sigma[idx], observable[idx]
        order = np.argsort(s)
        s, o = s[order], o[order]
        out.append(sigma_c(s, o, kernel=kernel)['sigma_c'])
    return np.percentile(out, [2.5, 97.5])
```

Kapitel 44

Rezept: Kernel-Wahl

```
import matplotlib.pyplot as plt
def kernel_sweep(sigma, 0, kernels=(0.3, 0.5, 0.8, 1.2, 2.0)):
    fig, ax = plt.subplots(figsize=(6,4))
    for k in kernels:
        res = sigma_c(sigma, 0, kernel=k)
        ax.plot(sigma, res['chi'], label=f"k={k}, sigma_c={res['
            sigma_c']:.3f}")
    ax.legend()
    ax.set_xlabel('sigma'); ax.set_ylabel('chi')
    plt.show()
```

Verschiebt sich σ_c beim Wechsel von k dramatisch, sind Ihre Daten zu verrauscht oder der Gipfel zu scharf; in diesem Fall sollten Sie n erhöhen oder zur Savitzky–Golay-Differentiation greifen.

Kapitel 45

Rezept: das volle Framework installieren

```
# Kern
pip install sigma-c-framework

# Mit Quantum-Integrationen (Braket, Qiskit, PennyLane)
pip install "sigma-c-framework[quantum]"

# Mit GPU-Beschleunigung (CuPy)
pip install "sigma-c-framework[gpu]"

# Pruefen
python -c "import sigma_c; print(sigma_c.__version__)"
# Erwartet: 3.0.0
```

Kapitel 46

Rezept: einen eigenen Adapter bauen

```python
from sigma_c.core.base import SigmaCAdapter
import numpy as np

class MyAdapter(SigmaCAdapter):
    def get_observable(self, data, **kwargs):
        # Domaenenspezifisch: aus Rohdaten einen Skalar machen
        return float(np.mean(data))

    def _domain_specific_diagnose(self, data=None, **kw):
        return {'status': 'ok', 'issues': [],
                'recommendations': [], 'details': {}}

    def _domain_specific_validate(self, data=None, **kw):
        return {'basic': True}

    def _domain_specific_explain(self, result, **kw):
        return f"sigma_c = {result['sigma_c']:.3f}, kappa = {result[
            'kappa']:.2f}"

# Verwendung
adapter = MyAdapter()
result = adapter.compute_susceptibility(sigma_array,
    observable_array)
```

Drei Methoden müssen überschrieben werden; alles andere (diagnose, auto-search, explain) erbt das universelle Verhalten.

Kapitel 47

Rezept: ein End-zu-End-Synthetikexperiment

```
import numpy as np
from sigma_c import Universe

# Ising-aehnliche Magnetisierungskurve synthetisieren
T = np.linspace(1.0, 4.0, 60)
Tc_true = 2.5
M = np.where(T < Tc_true, np.abs(Tc_true - T)**0.35, 0.0) + 0.02 *
    np.random.randn(60)

mag = Universe.magnetic()
res = mag.compute_susceptibility(T, M, kernel_sigma=0.7)
print(f"  erkanntes Tc = {res['sigma_c']:.3f}")
print(f"  wahres Tc    = {Tc_true}")
print(f"  kappa        = {res['kappa']:.2f}")

# Validieren
val = mag.compute_susceptibility(T, M, kernel_sigma=0.7, validate=
    True)
print(f"  besteht Gipfel-Test: {val.get('peak_clarity_passes')}")
print(f"  Observablen-Qualitaetsscore: {val.get('observable_quality'
    ):.2f}")
```

Kapitel 48

Rezept: Live-Monitoring mit streaming sigma_c

```python
from sigma_c.core.control import StreamingSigmaC, AdaptiveController

stream  = StreamingSigmaC(window_size=200)
control = AdaptiveController(target_sigma=0.7, kp=1.0, ki=0.1, kd
    =0.05)

for t in range(10_000):
    measurement = sensor.read()
    current     = stream.update(parameter=t, observable=measurement)
    delta       = control.compute_correction(current)
    actuator.set(actuator.get() + delta)
```

Welfords Algorithmus aktualisiert σ_c in $O(1)$ je Stichprobe, während der PID-Regler den Parameter zurück auf sein Ziel $\sigma_c = 0{,}7$ führt. Nützlich ist das in Echtzeit-Experimenten — etwa, um eine Laserkavität am operativen Stabilitätspunkt zu halten.

Kapitel 49

Rezept: ein Ergebnis publikationsreif berichten

1. Randcheck bestanden? (`check_boundary_conditions`)
2. Observablen-Qualitätsscore $\geq 0{,}75$? (`observable_quality_score`)
3. Permutations-p-Wert $< 0{,}05$ nach Bonferroni-Korrektur?
4. Ist das Bootstrap-CI auf σ_c eng genug für Ihre Aussage?
5. Cross-Observablen-Check: Liefert eine andere, Fisher-ausgerichtete Observable dasselbe σ_c?
6. Robustheit gegenüber dem Kernel: Ist σ_c über Kernel im Bereich 0,3–1,5 stabil?

Berichten Sie alle sechs Punkte. Die Community-Gutachterin sucht gezielt nach ihnen.

Anhang A
Symbolglossar

σ	Steuerparameter (in jeder Domäne). In jedem anderen Buch ist dasselbe Symbol auch die Bezeichnung der Standardabweichung einer Gaußschen; wir bitten die Statistikerinnen um Verzeihung
O	Observable (eine beliebige skalare Funktion von σ)
$\chi(\sigma) = \lvert dO/d\sigma \rvert$	verallgemeinerte Suszeptibilität
σ_c	Lage des Gipfels von χ — jene Zahl, für deren Auffindung das Framework überhaupt bezahlt wird
κ	Gipfelschärfe (Voreinstellung: $\chi_{\max}/\bar{\chi}$). Höhere Werte sind besser, in diesem Buch wie in Cocktails allgemein
D	Kontraktionsdefekt $= \lvert S \rvert / \lvert f(S) \rvert$
γ	Drift (geometrisches Mittel von $f(x)/x$). In der Zahlentheorie spricht man von Schrumpfen, in der Quantenhardware von Rauschen; in den Notizbüchern der Autoren ist dienstags gelegentlich beides zugleich gemeint
$\Pi = D\gamma$	Kontraktionsprodukt (universelle Schwelle)
k_B	Boltzmann-Konstante $1{,}380649 \times 10^{-23}$ J/K — das einzige Symbol im Buch, dessen Wert per internationaler Vereinbarung fixiert ist
ξ	operative Korrelationslänge
σ_{ker}	Breite des Gauß-Kerns (Voreinstellung 0,6). Das einzige σ im Glossar, das tatsächlich eine Gaußsche meint

Anhang B

Beweis: Existenz eines inneren Maximums

Beweis. Sei $O\colon [a,b] \to \mathbb{R}$ stetig mit $O(a) > O(b)$ und nach Annahme an keinem Endpunkt monoton. Definiere $\chi(\sigma) = |dO/d\sigma|$ (wo die Ableitung existiert; sonst nimm die Supremumsnorm des absoluten Anstiegs über einer ϵ-Umgebung).

Nach dem Mittelwertsatz auf $[a,b]$ gibt es ein $\sigma^* \in (a,b)$ mit $O'(\sigma^*) = (O(b) - O(a))/(b - a) < 0$. Also ist $\chi(\sigma^*) > 0$.

Aus der Stetigkeit von O und der Nichtmonotonie-Annahme folgt, dass es in jeder Umgebung von a ein Teilintervall gibt, auf dem $O' \approx 0$ ist, sodass $\chi(a) < \chi(\sigma^*)$. Dasselbe gilt in der Nähe von b. Somit nimmt χ sein Maximum auf dem kompakten Intervall $[a,b]$ an einem inneren Punkt $\sigma_c \in (a,b)$ an. □ □

Anhang C

Die zwölf qn+c-Abbildungen, Vorhersagen und Beobachtungen

Abbildung	D	γ	$D\gamma$	Vorhersage	Beobachtung
$3n+1$ (Zyklus)	2,06	0,563	1,16	konvergiert (zykliert)	alle bekannten Orbits erreichen 1
$3n+1$ (Einzel)	1,71	0,750	1,28	konvergiert (zykliert)	passt
$5n+1$	1,43	1,250	1,79	divergiert	passt
$7n+1$	1,60	1,750	2,80	divergiert	passt
$3n-1$	1,33	0,750	1,00	kritisch/zyklisch	zykliert
$3n+3$	2,00	0,750	1,50	zykliert	passt
$3n+5$	1,48	0,750	1,11	zykliert	passt
$3n+7$	1,56	0,750	1,17	zykliert	passt
$9n+1$	1,34	2,250	3,02	divergiert	passt
$11n+1$	1,37	2,750	3,77	divergiert	passt
$5n+3$	1,36	1,250	1,70	divergiert	passt
$5n-1$	1,43	1,250	1,79	divergiert	passt

Anhang D
Kommentierte Leseliste

Eine Handvoll Arbeiten trägt alles, was in diesem Buch steht. Wir führen sie hier mit einzeiligen Anmerkungen auf, damit Sie selbst entscheiden können, wohin Ihre Lektüre als Nächstes gehen soll.

Die Keimarbeit. **M. C. Wurm,** *Operational scale detection in quantum magnetism via susceptibility analysis* (Wurm, 2026). AVS Quantum Science **8**, 013804 (2026). DOI 10.1116/5.0312410. Der begutachtete empirische Anker des Frameworks; Kapitel 21 ist eine Leseanleitung dazu.

Klassischer Kontext. **H. E. Stanley,** *Introduction to Phase Transitions and Critical Phenomena* (Stanley, 1971). Das Lehrbuch, von dem das Wort „Suszeptibilität“ sein gesamtes Erbe bezieht; nur nötig, wer den Ursprüngen in der Gleichgewichts-Statistikmechanik nachgehen will.

L. Onsager, *Crystal Statistics, I* (Onsager, 1944). Die exakte Lösung des 2D-Ising-Modells von 1944, die uns $T_c = 2J/\ln(1+\sqrt{2})$ liefert — der Maßstab, an dem Kapitel 22 kalibriert ist.

J. M. Kosterlitz & D. J. Thouless, *Ordering, metastability and phase transitions in two-dimensional systems* (Kosterlitz and Thouless, 1973). Eine Ergänzung zur Suszeptibilitätssicht: topologische Übergänge ohne divergierendes χ.

Information und Fisher. **R. Landauer,** *Information is Physical* (Landauer, 1991). Der einseitige Artikel, dem die $\log_2 D$-Verbindung in Kapitel 31 ihre thermodynamische Schärfe verdankt.

C. W. Helstrom, *Quantum Detection and Estimation Theory* (Helstrom, 1976). Quelle für die Fisher-Schranke auf der Suszeptibilität, wie sie im Fisher-Information-Kapitel von Teil VI verwendet wird.

S. L. Braunstein & C. M. Caves, *Statistical distance and the geometry of quantum states* (Braunstein and Caves, 1994). Die Arbeit von 1994, die die QFI mit der Differentialgeometrie des Zustandsraums verbindet und so der Identität $\chi \approx \sqrt{F_Q}$ des Frameworks ihre rigorose Grundlage verschafft.

Hardware-Ära. **J. Preskill,** *Quantum computing in the NISQ era and beyond* (Preskill, 2018). Liefert den Kontext, warum die in Kapitel 21 detektierte operative Skala überhaupt von Bedeutung ist.

Methodik. **B. Efron,** *Bootstrap methods: another look at the jackknife* (Efron, 1979). Die ursprüngliche Bootstrap-Arbeit, Grundlage für Kapitel 8.

A. Savitzky & M. J. E. Golay, *Smoothing and differentiation of data by simplified least squares procedures* (Savitzky and Golay, 1964). Die Arbeit von 1964, der die Savitzky–Golay-Ableitungsoption des Frameworks ihren Namen verdankt.

S. Williams, A. Waterman & D. Patterson, *Roofline* (Williams et al., 2009). Die Arbeit aus dem Jahr 2009, die die Performance-Decke aus Kapitel 26 definiert hat.

Konkrete Domänen. **B. Gutenberg & C. F. Richter,** *Frequency of earthquakes in California* (Gutenberg and Richter, 1944); **F. Omori,** *On the aftershocks of earthquakes* (Omori, 1894). Die beiden empirischen Gesetze, auf denen Kapitel 24 aufbaut.

G. D. Nastrom & K. S. Gage, *A climatology of atmospheric wavenumber spectra of wind and temperature observed by commercial aircraft* (Nastrom and Gage, 1985). Der Datensatz hinter dem Mesoskalen-Übergang in Kapitel 25.

L. N. Smith, *Cyclical learning rates for training neural networks* (Smith, 2017); **S. McCandlish et al.,** *An empirical model of large-batch training* (McCandlish et al., 2018). Referenzen für den LR-Range-Test und das Phänomen der kritischen Batchgröße in Kapitel 27.

J. C. Lagarias, *The 3x+1 problem and its generalizations* (Lagarias, 1985); **T. Tao,** *Almost all orbits of the Collatz map attain almost bounded values* (Tao, 2022). Hintergrund und Stand der Forschung zu Kapitel 30.

Das Framework selbst. **ForgottenForge,** `sigma-c-framework` v3.0.0 (ForgottenForge, 2026). Die begleitende Open-Source-Bibliothek.

Literaturverzeichnis

S. L. Braunstein and C. M. Caves. Statistical distance and the geometry of quantum states. *Physical Review Letters*, 72(22):3439, 1994.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 7(1): 1–26, 1979.

ForgottenForge. sigma-c-framework v3.0.0: Universal criticality analysis and active control. https://pypi.org/project/sigma-c-framework/, 2026.

B. Gutenberg and C. F. Richter. Frequency of earthquakes in california. *Bulletin of the Seismological Society of America*, 34(4):185–188, 1944.

Carl W. Helstrom. *Quantum Detection and Estimation Theory*. Academic Press, 1976.

J. M. Kosterlitz and D. J. Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C: Solid State Physics*, 6(7):1181–1203, 1973.

Jeffrey C. Lagarias. The 3x+1 problem and its generalizations. *American Mathematical Monthly*, 92(1):3–23, 1985.

Rolf Landauer. Information is physical. *Physics Today*, 44(5):23–29, 1991.

Sam McCandlish, Jared Kaplan, and Dario Amodei. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

G. D. Nastrom and K. S. Gage. A climatology of atmospheric wavenumber spectra of wind and temperature observed by commercial aircraft. *Journal of the Atmospheric Sciences*, 42 (9):950–960, 1985.

F. Omori. On the aftershocks of earthquakes. *Journal of the College of Science, Imperial University of Tokyo*, 7:111–200, 1894.

Lars Onsager. Crystal statistics. I. a two-dimensional model with an order-disorder transition. *Physical Review*, 65:117–149, 1944.

John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. doi: 10.22331/q-2018-08-06-79.

Abraham Savitzky and Marcel J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

Leslie N. Smith. Cyclical learning rates for training neural networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

H. Eugene Stanley. *Introduction to Phase Transitions and Critical Phenomena*. Oxford University Press, 1971.

Terence Tao. Almost all orbits of the Collatz map attain almost bounded values. *Forum of Mathematics, Pi*, 10:e12, 2022.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

M. C. Wurm. Operational scale detection in quantum magnetism via susceptibility analysis: Critical-like behavior at the quantum-classical crossover on nisq hardware. *AVS Quantum Science*, 8(1):013804, 2026. doi: 10.1116/5.0312410.

Stichwortverzeichnis

Anhang E
Reproduzierbarkeit

Sämtliche in Teil IV zitierten Zahlen stammen aus den entsprechenden Dateien des Release v3.0.0 von `sigma-c-framework`. Um das Wurm-2026-Ergebnis aus den Rohdaten zu reproduzieren, genügen drei Zeilen:

```
git clone https://github.com/forgottenforge/magneto
cd magneto
python magnetvali.py --experiment E3 --bootstrap 1000
# Ausgabe: gamma_c = 0.6737 +/- 0.036, kappa = 8.58
```

Für alle zwölf qn+c-Abbildungen:

```
from sigma_c import Universe
nt = Universe.number_theory(map_type='collatz')
print(nt.verify_twelve_predictions(M=12))
```

Anhang F

Wo man die Kapitel-9-Daten bekommt

Kapitel 9 führt das Rezept an drei Datensätzen durch: an der 2D-Ising-Magnetisierung, den S&P-500-Renditen des Jahres 2008 und dem südkalifornischen Erdbebenkatalog rund um die Ridgecrest-Serie von 2019. Wer einen davon zu Hause reproduzieren möchte, findet hier den Einstieg. Der gesamte Zeitaufwand liegt bei zehn bis zwanzig Minuten pro Datensatz, ohne GPU.

F.1 Curie-Punkt (Ising-Magnetisierung)

Die Ising-Daten in Kapitel 9 wurden *simuliert*, nicht heruntergeladen: der Metropolis-Monte-Carlo aus Kapitel 22 läuft bei jeder der zehn Temperaturen einmal. *Die Daten erzeugen Sie sich also selbst.*

- **Code**: die Funktion `ising_mc` aus Abschnitt 22, etwa 20 Zeilen NumPy.
- **Rechenzeit**: ein Laptop fährt den vollständigen Sweep bei $L = 16$ in rund neunzig Sekunden durch.
- **Excel-Alternative**: nicht praktikabel — die Monte-Carlo-Schleife verlangt eine echte Programmiersprache. Wer noch kein Python installiert hat, holt es sich von `python.org`; das kostet nichts und dauert zehn Minuten.

F.2 S&P-500-Tagesrenditen (das 2008er Beispiel)

- **Quelle**: Yahoo Finance, Ticker `^GSPC`. Kostenlos, kein Login.
- **Python-Einzeiler**:

```
import yfinance as yf
df = yf.download('^GSPC', start='2008-01-01', end='2009-06-30')
```

- **Excel**: Bei `finance.yahoo.com` anmelden, nach `^GSPC` suchen, auf „Historical Data“ klicken, den Datumsbereich wählen und auf „Download“ klicken. Sie erhalten ein CSV mit Date, Open, High, Low, Close, Adj Close, Volume. In Excel berechnen Sie `=LN(Close_heute) - LN(Close_gestern)` für die Log-Renditen; das ist die Spalte, auf der das Rezept aus Abschnitt 9.2 arbeitet.
- **Hinweis**: Derselbe Workflow funktioniert mit jedem beliebigen Ticker; probieren Sie zum Vergleich `BTC-USD` oder `GC=F` (Gold).

F.3 Südkalifornischer Erdbebenkatalog (Ridgecrest)

- **Quelle**: SCEC Earthquake Data Center unter `scec.org/research-tools/downloadable-catalogs`. Wähle den *Hauksson-Katalog*, Datumsbereich 2018-01 bis 2021-06, Magnituden-Schwelle 2,5.
- **Format**: Tabulator-separierte Textdatei mit Spalten date, time, latitude, longitude, depth, magnitude.
- **Python-Laden**:

```
import pandas as pd
cat = pd.read_csv('hauksson.txt', sep='\s+', skiprows=1,
                  names=['date','time','lat','lon','depth','mag'])
```

- **Excel**: Öffnen Sie die Datei direkt und teilen Sie sie mit „Text in Spalten“ auf. Die b-Wert-Berechnung lautet `=0.4343/(MITTELWERT(F:F)-MIN(F:F))`, wobei Spalte F die Magnitude in einem gefilterten Sechsmonatsfenster ist. Für jeden Fenstermittelpunkt wiederholen, um Abschnitt 9.3 zu reproduzieren.

Falls Sie noch nie Python installiert haben. Gehen Sie zu `python.org/downloads` und installieren Sie die neueste 3.x-Version. Öffnen Sie anschließend ein Terminal und führen Sie `pip install numpy scipy yfinance pandas matplotlib` aus. Damit ist jeder Codeschnipsel dieses Buches abgedeckt. Sollte trotzdem etwas schiefgehen, hält die README des Repositorys `sigma-c-framework` Ersatzanleitungen für fünf Betriebssysteme bereit.

Falls Sie gar nichts installieren möchten. `kaggle.com`, `colab.research.google.com` und `deepnote.com` bieten Python direkt in einem Browser-Tab, mit den obigen Paketen bereits vorinstalliert. Der Gratis-Tarif reicht für jedes Kapitel des Buches aus — abgesehen von einem echten Hardware-Braket-Lauf.

Anhang G
Danksagung

Dieses Kompendium gäbe es nicht ohne die Vorarbeit, die im Repository `sigma-c-framework` (v3.0.0) dokumentiert ist, nicht ohne die empirischen Daten, die auf den Rigetti-Quantenprozessoren Ankaa-3 und Cepheus-1 erhoben wurden, und nicht ohne das Begutachtungsverfahren, das die Methodik in ihre heutige Form geschliffen hat.

Die Gutachter, namentlich. Drei Stimmen haben dieses Manuskript über zwei Revisionsrunden geprägt. *Sabine*, Programmlektorin eines bayerischen Universitätsverlags, hat mich den langen Titel beerdigen und die Karte mit den sieben Symbolen an ihren Platz rücken lassen. *Linus*, sechzehn und dann siebzehn, sagte mir, Teil III sei unlesbar, bis die Kaffeetasse zurückkomme — und hatte recht. *T.P.*, der anonyme Theoretiker, dessen zwei Briefe in einer Schreibtischschublade liegen, hat vier Vermutungen ausdrücklich benannt, die das Manuskript nicht einräumen wollte. Wenn irgendetwas in diesem Buch ehrlich klingt, geht das Verdienst zu vieren auf. Die Fehler sind meine.

Dank an meine Familie für die endlose Inspiration.

Anhang H
Kolophon

Titel. *Der Gipfel: ein universelles Rezept für Phasenübergänge, in zwölf Welten.* Der Titel wurde über zwei Manuskriptrunden und zwölf Monate hinweg auf den Punkt gebracht. Frühere Entwürfe liefen unter einer längeren Arbeitsbeschreibung; mit dieser Ausgabe wird sie zur Ruhe gelegt und nirgends sonst weiter aufbewahrt.

Satz. Gesetzt in LaTeX mit `lmodern`. Fließtext in Latin Modern Roman, 11 pt; Abschnittsüberschriften in Latin Modern Sans Serif, Kapitelüberschriften kursiv. Abbildungen in TikZ. Code in Latin Modern Mono über das `listings`-Paket bei aktivem `upquote`.

Lizenz. Open Source unter der GNU Affero General Public License, Version 3 oder später (AGPL-3.0-or-later). Kommerzielle Lizenzen ohne die AGPL-Verpflichtung — insbesondere für Closed-Source-Derivate, die das begleitende Framework einbinden — sind über nfo@forgottenforge.xyz erhältlich. Der Lizenztext der AGPL-3.0 ist im Quellrepository des Frameworks wörtlich wiedergegeben.

Reproduzierbarkeit. Jedes numerische Ergebnis dieses Buches lässt sich aus dem Release v3.0.0 von `sigma-c-framework` reproduzieren. Durchgerechnete Beispiele, die auf Quantenhardware-Daten beruhen, sind an einen konkreten Commit des Repositorys `magneto` gebunden. Die getaggten Git-Revisionen beider Repositorys stehen in Anhang E.

Errata. Errata werden unter `https://forgottenforge.xyz/compendium/errata` gepflegt und in spätere Drucke eingearbeitet. Korrekturen bitte an nfo@forgottenforge.xyz mit dem Betreff `[compendium errata]`.

Auflage und Kontakt. Erste Auflage, Mai 2026, Buckenhof in Deutschland. Der Autor ist unter der obigen Adresse erreichbar; Fragen jeder Tiefe sind willkommen, die meisten werden innerhalb einer Woche beantwortet.

Klappentext (gedruckte Ausgabe). Für Rezensentinnen und Bibliothekskataloge folgt hier der Klappentext der gedruckten Ausgabe im Wortlaut.

Ein Rezept. Drei Zeilen Python, die ein System mit einem verstellbaren Knopf und einer messbaren Antwort nehmen — und dir sagen, wo es kippt.

Dieselben drei Zeilen lokalisieren den Curie-Punkt von Eisen bei ungefähr $1043\,K$ *(Kapitel 22), das Volatilitätsregime, das in der Woche nachgab, in der Lehman Brothers im August 2008 fiel (Kapitel 23), und den Abfall des b-Werts im südkalifornischen Erdbebenkatalog, der der Ridgecrest-Serie im Juli 2019 vorausging (Kapitel 24). Zwölf Welten, ein Rezept.*

Ein Lehrbuch, das mit Arithmetik beginnt und mit veröffentlichungsreifer Diagnostik endet. Wir versprechen keine Revolution. Wir versprechen einen guten Butler.