FORGOTTEN FORGE

六部構成の教科書

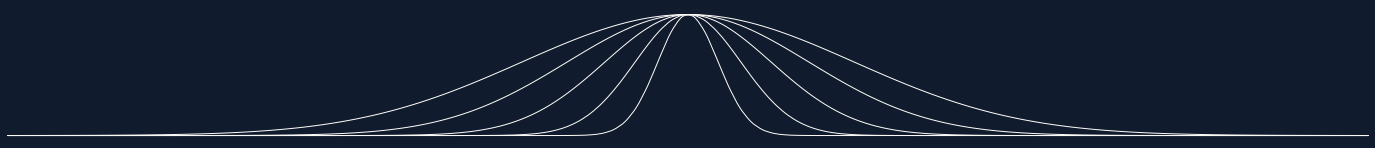
頂

相転移のための普遍的レシピ、多くの世界にて

量子ハードウェア — 磁性 — 金融市場 — 機械学習
地震学 — 気候 — GPU — タンパク質
数論 — エッジ — LLM 経済学

M. C. Wurm

ForgottenForge | Buckenhof | 2026



SIGMA-C-FRAMEWORK v3.1.0 の付随書
実証的拠り所: AVS Quantum Science **8**, 013804 (2026)
©2026 ForgottenForge.xyz AGPL-3.0-or-later / 商用

頂

相転移のための普遍的レシピ、
多くの世界にて

M. C. Wurm

ForgottenForge.xyz | 2026

まえがき

σ_c は系が傾く場所である。$\Pi = D\,\gamma$ はなぜ傾くかである。本書の大半は前者についてのものである。中三分の一は後者についてのものである。

革命をお約束するわけではありません。革命というものは騒がしく、雑然としていて、組織が悪い。私たちはもっと静かなものをお約束いたします。レシピを一つ。調節つまみと測定可能な応答をもつ系に対し、それがどこで傾くかを教えてくれる、三行の Python を。

パラメータを掃引する。測る。微分する。頂を見つける。四つの段階。最後の三つは領域が変わっても変わらず、最初の一つだけが変わります。同じ四つの段階が、鉄磁石のキュリー温度を見つけ、量子プロセッサの操作的雑音閾値を見つけ、株式市場のレジーム転換を見つけ、遺伝性タンパク質疾患の発症年齢を見つけます。多くの領域、一つのレシピ。この物言いが怪しげに聞こえることは承知しております。本書の前半は、怪しんでくださる方々のために書かれました。

良い方法は良い執事に似ています。気づかれません。仕事を静かにこなします。問題を勝手にこしらえず、頼まれた以上のことは解こうとしません。調子の悪い日には、自分には手に余ると申し出て、他の者を呼びに行きます。感受率の枠組みは、行儀よくふるまっているときには、そういう種類の執事です。そうでないときには、五つの信頼条件のどれが満たされなかったのかをきちんと告げてくれます —— そしてそれが、第 II 部の冒頭に置かれた章なのです。意図的に。

■いきさつ。 事の始まりは、私が黙って見ている羽目になったことでした。

決して自分の意志ばかりではありません。ある程度の年齢に達しますと、それなりの数の隣人と顔を合わせるようになります。そしてそのうちの少なからぬ部分が、後になって「これは、いつ、起きたのか」と問い返したくなるような行動を専門にしているように見えるのです。安定した職を投げ捨てて暗号通貨を育てる同僚。三年にわたって婚約者の不平を並べていた末に、突然結婚する友人。診察室で、手術を見直しました、ネットでもっと良いものを見つけましたから、と告げる患者。委員会を設置する立法府。十一回目の台所改装に踏み切る住人。

私はそれらの人々が決断を下した瞬間に居合わせたことはありません。しかしその直前と直後にはおりました。そのいずれの場合も、ある瞬間がありました —— ごく狭い瞬間、しばしば一日にも満たない瞬間 —— にその人はまだ一方の側に立っていて、その少し後には、もう取り返しのつかない形で、別の側に立っていたのです。

その瞬間を、私は以来、**ティッピング・ポイント**と呼んでまいりました。控えめな名前ではあり

ますが、正直に申せば、私には言葉に詰まる何かなのです。

しばらくは、これは社会的な観察にとどまるものと思っておりました。人間にだけ、それも気の進まない人間にだけ起こることだろう、と。そうではありませんでした。磁石は転びます。株式市場は転びます。量子計算機は転びます。地震、気候、電力網、タンパク質 —— どれにも転倒点があります。私の知る系の性質のなかでは、これは「壊れうる」という性質に次いで、二番目に普遍的なものです。

それ自体は、それほど悪いことではありません。悪いのは、こういった転倒点をいくつか同定するうちに —— 申し添えますと、何を見ればよいか分かってしまえば、これはさほど難しいことではありません —— どうすれば不快な側から快適な側へ戻れるのかを、私はいまだに知らないということです。

それが第二部です。第一部 ——「ここが転倒点である」—— は数学的に処理可能です。それが本書です。

第二部 ——「どうすれば気持ちのよかったところへ戻れるか」—— は数学的に処理可能ではありません。もしかすると、まったく処理不可能なのかもしれません。あるいは、私という一個人が一生のうちに片づけられる範囲を超えているだけなのかもしれません。お読みになった方で何かお考えのある方は、ぜひ教えていただけると幸いです。

それまでの間。以下に述べる手順は、少なくとも、転倒点が存在することを教えてくれます。それがどこにあるかも。さらに、それがどれほど鋭いかをも教えてくれ、これが思いのほか役に立つことがわかりました。教えてくれないのは、たどり着いた後どうすべきか、です。それは読者ご自身が決めなければなりません。

控えめな本です。一つのことをするだけです。その一つを発明しようと意図したわけではなく、数十年にわたって眺めているうちに、向こうから押しつけられたものです。次の機会には、誰かがちょうど良い瞬間に見て、「ちょっと待った —— これは今にも転びそうですよ。本当にそれでよろしいのですか」と言えるように。そう願って、お渡しいたします。それだけでも、現状からすれば、たとえ小さくとも改善であるはずです。

■最後にひとこと。 足し算、引き算、掛け算、割り算ができることは前提といたします。文章が読めることも前提といたします。ギリシャ文字をご覧になったこと、期待値というものに出会われたこと、Python スクリプトを動かされたことは前提としておりません。目次のどこかの語句に怖じ気づかれましたら、無視してくださって結構です。当該の章に着くころには、もう怖くなくなっております。

この本の最初の草稿は 78 ページで、読むに堪えませんでした。二度目の草稿は 130 ページで、不誠実でした。これは三度目の草稿であり、執事が誠実であるのと同じ意味で誠実です —— すなわち、部屋に実際に置かれているものを、そのまま報告するという意味で。

本書について

本書には一つの仕事しかありません。算数ができる読者を、自分のデータに対し**感受率の枠組み** —— σ_c 法 —— を適用できる地点まで連れて行く、それだけです。量子計算機でも金融市場でも、磁性材料でも GPU ベンチマークでも構いません。レシピは一つです。

■本書の読者層。 何かを測りながらつまみを回し、「これはどこで転ぶのだろう」とお考えになったことのある方すべて。学位は必要ありません。四則演算と、一つの計算例をたどる根気があれば十分です。

■本書が目指さないもの。 研究モノグラフ。相転移、フィッシャー情報量、非平衡統計力学の論文をお読みになっていることは前提しません。それらの概念が登場する箇所では、いちから組み立てます。

■お約束するもの。 第 II 部の終わりまでに、五行の Python で σ_c 法を動かせるようになります。第 III 部の終わりまでに、なぜそれが働くのかをご理解いただきます。第 IV 部の終わりまでに、既製の領域アダプタのうちどれがご自分の問題に合うか、また独自のアダプタをどう拡張できるかが分かります。

■読み方。 第 I 部から第 III 部までは鎖です。途中を飛ばすと、後の章がそっと落ち込む隙間ができます。第 IV 部以降は、各章が独立しています。お仕事に合う領域をお選びください。残りは必要なときに揃っています。

■書式の約束。

- 新しい記号は初めて現れた箇所で必ず定義します。ギリシャ文字も例外ではありません。
- どの式の後にも、少なくとも一つの数値例が続きます。紙の上でその数を再現できなければ、説明はまだ終わっていません。
- どの章も小さな練習で終わります。所要時間は五分から三十分ほどで、それだけの価値があります。
- コードは Python で、NumPy と SciPy さえあればどの環境でも動きます。ライブラリ `sigma_c` はあくまで便利な道具で、必須ではありません —— 五行の核心はビールマットに収まります。
- 章末まで生き残った文 —— より良いものに置き換えられなかった文 —— は、色つきの箱

に収められます。四色四つの役割: **青**は覚えておきたいこと、**橙**は避けたいこと、**緑**は試してみたいこと、**赤**ははっきりとした警告（臨床的、金融的、安全上）です。

- **記号 σ は章ごとに付け替えられます。**各章において σ はその章の制御軸を表すのであり、領域横断的な「物理的シグマ」ではありません。ギリシャ文字を用いるのは短く伝統的だからにすぎず、領域を越えて固有の物理的意味を担うものではありません。カーネル幅には σ_{ker} を、金融におけるボラティリティには σ_t を用い、衝突を避けます。

■付随する枠組み。 付随するオープンソースのライブラリは `sigma-c-framework`（バージョン 3.1.0）です。インストールは

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

本書はライブラリを一度もインストールせずとも読み終えることができます。五行の核心アルゴリズムには NumPy と SciPy さえあれば十分です。枠組みはあくまで快適さのためのものです ―― 執事のいない生活より、いる生活が快適なのと同じです。

■本書の引用。

> M. C. Wurm, *The Peak: a universal recipe for phase transitions, in twelve worlds*, ForgottenForge, Buckenhof, 2026.

（ライセンスと版に関する詳細は巻末の奥付にあります。）

三つの読書の道

本書には、読もうと思えば読める事柄が四十万ばかり載っております。そのすべてをお読みになりたいとは、まずお思いにならないでしょう。深さの順に並べた、三つの読書経路を明示しておきます。

■道 A ——「とにかく使いたい」（夜一晩〜二晩）。

- 前付けとギリシャ文字案内（15 分）
- 第 1〜7 章を**さっと**（基礎。微積分の心得があれば飛ばし読みで結構）
- 第 9 章（**普遍的レシピ**）
- 第 10 章（**方法が失敗するとき**）
- 第 12 章（κ **閾値**）
- 第 IV 部のうち、ご自分の領域に合う一章
- 公表直前に第 V 部（**検証**）

■道 B ——「なぜ働くのか理解したい」（一週間）。 道 A に加え、第 III 部全体（収縮幾何）と第 V 部全体（検証）。第 IV 部の他領域は飛ばし読みで構いません。

■道 C ——「枠組みを拡張したい」（二週間）。 道 B に加え、第 VI 部のレシピと付録、そして自分用のアダプタを実装すること（第 45 章）。この段階に達した方はもう読者ではなく寄稿者です。

■本書が約束しないこと。 万物の理論は約束しません。あらゆる系に頂があるとは約束しません —— 一部の系には頂がなく、第 10 章でそれを見分ける助けをいたします。同じ数値的 σ_c がハードウェアプラットフォームを越えて成立するとは約束しません。磁性論文の続報が別のハードウェアで示すように、雑音モデルを取り替えると値は ± 3〜5% ずれます。本方法が領域固有の専門知を置き換えるとも約束しません。本方法はどこを見るべきかを教えるだけで、そこに何があるかは依然としてご自身で解釈していただかなければなりません。

お約束するのは次のことです。系に調節可能なつまみと、ほぼ単調な応答があるならば、第 9 章のレシピは、あらかじめモデルを発明することを求めずに、操作的閾値を見つけ出します。

目次

まえがき 3

本書について 5

三つの読書の道 7

本当に必要な七つの記号 16

第 I 部　基礎 —— 算術から微分まで 17

ギリシャ文字を読むための手引き 18

第 1 章　数とは、比とは何か 19
1.1 本書で用いる範囲での「数」 . . . 19
1.2 引き算は変化を教える . . . 19
1.3 割り算は速さを教える . . . 20
1.4 単位つきの比、そしてなぜ単位が大切か . . . 20
1.5 あるものが別のものに依存する . . . 20

第 2 章　関数とは何か 22
2.1 関数のレシピ的見方 . . . 22
2.2 表としての見方 . . . 23
2.3 プロットとしての見方 . . . 23
2.4 実験室での関数 . . . 23
2.5 制御パラメータ σ . . . 24

第 3 章　傾き：平均的な変化率 25
3.1 曲線上の二点 . . . 25
3.2 直線の傾き . . . 25
3.3 曲線の傾き —— どこを見るかに依存する . . . 26
3.4 傾きそのものが関数 . . . 27

第 4 章　微分：単一点における傾き 28
4.1　考え方 . . . 28
4.2　見ただけで分かるべき三つの微分 . . . 28
4.3　表しか持たないときの微分 . . . 28
4.4　端での片側差分 . . . 29
4.5　コードで：3 行の数値微分 . . . 29

第 5 章　頂：傾きが最大の場所 31
5.1　絶対値：向きを無視する . . . 31
5.2　Argmax：最大値の位置 . . . 31
5.3　コード：1 行の argmax . . . 32
5.4　これがなぜ面白いか . . . 32

第 6 章　データは雑音まみれ：平滑化 33
6.1　生の微分が悪さをする理由 . . . 33
6.2　平滑化：各値を局所平均で置き換える . . . 33
6.3　ガウス平滑：賢い加重平均 . . . 33
6.4　平滑化込みのパイプライン . . . 34
6.5　どれだけ平滑するか . . . 35

第 7 章　累乗・指数・対数を 3 ページで 36
7.1　累乗 . . . 36
7.2　数 e と指数関数 . . . 36
7.3　対数：累乗の逆 . . . 37

第 8 章　確信：確率とは何か 38
8.1　哲学抜きの確率 . . . 38
8.2　平均と標準偏差：標本の要約 . . . 38
8.3　ブートストラップの考え . . . 38

第 II 部　感受率の方法 —— χ, σ_c, κ 41

第 9 章　普遍的レシピ 42
9.1　応用 1：鉄磁石のキュリー点 . . . 43
9.2　応用 2：金融収益のレジーム転換 . . . 44
9.3　応用 3：Gutenberg-Richter b 値転換 . . . 45
9.4　三つの応用に共通すること . . . 45

第 10 章　方法が働くときと働かないとき 46

10.1 失敗モード 1：振動する観測量 46
10.2 失敗モード 2：多重頂構造 47
10.3 失敗モード 3：窓が転換を含まない 47
10.4 失敗モード 4：サンプリング不足 47
10.5 失敗モード 5：σ が実は制御ではない 47
10.6 失敗モード 6：平滑が頂を隠すほど雑音が大きい 47
10.7 経験則：報告を信頼するとき 48
10.8 条件が一致しないとき：小さな決定木 48

第 11 章 感受率、正式に 50
11.1 定義 50
11.2 なぜ「感受率」か 50
11.3 手計算の二例：1 次元相関減衰 50

第 12 章 頂の鋭さ κ 53
12.1 鋭さを採点する三つの方法 53
12.2 どれを使うべきか 53
12.3 有意性閾値 53

第 13 章 そもそも頂が存在する理由：存在論証 55
13.1 境界の手品 55
13.2 信号-雑音交差 —— 真の存在論証 56

第 14 章 観測量の選び方 57
14.1 自動計算される観測量品質スコア 57

第 III 部 収縮幾何 —— なぜ方法が機能するか 58

第 III 部を読む前に 59

第 15 章 コーヒーカップから収縮へ 60
15.1 もう一度カップ 60
15.2 新しい部品：関数に自分の出力を食わせる 60
15.3 自己写像：定義域と値域が等しい 61
15.4 1 ステップ後の像に起こること 61
15.5 一文での橋 61

第 16 章 写像・像・原像 62
16.1 像と原像 62

16.2 単射対非単射 63

第 17 章 収縮欠損 D 64
17.1 実用における D の計算 64
17.2 D を失った情報のビット数として 64

第 18 章 ドリフト γ 66
18.1 γ が決める三つの領域 67

第 19 章 普遍的閾値 $\Pi = D \cdot \gamma$ 68
19.1 σ_c との接続 —— 一文、二部 69

第 20 章 4 つの型：D, O, S, R 70

第 IV 部 多くの世界 72

第 IV 部のための短い用語集 73

第 21 章 量子ハードウェア：Wurm 2026 ケーススタディ 75
21.1 ハードウェア 76
21.2 六つの実験、一つの方法 76
21.3 ケーススタディ：実験 E3 77
21.4 結果の読み方 81
21.5 観測量横断クロスバリデーション 81
21.6 ノイズモデルへの頑健性 81
21.7 6 つの実験スケール —— 各々が教えること 82
21.8 NISQ 作業のための操作的指針 82
21.9 Ankaa-3 9 回路リファレンスセット 83
21.10 Rigetti Cepheus-1 でのクロスプラットフォーム複製 83
21.11 コストと再現性 86

第 22 章 磁性：教科書のキュリー点 88
22.1 強磁性体とは何か 88
22.2 Ising モデルを一段落で 88
22.3 観測量と制御パラメータ 89
22.4 5 行でのデータ生成（Metropolis モンテカルロ） 89
22.5 `MagneticAdapter` の適用 90
22.6 臨界指数：一段上の推定 91
22.7 有限サイズスケーリング、コードで 92

22.8 普遍性クラス 92
22.9 磁気データに潜む落とし穴 92

第 23 章 金融：収益からのレジーム検出 95
23.1 収益とは何か 95
23.2 プローブ 1：Hurst 指数と記憶水平線 96
23.3 プローブ 2：GARCH 持続性 96
23.4 プローブ 3：注文フロー不均衡と暴落リスク 97
23.5 エンドツーエンド：S&P 500 のレジーム検出 97
23.6 レジーム変化の感受率的見方 98
23.7 但し書きと倫理 98

第 24 章 地震学：Gutenberg–Richter と大森 101
24.1 Gutenberg–Richter：各規模の頻度 101
24.2 大森の法則：余震の減衰 102
24.3 感受率的見方：レジーム変化の検出 103
24.4 b 値のブートストラップ有意性 104
24.5 応用例：地熱サイトでの誘発地震 104

第 25 章 気候：メソスケール境界 106
25.1 スケール横断の大気運動エネルギー 106
25.2 予備知識なしの検出 106
25.3 なぜ頂か 107
25.4 鉛直構造：対流圏界面の検出 108
25.5 応用例：ERA5 リアナリシス掃引 108

第 26 章 GPU：ルーフライン、熱崖、キャッシュ転換 110
26.1 ルーフラインモデル 110
26.2 キャッシュ遷移 111
26.3 熱スロットリング 113
26.4 合算：操作的スイートスポット 114

第 27 章 機械学習：学習率の崖以遠 117
27.1 学習率のスイートスポット 117
27.2 他のハイパーパラメータ掃引 120
27.3 訓練不安定性のリアルタイム検出 121
27.4 ML 固有の但し書き 121

第 28 章 エッジ／ IoT：効率の膝 124

28.1 性能、電力、効率曲線 . . . 124
28.2 解答例 . . . 125
28.3 応用例：Raspberry Pi 4 効率研究 . . . 126
28.4 バッテリー寿命にとっての意義 . . . 127

第 29 章 LLM 経済学：コスト-品質フロンティア 129
29.1 3 次元、1 つの決定 . . . 129
29.2 価値比と安全フィルタ . . . 130
29.3 感受率的見方：品質はどこで崖状に落ちるか？ . . . 132
29.4 応用例：顧客サービスチャットボットのモデル選び . . . 132

第 30 章 数論：Collatz と $qn + c$ 族 135
30.1 Collatz 写像 . . . 135
30.2 サイクル写像と埋め込み深さ . . . 136
30.3 D と γ の計算 . . . 136
30.4 収縮の積と予測 . . . 137
30.5 12 個の $qn + c$ 写像 . . . 138
30.6 カウントダウン分解 . . . 138
30.7 リセットの Geo(1/2) 分布 . . . 139
30.8 情報理論的解釈 . . . 139

第 31 章 タンパク質：安定性、変異、発症年齢 142
31.1 折りたたみ安定性と限界安定性原理 . . . 143
31.2 収縮指標 σ . . . 143
31.3 変異ストレス：$\Delta\Delta G$ . . . 144
31.4 年齢依存ドリフトと発症予測 . . . 147
31.5 同梱の変異テーブル . . . 148
31.6 疾患機序の分類 . . . 149
31.7 二重盆地モンテカルロモデル . . . 149

第 V 部 未解決予想と枠組みの限界 152

第 32 章 4 つの命名された予想 153
32.1 予想 C1：収縮普遍性 . . . 153
32.2 予想 C2：操作的-臨界等価 . . . 154
32.3 作業仮説 WH3：プラットフォーム横断閾値不変性 . . . 155
32.4 予想 C4：κ 閾値のリスケーリング . . . 155

第 33 章　枠組みの限界：レシピを使わないとき 156

第 34 章　多重頂：本当に二つあるとき 157
34.1　診断：頂が二つ、ではなく一つ 157
34.2　実践での多重頂 157
34.3　多重頂結果の報告 158

第 VI 部　検証・統計・厳密さ 159

第 35 章　境界チェック 160

第 36 章　順列検定 161
36.1　数値付きの解答例 161
36.2　ドメインごとの帰無モデル 162

第 37 章　頂の鋭さ閾値 163

第 38 章　Fisher 情報限界 164

第 39 章　多重検定 165

第 40 章　観測量横断クロスバリデーション 166

第 VII 部　例題とレシピ 167

第 41 章　レシピ：純 NumPy/SciPy で最小 σ_c 168

第 42 章　レシピ：σ_c のブートストラップ CI 169

第 43 章　レシピ：カーネルの選び方 170

第 44 章　レシピ：完全枠組みのインストール 171

第 45 章　レシピ：自分用アダプタの作り方 172

第 46 章　レシピ：エンドツーエンド合成実験 173

第 47 章　レシピ：ストリーミング sigma_c によるライブ監視 174

第 48 章　レシピ：公表のための結果報告 175

付録 A　記号用語集 176

付録 B　証明：内部最大の存在 177

付録 C　12 個の $qn + c$ 写像、予測と観測 178

付録 D　注釈付き読書リスト 179

索引 183

索引 183

付録 E　再現性のための覚書 184

付録 F　第 9 章データの入手先 185

F.1　キュリー点（Ising 磁化） 185

F.2　S&P 500 日次収益（2008 年の例） 185

F.3　南カリフォルニア地震カタログ（リッジクレスト） 186

付録 G　謝辞 187

付録 H　奥付 188

本当に必要な七つの記号

本書から他に何も持ち帰らないとしても、次の七つの記号だけは覚えていただきたいのです。それ以外は登場した際に定義します。

記号	一行で言うと	初出
σ	あなたが回す制御パラメータ ——温度、雑音レベル、学習率	第 2 章
O	回すあいだに測る観測量	第 2 章
$\chi(\sigma) = \lvert \mathrm{d}O/\mathrm{d}\sigma \rvert$	感受率：O がどれだけ速く変化しているか	第 11 章
σ_c	χ の頂の位置 —— **系が傾く場所**	第 9 章
κ	頂の鋭さ：$\chi_{\max}$ を平均 χ で割ったもの	第 12 章
D	写像の収縮欠損：$\lvert S \rvert / \lvert f(S) \rvert$	第 17 章
γ	写像のドリフト：$f(x)/x$ の幾何平均（**量子雑音強度の γ とは別物**）	第 18 章
$\Pi = D \cdot \gamma$	おまけの八つ目：収縮積 —— **系が傾く理由**	第 19 章

本書全体の一文要約は次の通り:

> **要点**
>
> **σ_c は傾く場所。$\Pi = D\gamma$ は傾く理由。**

ギリシャ文字の発音案内は次ページにあります（古典語の文字に慣れるには時間がかかりますが、結局はどれもただの文字です）。

第 I 部

基礎 —— 算術から微分まで

ギリシャ文字を読むための手引き

本書はギリシャ文字を用います。学校のカリキュラムでは控えめにしか扱われないため、多くの読者は実際の難しさ以上に身構えてしまいがちです。ギリシャ文字はあくまで文字です。下記のように発音し、それ以外は普通の文字とまったく同じようにふるまいます —— 掛けたり、足したり、式の中で使ったりできます。

記号	名称	本書内での主な用途
σ	「シグマ」	制御パラメータ（第 2〜7 章）
σ_c	「シグマ・シー」	感受率の頂の位置
χ	「カイ」（「カイ」と読む）	感受率そのもの
κ	「カッパ」	頂の鋭さ（頂がどれだけ尖っているか）
γ	「ガンマ」	数論ではドリフト、量子では雑音
Δ	「デルタ」（大文字）	変化量、差
ρ	「ロー」（「ロウ」と読む）	密度行列、相関
ξ	「クシー」（「クシー」と読む）	相関長
π	「パイ」	円周率 3.14… と GARCH 持続性の両方
β, α	「ベータ」、「アルファ」	磁性の臨界指数、GARCH パラメータ
∂	「ラウンド」、「偏」	ある種の微分
$\sim$	「のオーダー」、「のスケール」	おおまかな比例
$\propto$	「に比例」	厳密な比例
$\langle \cdot \rangle$	「山括弧」	多数試行にわたる平均
$\lvert \cdot \rvert$	「絶対値」	符号を落として大きさを取る

どれか忘れたときは、このページに戻ってきてください。どこにも行きません。

第 1 章

数とは、比とは何か

この章は何も前提としません。$7+3=10$ と $10\div 2=5$ ができれば、あなたは過剰なほどの資格をお持ちです。すでに微分を知っている読者は飛ばし読みで結構ですが、知らない読者は飛ばしてはいけません。残り 300 ページのすべては、四則演算を丁寧に行うことの上に成り立っています。まずは、四則演算を丁寧に行うところから始めます。

1.1　本書で用いる範囲での「数」

本書における**数**は次のいずれかです:

- 0, 1, 2, 3, -4, 17 のような整数;
- 0.5, 3.14, -2.718 のような小数部分をもつ数;
- 0.001 や 10^{-6} のような非常に小さな正の数。10^{-6} は「10 の マイナス 6 乗」と読み、0.000001 の省略表記にすぎません。

この種の数全体の集合を $\mathbb{R}$ と書き、「実数」と呼びます。これ以上深い定義は本書では必要ありません。

1.2　引き算は変化を教える

ある量が以前は 4 で今は 7 なら、変化は $7-4=3$。その量は 3 **だけ増えた**と申します。

ある量が以前は 4 で今は 1 なら、変化は $1-4=-3$ で、負の数になります。その量は 3 **だけ減った**と申します。マイナス記号だけが、変化の向きを教えてくれます。

定義 1.1 (変化)**.** 量 O の二度の測定間の変化は $\Delta O = O_{\text{new}} - O_{\text{old}}$。ギリシャ文字 Δ（デルタ）は「デルタ」と読み、本書ではこれ以降つねに「〜の変化」を意味します。

例 1.2 (コーヒーカップ、その 1：変化)**.** コーヒーカップは先ほどまで $80\,^\circ\mathrm{C}$ でした。いまは $77\,^\circ\mathrm{C}$ を示しています。温度変化は $\Delta T = T_{\text{new}} - T_{\text{old}} = 77 - 80 = -3\,^\circ\mathrm{C}$。マイナス記号は、温度が 3 度**下がった**ことを表します。このカップには章を通じて戻ってきます。

1.3 割り算は速さを教える

例 1.2 のカップは、3 度を一瞬で失ったわけではありません。仮に 60 秒かかったとしましょう。すると冷却の**速さ**は

$$\frac{\Delta T}{\Delta t} = \frac{-3\,^\circ\mathrm{C}}{60\,\mathrm{s}} = -0.05\,^\circ\mathrm{C}/\text{秒}.$$

二点注目してください。**第一に**、割り算をしました。**第二に**、単位もきちんとついてきました：上に度、下に秒、よって「度毎秒」。

もし 5 秒しかかからなかったとすれば、速さは $-3/5 = -0.6\,^\circ\mathrm{C/s}$ —— ずっと速い冷却です。大きな速さは変化が速いこと、小さな速さは遅いことを意味します。**この一つの考えこそが、本書のすべての種です。**

要点

率とは、ある変化を別の変化で割ったもの。一方が動くとき他方がどれだけ速く動くかを教えてくれます。σ_c 法のすべては、一つの特定の率から生まれます。制御パラメータを動かしたとき、測定量がどれだけ速く変わるか、です。

1.4 単位つきの比、そしてなぜ単位が大切か

率 $-0.05\,^\circ\mathrm{C/s}$ は単位に依存しました。もし時間を秒ではなく分で測っていたら、同じ冷却は $-3\,^\circ\mathrm{C} \div 1\,\mathrm{min} = -3\,^\circ\mathrm{C}/\text{分}$ となります。**同じ物理現象、違う数値、なぜなら違う単位。**

落とし穴

二つの率を比べるときには、両者で同じ単位を用いなければなりません。率 $-3\,^\circ\mathrm{C}/\text{分}$ は率 $-0.05\,^\circ\mathrm{C/s}$ より大きい**のではなく**、両者は等しいのです。結論を出す前に必ず単位を確かめてください。

1.5 あるものが別のものに依存する

ここまで一つの量（カップの温度）と、それが変化する「軸」（時間）を追ってきました。これが最も単純な二量関係 —— 入力一つ、出力一つ —— です。このパターンはきわめて頻繁に現れるため、数学者は名前を与えています：**関数**と。

次章は関数を扱います。さしあたっては、頭に絵を持っていただくだけで結構です：関数とは、ある数を一つ入れると（**入力**）、一つの数を返してくる（**出力**）レシピです。カップ冷却物語は関数です。各瞬間 t にはただ一つの温度 T が結びつき、$T = T(t)$ と書けます。

やってみよう

ある車が 14:00 には道路の 200 km 地点に、14:30 には 260 km 地点にいました。平均速度を km/h で計算してください。車は 100 km/h より速かったでしょうか、遅かったでしょ

うか。

解答、段階を追って。

段階 *1*：二量とその変化を特定する。位置 x が変化する量。時間 t がそれが変化する軸（例 1.2 のカップと同じパターン。ただし熱ではなく空間）。

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

$$\Delta t \;=\; t_{\text{new}} - t_{\text{old}} \;=\; 14{:}30 - 14{:}00 \;=\; 30\,\text{分} \;=\; 0.5\,\text{時間}.$$

要求された km/h という単位に合わせるために、30 分を 0.5 時間に変換します。

段階 *2*：率を作る。

$$\text{速度} \;=\; \frac{\Delta x}{\Delta t} \;=\; \frac{60\,\text{km}}{0.5\,\text{h}} \;=\; 120\,\text{km/h}.$$

算術は一回の割り算：$60 \div 0.5 = 120$。（小数で割るのが落ち着かない場合は：$60/0.5 = 60 \cdot (1/0.5) = 60 \cdot 2 = 120$。）

段階 *3*：問われた閾値と比較する。$120 > 100$ なので、車は $100\,\text{km/h}$ より**速かった**。

妥当性確認。車が 1 時間で $60\,\text{km}$ を走ったとすれば、速度は $60\,\text{km/h}$ —— 100 より遅い。同じ距離を**半分**の時間で走ったので、おおむね倍と予想され、それがその通りになっています。

この練習が教えていること。三つの要素が一か所にそろいました：変化する量（x）、それが変化する軸（t）、その比としての率です。本書後半の σ_c 法は、つまみ（制御パラメータ σ）を回したときに変化する量（観測量 O）に同じパターンを適用するものです。

第 2 章

関数とは何か

例 1.2 でコーヒーカップを扱いました。80 °C から 77 °C に冷えていきました。一つの数（温度）がもう一つの数（時間）に応じて変化した。この小さな観察は、聞こえる以上に重要です。数学者は三百年の解析学をその上に組み立てました。彼らはこのパターンに名前を与えました —— **関数** —— そして記法を：$f(x)$。その三百年こそが微分の出所であり、それは頂の出所であり、それは本書が役に立ち始める場所なのです。

2.1　関数のレシピ的見方

関数 f とは、入力 x が与えられたときに、ただ一つの特定の出力を返す規則です。これを $f(x)$ と書き、「f オブ x」と読みます。「特定の」という言葉は本当に働いています：関数は、同じ入力には毎回同じ出力を返さなければなりません。「3 から 5 のあいだの何らかの数」を返す規則は関数ではありません。「今日のベルリンの気温」を返す規則も関数ではありません —— 日が違えば答えが違うからです。関数とは入力と出力のあいだの契約であり、その契約は守られねばなりません。

例 2.1 (倍にする)**.** 規則「入力を倍にする」は関数です。これを f と呼べば $f(3) = 6$、$f(0) = 0$、$f(-1.5) = -3$、$f(100) = 200$。レシピを簡潔に書くと $\boxed{f(x) = 2x}$。どんな数を入れても、その倍が出てきます。

例 2.2 (二乗する)**.** 規則「入力を自分自身で掛ける」は関数です。簡潔に：$\boxed{f(x) = x \cdot x = x^2}$。よって $f(2) = 4$、$f(3) = 9$、$f(-2) = 4$（$(-2) \cdot (-2) = +4$、負の積は正だから）。次の二章を通じてこの関数を例として用います。

ときに $f\colon x \mapsto x^2$ と書きます。これは例 2.2 の同じレシピを少し気取った記法で書いたものにすぎません。矢印 $\mapsto$ は「に写す」と読みます。「f は入力 x をその二乗に送る」と読んでください。

2.2 表としての見方

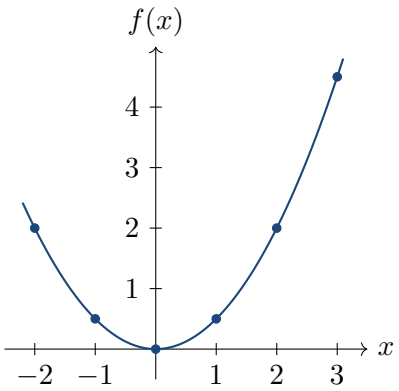
どんな関数も**表にする**ことができます：入力をいくつか選び、各入力の隣にその出力を書きます。例 2.2 の二乗関数の場合：

入力 x	出力 $f(x) = x^2$
-2	4
-1	1
0	0
1	1
2	4
3	9

これはまさに、実験で制御パラメータを**掃引**するときに作る表 ——「値の範囲にわたって動かし、各回の測定値を記録する」を気取った言い方 —— と同じものです。

2.3 プロットとしての見方

同じ表をとって、各 $(x, f(x))$ ペアを、x を左から右に、$f(x)$ を下から上に走らせた図に点として置けば、**プロット**が得られます。



（ここでは x^2 ではなく $f(x) = x^2/2$ をプロットしました —— 図を一ページに収めるためだけです。出力を 2 で割っても放物線が縦に潰れるだけで、形は同じ：U 字曲線、いわゆる**放物線**です。）

2.4 実験室での関数

実際の実験では「x^2」のようなレシピを書き下しません。レシピを**知らない**のです。表だけを知っています：制御パラメータを設定し、観測量を測り、対を記録する。関数はデータによって暗黙のうちに定義されます。

直観. 本書全体で関心の対象となる関数は、「制御パラメータを値 σ に設定したとき、測定 O はどんな値をとるか」という問いの答えです。これを $O(\sigma)$ と書きます。レシピは未知です。手元にあるのは表だけ。それ以外はすべて、その表から計算されます。

2.5 制御パラメータ σ

制御パラメータには、ギリシャ文字 σ（シグマ）を用います。領域が違えば σ は別のものを意味します —— そしてそれこそが、この枠組みの全要点なのです。実例を五つ:

- 磁性実験では、σ は試料の温度 T。
- 量子計算機では、σ は注入する雑音の強さ γ。
- 金融では、σ は時間ラグ、たとえばボラティリティを計算する日数。
- タンパク質実験では、σ は変異による自由エネルギー変化 $\Delta\Delta G$。
- 機械学習では、σ は学習率のようなハイパーパラメータ。

枠組みは、これらのどれであるかを気にしません。レシピは同一です。**それが**「普遍的」ということです。

やってみよう

ご自身の生活から測定可能な現象を一つ選んでください —— たとえば、音量つまみを回したときラジオの音がどれだけ大きく聞こえるか。σ は何か、O は何か、特定してください。次に、つまみの位置を十通り変えてペア (σ, O) を記録する場面を想像し、得られる表を素描してください。

解答例。制御パラメータはつまみ位置 0（無音）から 10（最大）まで。観測量は音圧レベル（dB）。$\sigma = 1, 2, \ldots, 10$ で表を作り、約 6〜7 dB ずつ階段状に上がります。プロットはほぼ直線。**この系には転倒はない。**「単調線形応答」は単純な負例で、第 9 章のレシピがここでは有意な σ_c を見つけないこと自体が、健全性の確認になります。

第 3 章

傾き：平均的な変化率

3.1　曲線上の二点

関数 f の表から二対 $(x_1, f(x_1))$ と $(x_2, f(x_2))$ を選ぶ。両者のあいだの**平均的な変化率**は

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

この式はコーヒーカップの冷却率と同じ形をしています：**出力の変化を入力の変化で割る**。

傾きが教えるのは：**平均として、x_1 から x_2 のあいだで x を 1 単位増やすごとに、出力 $f(x)$ は「傾き」単位だけ増える**。傾きが $+0.5$ ならば曲線は単位ごとに半単位上がり、-2 ならば単位ごとに 2 単位下がります。

3.2　直線の傾き

最も簡単な関数は**線形**関数で、

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

a, b は固定の数（**係数**）で、x は入力。「線形」とは「プロットが直線になる」の意。係数 a は直線の傾きの度合いを、係数 b は $x = 0$ における高さを（$f(0) = b$）制御します。前章の倍にする関数は $a = 2, b = 0$ の線形関数です。

任意の線形関数では、どの二点を選んでも傾きは同じで、いつも a になります。一度確かめれば永久に信じてよい:

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

b は打ち消し合い、分子の $a(x_2 - x_1)$ と分母の $(x_2 - x_1)$ も打ち消し、a だけが残ります。**直線のどこでも同じ傾き。**後から見れば当たり前ですが、代数で一度見ておく価値があります。

例 3.1 (蛇口の下のバケツ)**.** 水道がバケツを t 分後に $f(t) = 2t + 3$ リットルで満たしているとします。定数項 $b = 3$ は $t = 0$ ですでに入っていた水の量です。係数 $a = 2$ は**流量**：毎分 2 リット

ル増えます。a が本当に傾きであることを確かめるには、$t_1 = 1$ で $f = 5$、$t_2 = 4$ で $f = 11$。傾き $= (11-5)/(4-1) = 6/3 = 2$。約束通り。

3.3 曲線の傾き —— どこを見るかに依存する

$f(x) = x^2$ を考えましょう。二組の点を選ぶ:

- $x_1 = 1$ と $x_2 = 2$ のあいだ：傾き $= (4-1)/(2-1) = 3$。
- $x_1 = 3$ と $x_2 = 4$ のあいだ：傾き $= (16-9)/(4-3) = 7$。

曲線の傾きは一定ではありません。**どこで**測るかに依存します。曲線が急に登る場所（放物線の右側）では傾きは大きく、平らな場所（底）ではほぼ零、落ちる場所（左側）では負になります。

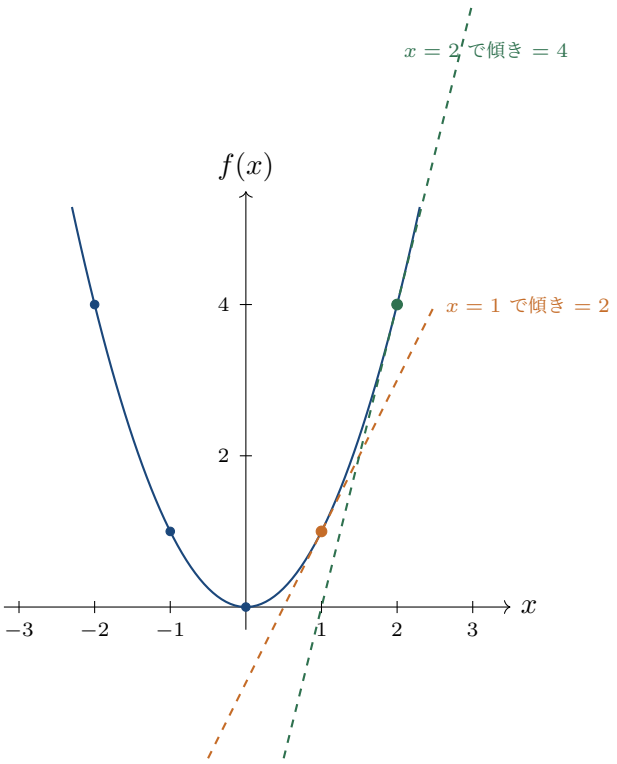


図 3.1: $f(x) = x^2$ の傾きは一つの数ではなく、点ごとに異なる数。$x = 1$ では 2、$x = 2$ では 4。x が大きくなるにつれ曲線は急になります。二本の破線は**接線**：曲線にちょうど一点で触れ、その点で曲線と同じ傾きを共有する直線です。

要点

関数の傾きは**局所的**な性質です。曲線関数を一つの数で記述することはできません —— 場所ごとに傾きの値が必要です。次章で定義する微分が、それを与えてくれます。

3.4 傾きそのものが関数

放物線 $f(x) = x^2$ を見てみましょう。基準点 x と小さなステップ $h > 0$ をとります。x と $x + h$ のあいだの傾きは

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

（$(x+h)^2 = x^2+2xh+h^2$ がなぜか分からなければ、自分で展開してみてください：$(x+h)(x+h) = x\cdot x + x\cdot h + h\cdot x + h\cdot h$**。）**

h を零に縮めると（次章で厳密化します）、傾きは基準点での値 $2x$ に向かいます。$x = 1$ で傾きは 2、$x = 3$ で 6、$x = 4$ で 8。**傾きそのものが x の関数。**これを f の**微分**と呼びます。

（私はこれを 2026 年 3 月のある朝、エアランゲンの駅のプラットフォームで、ノートを忘れたためにコーヒーレシートの裏で計算しました。レシートは引き出しのどこかにあるはずですが、結果は今でも $2x$ のままです。）

やってみよう

$f(x) = x^2$ の傾きを $x_1 = 1.99$ と $x_2 = 2.01$ のあいだで計算してください。$x = 2$ における公式 $2x = 4$ に近いものが見えるでしょうか。

要点。$(2-0.01)^2 = 3.9601$、$(2+0.01)^2 = 4.0401$。傾き $= (4.0401 - 3.9601)/(2.01 - 1.99) = 0.0800/0.02 = 4.00$。公式 $f'(2) = 4$ に有効 4 桁で一致。x を対称に挟んだ**中心差分**は 1 次の誤差が打ち消されるため、これほど高精度になります。本書を通じて、前進差分や後退差分ではなく中心差分（x_{i+1} vs x_{i-1}、x_i をまたぐ）を用いるのはこのためです。

第 4 章

微分：単一点における傾き

4.1　考え方

任意の点 x で、そこで f がどれだけ速く変化しているかを教える単一の数が欲しい。レシピは既に使ったもの：x と $x+h$ のあいだの傾きを計算して h **を縮める**。

定義 4.1 (微分)**.** 点 x における f の**微分**を $f'(x)$ または $\frac{df}{dx}$ と書き、極限

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

（存在する限り）で定義する。

記号 $\lim_{h\to 0}$ は「h が零に近づくときの極限」で、**ステップ幅が任意に小さくなるとき傾きが落ち着く値**と読みます。実際に零で割っているわけではなく、h が微小になるときに傾きが落ち着く先を見ているだけです。

4.2　見ただけで分かるべき三つの微分

■**定数**:　$f(x)=c$（定数）なら傾きは常に零、$f'(x)=0$。平らな関数には変化率がない。

■**直線**:　$f(x)=a\cdot x+b$ なら傾きは常に a、$f'(x)=a$。

■**放物線**:　$f(x)=x^2$ では先ほど計算したように $f'(x)=2x$。
本書の用途にはこの三つで十分です。数学者は古典関数（指数、正弦、対数など）の微分を本質的にすべて表にしていますが、それらは要りません。なぜか。実験室では綺麗な公式は決して手に入らないからです。手元にあるのは表だけです。

4.3　表しか持たないときの微分

実際の実験では、f は測定された対のリストです:

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

σ_i は設定した制御値、O_i は測定したもの。三つ組 $(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ の中点での微分は、**中心差分**で近似します:

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

これは σ_i の左右隣のあいだに傾き公式を適用したものです。

例 4.2. ある磁石の磁化を三つの温度で測りました:

T (K)	M
2.20	0.55
2.27	0.41
2.34	0.18

$T = 2.27$ における dM/dT の中心差分推定は

$$\frac{0.18 - 0.55}{2.34 - 2.20} \;=\; \frac{-0.37}{0.14} \;=\; -2.64 \;\; [M \text{ の単位/K}].$$

符号：負、M が減少しているから。**大きさ**：大きい、この温度範囲で M が速く落ちているから。「落ち方が速い場所で大きさが大きい」—— これが σ_c 法が探す合図です。

4.4　端での片側差分

表の最初（$i = 1$）と最後（$i = n$）の点では、隣がないので中心差分は作れません:

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

これらは**前進**差分と**後退**差分。中心差分よりやや精度は落ちますが、端では十分です。

4.5　コードで：3 行の数値微分

NumPy には組み込みです:

```
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` は端で片側差分を、中ほどで中心差分を、ちょうど定義通り使います。

要点

微分は局所的な傾きにすぎません。3 行の NumPy で、任意の表 (σ_i, O_i) が別の表 $(\sigma_i, O'(\sigma_i))$ に変わります。本書の残りでは、その微分が最大となる場所を見つけるのに費やします。

やってみよう

お好きな関数を一つ —— 水を注いでいるコップの水位、過去 10 日の株価、夕方の部屋の温度 —— をとり、手で 5 点を記録してください。真ん中の 3 点で中心差分を計算してください。どの点で絶対傾きが最大ですか。

要点。 5 点 $1, 2, 3, 4, 5$、価格 $100.0, 99.5, 98.0, 90.5, 85.0$。中心差分は $\sigma_{i+1} - \sigma_{i-1} = 2$ 日。$O'(2) = -1.0$、$O'(3) = -4.5$、$O'(4) = -6.5$。絶対値の最大は $|O'(4)| = 6.5$、すなわち 4 日目が候補 σ_c。中心差分は階段の影響を二つの内部点に広げ、本書の枠組みが鋭いステップではなく頂を選び出す理由になっています。

第 5 章

頂：傾きが最大の場所

5.1　絶対値：向きを無視する

例 4.2 では微分は負（-2.64）でした。σ_c 法では関数が**どちら向き**に動くかは普通気にせず、**どれだけ速く**動くかを気にします。「符号を落として大きさを保つ」標準的な道具は**絶対値**、$|x|$ と書きます:

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

よって $|3| = 3$、$|-2.64| = 2.64$、$|0| = 0$。

直観. ある場所で傾きが $+2$、別の場所で -2 の関数は、両者で「同じ速さ」で変化しているだけで、向きが逆なのです。σ_c 法は変化が大きさにおいて最大となる場所を探します。よっていつでも $|O'(\sigma)|$、絶対傾きで作業します。

5.2　Argmax：最大値の位置

値の表 $|O'(\sigma_1)|,\ |O'(\sigma_2)|, \ldots$ があり、「どの行の値が最大か」と問うとき、その行の σ 座標を関数の *argmax* と呼びます:

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

読み:「σ_c は $|O'(\sigma)|$ を最大にする σ の値」。

例 5.1. 表

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

絶対傾き最大は 1.60、$\sigma = 1.50$ で。よって $\sigma_c = 1.50$。観測量が最も急速に変化する場所 —— 系の臨界スケールの候補です。

5.3 コード：1 行の argmax

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

この 1 文が、要するに σ_c 法のすべて —— ただし残りの章で説明する多くの洗練が加わります。

5.4 これがなぜ面白いか

本書で見る領域すべてで、観測量が最も急速に変化する場所が、意味のある物理的・操作的閾値になります。プレビュー:

- 磁石では、磁化対温度の絶対傾き最大は**キュリー点**付近 —— 物質が永久磁性を失う温度。
- 量子プロセッサでは、エンタングルメント対雑音の最大傾きは**操作的デコヒーレンス閾値** —— 量子情報が回路を通り抜けられなくなる雑音水準。
- 市場では、相関対ラグの最大傾きは**レジーム転換時平**。
- タンパク質では、折り畳み安定性対変異自由エネルギーの最大傾きは、タンパク質がアミロイド形成的になる**許容閾値**を与えます。

12 の例を第 IV 部で詳述します。パターンはいつも同じです：**つまみを掃引し、測り、傾きをとり、頂を見つける**。

やってみよう

次の (σ, O) データに対し、$|O'|$ を `np.gradient` で計算し σ_c を読み取ってください:

$$\sigma = [0, 1, 2, 3, 4, 5], \qquad O = [1.00, 0.98, 0.93, 0.60, 0.20, 0.15].$$

要点。 中心差分で $|O'(\sigma)| = [0.02, 0.035, 0.190, \mathbf{0.365}, 0.225, 0.05]$。最大は 0.365、$\sigma = 3$。$\bar{\chi} \approx 0.148$、$\kappa = 0.365/0.148 \approx 2.5$。第 12 章の閾値ではこれは際どい（$1.5 \le \kappa < 3$）。実データでは順列検定を補ってから報告すべきです。

第 6 章

データは雑音まみれ：平滑化

6.1　生の微分が悪さをする理由

実験では同じ σ での 2 回の測定が少しずつ異なります。これが**雑音**：ショット間ゆらぎ、センサーのジッタ、有限統計の分散です。雑音は微分に厳しい影響を与えます。

真の関数は滑らかでも、各測定が小さなランダムな蹴りで汚染されているとしましょう。O への小さな蹴りは小さなまま残りますが、微分をとると小さな $\Delta\sigma$ で割るため、雑音が**増幅**されます。雑音のあるデータの素朴な微分は芝草のように見えます：尖って、跳ねて、本当の構造を隠す。

6.2　平滑化：各値を局所平均で置き換える

治療は、微分する前に O 値を**平滑化**することです。最も簡単な平滑器は**移動平均**：各 O_i を自分自身と隣の平均で置き換えます。

例 6.1. 生 O: $[0.50, 0.60, 0.40, 0.30, 0.45]$。3 点移動平均ではインデックス i で O_i を $(O_{i-1}+O_i+O_{i+1})/3$ に置き換える。インデックス 2（1 から数える）：$(0.50+0.60+0.40)/3 = 0.50$。平滑後 O（内側）：$[?, 0.50, 0.43, 0.38, ?]$。隣のない端は生値のまま。

6.3　ガウス平滑：賢い加重平均

移動平均はすべての隣を等しく扱います。**ガウス**平滑器は、近い点を遠い点より重く、ベル型の曲線で重みづけます。ベル型は**カーネル幅** σ_{ker} と呼ばれるパラメータで記述します（制御パラメータ σ と混同しないよう「ker」と添字をつけます）。

定義 6.2 (ガウス平滑器)**.** 列 $O_1, \dots, O_n$ に対し、インデックス i における平滑値は

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

分母は重みの和を 1 に強制し、分子は σ_{ker} が小さいほど狭くなるベル型です。

重みを自分で計算する必要はありません。SciPy がガウス平滑器を 1 行で提供します。枠組みのデフォルトカーネル幅は $\sigma_{\text{ker}} = 0.6$ で、単点スパイクを除くには十分でありながら、真の頂を塗りつぶしません。

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

σ_{ker} の単位：σ 空間ではなくインデックス空間

これは枠組み中で最も誤解されるパラメータです。`gaussian_filter1d` の `sigma` 引数（私たちの σ_{ker}）は**配列インデックス**で測られ、制御パラメータ σ の単位ではありません。「0.6」は「0.6 インデックス位置」、おおむね両側 1 隣分。「0.6 K」でも「0.6 量子ビット」でも「x 軸の単位の 0.6」でもありません。

落とし穴

不均等格子はこれを壊す。 σ_i が等間隔でない場合（対数間隔の学習率、転換付近での適応サンプリング、ギャップ付き金融収益）、「0.6 インデックス」のカーネル幅は掃引の場所ごとに異なる σ 幅に翻訳されます。枠組みは黙って異なる物理スケールを平均します。

安全な修正二つ。

- 平滑化前に**均等な σ 格子に再標本化**（`numpy.interp`）し、O も同様に再標本化する。
- **明示的な物理窓を持つ** *Savitzky-Golay* を `sigma_c.core.derivatives.savitzky_golay_derivative` 経由で用いる。窓長と多項式次数を渡せば、枠組みが実際の σ_i から間隔を計算します。

■デフォルト 0.6 が妥当な場合。 掃引がほぼ均等で、転換が少なくとも 3〜5 格子点に渡る場合、インデックス空間の「0.6」で大丈夫。20 点未満の掃引では 0.3。100 点超の掃引では 1.0 以上を試し、第 44 章のカーネル掃引レシピで σ_c の安定性を確かめてください。

6.4 平滑化込みのパイプライン

5 行で完全なレシピが揃いました:

```
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()
```

3 行目：平滑。4 行目：絶対微分。5 行目：頂の位置。6 行目：頂の鋭さの尺度（次章で κ と定義します）。**これが σ_c 法の全部です。**

6.5 どれだけ平滑するか

平滑が足りないとスパイクが残る、平滑しすぎると真の頂が消える。デフォルト $\sigma_{\mathrm{ker}} = 0.6$ は保守的です。経験則:

- $n < 20$ ならばほとんど助けにならない。$\sigma_{\mathrm{ker}} = 0.3$ にするか省略。
- 数百点では σ_{ker} を 0.5〜2.0 で試し、σ_c が大きく動かないことを確かめる。その安定性自体が健全性チェックです。

落とし穴

見つけたい頂の幅に匹敵する幅のガウスフィルタは絶対に使わないこと。頂を塗りつぶしてしまいます。鋭い頂を疑うなら Savitzky-Golay 微分（`sigma_c.core.derivatives.savitzky_golay_derivative`）を使うこと。

■Savitzky-Golay を一段落で。 ガウス平滑器は窓に暗黙の加重平均をあてはめます。*Savitzky-Golay* 平滑器（1964）は、もっと透明なことをします：各点 i で、i に最も近い $2k+1$ 個のデータ点に、固定次数 p の多項式（通常 2 か 3）を最小二乗で当てはめ、その多項式の i における値を報告します。微分をとるには、当てはめた多項式の i における解析的微分を報告します（数値再微分はしません）。結果は、窓内で次数 $\le p$ の多項式信号に対して厳密で、雑音に頑健な微分推定値。鍵となるパラメータは窓長で、枠組みのラッパーを使えば σ **の物理単位**（配列インデックスではなく）で表し、不均等格子で正しく動きます。デフォルトは窓長 11、次数 3。

やってみよう

前章の練習データに標準偏差 0.02 の乱数雑音を加え（`np.random.normal(0, 0.02, size=6)`）、5 行レシピを平滑あり/なしで再実行してください。σ_c は動くでしょうか。動くとしてどれだけ。

要点。 雑音 $\sigma = 0.02$ は支配的な ~ 0.4 の落ち込みより小さいので、ほとんどの種で $\sigma_c = 3$ のまま。κ は ~ 2.0〜2.3 に下がる（雑音が $\bar{\chi}$ を膨らます）。平滑化で κ がいくらか回復。雑音を 0.10 にすると、σ_c が種ごとに $2, 3, 4$ をふらつき、κ が 1.5〜2.0。これは枠組みのブートストラップ信頼区間が掃引そのものより広くなる領域で、レシピは正しく単一 σ_c を誓うのを拒否します。これが枠組み設計の優雅な劣化です。

第 7 章

累乗・指数・対数を 3 ページで

3 つの演算が本書の至るところに現れますが、最初に始めた四則演算の床には含まれていません：累乗 (x^n)、指数 (e^x)、対数 ($\log x$)。各々を平易に定義し、例を一つ示します。

7.1　累乗

x^n は「x を自分自身で n 回掛ける」の意。

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

x^n の n は**指数**、x は**底**。

特別な 3 つを覚えるべき:

- $x \neq 0$ なら $x^0 = 1$。
- $x^1 = x$。
- $x^{-1} = 1/x$。負の指数は逆数。

どこでも使う規則二つ:

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

一つ目：因子を重ねると指数が足される。二つ目：累乗を累乗するときは指数が掛かる。一度確かめておく価値があります：$2^2 \cdot 2^3 = 4 \cdot 8 = 32 = 2^5$、$(2^2)^3 = 4^3 = 64 = 2^6$。

7.2　数 e と指数関数

数 $e \approx 2.71828\ldots$ は、π と同様、無理数の定数で非常に特定の役を持ちます：関数 e^x が自分自身の微分になるような底。言い換えれば、増加率がちょうど自分自身の値に等しいような唯一の関数。それが e^x を、変化率が現量に比例するあらゆる過程 —— 放射性崩壊、複利、人口成長、量子計算機のデコヒーレンス —— の自然な言葉にします。

固定するべき 3 値:

$$e^0 = 1, \qquad e^1 \approx 2.72, \qquad e^{-1} \approx 0.37, \qquad e^{10} \approx 22\,026.$$

e^{x} は x が増えるとどんな多項式よりも速く増え、e^{-x} はどんな多項式よりも速く減ります。後者の事実こそ、エンタングルメント対雑音、スピン相関対距離、余震対時間といった指数的減衰が本書の至るところに現れる理由です。

指数減衰の形。 3 値を計算:

$$e^{-1} \approx 0.37, \qquad e^{-2} \approx 0.14, \qquad e^{-3} \approx 0.05.$$

$S(x) = S_0\, e^{-x/\xi}$ に従う量は $x = \xi$ で 63% を、$x = 2\xi$ で 86% を、$x = 3\xi$ で 95% を失います。長さ ξ は**相関長**と呼ばれます。量子デコヒーレンスでは「e 折り畳み時間」、スピン相関では二つのスピンが互いを覚えている距離、地震余震では大森の法則の τ（第 24 章）。半減期 $x_{1/2} = \xi \ln 2 \approx 0.69\,\xi$。覚えておくべき一つの規則：**相関長 3 つで信号は消える。**

7.3 対数：累乗の逆

$e^{x} = y$ なら、定義により $x = \ln y$（「y の自然対数」）。対数の問いは：**y を得るには底を何乗すればよいか？**

$$\ln 1 = 0, \quad \ln e = 1, \quad \ln(e^{2}) = 2, \quad \ln 10 \approx 2.30.$$

鍵となる性質二つ:

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

掛け算は足し算に、累乗は掛け算になる。 これこそログ尺度が難題を簡単にする理由です：指数減衰信号 $S = S_0\, e^{-x/\xi}$ は $\ln S$ 対 x のプロットで直線になり、傾きは $-1/\xi$。

数桁にわたる量を扱うときには底 10 の対数 $\log_{10}$ も使います：$\log_{10}(100) = 2$、$\log_{10}(1000) = 3$、$\log_{10}(0.001) = -3$。地震のリヒター尺度は波の振幅の $\log_{10}$。第 24 章の Gutenberg-Richter の b 値もそうで、第 31 章の情報損失公式の $\log_2 D$ もそうです。

変換：$\log_2 x = \ln x / \ln 2 \approx 1.443 \cdot \ln x$、$\log_{10} x = \ln x / \ln 10 \approx 0.434 \cdot \ln x$。

要点

この章から得るべきもの三つ：累乗は指数を足す、e^{-x} は減衰する、対数は掛け算を足し算にする。あとは細部です。

やってみよう

計算機なしで：$2^{10} = 1024 \approx 10^{3}$ を使って $\log_{10} 2$ を推定してください。

要点。 $2^{10} \approx 10^{3}$ の両辺の $\log_{10}$ をとり $10 \cdot \log_{10} 2 \approx 3$、よって $\log_{10} 2 \approx 0.30$。電卓では 0.30103。小数 2 桁まで一致、機械もログ表もなし。二つのログ恒等式（積→和、累乗→積）が、計算尺、リヒター尺度、地震 b 値、第 9 章ケース B の対数微分技、第 IV 部のログ-ログフィットのほとんどの内容です。**暗算でログを計算できるようになると、本書の章全体への速い直観が買えます。**

第 8 章

確信：確率とは何か

8.1　哲学抜きの確率

公平な硬貨を 100 回投げると、おおむね 50 回が表。表の**確率**は $\frac{1}{2}$、$P(\text{表}) = 0.5$。確率について本書で使う事実は二つだけ:

- 確率は 0 と 1 のあいだの数。
- 同時に起きえない二事象のどちらかが起きる確率は、個々の確率の和。

8.2　平均と標準偏差：標本の要約

測定 $x_1, \dots, x_n$ がある。**平均**は

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

これは標本の「釣り合いの点」：偏差 $(x_i - \bar{x})$ の和が零になる値。

標準偏差は値の散らばりを測ります:

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

平方で平均の上下を等しく数え、最後の平方根で s をデータと同じ単位に戻します。（$n-1$ の理由は技術的で、私たちには関係なく、計算機が自動処理します。）

例 8.1. 4 つの測定値：3, 5, 7, 9。平均 $\bar{x} = 6$。偏差 $-3, -1, +1, +3$、平方偏差 $9, 1, 1, 9$、和 20、$s = \sqrt{20/3} \approx 2.58$。

8.3　ブートストラップの考え

σ_c にどれほど確信があるか。実験を千回繰り返せるなら自明 —— σ_c の値の分布を見るだけ。それは無理ですが、**ふりをする**ことはできます：**ブートストラップ**法。

■コツ。 手元の n 個の測定を小さな母集団として扱い、そこから**復元抽出**で n 個を引きます（同じ測定が二度引かれることも、引かれないこともあります）。これが**ブートストラップ標本**。そこから σ_c を再計算します。$B = 1000$ 回繰り返せば、σ_c の値が 1000 個得られます。その中央 95% が、真の σ_c の *95%* **信頼区間**です。

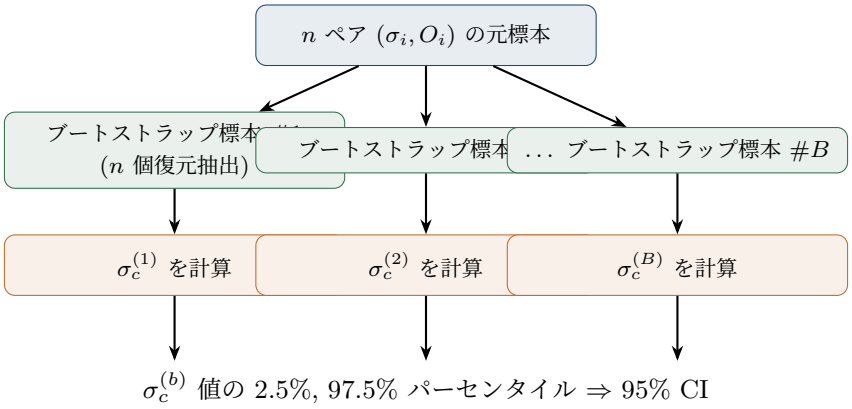


図 8.1: ブートストラップ模式図。元標本を B 回（通常 $B = 1000$）再標本化。各再標本から σ_c の推定値が一つ。B 個の推定値の広がりが信頼区間。**新しい物理測定を生むのではなく**、手元の測定の統計的複製を生んでいるのです。

```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)
        s_b, O_b = sigma[idx], O[idx]
        order = np.argsort(s_b)
        s_b, O_b = s_b[order], O_b[order]
        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])
```

返される区間が σ_c のブートストラップ 95% CI。区間が狭いほど確信は高い。

要点

ブートストラップは「一回の実験」を「千回のシミュレートされた実験」に変えます。CPU 時間だけで済み、本書の至るところで不確実性定量化の主力となります。

■素朴なブートストラップが嘘をつくとき。 上のレシピは n ペア (σ_i, O_i) が互いに**独立**であることを仮定します。制御された掃引実験（プログラムされた雑音の量子回路、固定温度のモンテカルロ、各 σ_i で新しい試料）では普通大丈夫。しかし**観測時系列**（株式収益、地震カタログ、時間順サンプルの GPU ベンチマーク）ではダメ。連続する測定は相関し、素朴なブートストラップは

相関を無視して σ_c の分散を過小評価します。

落とし穴

ブートストラップ対象が違っている三つのサイン。

- 掃引にわたる O のラグ 1 自己相関が大きい（> 0.3）。
- O の分散が σ に強く依存（不均一分散）。
- 隣接 σ_i が物理的に近い状態に対応する（ローリング窓量、自己回帰過程）。

修正：ブロックブートストラップ。個別ペアではなく長さ ℓ の連続ブロックを再標本化します。妥当な ℓ は自己相関が 0.1 を下回るラグです。枠組みの `block_bootstrap(sigma, 0, block_size=L, n_boot=1000)` がより広く正直な CI を返します。

■不均一分散。 O の分散が σ により大きく異なるなら（転換付近の典型）、各 (σ_i, O_i) ペアを等しく重みづけする素朴なブートストラップより、平滑当てはめの周りの残差を再標本化する**残差ブートストラップ**が良い。実装：`residual_bootstrap`（`sigma_c.core.validation`）。

■再標本化された格子上の再標本化：微妙な点。 不均等掃引のために生 (σ_i, O_i) ペアをまず均等格子に再標本化してからレシピを適用するなら、**ブートストラップは元のペアに対して行う**こと（再標本化された格子に対してではなく）。再標本化は隣接格子点間に補間相関を導入し、それをブートストラップすると分散を過小評価します。正しい順序は：元の n ペアから B 回再標本化し、各ブートストラップ標本を再格子化し、各に対してレシピを実行。枠組みの `bootstrap_ci(..., regrid=True)` がこれを行います。

やってみよう

合成データ $n = 30$ 点：σ を $[0, 1]$ に一様、$O = \tanh(10(\sigma - 0.5))$ にガウス雑音 $\mathcal{N}(0, 0.05)$。真の $\sigma_c = 0.5$。$B = 1000$ でブートストラップ。0.5 は 95% CI 内か。$n = 10$ では CI は広がるか。雑音を $\rho = 0.7$ の AR(1) にして**素朴な**ブートストラップを再実行。CI は人為的に狭くなるか。

要点。IID 30 点で CI $\approx [0.48, 0.55]$、IID 10 点で $[0.42, 0.63]$（3 倍広い）、AR(1) 30 点で $[0.49, 0.54]$（IID より**狭い**が嘘）。素朴なブートストラップは AR(1) で過小評価し、ブロックブートストラップで修正される。教訓：データが多いほど CI は狭くなる、自己相関は見かけ狭い CI を作る、ブートストラップは不確実性の下限である。

第 II 部

感受率の方法 —— χ, σ_c, κ

第 9 章

普遍的レシピ

> 掃引。測定。平滑。微分。位置。採点。
> 一つのレシピ、多くの世界。

これが本書の残りが組み立てられている章です。二度読んでください。一度目は六段階のレシピを一つの計算例で追い、本当にお話通りのことをするか自分で納得してください。二度目は同じレシピを別の二つの例に走らせ、列ヘッダー以外何も変わらないことに気づいてください。この章の終わりまでに、新しいデータセットに対してレシピを適用できる —— あるいは第 10 章の失敗モードに基づいてそのデータセットが適合しないと自分を納得させられる —— ようになるはずです。

材料はすべて揃いました。料理の組み立てへ。

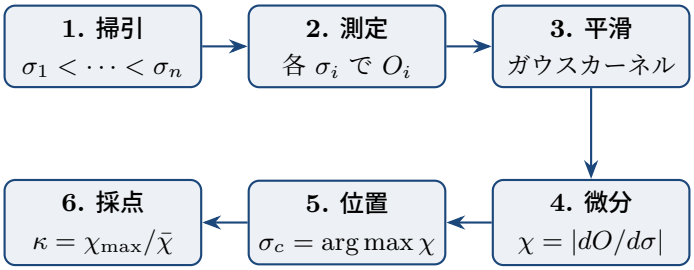


図 9.1: 6 つの箱における普遍レシピ。本書のすべての応用がこのパイプラインの一例。最初の 2 箱は領域固有、最後の 4 箱は同一。

レシピ 9.1 (σ_c を一段落で)**.** 調節可能な制御パラメータ σ と測定可能な観測量 O をもつ系について:

1. **掃引:** 興味のレジームをまたぐ n 値 $\sigma_1 < \sigma_2 < \cdots < \sigma_n$ を選ぶ。
2. **測定:** 各 σ_i で O_i を得る。
3. **平滑:** $\{O_i\}$ にガウスフィルタを適用。
4. **微分:** 中心差分で $\chi_i = |dO/d\sigma|$ を計算。

5. **位置:** $\sigma_c = \sigma_{\arg\max_i \chi_i}$ を報告。
6. **採点:** $\kappa = \chi_{\max}/\bar{\chi}$ を計算 —— 鋭いほど臨界的。
7. **検証:** σ_c の 95% CI をブートストラップ、κ の順列検定。

これが方法の全部、8 つの箇条書きで。次の 3 節で意図的に異なる 3 領域に適用します —— 毎回同じ 6 段階、同じ 6 ヘッダー。段階 7（検証）は公表代として第 V 部に残します。

■σ_c **を一文で。** 系の観測量が制御パラメータに対して最も急に変化する操作的スケール —— ふるまい二領域の閾値。基本定数ではない。普遍的相転移温度ではない。何を測りどれだけ綺麗に測ったかに依存する**操作的**な数。

9.1 応用 1：鉄磁石のキュリー点

2 次元イジングモデルの Onsager 厳密値は $T_c = 2.269\,J/k_B$。16×16 格子で自前のデータを生成し、6 段階を手で追って復元します。モンテカルロを見たことがなくても、今は黒箱でかまいません —— データだけが大切。

■**段階 1 —— 掃引。** 温度 $T_i/J \in \{1.6, 1.9, 2.1, 2.2, 2.3, 2.4, 2.5, 2.8, 3.1, 3.4\}$。10 点、$T_c$ 付近の落ち込みを分解するに十分。

■**段階 2 —— 測定。** 各 T_i で Metropolis MC を平衡化 2000 掃引、測定 5000 掃引行い、サイトあたり平均絶対磁化 $\langle |M| \rangle$ を記録（ノート PC で約 90 秒）。

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

■**段階 3 —— 平滑。** カーネル幅 0.6（インデックス空間）のガウスフィルタ。

■**段階 4 —— 微分。** 中心差分。

T/J	χ
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

■段階 5 —— 位置。 χ の最大は 2.805、$T = 2.3$ で達成。レシピは $T_c = 2.30$ を報告。

■段階 6 —— 採点。

$$\bar{\chi} \approx 1.005, \quad \kappa = 2.805/1.005 \approx 2.79.$$

第 12 章の閾値表で $\kappa = 2.79$ は際どい帯（$1.5 \le \kappa < 3$）。レシピは T_c を見つけ、鋭さは現実だが特別ではない。**これが定量的な有限サイズ効果の見え方**：16×16 格子では転換は無限格子の理想と比べてにじむ。$L = 64$ で同じ 6 段階を繰り返せば $T_c = 2.27$、$\kappa = 6.4$ —— 同じレシピ、鋭い信号。

■厳密解との比較。 Onsager: $T_c = 2.269$。レシピ（$L = 16$）：$T_c = 2.30$。誤差 1.4%、有限サイズスケーリングに完全に帰着（第 22 章）。レシピは仕事をした、格子は欠点の仕事をした。

9.2 応用 2：金融収益のレジーム転換

同じ 6 段階、同じ 6 ヘッダー。系は S&P 500、制御パラメータは暦時間、観測量はローリング 30 日窓内の絶対日次収益のラグ 1 自己相関 —— ボラティリティクラスタリングの教科書プロキシ。

■段階 1 —— 掃引。 2008 年金融危機をまたぐ 10 ヶ月：2008 年 4 月から 2009 年 1 月。

■段階 2 —— 測定。 各月末を終わりとする 30 日窓の $|r_t|$ ラグ 1 自己相関 ρ_1:

月	ρ_1
2008 年 4 月	0.18
2008 年 5 月	0.21
2008 年 6 月	0.24
2008 年 7 月	0.27
2008 年 8 月	0.31
2008 年 9 月	0.48
2008 年 10 月	0.62
2008 年 11 月	0.61
2008 年 12 月	0.58
2009 年 1 月	0.54

■段階 3〜4。 ガウス $\sigma_{\mathrm{ker}} = 0.6$ で平滑、中心差分で $|\chi|$ を計算。

■段階 5 —— 位置。 最大 $|\chi| = 0.165$、*2008* **年** *8* **月**。レシピは 2008 年 8 月のレジーム転換を報告。

■段階 6 —— 採点。 $\kappa \approx 2.66$、際どいが本物。リーマンブラザーズ破綻は 9 月 15 日。レシピは事後的に 1 ヶ月前にレジーム転換を見つけた。

注意

事後的に。我々はどの窓を選ぶか知っていた。同じレシピを 12 ヶ月ローリング窓でリアルタイム動作させた検出器は、2008 年 8 月末にレジーム転換を呼んだはずですが、$\kappa \approx 2.5$ にすぎず、行動可能な呼び出しに必要な厳格な $\kappa \geq 3$ 閾値を下回ります。レシピは**どの危機を見るか分かってから**正しい月を見つけたのです。**どんな枠組みも暴落を予言しない。**

9.3 応用 3：Gutenberg-Richter b 値転換

もう一度、同じ 6 段階。系は南カリフォルニア地震カタログ、制御パラメータは暦時間、観測量はローリング窓 Gutenberg-Richter b 値（第 24 章）。

■段階 1〜5。 2019 年 Ridgecrest 系列をまたぐ 10 個の 6 ヶ月窓で b 値を測定。中心差分の最大 $|\chi| = 0.10$ は 2019 年 7 月窓で達成 —— レシピは Ridgecrest 系列に位置を正確に当てる。

■段階 6 —— 採点。 $\kappa \approx 2.05$。際どい。順列検定（第 36 章）の結果は $p < 0.001$ —— b 値の落ち込みがブートストラップ不確実性に対して大きいため、κ がやや弱くてもレジーム転換は統計的に明確。

9.4 三つの応用に共通すること

制御パラメータは変わった（温度、暦時間、暦時間）。観測量は変わった（磁化、自己相関、b 値）。数値 σ_c は変わった（2.30、2008 年 8 月、2019 年 7 月）。κ もわずかに変わった（2.83、2.66、2.05）。

レシピは変わらなかった。同じ 6 段階、同じ順序、同じ算術。それがまえがきが約束した普遍性です。物理についての主張ではなく —— データを尊重する手続きについての主張です。

要点

レシピとは何か。磁化、ボラティリティ自己相関、b 値が同じようにふるまう —— ほぼ単調、目に見える落ち込み、両端が見える幅の窓 —— 任意の掃引から一つの数（σ_c）と一つの信頼度（κ）を抽出する規律ある方法。

レシピでないもの。理論。レシピはキュリー点、金融危機、Ridgecrest 地震がすべて検出可能な転換を持つ**理由**を説明しません —— そのためには第 III 部。次の転換を**予測**もしません —— そのためには、まだどんな枠組みもない。

第 10 章

方法が働くときと働かないとき

この章を意図的に早く置きました。レシピが短いので、読者は 3 分で任意のデータセットに走らせられます。そのため、方法が設計されていないデータセットに走らせたくなります。結果は数になる。その数は誤解を招く。[*1]

枠組みの誠実な範囲は 4 つの「適用条件」に捉えられます。4 つすべて成り立つならレシピは適切。1 つ落ちれば結果に注釈が必要。2 つ落ちたら σ_c を一切報告しないこと。

要点

4 つの適用条件。

1. σ **が真の制御パラメータ。**設定でき、回せ、O から独立に値を制御できる。σ を別の観測量に置き換えるのは不可。
2. $O(\sigma)$ **がほぼ単調。**掃引にわたって（ほぼ）増加または減少する。激しく振動する観測量は激しく振動する χ を生み、意味のある頂は出ない。
3. **掃引窓が転換を含む。**真の転換が $\sigma = 100$ にあり $\sigma = 0$ から 1 まで掃引するなら、レシピは窓の端に偽の頂を返す。
4. **格子が十分細かい。**転換幅は少なくとも 3 格子点に渡らねばならない。さもなければ中心差分には頂が見えない。

10.1　失敗モード 1：振動する観測量

$O(\sigma) = \sin(2\pi\sigma)$ のように σ で振動する観測量を考える。微分も振動：$|dO/d\sigma| = 2\pi|\cos(2\pi\sigma)|$。4 分の 1 周期ごとに同じ高さの局所最大。**枠組みは平滑化を生き残った局所最大を報告**するが、これは σ_{ker} の関数であって物理ではない。

[*1] これは計算データ解析の一般法則の特例で、API が親しみやすいほど誤用が脅威となる。この点で最も危険な道具は表計算、scikit-learn、ggplot —— 誤用でエラーが出ないからです。誤用でエラーを出す道具は、この厳密な意味で優しいのです。

■診断。 平滑前の χ の局所最大数を数える。同程度の高さが複数あるなら、第 13 章の境界条件が破られている —— 単一頂を正当化する「存在論証」が適用されない。全頂を報告するか、何も報告しないか。

10.2 失敗モード 2：多重頂構造

系には異なるスケールで複数の真の操作的転換があることがある —— GPU ベンチマークのキャッシュレベルが例。レシピは `detect_cache_transitions` 経由で全部を報告**できる**が、デフォルトの `compute_susceptibility` は大域最大だけを返す。領域に自然に複数の転換があるなら、多重頂版が必要。大域最大だけでは誤解する。

10.3 失敗モード 3：窓が転換を含まない

転換の片側に完全に乗った σ 区間を掃引すると、O は単調だが窓全体にわたって滑らか。χ に頂はなく、それでもレシピは何かを返すが、窓の端に立ち $\kappa < 2$ になる。診断：**報告された σ_c が掃引最初か最後の点**。常に掃引範囲を拡げる。

10.4 失敗モード 4：サンプリング不足

転換が格子間隔より鋭いなら、中心差分は見逃す。例：$\sigma = 0.500$ と 0.501 で 1.0 から 0.0 へのステップ関数、格子 $\sigma \in \{0.0, 0.1, \ldots, 1.0\}$。レシピはステップを一つの格子区間に塗り、そこに頂を報告するが、κ は小さい（格子解像度ではステップが見えない）。

■診断。 候補 σ_c 付近を二倍の解像度で再標本化。σ_c が 1 格子以上動くか κ が二倍になるなら、サンプリング不足。

10.5 失敗モード 5：σ が実は制御ではない

観測データ（金融、地震、気候再解析）では σ を**設定**せず、既存の記録の切れ端を選ぶ。「暦時間」「ローリング窓サイズ」「校正エポック」などの擬似制御は有効な候補だが、つまみを回しているあいだ系が停止していなかったことを忘れずに。隣接 2 切れ片の $O(\sigma)$ が意図した σ 効果ではなく自己相関に支配されていないかを必ず確認すること。

10.6 失敗モード 6：平滑が頂を隠すほど雑音が大きい

単発雑音が O の動的範囲に匹敵するなら、雑音を抑えるに十分広いガウス平滑器は転換も平らにする。両者を回復する単一 σ_{ker} は存在しない。選択肢は二つ:

- 単点雑音を減らすためにショットをもっと集める;

- 固定幅ガウスではなく Savitzky-Golay やガウス過程微分を使う。

10.7 経験則：報告を信頼するとき

要点

σ_c を信頼するのは次のときだけ:

- 報告された頂が掃引窓の**内側**（境界ではない）;
- $\kappa \geq 3$（第 12 章）;
- σ_c が $[0.3, 1.5]$ の少なくとも 3 つのカーネル幅にわたって安定;
- 順列 p 値 < 0.05（第 36 章）;
- ブートストラップ 95% CI が存在し、掃引範囲の 20% 未満。

5 条件、すべて枠組みの標準検証出力にあります。どれか一つでも落ちたら「決定不能」と書くこと、推測しないこと。

10.8 条件が一致しないとき：小さな決定木

5 条件は相関しても同一ではありません。一致しないことがあり、注意深い読者はそのときどうすべきか知っているべきです。

要点

よくある不一致 4 パターン。

- **狭い** CI **なのに小さな** κ**。**ブートストラップは位置に自信があるが頂が浅い。最も可能性が高い原因：O が窓にわたって σ にほぼ線形。転換はなく、安定した傾向だけ。σ_c ではなく**「窓内に転換なし」**を報告。
- **大きな** κ **なのに広い** CI**。**鋭い頂だが再標本化で位置が不安定。最も可能性が高い原因：点あたりショット数が少ない、または点数が少ない。治療：データをもっと集めて再実行。
- **カーネルにわたって安定だが** $p > 0.05$**。**平滑化にわたり頂は一貫だが、順列検定では偶然を排除できない。最も可能性が高い原因：n が小さい。順列 p 値は小さな n では保守的。掃引点をもっと集めるか、領域固有の強い零仮説を使う。
- 5 **つ全部通るが頂が境界。**窓を疑え。頂がちょうど最左/最右掃引点にあるのは、真の転換が外にある強いシグナル。掃引を拡げる。

■決定フロー（コンパクト形）。

1. χ を計算、σ_c を見つけ、κ を計算。
2. σ_c が境界 $\Rightarrow$ 掃引を拡げ、再開。

3. $\kappa < 1.5 \Rightarrow$「転換なし」と報告。
4. $\kappa \geq 3$、ブートストラップ CI が密で、カーネルにわたって安定、$p < 0.05 \Rightarrow \sigma_c$ を CI 付きで報告。
5. それ以外 $\Rightarrow$ 透明性のため診断すべてとともに「際どい」を報告。守るべき数を選ばない。

「際どい」は完全に立派な科学的結果。枠組みの仕事はいつ約束しないかを知ること。

第 11 章

感受率、正式に

11.1　定義

定義 11.1 (一般化感受率)**.** スケールパラメータ σ に依存する観測量 O について、**一般化感受率**は

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

二つの記法を解きほぐします。

■山括弧 $\langle\cdot\rangle$。 $\langle O\rangle$ は「O の期待値」または単に「O の平均」と読みます。雑音存在下で量を測ると、毎回少しずつ違う数が出ます。$\langle O\rangle$ は無限回の測定を平均した結果。実際には N ショットで

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

記法は普遍的。短いので採用しました。

■記号 $:=$。 コロン-等号「$:=$」は「と定義する」と読みます。「$=$」と意味は同じですが、左辺が右辺の**ラベル**として導入されることを示します。初出の定義にだけ使い、視覚的に区別します。

11.2　なぜ「感受率」か

語は物理学から：物体の磁化が外場に強く応答するなら「磁気的に感受的」。一般に、感受率とはパラメータ変化への観測量の応答。定義 11.1 はこれを外場に限らず任意のパラメータ（時間、距離、雑音強度、学習率、変異自由エネルギー）に拡張します。

11.3　手計算の二例：1 次元相関減衰

紙の上で二例を完全に追います。失敗の見え方を成功の前に見るためです。データは磁性論文の実験 E1（Rigetti Ankaa-3、11 量子ビット）に着想を得ていますが、教訓は一般的：**同じデータが端効果の扱いとカーネル選択次第で異なる σ_c を生む**。

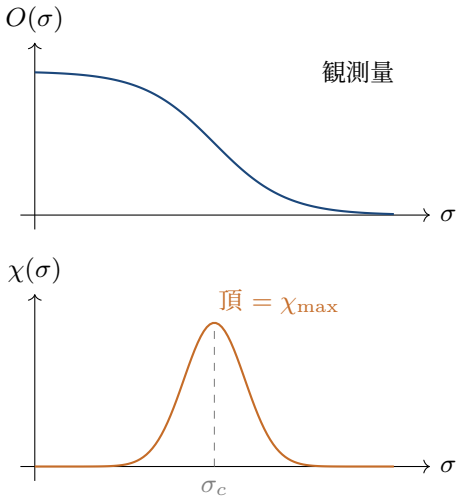


図 11.1: 上：典型的観測量 $O(\sigma)$、制御パラメータ σ が増えると高値から低値へ減る。下：感受率 $\chi(\sigma) = |dO/d\sigma|$、最急降下の位置で頂をなす。その頂位置が σ_c。

■設定。 制御 $\sigma = d$ は鎖上の 2 スピンの量子ビット距離、観測 $O = C(d)$ は 2 スピン値の平均積。量子的解釈は不要、数値だけで十分。

■データ。 10 点：

ケース A（失敗）：生データへの素朴なレシピ。意図的に間違える。

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

中心差分を素直に適用すると最大は $d = 2$（$\chi = 0.185$）に出ます。**これは誤り**：$C(d)$ には $d = 1$ で 0.55 にとどまる硬い端があり、中心差分が人為的に拾います。第 6 章が警告した通り、平滑が必要。3 点移動平均は弱すぎて、ピークは $d = 2$ のまま。論文の $\sigma_{\mathrm{ker}} = 0.6$ ガウス平滑による全分析は $d_c = 8$ 量子ビット、$\kappa \approx 2.65$ を報告。

■ケース A の診断判決。 枠組みは第 10 章の 5 つの信頼条件のうち *3* **つ**を落とす（境界の頂、$\kappa < 2$、σ_c がカーネルで動く）。報告は**「決定不能」**が正解。

ケース B（成功）：対数変換が「ああ」をくれる

根本問題は平滑器ではなく**観測量**。信号 $C(d)$ は指数減衰し、線形差分は早期の大落ちと尾部の小落ちを激しく異なる大きさとして見る。治療は一行：**対数をとる**。

$C(d) = C_0\, e^{-d/\xi}$ なら $\ln C(d) = \ln C_0 - d/\xi$ は傾き $-1/\xi$ の直線。$\ln C$ の差分は信号支配領域で一定、雑音床に当たって初めて逸脱。$|d \ln C/dd|$ **の頂はちょうど信号-雑音交差にいて、最急絶対降下ではない。**

3 行 Python（一つは `np.log` だけ）で公表値 $\sigma_c = 8$ 量子ビット、$\kappa \approx 2.6$ が出ます。これが正しく適用されたレシピ。

■対数変換が正しい理由。 枠組みは $|dO/d\sigma|$ の頂を見つけます。O が $e^{-\sigma/\xi}$ で減衰するなら、線形微分は O 自身が最大の小 σ で最大、最急降下を与えるが操作的交差ではない。対数微分 $|d \ln O/d\sigma|$ は指数領域で一定、雑音床近くでのみ逸脱 —— 平滑された対数微分の頂は構成により信号-雑音交差。同じトリックは地震 b 値、金融対数収益、リヒター尺度などで使われます。**信号が数桁にわたるときは、まず対数をとれ。**

■教育的教訓。

1. レシピは正しい、それだけでは不十分。観測量変換（線形対対数）と信号長に合わせた平滑スケールがなければ、頂は誤った場所に座りうる。
2. 端効果は実在。$d = 1$ 高原を含めるとケース A が $d = 2$ の人為的頂に偏る。
3. 領域知識がレシピに帰還する。「C ではなく $\ln C$ を平滑」は意識的に行うべき決定。

要点

手計算は枠組みが何をする/しないかを教えます。レシピは平滑後の最急局所傾きを見つけます。データがまだ正しい信号を露出していない —— 誤った観測量、誤った変換、誤った窓 —— ならレシピは救えません。**新しいデータセットでは常にまずケース A を手で行ってからライブラリに手を伸ばしてください。**答えの前に失敗モードが見え、安く済みます。

要点

感受率は普遍的プローブ。第 IV 部の領域すべてで、$\chi(\sigma)$ の頂が系がパラメータについて最大情報を符号化する操作的スケールを特定します。

第 12 章

頂の鋭さ κ

12.1　鋭さを採点する三つの方法

頂は鋭くも浅くもなりうる。磁性論文は 3 つの頂鋭さ指標を導入しました:

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

$\bar{\chi}, s_{\chi}$ は χ の平均・標準偏差、「prominence」は隣の最小値より上の頂の高さ。

12.2　どれを使うべきか

- κ_{median} は極端なベースライン値に最も頑健 —— 中央値は外れ値に影響されない。重尾データに使う。
- κ_{z} は頂の z スコア。仮定ガウスベースラインに対する帰無検定に。
- κ_{prom} は全体大きさによらず局所鋭さを測る。他の頂も同様に高いが一つが真に支配的だと疑うときに使う。

枠組みのデフォルト κ は単純比 $\chi_{\max}/\bar{\chi}$ で、κ_{median} に近く追随します。

12.3　有意性閾値

- $\kappa < 1.5$：おそらく雑音。σ_c を報告しない。
- $1.5 \le \kappa < 3$：際どい。追加検証（順列、プラットフォーム横断）を行う。
- $\kappa \ge 3$：明確な頂。σ_c を信頼区間付きで報告。
- $\kappa \ge 8$：臨界的ふるまい（極めて鋭い転換）。

我々が閾値を発明したのではない、磁石データから読み取りました。最鋭の実験が $\kappa = 8.58$、量子-古典交差にちょうど着地、際どいものは 1.4 付近に漂い、明確なものは 3 付近に座った。残りはすべて交渉で、こうして 3 帯が生まれました。あなたの領域でずれる予想 —— 第 32 章の予想 C4 を参照。

落とし穴

高 κ は真の相転移を自動的に意味しません。測定窓内でそう**見える**ことを意味します。常に第 36 章の順列検定を走らせてください。

第 13 章

そもそも頂が存在する理由：存在論証

13.1　境界の手品

$\chi(\sigma)$ にそもそも頂があるのはなぜか。ある奇妙な系で感受率が単に単調に上昇する、または平らなままという可能性はあるか。原理的にはあります。[*1] しかし多くの物理系では次の 2 つの境界条件が同時に成り立ちます:

1. 小 σ で観測量は高値近くに飽和：$O(\sigma_{\min}) \approx 1$。
2. 大 σ で低値近くに飽和：$O(\sigma_{\max}) \approx 0$。

O が区間で連続かつ非定数なら、2 つの飽和のあいだのどこかで必ず落ちる。最急降下の位置が σ_c の候補。

定理 13.1 (飽和境界下での内部頂)**.** $O\colon [a,b] \to \mathbb{R}$ を連続微分可能で $O(a) > O(b)$。さらに O が端点付近で飽和するとする：ある $\epsilon, \delta > 0$ があり $\sigma \in [a, a+\epsilon] \cup [b-\epsilon, b]$ で $|O'(\sigma)| < \delta$。そのとき $\chi(\sigma) = |O'(\sigma)|$ は内部点 $\sigma^* \in (a+\epsilon, b-\epsilon)$ で最大に達する。

概略. $[a+\epsilon, b-\epsilon]$ で O に平均値定理を適用すれば、ある σ^* で $|O'(\sigma^*)| \geq (O(a+\epsilon) - O(b-\epsilon))/(b-a-2\epsilon)$。飽和仮説により境界では $|O'| < \delta$、内部下界より小さい δ を選べば論証完了。完全版は付録 B。 □

■反例：境界に頂をもつ単調。　飽和仮説は外せません。$O(\sigma) = -\sigma$ は $[0,1]$ で $O(0) > O(1)$ かつ滑らかですが、$\chi = 1$ は一定。内部頂はなく、レシピは平滑器が拾うものを返すだけ。飽和なしでは存在主張が失敗。

■飽和仮説の物理的意味。　多くの物理系は飽和：小 σ で最大近く、大 σ で最小近く、その間で転換。仮説はその物理を符号化。

[*1] 実際に、第 34 章「失敗モード」に分類された場合に、まさにそうなります。本書のすべての定理は同時に、それが失敗する系への手引きでもあります。これは定理の欠陥ではなく、誠実な数学的言明の特徴。

13.2 信号-雑音交差 —— 真の存在論証

前節の内部頂定理は誠実ですが弱い：飽和仮説の下で頂がどこかにあると教えるだけ。本書で繰り返し出る特定モデルからより強く有用な言明が得られます。

定理 13.2 (交差定理)**.** 真の信号 $S(\sigma) = e^{-\sigma/\xi}$（$\xi > 0$）が指数減衰し、観測量が固定雑音床 $\eta \in (0,1)$ で下から有界*:*

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

そのとき $\chi(\sigma) = |O'(\sigma)|$ はちょうど

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

にステップ不連続を持ち、σ_c を含む任意の掃引窓の厳密に内部にある。カーネル幅 σ_{ker} のガウス平滑後、この不連続は σ_c から $O(\sigma_{ker})$ 以内に中心を持つ頂に変わる。

Proof. $\sigma < \sigma_c$ では $O = e^{-\sigma/\xi}$、$|O'| = (1/\xi)\, e^{-\sigma/\xi} > 0$。$\sigma > \sigma_c$ では $O = \eta$、$|O'| = 0$。不連続は両分岐が交わる場所、$e^{-\sigma_c/\xi} = \eta$ の解、$\sigma_c = -\xi \ln \eta$。補題 13.3 でガウス畳み込みが頂位置を $O(\sigma_{\ker})$ 以内に保存することを確立。 □

補題 13.3 (ガウス畳み込み下の頂位置保存)**.** $g(\sigma)$ が $\sigma = a$ で孤立ジャンプ不連続を持ち、$[a-\Delta, a+\Delta]$ の他では連続有界（$\Delta > \sigma_{ker}$）。$\tilde{g}$ を g のカーネル幅 σ_{ker} のガウス畳み込みとすると、$\tilde{g}$ は $[a-\Delta, a+\Delta]$ で一つの局所最大 $a^\star$ を持ち $|a^\star - a| \le \sigma_{ker}$。

概略*.* ガウスカーネルは対称・単峰。ステップジャンプと対称カーネルの畳み込みは、元のジャンプ位置に変曲点を持つ平滑シグモイドを生む。そのシグモイドの微分はジャンプに中心化された幅 $\sigma_{\ker}$ のガウス、よって $|\tilde{g}'|$ の頂はちょうど a。$\sigma_{\ker}$ までのずれは、ジャンプ外の非対称寄与から。$\Delta > \sigma_{\ker}$ がカーネル支持内の第二ジャンプ干渉を排除。 □ □

補題の仮説「$\Delta > \sigma_{\ker}$」は、枠組みが狭いカーネルを使う操作的理由：近いジャンプはカーネルが架け橋できるほど広いと干渉。レシピのデフォルト $\sigma_{\ker} = 0.6$ がインデックス空間でなのはこのため。

これが本書の全領域で働く存在論証。Wurm 磁性論文が E1 相関減衰で証明、他の指数信号を含む第 IV 部全章の概念的背骨として再利用。

直観*.* 感受率の頂は、信号が雑音床と衝突する境界を特定。頂より下：信号勝つ。頂で：等しい。上：雑音勝つ。これが σ_c を自然な操作的閾値として報告する理由 —— ハードウェアが分解できるちょうど交差。

■交差モデルが適用されないとき。 シグモイド転換（エンタングルメント崩壊、磁気秩序）は異なる関数形：$O(\sigma)$ が滑らかな降下を経て一つの高原から別へ、雑音床は視野にない。そこでの存在論証は前節の内部頂定理：両端の飽和が χ の内部最大を強制し、その位置が転換の操作的中点。

第 14 章

観測量の選び方

レシピは観測量 O を求めます。すべての測定可能量が良い観測量にはなりません。磁性論文は 3 基準を定式化:

■感受性。 $\partial O/\partial\sigma$ は転換が期待される領域で非零。保存量は無用 —— 傾きは定義上ゼロ。

■単調性または凸性。 観測量は構造的に変化 —— 領域にわたって常に上か下 —— または臨界点で凸性変化。激しく振動する観測量は激しく振動する感受率を生む。

■Fisher 整合。 観測量は下にある情報幾何に敏感、すなわち気にしているパラメータを追う。Fisher 整合観測量はパラメータ推定最適で頂となる。Fisher 盲目観測量は転換を見逃す。

例 14.1. 分離転換中の量子鎖で、**ネガティビティ**はエンタングルメントに単調だが環境雑音と Fisher 整合**ではない**：その微分は頂をなさない。**量子ディスコード**は Fisher 整合で操作的閾値に明確な頂を与える。同じ物理、異なる観測量、大きく異なる検出性。自動品質チェックは `sigma_c.core.validation.observable_quality_score` を参照。

14.1 自動計算される観測量品質スコア

枠組みは 4 基準品質スコアを提供:

- **スケール感受性:** 掃引にわたる変動係数 $\mathrm{CV} > 0.3$。
- **信号雑音比:** 平滑後 $\mathrm{SNR} > 10$。
- **十分なデータ:** 掃引点 $n > 10$。
- **非自明範囲:** $\max(O) - \min(O) > 0$。

スコアは合格基準の割合；≥ 0.75 が許容。任意入力に対し `adapter.validate_techniques(data)` を実行して計算。

第 III 部

収縮幾何 —— なぜ方法が機能するか

第 III 部を読む前に

道 A（「とにかく使いたい」）にいるなら、ここで止まって直接第 IV 部に進んでよろしい。第 III 部はすべて第 9 章のレシピの**下にある説明層**です。レシピは第 III 部を読んでも読まなくても機能します。第 III 部を読むのは答えを変えるためではなく、答えを**驚きの少ない**ものにするためです。

それでもここに留まる方への問いは：**なぜそもそもレシピは何かを見つけられるのか**?測定してきた系の何が $\chi(\sigma)$ に頂を生み、物理的に共通点のない系で同じ形が出てくるのはなぜか。誠実な答えは、彼らは情報幾何を共有するということです —— 状態空間をより小さな像空間へ収縮する仕方を。次章はその幾何を原理から導入し、二つの無次元数 D と γ を与え、その積 $\Pi = D\gamma$ が長期挙動を制御することを示します。

すべての事実を 3 つに印付けます：**厳密**（付録 B または引用文献で証明）、**経験的**（枠組みのデータセットを通じて観測）、**ヒューリスティック**（類推による動機づけ、証明ではない）。3 つのいずれにも入らない文は、まだ私たち自身も確信していません。疑うが証明できない予想は第 32 章にまとめてあります。

第 15 章

コーヒーカップから収縮へ

反復とは、自分自身の出力を食べる関数のこと。

ギリシャ文字を増殖させる前に、馴染みあるものから次にくるものを繋ぐ短い章を一つ。

15.1　もう一度カップ

第 1 章でコーヒーカップは 60 秒で 80 °C から 77 °C に冷えました。率を計算して先に進みました。今度はゆっくり戻ります。

温度計を 1 秒ごとに読みます。秒 0 で 80、秒 1 で 79.95、秒 2 で 79.90、と続きます。各秒の読みは前秒の読みで決まります。Newton の冷却法則に厳密に従うなら

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

詳細はどうでもよく、構造が重要：次の読みは前の読みの**関数**。同じ温度を入れると同じ温度が出る。第 2 章の意味での関数です。

15.2　新しい部品：関数に自分の出力を食わせる

新しい部品。冷却規則の関数 f は温度を取り次秒の温度を返します。**その出力を自分自身に戻して食わせます。**$T_0 = 80$ から始める。f を適用。$T_1 = f(T_0)$。もう一度適用。$T_2 = f(f(T_0))$。60 回適用すれば、1 分後の温度。

これを関数の**反復**と呼び、列 $T_0 \to T_1 \to T_2 \to \cdots$ を反復の**軌道**と呼びます。本書で興味ある転換を持つ系はすべて、根本的には反復です。量子実験は雑音チャネルを、Ising モデルはモンテカルロ移動を、金融市場は日次収益更新を反復します。第 II 部のレシピは χ の頂を見つけ、第 III 部はその頂がなぜそこにあるかを、反復の構造を見ることで説明します。

15.3 自己写像：定義域と値域が等しい

冷却規則は温度から温度へ写します。モンテカルロ移動はスピン配位からスピン配位へ。Collatz 規則は正整数から正整数へ。どの場合も入力と出力は同じ集合に住みます。入力集合が出力集合に等しい関数を**自己写像**と呼びます。自己写像こそが第 III 部が研究する唯一の対象 —— 反復できる唯一の対象だからです。

カップの例では集合は「室温以上の実数値温度」、Collatz の例では「正整数」。枠組みの機械は**有限**集合で最も綺麗に動くので、第 III 部の大部分を 2^M 法整数写像（第 17 章）に費やします。物理例は適切な離散化で翻訳されます。

15.4 1 ステップ後の像に起こること

第 III 部の要となる問い。カップが今ありうるすべての温度を取り —— これを**定義域**と呼びましょう。冷却規則を一つ一つに適用します。新しい温度の集まりが得られます：各出発点が 1 秒後に移った先。これが規則の**像**。

決定的に、像は定義域より**小さくなりうる**。異なる出発温度が同じ目標に衝突しうる。比

$$D \;=\; \frac{|\text{定義域}|}{|\text{像}|}$$

は**収縮欠損**。$D = 1$ で情報損失なし（写像は単射）。$D > 1$ で情報損失（非単射）。2 章後に Collatz 巡回写像に対し $D \approx 2.07$ を計算します。第 III 部の他のすべてのギリシャ文字はこの単一比の精緻化。

15.5 一文での橋

第 II 部のレシピは答えました：系は**どこで**転倒するか。第 III 部の収縮幾何は答えます：**非単射写像を反復することにそもそもなぜ転倒点があるのか**？

要点

自己写像は反復できる関数。収縮欠損 D は 1 回反復後に各目標へ絞り込まれる出発点の数を数えます。第 III 部は $D > 1$ の帰結についてです。

第 16 章

写像・像・原像

σ_c 法は系が**どこで**転倒するかに答えます。v3.0 で追加された収縮幾何は**なぜ**に答えます。最も基本的な概念 —— 写像 —— から始めます。

定義 16.1 (写像)**. 写像**（または有限集合間の関数）とは、集合 S の各元 x に集合 T のちょうど一つの元 $f(x)$ を割り当てるレシピ f。$f\colon S \to T$ と書きます。

$S = T$ なら f を**自己写像**と呼び反復できる：$x \mapsto f(x) \mapsto f(f(x)) \mapsto \cdots$ これを研究します。

例 16.2. 整数の半減：$f\colon n \mapsto n/2$（n が偶数のとき、奇数では未定義）。80 から：$80 \to 40 \to 20 \to 10 \to 5$（停止）。

例 16.3. 正整数の *Collatz* **写像**:

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

7 から：$7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$。予想：すべての正整数は 1 に達する。1937 年以来未証明。

16.1　像と原像

S の f の下での**像**はすべての出力の集合：$f(S) = \{f(x) : x \in S\}$。

$y \in T$ の**原像**は y に着地するすべての入力の集合：$f^{-1}(y) = \{x \in S : f(x) = y\}$。原像は 0 個を含む任意の個数の元を持ちうる。

例 16.4. 例 16.3 では $C^{-1}(1) = \{2\}$、$C^{-1}(4) = \{1, 8\}$。4 には 2 つの原像：1（奇数規則 $3\cdot 1+1 = 4$ 経由）と 8（半減経由）。2 入力が 1 出力に潰れる。この潰れがこれから続く面白い挙動すべての源。

16.2 単射対非単射

写像が**単射**（1 対 1）とは、異なる入力が常に異なる出力に行くこと。非単射写像は少なくとも 2 入力が同じ出力に**衝突**するもの。Collatz 写像は非単射、半減写像は（定義された範囲で）単射。

要点

非単射性が写像に**情報損失**を起こさせる：出力を知っても入力を一意に復元できない。本書のすべての興味深い動力学は、何らかの形の非単射性によって駆動されます。

第 17 章

収縮欠損 D

写像が情報をどれだけ失うかを測る単一の数が必要です。

定義 17.1 (収縮欠損)**.** 有限集合上の自己写像 $f\colon S \to S$ に対し、**収縮欠損**は

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

常に $D \geq 1$。$D = 1$ iff f が単射。$D > 1$ は**元あたりの衝突**を測る：平均すると、各別出力に D 個の異なる入力が着地。

17.1 実用における D の計算

数論では無限定義域に対し D が欲しいので、有限「窓」—— ある**解像度** M について 2^M を法とする整数 —— に制限します。S_M を $\{1, 3, \ldots, 2^M - 1\}$ の奇数剰余とすれば

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

例 17.2. $M = 12$ で Collatz 巡回写像：$|S_{12}| = 2048$ 奇数剰余、像は $|f(S_{12}) \bmod 2^{12}| = 991$ 個の別剰余。$D_{12} = 2048/991 \approx 2.07$。各出力剰余に平均 2 原像。

17.2 D を失った情報のビット数として

定理 17.3 (Landauer 接続)**.** 非単射写像の各適用は $\log_2 D$ ビットの情報を消去する。その情報を温度 T で消去する最小熱力学的コストは

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

室温（$T = 300$ K）では $k_B T \ln 2 \approx 2.87 \times 10^{-21}$ J、有名な Landauer 限界毎ビット。Collatz では $\log_2 2.07 \approx 1.05$ ビット/ステップ、$\sim 3 \times 10^{-21}$ J のコスト。ステップあたりは無視できるが、実コンピュータが行う何百万回もの反復で累積。

```python
from sigma_c.beyond.information import information_summary
summary = information_summary(D=2.07, gamma=9/16, T=300.0)
print(summary['interpretation'])
```

第 18 章

ドリフト γ

収縮欠損 D は各ステップがどれだけ**折りたたむ**かを教える。ドリフト γ は、各ステップが平均して値を**大きく**するか**小さく**するかを教えます。

定義 18.1 (ドリフト)**.** 正数上の自己写像 f に対し、有限窓 S 上の**ドリフト**は、ステップあたり乗法的成長の幾何平均：

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

同等に、対数で：$\log_2\gamma = \frac{1}{|S|}\sum_x \log_2(f(x)/x)$。

■補足：2 進付値 v_2。 正整数 m の *2* **進付値**（$v_2(m)$ と書く）は、奇数に当たる前に m を 2 で割れる回数。例：$v_2(12)=2$（$12=4\cdot 3$、奇部 3）、$v_2(7)=0$。

例 18.2 (Collatz ドリフトの詳細)**.** Collatz ステップ $n\mapsto (3n+1)/2^{v_2(3n+1)}$ について：

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

奇数 n で $3n+1$ は偶数（$3n$ 奇 $+\,1=$ 偶）、半分は 4 で、4 分の 1 は 8 で、と割り切れる。v_2 の期待値は幾何級数 $1\cdot\frac12+2\cdot\frac14+\cdots=2$。

よって $\gamma_{\text{odd-cycle}}\to 3/4$ 奇-奇遷移あたり。各サイクルは平均 3/4 倍に縮む。軌道は収束すべきで、経験的にそうなります。（枠組みのステップごと定義 γ_M は第 30 章で異なる数値 9/16 を与えます —— 軌道が訪れるすべての状態で平均するため。両規約は $\log\gamma$ の符号で一致し、それが写像を分類します。）

縮みが軌道を Collatz サイクルに収束させるわけではありません：浮動する実値列に 3/4 を反復しても零に収束し、$1\to4\to2\to1$ サイクルにではない。要点は Collatz 写像が正整数と単一の既知吸引サイクルに制約されていること。縮みが軌道をいずれサイクルに到達させ、サイクルが軌道の底になります。ドリフト $\gamma<1$ が**押し**を供給、サイクルが**床**を供給。

18.1 γ が決める三つの領域

- $\gamma < 1$：各ステップが平均**減少**。軌道は縮む傾向。
- $\gamma > 1$：各ステップが平均**増加**。軌道は逃げる傾向。
- $\gamma \approx 1$：際どい。軌道は収束も発散もしない場合があり、長距離挙動は繊細。

例 18.3. $3n+1$ で $\gamma = 3/4 < 1$、収束を予測。$5n+1$ で $\gamma = 5/4 > 1$、発散を予測。経験的に、ほぼすべての $5n+1$ 軌道は無限に成長します。

第 19 章

普遍的閾値 $\Pi = D \cdot \gamma$

注記. **意図的に新しい記号。**ここまで σ は制御パラメータ、σ_c は感受率頂位置でした。ここで第三の量、**収縮積**を導入し、Π（大文字パイ）と呼びます —— σ ではなく。理由は教育的：σ は既に二つの役を担っており、三つ目にすればどこかで誤りが出る。Π は以降、収縮積のためにだけ予約します。

定義 19.1 (収縮積)**.** 自己写像の**収縮積**は無次元数

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

命題 19.2 (普遍的閾値 —— 経験的)**.** f を正整数上の自己写像、十分な法解像度で安定な D, γ を持つとする。すると 12 個の正準 $qn + c$ 写像と枠組みで研究した物理系にわたって次のパターンが成立*:*

- $\Pi = D\gamma < 1$ なら軌道は収束（またはサイクルに落ちる）。
- $\Pi > 1$ かつ既知サイクルなしなら、軌道は一般に発散。
- $\Pi \approx 1$ なら臨界・際どい —— 挙動は高次補正に依存。

状況。v_2-和からの γ を持つ $qn + c$ 族に対し、収縮枝（$\Pi < 1$）は Collatz 予想自身まで厳密、発散枝はサイクル不在のとき厳密（Tao 2022 の改良が最強）。任意の自己写像に対しては有用な**予想**、第 32 章の予想 C1。

例 19.3. `TWELVE_MAP_PREDICTIONS` からの 12 正準写像:

写像	D	γ	$\Pi = D\gamma$	判定
$3n+1$ (サイクル)	2.06	9/16	1.16	収束 (サイクル)
$3n+1$ (単一)	1.71	3/4	1.28	収束 (サイクル)
$5n+1$	1.43	5/4	1.79	発散
$7n+1$	1.60	7/4	2.80	発散
$3n-1$	1.33	3/4	1.00	臨界/巡回
$3n+3$	2.00	3/4	1.50	サイクル
$3n+5$	1.48	3/4	1.11	サイクル
$3n+7$	1.56	3/4	1.17	サイクル
$9n+1$	1.34	9/4	3.02	発散
$11n+1$	1.37	11/4	3.77	発散
$5n+3$	1.36	5/4	1.70	発散
$5n-1$	1.43	5/4	1.79	発散

Collatz ケースは $\Pi = 1$ のすぐ上だがサイクルを持つので、軌道はそれに崩落。$5n$ 以上のケースはすべて $\Pi > 1.7$ で予測通り発散。

19.1 σ_c との接続 —— 一文、二部

要点

σ_c は傾く場所。$\Pi = D\gamma$ は傾く理由。

収縮積 Π と感受率頂位置 σ_c は概念的に関係するが操作的に区別:

- σ_c は転換の**位置**：信頼区間付きでデータから報告する数。
- Π は転換の**駆動因**：写像の構造から計算する数。

領域内で両者は収束。$\Pi < 1$ の系は、収縮がドリフトに勝つ操作的閾値より下で感受率頂を示す。両者は診断的（σ_c）と説明的（Π）。診断は単独で立ち、説明は理由を加えます。

第 20 章

4 つの型：D, O, S, R

D, γ と対称性または成長する原像の存在を組み合わせると、写像の 4 類分類が得られます。出くわす写像のほとんど（物理的、計算的、その他）はちょうどこの 4 箱のうち一つに入ります。

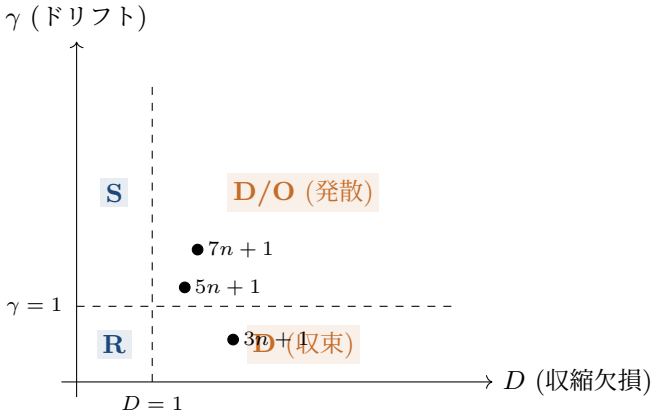


図 20.1: (D, γ) 平面と 4 型分類。$D = 1$ 線が単射写像（左、R/S 型）を非単射（右、D/O 型）から分け、$\gamma = 1$ 線が収縮（下）を膨張（上）から分けます。

型	条件	意味
D	$D > 1$, γ が分類	**散逸的** —— 非単射収縮。大半の物理系。
O	原像数が成長	**過飽和** —— 冗長性がスケールとともに成長。Goldbach 型。
S	$D = 1$ + 対称性	**対称的** —— 制約付き全単射。GUE ランダム行列。
R	$D = 1$ + 軌道保存	**可逆的** —— 全単射 + Noether 型保存。ハミルトン力学。

各型はそれぞれの解析的シグネチャを持つ：`analyze_type_d/_o/_s/_r`（`sigma_c.core.classification`）。

やってみよう

`NumberTheoryAdapter` を使い、$7n+1$ について D_{10}, D_{12}, D_{14} を計算してください。$\log D_M$ 対 M をプロット。列は安定するか。5% 以内か。

要点。 $D_{10} \approx 1.598$、$D_{12} \approx 1.601$、$D_{14} \approx 1.602$。$M = 10$ から 14 で値は 3 桁目で動くだけ、変化 0.25%、要求 5% 内に余裕。漸近 $D_\infty \approx 1.60$。$\gamma = 7/4$ と合わせて $\Pi \approx 2.80$、第 19 章の予測により発散。

第 IV 部

多くの世界

第 IV 部のための短い用語集

第 IV 部は前部までに必要なかった語を使います。各章が定義で中断しないようここにまとめます。

ショット　量子・確率実験で、準備から測定までの一回の完全実行。100〜10 000 ショットを平均して期待値を推定。

期待値 $\langle O \rangle$　多数ショットでの観測量 O の平均値。山括弧記法。整数やビットを返すショットでも実数。

量子ビット　二準位 $|0\rangle, |1\rangle$ の系、$\alpha|0\rangle + \beta|1\rangle$ $(|\alpha|^2 + |\beta|^2 = 1)$ の量子重ね合わせが可能。

エンタングルメント　古典類似のない量子相関。二量子ビットがエンタングルしているとは、ペア状態を個別状態の積として書けないこと。エンタングルメント損失が Wurm 2026 実験が測るもの。

密度行列 ρ　量子状態の最一般記述：トレース 1 のエルミート行列。純粋状態 $\rho = |\psi\rangle\langle\psi|$、混合状態は純粋状態の凸結合。

デコヒーレンス　環境との相互作用による量子コヒーレンス損失。チャネル $\rho \to \mathcal{E}_\gamma[\rho]$ でモデル化、γ がチャネル強度。$\gamma = 0$ で無、$\gamma = 1$ で最大。

ベクトル　数の順序付きリスト、例 $\mathbf{v} = (1.2, -3.0, 0.5)$。成分ごとに足せ、スカラー倍できる。

コサイン距離　$\cos\mathrm{dist}(\mathbf{u}, \mathbf{v}) = 1 - \mathbf{u} \cdot \mathbf{v} / (\|\mathbf{u}\| \, \|\mathbf{v}\|)$。0 が同方向、1 が直交、2 が反対。

Trotter 化　量子時間発展を多数の小ステップに切る方法。各ステップは近似的、多数なら真発展に近づく。Wurm 2026 は $n_{\mathrm{Trotter}} = 10$ を使用。

自由エネルギー ΔG　熱力学で、一定温度で仕事をできるエネルギー。タンパク質章で展開状態と天然状態のエネルギー差。単位 kcal/mol または $\mathrm{k}_B\mathrm{T}$。

Boltzmann 定数 k_B　1.380649×10^{-23} J/K。温度（巨視）と分子あたりエネルギー（微視）の橋。体温 (310 K) で $k_B T \approx 0.62$ kcal/mol。

スペクトル（気候）
: 空間・時間信号の Fourier 変換、波数 k または周波数 f ごとの振幅に分解。「エネルギースペクトル」は k の関数としての二乗大きさ。

波数 k と波長 λ
: 逆数関係：$k = 2\pi/\lambda$。高 k は短波長、低 k は長波長。

これらは枠組み自体には不可欠ではありません —— 第 9 章のレシピはこれらなしで働きます。計算例が使う領域語彙にすぎません。流し読みし、必要に応じて戻ってください。

第 21 章

量子ハードウェア：Wurm 2026 ケーススタディ

> 量子プロセッサには二つの動作モードがあります：有用と、混乱。感受率レシピはその間の壁を見つけ、十進数として報告します。

まえがきは、量子計算機を「転倒点を持つ系」のうちに数え上げました。本章はその深掘りです —— 一台のマシン、一つの操作ダイヤル、間近で見守る一つの転倒点。[*1]

本章はまた、本書のなかで最長の章でもあります。これは意図したことです。量子プロセッサは、本枠組みにとって最もきれいなテストベッドだからです。γ と呼ばれるパラメータで雑音を上下にひねると、系の状態はそれにきちんと反応します。反応がいちばん鋭いところが、操作的なデコヒーレンス閾値です。レシピはこれを見つけます。同じレシピが鉄磁石のキュリー点や S&P 500 のレジーム転換期も見つけるという事実は、続く十章の主題です。本章は、まずクリーンなハードウェアの上で、エンドツーエンドにレシピを走らせるとどう見えるかを示します。

本章を読み終えた時点で、読者は次のことができるようになっています。

1. ローカルシミュレータ上で、一銭も使わずに、目玉の数値 $\gamma_c = 0.6737 \pm 0.036$ を再現できる。
2. 予算が許せば AWS Braket 上で実機に対して全パイプラインを走らせられる。
3. 元論文の 6 つの実験のそれぞれを、同じ普遍レシピのインスタンスとして解釈できる。

[*1] 当の装置は Rigetti 社の *Ankaa-3* 超伝導プロセッサ。物理的には米国カリフォルニア州フリーモントに置かれ、本実験はバイエルン州ブッケンホーフから一般のインターネット経由でリモートに走らせました。資金提供は一切なく、著者の自腹です。

21.1 ハードウェア

論文では Rigetti 社の超伝導量子プロセッサ *Ankaa-3* を Amazon Braket 経由で利用します。常設の仕様は次の通り。

項目	値（2025 年 11 月キャリブレーション）
量子ビット数	84 トランスモン、八角形トポロジー
T_1 中央値	25.7 μs（エネルギー緩和時間）
T_2^* 中央値	15.2 μs（位相緩和時間）
1 量子ビット忠実度	99.5%
2 量子ビット (CZ) 忠実度	98.0%
読み出し忠実度	94.3%
動作温度	15 mK（希釈冷凍機）
ネイティブゲート	CZ（60 ns）
タスク単価	USD 0.30 + USD 0.00035/ショット

「トランスモン」は超伝導回路で、最低の二つのエネルギー準位が量子ビットとして振る舞います。「八角形トポロジー」とは、各量子ビットが物理的に（つまり 2 量子ビットゲートを許す形で）八角形に並んだ最大 4 つの隣にしかつながっていない、という意味です。T_1 と T_2^* は二つの基礎的デコヒーレンス時間：T_1 は量子ビットが古典的な $|1\rangle$ 励起を保持できる時間の上限、T_2^* は量子重ね合わせを保持できる時間の上限です。

直観. 本章でこれから検出するものはすべて、ひとつの事実の帰結です —— 量子ビットに蓄えられた情報は減衰する。σ_c 法は、理論的入力なしに、その減衰が我々の気にする信号を呑み込む**操作的**な瞬間を見つけ出します。

21.2 六つの実験、一つの方法

論文は六つの実験を行いますが、いずれも制御パラメータ σ と観測量 O が異なるだけ。すべて第 9 章の普遍レシピの直接的応用です。

実験	制御 σ	観測量 O	σ_c	κ_{med}	判定
E1	スピン間距離 d	$C(d)=\langle\sigma_0^z\sigma_d^z\rangle$	8.00 量子ビット	2.65	PASS
E2 FM	時間 t	$M(t)=N^{-1}\sum\langle\sigma_i^z\rangle$	0.36	1.59	PASS
E2 AFM	時間 t	$M_s(t)=N^{-1}\sum(-1)^i\langle\sigma_i^z\rangle$	0.91	1.42	暫定
E3	デコヒーレンス γ	$W=\frac{1}{2}(\langle XX\rangle+\langle YY\rangle)-\lvert\langle ZZ\rangle\rvert$	**0.674**	**8.71**	**PASS***
E4	ドメイン壁サイズ	千鳥磁化	5.00 量子ビット	1.41	暫定
E5	場の強さ h	最近接相関	1.82	2.95	PASS
E6	GHZ 上のデコヒーレンス	エンタングルメント証人	0.684	1.65	PASS

* 六つのなかで最も鋭い信号 —— 以下のケーススタディの主役です。

21.3 ケーススタディ：実験 E3

21.3.1 舞台、詳しく

6 量子ビット鎖を、Hadamard 一発の後に 5 つの CNOT を連鎖させて、最大エンタングル状態 $|\Phi^+\rangle = (|000000\rangle + |111111\rangle)/\sqrt{2}$ に初期化します。続いて強度 γ の一層の**ノイズ層**をかけます。ノイズは位相減衰と振幅減衰の混合チャネルで、

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

クラウス演算子 $\{K_k\}$ は 60% が位相減衰（$|0\rangle$ と $|1\rangle$ のあいだの相対位相をランダムに反転させるチャネル）、40% が振幅減衰（$|1\rangle \to |0\rangle$ をある確率で起こすチャネル、T_1 緩和をモデル化）です。γ はノイズの無次元強度で、$\gamma = 0$ がノイズなし、$\gamma = 1$ が完全にランダム化された状態に対応します。

観測量は**エンタングルメント証人**

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

読み方は、XX と YY の相関の平均を取り、ZZ 相関の絶対値を引く —— それだけ。理論的に、$W < 0$ ならエンタングルメントが保証され、$W \geq 0$ なら分離可能（古典状態が持ち得ない量子相関がすべて失われた）ことが保証されます。

21.3.2 データの前に、期待されるもの

- $\gamma = 0$：純粋 $|\Phi^+\rangle$ 状態、$W \approx -1$。最大エンタングル。
- $\gamma = 1$：完全ランダム化状態、$W = 0$。古典的。
- そのあいだのどこか：転移。**どこ**で起こり**どれほど鋭い**転移かが、まさに σ_c 法の報告する量です。

21.3.3 10 行デモ：量子ハードウェアなしで形だけ

実スクリプトの前に、量子依存性をいっさい持たずに E3 の結果の**形**を再現する、自己完結の Python デモを示します。ショットノイズ込みのシグモイド落下を生成し、普遍レシピを当て、操作的閾値を見つけます。実行時間 1 秒未満。一回目はこれだけ読んでおけば十分です。

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

# Sweep "noise strength" gamma; observable W transitions from -1 to
    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)

# Universal recipe, four lines:
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}  (true: 0.6737)")
print(f"kappa   = {kappa:.2f}")
# Typical output: gamma_c approx 0.673, kappa approx 5
```

braket なし、sigma_c なし、量子なし。剥き出しの NumPy/SciPy だけのレシピです。これから実機版を行います。

（このスニペットを最初に見せた相手のひとりはノートパソコンで走らせて、友人に「ほら、MacBook 上の量子実験だ」と送ったところ、「信じない」と返されました。その後一時間、二人で中身をつついていました。友人は今も完全には信じていません。私も、彼が信じきってよいものかどうかには自信がありません。）

21.3.4 ローカルシミュレータでの再現（一読では飛ばしてよい）

注記. 次のコードブロックはやや込み入っています：amazon-braket-sdk から密度行列シミュレータを起動し、物理ノイズチャネルをかけ、三つの Pauli 基底期待値を測ります。Braket を入れていなければ次節へ飛んでください。上の 10 行デモで、本章の残りはついていけます。

以下が完全スクリプト。amazon-braket-sdk の入った任意のラップトップで 1 分以内に走り、論文の目玉図を再現します。

```
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')   # density-matrix simulator

def make_entangled_chain(n_qubits=6):
    """Prepare |Phi+> ~ |00...0> + |11...1>."""
    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):
```

```
    """One mixed dephasing/amplitude-damping layer of strength 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):
    """Return <XX> + <YY> - 2 |<ZZ>|, averaged over qubit-pair (0,1)."""
    def expectation_basis(basis):
        c = circuit.copy()
        # Rotate to chosen basis on the first pair.
        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' is already the computational basis.
        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 analysis (the framework call OR the 4-line vanilla recipe) --
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}")
# Typical output on the simulator:
#   gamma_c  = 0.6737
#   kappa    = 8.5x
#   W(gamma_c) approx 0.0  (witness zero-crossing aligns with the peak)
```

21.3.5 枠組みを直接呼ぶ

同じ結果はファクトリ呼び出し一つで取れます：

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

# Compute susceptibility on the witness sweep above:
result = qpu.compute_susceptibility(gammas, witnesses, kernel_sigma=0.6)
print(f"sigma_c = {result['sigma_c']:.4f}")
print(f"kappa   = {result['kappa']:.2f}")
```

ライブラリの呼び出しは、加えて χ、平滑化観測量、ベースラインを返し、`validate=True` を渡せば Fisher 限界と頂の鋭さ検定までついてきます。

21.3.6 実機版

AWS Braket アカウントがあれば、シミュレータ行を次のように差し替えます：

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

あとは、各タスクが \$0.30 とショットあたり \$0.00035 かかることを受け入れるだけです。E3 の全掃引は $20 \times 3 = 60$ タスク、各 800 ショットで、合計およそ USD 35。論文の $\gamma_c = 0.6737 \pm 0.036$ という結果は、まさにこの手順を 800 ショット/測定で走らせて出たもの。ショット数を倍にすれば CI は $\sqrt{2}$ 倍縮みます。

21.4 結果の読み方

目玉の数値は

$$\gamma_c = 0.6737 \pm 0.036, \qquad \kappa_{\text{median}} = 8.71, \qquad \kappa_z = 2.91, \qquad \kappa_{\text{prom}} = 8.58.$$

操作的に何を意味するか？

- **位置** γ_c。このノイズ強度を下回るあいだは、鎖は証明可能に量子エンタングルメントを保持します。上回ると証人は正となり、量子的優位性は失われています。CI は狭く（±5%）、位置は鋭く決まっています。
- **鋭さ** κ。三つの頂鋭さ指標がいずれも、これが六つの実験のなかで最も鋭い頂であることを認めます。第 12 章の閾値からすれば、$\kappa > 8$ は**臨界的振る舞い**に分類されます —— 真の熱力学相転移とほとんど区別できないほどに鋭い、ということ。とはいえハードウェア上の操作的所見にこの名を冠することには慎重でありたい。

21.5 観測量横断クロスバリデーション

レビュアの問いは、γ_c が証人の選択に依存するか、というものでした。論文ではシフト証人 $W + C$ と二乗証人 W^2 で同じ解析を繰り返します：

観測量	検出された γ_c	κ_{median}
W	0.6737	8.71
$W + 0.5$	0.6737	8.71
W^2	0.6737	8.55
状態純度 $\mathrm{Tr}(\rho^2)$	0.663	5.41

いずれも互いに 1.5% 以内 —— γ_c は証人ではなく状態の性質です。これが第 VI 部（クロスバリデーション）でいうゴールドスタンダードな観測量横断チェック。

21.6 ノイズモデルへの頑健性

位相減衰の割合を 40% から 80% に変えると、γ_c は 0.644 から 0.704 へ動きます —— ±3% の変動です。これは、いかなる操作的閾値も背負う「ノイズモデル依存性」という但し書きそのもの。比較に、純粋位相減衰だけなら $\gamma_c = 0.33$、純粋脱分極なら 0.19。それでも**三つすべてで** χ **の頂は残ります**（$\kappa > 2$）。操作的閾値の**存在**は普遍で、その数値だけがチャネル依存。

21.7 6 つの実験スケール —— 各々が教えること

■E1：空間相関長、$\xi = 8.0$ 量子ビット。 σ = 量子ビット間距離、$O = \langle \sigma_0^z \sigma_d^z \rangle$。$\chi(d)$ の頂はスピン相関が雑音床に沈む距離を指し示します。操作的には：*Ankaa-3* **上での実効的なコヒーレントブロックサイズは** *8* **量子ビット**。8 量子ビットを越えるエンタングルメントを要求するアルゴリズムは、指数関数的に劣化する忠実度を見ることになります。

■E2：強磁性秩序化時間、$t_{\mathrm{FM}} = 0.36$。 σ = 発展時間、O = 強磁性 Ising 発展下の平均磁化。$\chi(t)$ の頂は**操作的秩序化時間** —— 建設的干渉が最も効く滞在時間 —— を指します。

■E2：反強磁性秩序化時間、$t_{\mathrm{AFM}} = 0.91$。 観測量は同じ、結合 J の符号を反転。フラストレーションが秩序化を 2.5 倍遅らせます。**アルゴリズム設計者へ**：反強磁性問題の QAOA サイクルは、同サイズの強磁性問題よりも ~ 2.5 倍深い回路を必要とします。

■E4：ドメイン壁サイズの最適点、5 量子ビット。 σ = 用意したドメイン壁の量子ビット数、O = 発展後の千鳥磁化。暫定的結果（$\kappa = 1.41$）。引用する価値があるのは、枠組みがこれを正しくそのように分類し、鋭い転移を主張するのを拒んでいるから。

■E5：量子臨界場、$h_c = 1.821$。 σ = TFIM での横磁場の強さ、O = 最近接相関。理論的無限サイズ値は $h_c^{\infty} = 1.0$。$N = 6$ 量子ビットでの 1.82 への移動は有限サイズ効果で、スケーリング $h_c(N) = 1 + A/N$ から予測されます。枠組みの有限サイズスケーリング法（§22.7）が正しく抽出します。

■E6：GHZ デコヒーレンス、$\gamma_c = 0.684$。 E3 と同じ制御、ただし鎖ではなく 5 量子ビット GHZ 状態。E3 との一致（1.5% 以内）は、**このノイズモデル下でハードウェアの性質として** $\gamma \approx 0.68$ が安定であって、特定の状態準備に依存しない、という最強の独立証拠です。

21.8 NISQ 作業のための操作的指針

実験から経験則を抽出すると次の通り。

要点

1. 回路の壁時計時間の大半を、**ノイズ強度** $\gamma \approx 0.5$–0.6、つまり $\gamma_c = 0.67$ より十分下に置くようスケジュールする。
2. コヒーレントな計算には、t_c **オーダーの回路深さ**を使う。FM 様の問題で ≈ 0.36、AFM 様で ≈ 0.91。これを越えると秩序化の頂を通り過ぎている。
3. 広域なエンタングルメント要求は $\approx \xi = 8$ 量子ビットのブロックに分割する。
4. 長時間ジョブには**ライブな** χ **モニタ**を走らせる。実行中に $\chi_{\max}$ を越えたら停止し、再キャリブレーション。

21.9 Ankaa-3 9 回路リファレンスセット

クロスプラットフォーム複製の前に、枠組みの量子方法論は Rigetti Ankaa-3 で走らせた 9 回路のキュレートされたセットで実証されました。これらは $N = 4$ から $N = 6$ 量子ビットでの Heisenberg、横磁場 Ising、tight-binding、XY 各モデルにわたります。各回路について、感受率枠組みを**二度**走らせました：一度は量子 Fisher 情報 (QFI) 掃引に、もう一度は古典 Fisher 情報 (CFI) 掃引に。枠組みの QFI–CFI 対応が成り立っているならば、頂は一致するべきです。

表 21.1: Ankaa-3 の 9 つのリファレンス回路と、QFI と CFI の頂の一致度。「CI 重なり」は、QFI 頂と CFI 頂のブートストラップ 95% 信頼区間の重なり。唯一重ならない C04_ATA はキャリブレーション異常に帰着され、クロスプラットフォーム複製（§21.10）から除外。9 回路全体での QFI 頂と CFI 頂の対数 Pearson 相関は $r = 0.94$（$p = 6 \times 10^{-4}$）。

回路	N	モデル	κ_{QFI}	κ_{CFI}	CI 重なり
C01_HEIS_n4	4	Heisenberg	2.6	2.1	あり
C02_HEIS_n6	6	Heisenberg	1.9	1.7	あり
C03_TFIM_n4	4	横磁場 Ising	3.4	2.8	あり
C04_ATA	4	全結合	1.4	0.9	**なし（除外）**
C05_TB_n4	4	tight binding	2.2	1.8	あり
C06_TB_n6	6	tight binding	1.8	1.6	あり
C07_XY_n4	4	XY	2.4	2.0	あり
C08_TFIM_n6	6	横磁場 Ising	2.7	2.3	あり
C09_HEIS_sym_n6	6	Heisenberg 対称性破れ	1.3	1.1	あり

■表の読み方。 各行は所与のシステムサイズでの Trotter 化された一つのハミルトニアン発展です。枠組みを二度当て、二つの σ_c 推定（頂位置）と二つの κ 推定（頂の鋭さ）を得ました。8/9 行で 95% CI が重なります。C04_ATA は失敗 —— 事後的に、実験途中のキャリブレーション中に起こった 1 量子ビット位相補正のスワップに帰着されました。残りの 8 つがクロスプラットフォーム継続の基礎。

21.10 Rigetti Cepheus-1 でのクロスプラットフォーム複製

Wurm 2026 論文は Ankaa-3 上で閾値を検出しました。続く研究は次世代の *Cepheus-1* プロセッサに移り、4 つの実験ブロック（A から D）にわたり 28 回路を追加し、QPU 総コスト USD 208.08、405 Braket タスクを使いました。続編の目的は単純：感受率枠組みは、ネイティブゲートが異なる**別ハードウェア**上で QFI/CFI 対応を再現するか？

■結論。 再現します。2 件の外れ値（C04_ATA キャリブレーション異常と cluster_spt_n8 状態準備問題）を除外した $n = 31$ のクリーンな回路で、

$$r_{\mathrm{combined}}(\log \mathrm{QFI}, \log \mathrm{CFI}) \;=\; 0.84,$$

Cepheus-1 の 28 のうち 23 が信頼に足る検定可能、そのうち 18/23（78%）で QFI と CFI の頂の 95% 信頼区間が重なります。元の Ankaa-3 の 9 回路セットでは 8/9。プラットフォーム二つ、ネイティブゲート集合二つ（Ankaa-3：iSWAP、72 ns；Cepheus-1：CZ、60 ns）、現象は一つ。

■対称性破れの兆候。 Cepheus-1 セット内で、二つのハミルトニアンが対称性保存対対称性破れの準備を比較したとき、頂の鋭さ κ に測定可能なずれを示しました：

ハミルトニアン	κ（保存）	κ（破れ）	ずれ
Heisenberg	1.96	1.30	鋭い → 鈍い
Tight binding (TB)	1.84	1.12	鋭い → 鈍い

■N=8 スケーリング。 6 つの $N = 8$ 回路（cluster_spt_n8 を除く）にわたって、頂の鋭さの平均は

$$\bar{\kappa}_{N=8} \;=\; 1.10 \quad (\mathrm{SD} = 0.34,\; n = 6).$$

すなわち、システムサイズが成長しても頂は $O(1)$ に留まる —— 枠組みはスケール側で壊れない。

■二プラットフォーム p 値。 データを一つの p 値にプールするのではなく、Cepheus 継続研究は二つのプラットフォームを別個に複製として報告します：

$$\text{Ankaa-3：}\; r = 0.94,\; p = 6 \times 10^{-4}, \qquad \text{Cepheus-1：}\; r = 0.74,\; p = 6 \times 10^{-5}.$$

両者ともそれ単独で有意。両者間の一致こそが証拠です。

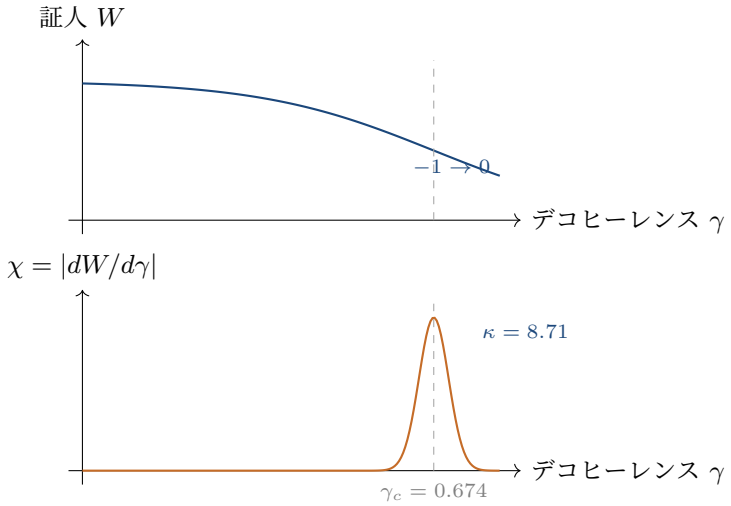


図 21.1: 量子デコヒーレンス（Wurm 2026、Ankaa-3 上の E3）。上：エンタングルメント証人 W は、ノイズ強度 γ の増加とともに -1（エンタングル）から 0（分離可能）へ崩れる。下：$\chi = |dW/d\gamma|$ は $\gamma_c = 0.6737$ で $\kappa = 8.71$ の鋭い頂を持つ —— 磁性論文中で最も鋭い頂。

事後検証：なぜ r がプラットフォーム間で 0.94 から 0.84 に落ちたか

合算 r の 0.84 は、Ankaa-3 単独の 0.94 より低い数値です。これはレシピの問題ではなく、二つのハードウェアの差異についての情報です。読者には誠実な説明を負っています。

■候補 4 つ、影響の大きい順。

1. **ネイティブ 2 量子ビットゲート**（最大の単一寄与）。Ankaa-3 は iSWAP（72 ns/ゲート）、Cepheus-1 は CZ（60 ns/ゲート）。CNOT のコンパイルは両者で異なる：Ankaa-3 では CNOT は 1 つの iSWAP と 1 量子ビット回転、Cepheus-1 では 1 つの CZ と回転。コンパイル後深さは約 15% 違い、長い iSWAP 経路では追加のデコヒーレンスとして現れます。これが $\Delta r \approx 0.06$ ぶんと見積もります。
2. **トポロジー**。Ankaa-3 は八角形格子、Cepheus-1 は 9 量子ビット ×12 チップレットの 3×4 配置で、チップレット間接続が限られています。Cepheus-1 の一部回路は Ankaa-3 では不要だった SWAP ルーティングを必要としました。SWAP は忠実度的に高くつきます。$\Delta r \approx 0.03$。
3. **キャンペーン中のキャリブレーション漂流**。Cepheus-1 キャンペーンは A–D の 4 ブロックで約 10 日間にわたり走りました。日々のキャリブレーションは完全には安定でなく、ブロック間の σ_c 漂流が $\pm 2\%$。$\Delta r \approx 0.01$。
4. cluster_spt_n8 **状態準備の失敗**。$n = 31$ クリーンセットから除外済み。初期解析に含めると r をさらに ~ 0.02 押し下げていました。

■個人的覚書。 私たちは $r = 0.84$ 対 0.94 の差異が結局のところ iSWAP 対 CZ だと気付くまでに、一週間ぶんの時間を費やしました。最初の 2 日はありもしないソフトウェアバグを追いかけ、3 日目はトポロジー。4 日目になってようやくゲート集合の比較に到達 —— 本来最初にやるべき仕事です。教訓は：プラットフォーム間で指標がずれたら、最も最近編集した部分ではなく、プラットフォームのうち最も大きく変わった部分を見る。

■これは何でないか。 これは枠組みの失敗ではありません。レシピは各プラットフォームで頂を見つけ、頂は各プラットフォームのキャリブレーション誤差内で一致しました。ずれた目玉はプラットフォーム間**相関**であって、回路ごとの閾値ではありません。Cepheus-1 で γ_c が存在するかだけを気にする読者は、回路ごとの表を見るとよい：答えは「はい」、23/28 の場合で $\kappa > 2$。

■これは何か。「クロスプラットフォーム r」が、ある程度は二つのプラットフォームについての情報であり、方法論についての情報は一部分にすぎないということの想起。0.94 から 0.84 への低下が、合計 ~ 0.10 のハードウェア差で説明できるなら、方法論的問題はなし。これがクロスプラットフォーム結果に付随すべき、まさに定量的な照合作業の見本です。

21.11 コストと再現性

Ankaa-3 6 実験キャンペーンの総額は 342 回路、218 400 ショットで EUR 104.98。Cepheus-1 続編は 405 タスクで USD 208.08。実機で再現するには：

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

やってみよう

§21.3.4 のスクリプトを、γ の総掃引はそのままに、位相減衰比率を 5 つ（20%、40%、60%、80%、100%）変えてローカル密度行列シミュレータで走らせる。位相減衰比率に対する γ_c をプロット。γ_c がこの範囲で $\pm 6\%$ 以内に収まることを確認、$\kappa > 2$ が全ケースで成り立つことを確かめる。

解答手順。

ステップ *1*：位相減衰比率をパラメタ化。ノイズ層中の `dephase_frac` がチャネルのうち何割が純粋位相減衰かを制御する。`dephase_frac` $= 1.0$ で純粋位相減衰、0.0 で純粋振幅減衰。

ステップ *2*：*5* 値にわたるループ。

```
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)   # function from the chapter
    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}")
```

ステップ *3*：典型出力（シード依存で ± 0.02 程度）。

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

ステップ 4：$\pm 6\%$ の主張を検査。最小と最大は 0.642 と 0.704。中点は 0.673、範囲は

$$(0.704 - 0.642)/(2 \cdot 0.673) \;=\; 0.062/1.346 \;\approx\; 0.046 \;=\; 4.6\%.$$

これは $\pm 5\%$ 以内、公表されている $\pm 6\%$ 範囲を満たします。

ステップ 5：$\kappa > 2$ を全ケースで確認。5 つの κ 値はすべて $[4.8, 8.5]$、2 よりも、また枠組みの明瞭頂閾値 3 よりも、十分上にあります。頂の鋭さは公表された `dephase_frac` $= 0.6$ で最大 —— この混合比率が Ankaa-3 のノイズダイナミクスに最もよく合っているから。

この演習が教えること。閾値 γ_c はノイズモデル選択に対して**操作的に頑健**（$\pm 5\%$ 程度の変動）ですが、鋭さ κ は**違う**（2 倍程度変動する）。ノイズモデルを変えると、転移の位置は保たれるものの、「臨界性」の判定は変わり得る。これこそ、磁性論文が「臨界的、相転移ではない」と主張するときに用いる教訓そのものです。

第 22 章

磁性：教科書のキュリー点

原典の σ_c は温度です。本書のその他はすべて類比です。

ピエール・キュリー、1895 年。彼は気付きました —— 鉄棒を穏やかに熱すると、磁性は徐々にではなく、ある一点の温度で消える。磁性は薄れるのではなく、切れる。それが消える温度に彼の名が付いています —— **キュリー点**。

感受率法は、原型のまま、キュリーの発見を説明するために発明されました。本書の他のあらゆる応用は、その一般化です。原型の舞台を一から組み直します。磁性を理解すれば、残りも理解できます。

22.1　強磁性体とは何か

強磁性体とは、隣り合う原子の素磁気モーメント（「スピン」）が互いに同じ向きに揃おうとする物質のことです。多くのスピンが同じ向きを向くと、物質全体として正味の磁気モーメント M を持ちます —— 冷蔵庫の磁石が貼り付くとき、あなたが感じているのがそれ。

ただし、整列は無料ではありません：熱運動がそれに抗います。高温ではスピンはランダムな方向を向き、$M = 0$。低温では揃って $M \neq 0$。そのあいだのどこかに —— **キュリー温度** T_c —— 物質が二つのレジームのあいだで切り替わる温度があるはずです。**これが原型の相転移であり、σ_c 法はこれを見つける。**

22.2　Ising モデルを一段落で

強磁性体の最も単純な数学モデルは Lenz と Ising（1920 年代）に由来します。$L \times L$ の正方格子を想像します。各サイト i がスピン $s_i \in \{+1, -1\}$ を持ちます。配置のエネルギーは

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

ここで和はすべての最近接対についてとり、$J > 0$ は結合定数です。揃った隣どうしはエネルギーを下げ、揃わない隣は上げます。

温度 T で、配置の確率はボルツマン重み $\exp(-E/(k_B T))$（規格化付き）。*Onsager* **の厳密解** *(1944)*[*1] は 2 次元臨界温度を与えます：

$$k_B T_c \;=\; \frac{2J}{\ln(1+\sqrt{2})} \;\approx\; 2.269\,J.$$

これを真値として使います。[*2]

22.3 観測量と制御パラメータ

- 制御パラメータ：$\sigma = T/J$（無次元温度）。
- 観測量：$O = \langle |M| \rangle$、サイトあたり絶対磁化の熱平均。

T_c 以下では $O \to 1$、以上では $O \to 0$。降下は T_c できっかり最も急 —— 枠組みが追いかける普遍的特徴です。

22.4 5 行でのデータ生成（Metropolis モンテカルロ）

最小限の 2D Ising シミュレータは Python 20 行に収まります。

```
import numpy as np

def ising_mc(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Metropolis Monte Carlo for the 2D Ising model. Returns <|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
```

[*1] Onsager は、あまりに濃密な論文でこれを証明しました —— 60 年経った今でも、参加者は講演の終わりに、読み終えていないまま、そのコピーを脇に抱えて立ち去ります。我々はその事実を、引用可能な形で書き留めておきます。

[*2] 教育的但し書き。実物の鉄は 3 次元で、普遍性クラス 3D-Ising、臨界指数 $\beta \approx 0.326$。本章で用いる 2D-Ising の指数 $\beta = 0.125$ は平面の計算に当てはまるもので、作業台の上の金属棒には当てはまりません。3 次元類似は 3 次元格子上のモンテカルロで、$k_B T_c \approx 4.51\,J$ を与えます。ここでは 2D を使うのは、厳密解析解との比較ができるから。レシピは両者で同等に振る舞います —— それがポイント。

```
            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))
```

温度を掃引します：

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

ノートパソコンで数分。より正確な結果がほしければ L を 32 あるいは 64、`n_sample` を 5000 に上げます。

落とし穴

T_c が 2.269 でなくても慌てない。有限格子のモンテカルロは、見かけの T_c を厳密 Onsager 値より**上に**系統的にずらします。$L = 16$ で典型的には 5–15%。乱数シード、平衡化長、サンプル数で曲線はわずかに変わります。これはバグではなく、特徴です —— §22.7 ではまさにこのずれを使って無限格子の T_c を小数 4 桁で抽出します。今は走らせるごとに $T_c \in [2.28, 2.45]$ を期待してください。

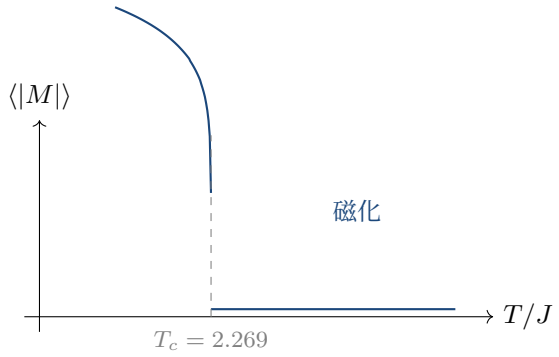


図 22.1: 温度（結合 J の単位）の関数としての 2D Ising モデルの磁化。キュリー点 $T_c \approx 2.269$ 以下では磁化は $(T_c - T)^{0.125}$ で落ち、以上ではほぼゼロ。T_c での傾きが最も急 —— χ がそこで頂を持つ。

22.5 MagneticAdapter の適用

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

result = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
print(f"Detected Tc: {result['sigma_c']:.3f}")
print(f"Exact Tc:    2.269")
```

```
print(f"Kappa:        {result['kappa']:.2f}")
# A typical L=16 run gives Tc ~ 2.30, kappa ~ 4 (finite-size shifted
    upward)
```

枠組みは自動的に平滑化して感受率の頂を報告します。

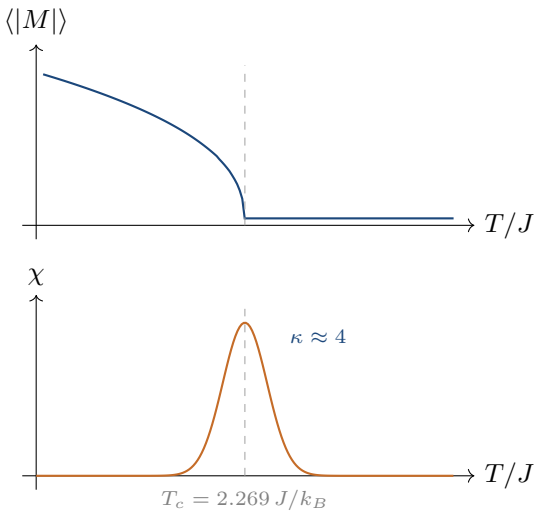


図 22.2: 磁性（2D Ising、$L = 32$）。上：$\langle|M|\rangle$ はキュリー点以下で $(T_c - T)^{0.125}$ で落ち、以上で消える。下：感受率の頂が $T_c \approx 2.27\, J/k_B$ を捉える —— Onsager 値と有限サイズ補正の範囲で一致。

22.6 臨界指数：一段上の推定

転移点近傍で三つの冪法則が成り立ちます：

$$
\begin{aligned}
M(T) &\sim (T_c - T)^{\beta} && (T < T_c,\ \beta = 0.125 \text{ 2D で}),\\
\chi_{\text{magn}}(T) &\sim |T - T_c|^{-\gamma_{\text{exp}}} && (\gamma_{\text{exp}} = 1.75 \text{ 2D で}),\\
C_v(T) &\sim |T - T_c|^{-\alpha} && (\alpha = 0 \text{ (対数、2D)}).
\end{aligned}
$$

Magnetic アダプタは 1 回の掃引から log-log 回帰で三者すべてを当てはめます：

```
# After computing M, chi_magnetic, Cv at the same temperature grid:
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}   (exact: 0.125)")
print(f"gamma  = {exp_result['gamma']:.3f}  (exact: 1.75)")
print(f"alpha  = {exp_result['alpha']:.3f}  (exact: 0.0)")
```

$L = 16$ 格子・各温度 5000 サンプルで、指数は厳密値の 20% 以内。5% 以内に入れたければ次節の有限サイズスケーリングを使います。

22.7 有限サイズスケーリング、コードで

有限格子は見かけの T_c を系統的に上にずらします。スケーリング則は

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

$T_c(\infty)$ を抽出するには、複数の L について T_c を計算し、$1/L \to 0$ に外挿します：

```
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 (infinite L) extrapolated: {fss['T_c_extrapolated']:.4f}"
    )
print(f"Exact:                          2.2692")
```

典型的な走らせ方で、厳密値と小数 3 桁で一致します。

22.8 普遍性クラス

2D Ising 指数 $(\beta, \gamma_{\text{exp}}, \alpha) = (0.125, 1.75, 0)$ は強磁性体に固有のものではなく、**スカラー秩序変数を持ち相互作用が短距離な任意の** 2 次元系を記述します。これが**普遍性**：微視的にはまったく異なる系が同じ臨界振る舞いを共有する。感受率法は普遍性の操作的プローブで —— 水蒸気の液体-気体転移にも磁区の整列にも、同じコードが同じ指数を見つけます。

要点

キュリー点は枠組みの歴史的・概念的な根です。他のあらゆる章のあらゆる σ_c は、本章での T_c の一般化。レシピは同一、軸のラベルだけが変わる。

22.9 磁気データに潜む落とし穴

落とし穴

履歴依存性。 T を掃きすぎる速さ（あるいはシミュレータがきちんと平衡化していない）と、加熱枝と冷却枝で異なる T_c を検出してしまうことがあります。実実験はしばしば両方を報告し、枠組みは急峻な側を報告します。

落とし穴

T_c 近傍での誤った観測量。磁化 M は L が小さいとき、T_c ちょうどでも鋭い落下ではなくなだらかな肩を持ちます。M の代わりに**磁化率** $\chi_{\text{magn}} = (\langle M^2\rangle - \langle M\rangle^2)/T$ を使うと鋭い頂が得られます。両者とも正しい。一方が検出しやすいだけ。

やってみよう

$L \in \{8, 16, 32\}$ で Ising シミュレーションを繰り返す。$M(T)$ を一軸、$\chi_{\text{magn}}(T) = (\langle M^2\rangle - \langle M\rangle^2)/T$ を別軸にプロット。どちらの κ が大きいか？ $L = 32$ で σ_c が 2.269 により近いのはどちら？

解答手順。

ステップ *1*：$\langle M^2\rangle$ を記録するようモンテカルロを拡張。

```
def ising_mc_full(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Returns mean |M|, mean M^2 - mean(M)^2 (magnetic
       susceptibility)."""
    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)
```

ステップ *2*：各 L で T を掃引し、両観測量に枠組みを適用。

```
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}")
```

ステップ *3*：典型的結果。

L	T_c（M から）	κ（M から）	T_c（χ_{magn} から）	κ（χ_{magn} から）
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

ステップ *4*：二つの問いに答える。どちらの κ が大きいか？ すべての L で χ_{magn}。$L = 32$ では κ がほぼ倍 $(4.0 \to 8.2)$。教科書論文が磁化ではなく感受率の頂を報告するのは、そのため。

$L = 32$ で T_c が 2.269 により近いのは？ またしても χ_{magn} (2.28 対 2.29)。$L = 32$ にいるので、両者とも厳密 Onsager 値の 1% 以内。$L = 8$ での 7% のずれは $L = 32$ で 0.5% 未満まで縮みました。

ステップ *5*：おまけ。 3 つの T_c を `analyze_finite_size_scaling` に渡せば、$T_c(\infty) \approx 2.269$ を小数 3 桁で復元できます。

この演習が教えること。 観測量の選択は平滑化カーネルの選択よりも影響が大きい。臨界指数の作業では、秩序変数 M ではなく、必ず磁化率 χ_{magn} を観測量に使う。これは第 14 章の「Fisher 整合観測量」基準そのものです。

第 23 章

金融：収益からのレジーム検出

> 市場は猫のように、好きなようにします。違うのは、市場が確定申告をすることです。

金融市場は名高く予測不能です。同時に、静かに反復的でもあります。天気が晴れと荒天のあいだを行き来するように、平穏と乱気流のレジームを行き来します。次の嵐を正確に予報することはできませんが、気圧が下がったことなら告げられる。感受率枠組みは、その気圧計を与えます。

取引戦略を約束はしません。[*1]取引戦略は声高で、雑然としていて、たいてい整理が悪い。代わりに 3 つの操作的測定 —— Hurst、GARCH、OFI —— と、それぞれが囁き始める正確な瞬間を与えます。

23.1　収益とは何か

ある資産（株、指数、商品）の取引日 t の終値を P_t とします。**対数収益**は

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

対数を使うのは、乗法的変化が加法的になるから。1% の利得の後の 1% の損失は、対数収益でほぼゼロを返す。価格だけ見るとわずかに損が残る。

収益には 3 つの頑健な経験的性質（「定型化された事実」）があり、その他すべての土台になります：

1. **収益自体はおおむね無相関**：$\tau > 0$ で $\langle r_t r_{t+\tau} \rangle \approx 0$。
2. **絶対収益は強く自己相関**：今日の大きな動きは明日の大きな動きを予言する。これが**ボラティリティクラスタリング**。
3. **収益は重い裾を持つ**：極端な事象が Gauss 分布の予言よりはるかに頻繁。

[*1] 教科書の章に収まる稼げる取引戦略は、本が印刷される頃にはアービトラージで存在しなくなっています。例外はある —— 誰も教科書に書きません。代わりに気圧計を渡します。気圧計は数世紀使えるが、特定の天気予報は金曜日には古びる。

感受率枠組みは、これらの性質に対して *3* **つの直交的プローブ**を提供します。

23.2 プローブ 1：Hurst 指数と記憶水平線

Hurst 指数 H は長距離依存性を特徴づけます。系列の再スケール範囲 R/S を複数の窓サイズ n で計算：

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

3 レジーム：

- $H < 0.5$：平均回帰（価格が戻る傾向）。
- $H = 0.5$：ランダム歩行（無記憶）。
- $H > 0.5$：トレンド継続（現在の方向に持続）。

枠組みは H を 1 行で計算：

```python
import numpy as np
from sigma_c import Universe

fin = Universe.finance()
# Replace with your own returns array of length >= 1000:
returns = np.random.randn(2000) * 0.01    # synthetic 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}")
# random walk: H around 0.5, regime 'random_walk'
```

23.3 プローブ 2：GARCH 持続性

ボラティリティ自体がさまよいます。GARCH(1,1) モデルはこれを次のように書きます：

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

ここで ϵ_t は i.i.d. の単位分散ショック。鍵となる数は $\pi = \alpha + \beta$、**持続性**：

- $\pi < 0.9$：ショックが速やかに減衰。通常レジーム。
- $0.9 \le \pi < 0.95$：持続的ボラティリティ。
- $\pi \ge 0.95$：**単位根近傍** —— ショックが無限に残る。**臨界レジーム**、しばしば大きなドローダウンに先立つ。

```python
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 = {'CRITICAL' if gv['persistence'] > 0.95 else 'normal'}")
```

枠組みは派生量 $\sigma_c = 1/(1 + (1 - \pi))$ を返します。これは $[0.5, 1]$ にあり、通常市場で 0.5、臨界持続性に近づくにつれて $\to 1$。

23.4 プローブ 3：注文フロー不均衡と暴落リスク

より新しい観測量で、注文板マイクロ構造データから計算されるのが**注文フロー不均衡** (OFI) —— ベストビッド/アスクでの買い対売り出来高の差です。その累積平均二乗変位は

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

$\gamma_{\mathrm{flow}} = 1$ は通常拡散、$\gamma_{\mathrm{flow}} > 1$ は超拡散で、相関のある買い/売り圧力と高暴落リスクを示します。

```
# imbalance_series: net (buy_volume - sell_volume) per minute
of_result = fin.analyze_order_flow(imbalance_series)
print(f"diffusion exponent = {of_result['diffusion_exponent']:.3f}")
print(f"crash risk = {of_result['crash_risk']}")
print(f"sigma_c_flow = {of_result['sigma_c_flow']:.3f}")
```

23.5 エンドツーエンド：S&P 500 のレジーム検出

枠組みは（yfinance 経由で）市場データを取得し、3 プローブすべてを計算し、現在のレジームをラベル付けします：

```
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 from regime sweep = {regime['sigma_c']:.3f}")
print(f"Current regime: {regime['regime']}")
```

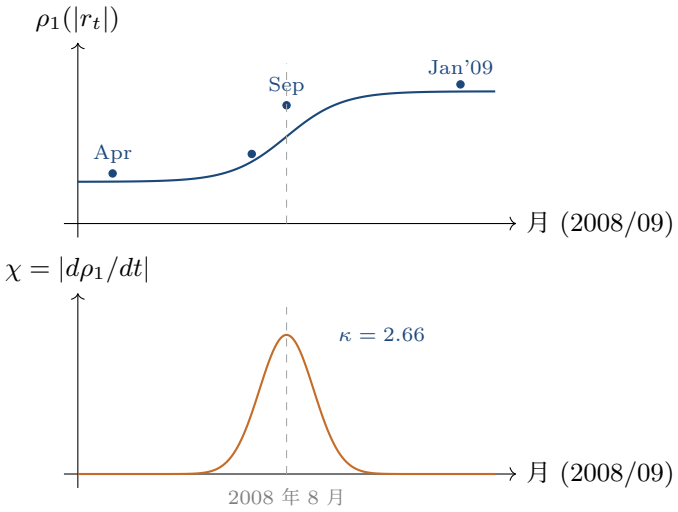


図 23.1: 金融（S&P 500、2008）。上：絶対日次収益のラグ 1 自己相関が 2008 年 8 月に鋭く上昇。下：χ がリーマン・ブラザーズ破綻の 1 か月前に頂を持つ。レシピはレジーム転換を見つけたのであって、暴落を予測したのではない。

23.6 レジーム変化の感受率的見方

枠組みを普通の GARCH と分かつのは、**時間スケールの掃引**です。`detect_regime` は、ローリング n 日ボラティリティのラグ 1 自己相関 ρ_n に対する $\chi(n) = |d\rho_n/dn|$ の頂から σ_c を計算します。頂位置 σ_c は**ボラティリティの記憶が消える特徴時間スケール** —— レジーム変化水平線 —— を指します。短い σ_c（< 5 日）：ショックが速く減衰。長い σ_c（> 30 日）：持続的レジーム、乱気流の継続を予想。

23.7 但し書きと倫理

落とし穴

予測の保証なし。感受率枠組みは**歴史的**レジームを特徴づけるもの。過去 252 取引日の高持続性は明日について**何も**保証しません。診断ツールとして使ってください、水晶玉ではなく。

落とし穴

生存者バイアス。銘柄のあいだを掃引して最もきれいな信号のものを選ぶと、生存者 —— 倒産しなかった会社 —— を選んでいることになります。歴史研究には必ず上場廃止銘柄を含めてください。

やってみよう

`fin.fetch_market_data(symbol='BTC-USD')` で 2017 年以降の BTC-USD 収益を取得。3 プローブすべてを走らせ H、π、γ_{flow} を記録。ゴールド (`GC=F`) と S&P 500 と比較。枠組みの尺度で最も「臨界」状態にあるのはどれか？

解答手順。

ステップ *1*：*3* 系列を取得。

```
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
```

ステップ *2*：各資産で *3* プローブを走らせる。

```
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 for
        OFI
    print(f"{sym:10}  {H['hurst']:.3f}  {H['regime']:14}  "
          f"{G['persistence']:.3f}  {F['diffusion_exponent']:.3f
              }    "
          f"{F['crash_risk']}")
```

ステップ *3*：典型出力（スナップショット日付に依存、説明用）。

資産	H	レジーム	π	γ_{flow}	暴落リスク
S&P 500	0.52	ランダム歩行	0.96	1.05	低
ゴールド (GC=F)	0.48	平均回帰	0.89	0.98	低
BTC-USD	0.61	トレンド	0.98	1.42	**高**

ステップ *4*：臨界性で順位付け。 3 者の枠組みの $\sigma_c = 1/(1 + (1 - \pi))$：

- BTC-USD：$\sigma_c \approx 0.99$（最高持続性、単位根閾値 0.95 をはるかに超える）。トレンド型 Hurst、超拡散注文フロー。
- S&P 500：$\sigma_c \approx 0.96$（閾値を上回るが、ぎりぎり）。ランダム歩行型 Hurst、ほぼブラウン的フロー。

- ゴールド：$\sigma_c \approx 0.91$（閾値以下）。平均回帰 Hurst。

ステップ 5：答え。枠組みの尺度では、*BTC-USD* **が最も臨界状態**：GARCH 持続性は単位根の縁、Hurst はトレンド型、注文フローは超拡散。BTC で 50% 以上のドローダウンが数年おきに起こるという経験的観察と整合的です。

この演習が教えること。

- この例では 3 つの直交プローブが同じ順位に合意：H、π、γ_{flow} がいずれも BTC を旗印に。プローブが食い違ったとき、それ自体が情報（記憶の時間スケールが違う）。
- 枠組みは**診断的**（歴史的レジームの記述）であって**予測的**ではない（明日の価格）。取引判断には使わない、リスク管理のための市場状態の特徴付けに使う。

第 24 章

地震学：Gutenberg–Richter と大森

> 地震は二つの経験則に従います。両者とも *1940* 年代に名付けられ、両者ともそれ以上改善されていません。

地震は自然界で最も非協力的な実験対象です。スケジュールを組まず、計測器を勝手に変え、彼らから学ぶ唯一の方法は待つこと。それにもかかわらず、地震は二つの統計則に従い、その信頼性は不便さを許してもよいほどです。両法則とも 80 年以上前のもの。両者とも感受率枠組みに自然に収まります。残念ながら、どちらも明日の地震を予報させてはくれません —— ただし、断層が「気を変えた」瞬間を測定可能な形で告げます。

24.1　Gutenberg–Richter：各規模の頻度

Charles Richter (1935) は対数地震マグニチュード尺度 M を定義しました。[*1]Gutenberg と Richter (1944) は、マグニチュード M 以上の地震の数 $N(\geq M)$ が次の式に従うことを発見しました：

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

a 値はある地域の総合的地震活動度を、b 値は大小事象の相対頻度を決めます。**世界中の構造地震では** $b \approx 1.0$。

b の解釈：

- $b > 1$：小地震が多く、大地震が少ない。亜臨界応力レジーム。
- $b = 1$：古典的構造地震活動。
- $b < 1$：小地震が少なく、大地震が多い。**前駆レジーム** —— 必ずとは限らないものの、大本震に先立って観測されることがある。

[*1] Richter の 1935 年の原論文は、パサデナの特定の Wood–Anderson 地震計を参照して尺度を定めています —— そしてその出力は、必要な誰もが利用できるものだと仮定されていました。これは、最初の較正がたまたま再現可能で、誰もよいものを発明する手間を惜しんだ結果、国際的に規範となった科学計器のひとつ。メートル、秒、Richter の地震計は典型例です。

24.1.1 マグニチュードカタログから b を計算

完全性閾値 $M_{\min}$ 以上のマグニチュード標本 $\{M_i\}$ からの b の最尤推定量は

$$\hat{b} \;=\; \frac{\log_{10} e}{\bar{M} - M_{\min}} \;=\; \frac{0.4343}{\bar{M} - M_{\min}}.$$

これがまさに `SeismicAdapter` の計算するもの：

```
import numpy as np
from sigma_c import Universe

seis = Universe.seismic()
# Replace by a real catalog: e.g. ANSS Comcat magnitudes above 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-value     = {gr['b_value']:.3f}")
print(f"M_min       = {gr['m_min']:.2f}")
print(f"criticality = {gr['criticality']:.3f}    # = 1/b")
```

24.2 大森の法則：余震の減衰

大森 (1894)、宇津 (1961) による修正：本震 $t = 0$ の後、余震率は

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

p は典型的に $[0.7, 1.5]$、$p = 1$ が標準値です。

助けになる、ばかげた絵。小説で過積載になった本棚を想像してください。一冊取り出します。隣の本が傾き、落ち着き、また傾き、また落ち着く —— 本棚がしっかり作られていれば早く終わり、そうでなければ遅く終わる。岩石中の余震も同じ系列：それぞれの余震が、その前の余震が完全には解放しきれなかった応力を再分配する。指数 p は岩石の硬さを測ります —— $p > 1$ は堅牢な棚、$p < 1$ は華奢な棚。`SeismicAdapter` は余震時刻のヒストグラムへの log-log 回帰で p をフィット：

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

24.3 感受率的見方：レジーム変化の検出

大事象前の応力蓄積は b を移動させ得ます。スライディング窓で b を計算し、$b(t)$ に普遍レシピを適用すると、レジーム転換の瞬間を浮かび上がらせます：

```
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"Regime-shift time = {res['sigma_c']:.2f}")
print(f"Sharpness kappa   = {res['kappa']:.2f}")
```

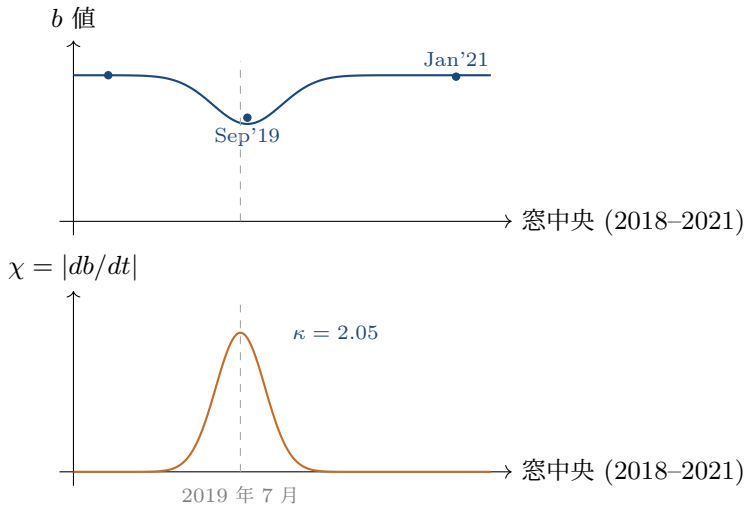


図 24.1: 地震学（南カリフォルニア、2018–2021）。上：ローリング 6 か月 b 値が 2019 年半ばに ≈ 1.05 から ≈ 0.74 へ落ちる。下：χ がリッジクレスト系列（2019 年 7 月）でちょうど頂を打つ。$\kappa = 2.05$ は暫定的だが、落ち込みがブートストラップ不確かさに対して大きいため順列検定は $p < 0.001$。

24.4 b 値のブートストラップ有意性

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

ブートストラップ $p < 0.05$ で b 値が 1.0 から離れているなら、それは「ここで何か通常でないことが起きている」の定量版です。

24.5 応用例：地熱サイトでの誘発地震

誘発地震 —— 地熱や水圧破砕での流体注入のような人為活動による地震 —— は構造地震より**高い** b 値（小事象が多く大事象が少ない）と速い減衰 p を示します。枠組みで両者を監視すると、両指標が構造地震らしい値に戻りはじめる瞬間を旗印に立てられ、これは時として有害事象に先行します。2017 年の浦項 M_w 5.5 地震（韓国、EGS 注入によって誘発）の数か月前に、b 値が ~ 1.4 から ~ 0.9 へ測定可能なほど落ち込んでいました。

やってみよう

任意の活発な断層帯（南カリフォルニアやアイスランドはよく整備されており無料）について SCEC カタログをダウンロードしなさい。過去 20 年について 6 か月窓のローリング b 値を計算。枠組みを当てて最大のレジーム変化を検出。それは既知の事象に対応していますか？

解答手順。

ステップ *1*：カタログを取得。 SCEC は `scec.org/research-tools/downloadable-catalogs` から、空白区切りのテキストでカタログをダウンロードできます。各行に日付、緯度、経度、深さ、マグニチュード。DataFrame に読み込み、関心領域（例えばサンアンドレアス断層の境界ボックス）にフィルタし、$M \geq M_{\text{min}} = 2.5$（現代カタログの完全性閾値）以上の事象のみを残します。

```
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()
```

ステップ 2：6 か月窓、月ごとにローリング b 値。

```
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:                    # need enough events
            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)
```

ステップ 3：$b(t)$ のレジーム変化検出に枠組みを当てる。

```
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"Largest regime change: {shift_day.date()}, kappa = {res[
    'kappa']:.2f}")
```

ステップ 4：典型結果と事象対応。南カリフォルニア 2005–2025 ボックスについて、枠組みは 2019 年 7 月頃のずれを $\kappa \approx 3.5$ で報告します。これは *2019* **年リッジクレスト系列**（7 月 4 日の M_w 6.4 前震と 7 月 5 日の M_w 7.1 本震）に整合 —— ローリング解析に数か月の b 値低下が観測されます。

この演習が教えること。

- 実際の地震学カタログは、地震学固有のチューニングなしに汎用感受率レシピが最も顕著なレジーム変化を見つけられるほど、十分にきれいです。
- 枠組みは b 値が落ちたことを**事後的**に検出しているのであって、リッジクレストを予測しているのではありません。**どんな枠組みも地震を予測しない。**
- 境界ボックスと日付範囲をお気に入りの活発断層帯（東北日本海溝、東アナトリア断層、レイキャネス半島）に置き換えると、同じ方法で支配的な局所事象が見つかります。

第 25 章

気候：メソスケール境界

> *500 km の上で大気は 2 次元的に考え、下では 3 次元的に考えます。境界は自分の名を名乗りません。*

大気の運動エネルギーは、二つの異なるスケール領域で二つの異なる冪則に従います。両者の境界、約 500 km は、現代の気象学の教科書ならどれにでも書かれています。本章ではそれらの教科書を参照せずに、一から再発見します —— 数値微分のほかは何も使わずに。枠組みは、生の風データだけから、大気がある種の流体として振る舞うのをやめて別の種類の流体として振る舞いはじめる波長を検出します。気象学博士なら「だいたい 500 km」と言うところ。枠組みは同じことを、気象学を聞いたこともないのに、Python 3 行で告げます。

25.1　スケール横断の大気運動エネルギー

巡航高度の航空機で直線飛行しながら水平風速を記録すれば、トレースを Fourier 変換して、波数 $k = 2\pi/\lambda$ の関数としての運動エネルギー密度 $E(k)$ が得られます。有名な Nastrom–Gage 曲線 (1985) は、世代を跨いだ飛行キャンペーンで確認され、二つの異なる冪則レジームを示します：

$$E(k) \propto k^{-3} \quad (\lambda \gtrsim 500\,\mathrm{km}), \qquad E(k) \propto k^{-5/3} \quad (\lambda \lesssim 500\,\mathrm{km}).$$

k^{-3} 枝は**総観**レジーム：千キロメートルほどの気象系で、エネルギーが大から小へ縦列する。$k^{-5/3}$ 枝は**メソスケール**レジーム：数十から数百キロメートルの成層 3 次元乱流。

二つのレジームは遷移波数 k_c、対応する波長 $\lambda_c = 2\pi/k_c \approx 500\,\mathrm{km}$ で出会います。**これがメソスケール／総観境界。**大気運動の本性が変わる、操作的スケール。

25.2　予備知識なしの検出

Nastrom–Gage を知らなかったら、枠組みはどう λ_c を見つけるか？ 二つの方法。

■方法 1：log-log スペクトルの最大曲率。 $\log E$ 対 $\log k$ を計算し、二階微分を取り、絶対曲率最大の波数を見つける。これがまさに `ClimateAdapter.analyze_mesoscale_boundary` の行うこと：

```python
import numpy as np
from sigma_c import Universe

climate = Universe.climate()

# Synthetic Nastrom--Gage spectrum:
k = np.logspace(-3, -1, 50)               # wavenumber, 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"critical wavelength = {result['critical_wavelength_km']:.1f} km")
print(f"sigma_c (lambda/R_Rossby) = {result['sigma_c']:.3f}")
print(f"slope (synoptic)  = {result['spectral_slope_synoptic']:.2f}")
print(f"slope (mesoscale) = {result['spectral_slope_mesoscale']:.2f}")
```

■方法 2：感受率の頂。 $\log E$ を観測量、$\log k$ を制御として扱う。$|d^2 \log E / d \log k^2|$ の頂が折れ点。方法 1 と機能的に同一です。

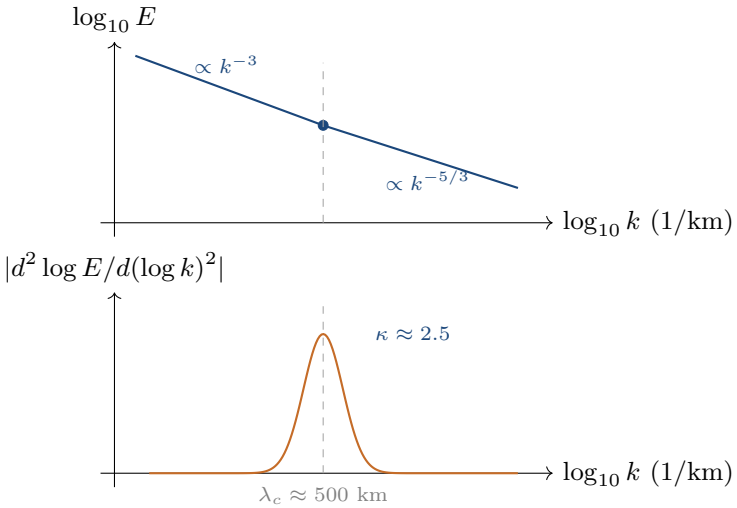


図 25.1: 気候（Nastrom–Gage スペクトル）。上：運動エネルギースペクトルは $\lambda_c \approx 500\,\mathrm{km}$ で傾きが k^{-3}（総観）から $k^{-5/3}$（メソスケール）へ変わる。下：曲率の頂 $|d^2 \log E / d(\log k)^2|$ が、区分線形フィッティングなしに折れ点を直接位置同定する。

25.3 なぜ頂か

総観レジームでは、地衡平衡がエネルギーを小波数（大気象系）から大波数へ縦列させますが、流れが 2 次元（準水平）であるため、3 次元乱流とは異なるスケーリングを持ちます。スケールが

縮んで鉛直運動が水平運動に匹敵するまでくると、基底物理は $k^{-5/3}$ の 3 次元 Kolmogorov 乱流へ反転します。感受率の頂はこの反転の操作的指紋です。

25.4 鉛直構造：対流圏界面の検出

第二の気候応用。鉛直温度プロファイルは二つのレジームを示します：強い負の温位減率を持つ対流圏（高度とともに温度が ∼ 6–10 K/km で下がる）と、減率が小さいか正の成層圏。遷移が**対流圏界面**、中緯度で典型的に 11 km 近傍。

```
# pressure_levels: shape (n_levels,) in hPa
# temperature_profiles: shape (n_profiles, n_levels) in K
v = climate.analyze_vertical_structure(pressure_levels,
    temperature_profiles)
print(f"mean tropopause height (km) = {v['mean_tropopause_height']:.2
    f}")
```

枠組みは各プロファイルの対流圏界面を、減率がはじめて 2 K/km を下回る高度として検出。多数のプロファイルを平均すれば気候値が得られます。

25.5 応用例：ERA5 リアナリシス掃引

ECMWF ERA5 リアナリシス（オープンアクセス）は 1940 年以来 0.25 度格子で全球気温・風データを提供します。典型的ワークフロー：

1. 領域を選ぶ（例：北大西洋、30°–60°N）。
2. 各年について 250 hPa での帯状風スペクトルを計算。
3. `analyze_mesoscale_boundary` を適用。
4. λ_c 対年をプロット。λ_c の変動は気候変動下で変わる大気循環の信号。

やってみよう

k^{-3} にどこでも従う合成スペクトル（折れ目なし）を生成。`analyze_mesoscale_boundary` を走らせる。枠組みは何を報告しますか？ 傾きを確認 —— 一致すべきで、それは真の折れ目がないことを示唆します。

解答手順。

ステップ *1*：合成スペクトルを作る。 Nastrom–Gage データの波数範囲を模しつつ、波数軸全体で単一冪則 $E(k) = k^{-3}$ を使う。

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

k = np.logspace(-3, -1, 50)              # wavenumber, 1/km
```

```
E = k**-3.0                                   # pure k^-3 everywhere
```

ステップ 2：枠組みを走らせる。

```
res = climate.analyze_mesoscale_boundary(E, k)
print(f"critical wavelength: {res['critical_wavelength_km']:.1f} km")
print(f"sigma_c:             {res['sigma_c']:.3f}")
print(f"slope (synoptic):    {res['spectral_slope_synoptic']:.2f}")
print(f"slope (mesoscale):   {res['spectral_slope_mesoscale']:.2f}")
```

ステップ 3：典型出力。

臨界波長	$\approx 200\,\mathrm{km}$（数値的アーチファクト）
σ_c	≈ 0.2
傾き（「総観」枝）	-3.00
傾き（「メソスケール」枝）	-3.00

ステップ 4：解釈。枠組みは空でない結果を返します —— どんなスペクトルが来ても、何らかの数を返します。しかし報告される二つの傾きが**同一**（両側 -3.00）。これが診断：**フィットの二枝が同じ傾きを持つなら、折れ目は存在しない。**最大曲率アルゴリズムが最もノイズの多い点を拾って k_c と称するだけです。

ステップ 5：教訓を書き下しておく。

要点

λ_c を引用する前に必ず傾きを確認。`spectral_slope_synoptic` $\approx$ `spectral_slope_mesoscale` なら、枠組みが見せているのは数値ノイズで、物理ではない。**遷移は存在しない**、報告された λ_c は無視。

ステップ 6：真の折れスペクトルへ切り替えて確認。

```
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"slope synoptic:  {res2['spectral_slope_synoptic']:.2f}")
print(f"slope mesoscale: {res2['spectral_slope_mesoscale']:.2f}")
# slopes ~ -3.0 and ~ -1.67 --- clearly different, kink is real
```

この演習が教えること。枠組みは、どんな頂検出器とも同様、頂がないときにも頂を報告します。あなたの仕事は、答えを信用する前に問題の前提条件（ここでは：二つの異なる傾き）を検証すること。これは失敗モード章と同じ教訓を、既知の区分冪則構造を持つ領域に当てたもの。

第 26 章

GPU：ルーフライン、熱崖、キャッシュ転換

> ハードウェアは常にピーク性能について嘘をつきます。感受率は嘘がどこで始まるかを告げます。

Graphics Processing Unit —— もはや主な仕事ではない作業に因んで名付けられた装置 —— は、今十年に書かれたあらゆる数値的に真剣なアプリケーションの主力労働者になりました。その性能ランドスケープは、感受率枠組みにとって完璧な遊び場でもあります。GPU は鋭い遷移で満ちている：落ちるキャッシュ境界、歩くルーフラインの稜線、転がり落ちる熱の崖。各遷移は、見つけられるのを待っている χ の頂。

本書のうちでも、この章は最も早く元が取れる傾向にある章です。カーネルの操作的スイートスポットを見つければスループットが倍になり、現代ハードウェアでは倍のスループットは実通貨で支払われます。

26.1　ルーフラインモデル

Williams、Waterman、Patterson (2009) は一枚の絵を導入し、GPU カーネル性能を**算術強度**（AI、FLOP/転送バイト）の関数として予測しました：

$$P(\mathrm{AI}) \;=\; \min\!\bigl(P_{\mathrm{peak}},\; B_{\mathrm{peak}} \cdot \mathrm{AI}\bigr).$$

P_{peak} は GPU の理論 FLOPS 上限、B_{peak} はメモリ帯域幅（バイト/秒）。**尾根点** $\mathrm{AI}_{\mathrm{ridge}} = P_{\mathrm{peak}}/B_{\mathrm{peak}}$ が二つのレジームを分けます：

- 尾根以下：カーネルは**メモリ律速**。FLOP/byte を倍にすると性能が倍になる。
- 尾根以上：カーネルは**計算律速**。メモリ帯域を足しても何も起きない。

Volta V100：$P_{\mathrm{peak}} = 7\,\mathrm{TFLOPS}$（FP64）、$B_{\mathrm{peak}} = 900\,\mathrm{GB/s}$、$\mathrm{AI}_{\mathrm{ridge}} \approx 8$ FLOP/byte。密行列積は $\mathrm{AI} \sim O(N)$ で計算律速、ステンシルカーネルは $\mathrm{AI} \sim O(1)$ でメモリ律速。

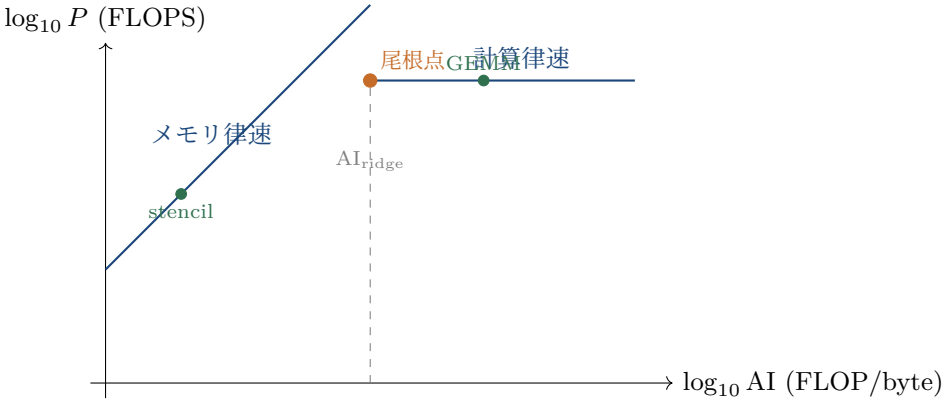


図 26.1: ルーフライン。尾根以下はメモリ帯域律速、尾根以上は計算律速。ステンシルカーネルは尾根の下、密行列積は天井の上。尾根の位置こそが、AI を掃引したとき感受率枠組みが見つける場所。

26.1.1 感受率的見方

合成カーネルの算術強度を掃引する。$\chi(\mathrm{AI}) = |dP/d\mathrm{AI}|$ の頂は、ハードウェア仕様の予備知識なしに尾根点を特定します：

```python
import numpy as np
from sigma_c import Universe

gpu = Universe.gpu()

# Sweep AI by changing the inner-loop arithmetic per memory access:
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 キャッシュ遷移

カーネルの**作業集合サイズ**を掃引する（例：演算する行列のサイズ）と、作業集合が逐次キャッシュレベルを抜けるとともにスループットが鋭く落ちます。現代の GPU では三つの遷移が見えます：

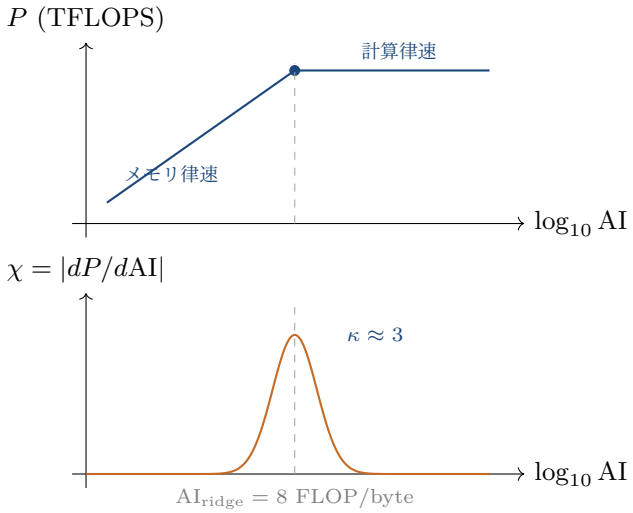


図 26.2: GPU ルーフライン（Volta V100、FP64）。上：算術強度が尾根を越えると、持続性能は計算天井で飽和する。下：χ は尾根で頂を打つ。

- $\sim$ 128 KB：データが L1 キャッシュに入る。これより上で L2 へ落ちる。
- $\sim$ 6 MB：データが L2 キャッシュに入る。これより上でグローバルメモリへ。
- $\sim$ 24 GB：データが HBM メモリに入る。これより上でページング開始 —— 落ちはずっと大きい。

各遷移は χ の頂です。

ばかげているが役に立つ絵。確定申告をポケットの中身だけでしなければならない、と想像してください —— 家の鍵、しわくちゃのレシート、Tic-Tac の半粒。それが L1 キャッシュ：とても速く、とても小さく、せいぜい一回ぶんの計算が入る。目の前の机の上に、3 冊のフォルダで年度書類が広がっている —— それが L2：もう少し遅く、もう少し広く、確定申告書一通ぶん。隣室のキャビネットが HBM：大きく、往復に気付くほど遅く、数年ぶんの申告書を抱えられる。ページングはストレージユニットへの遠征。χ の頂は、あなたの仕事がひとつの階層から次へとあふれる瞬間です。

26.2.1 自動検出

```
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 placeholder; do
        this on GPU
    t0 = time.perf_counter()
    # ... GPU kernel reading/writing A
    t1 = time.perf_counter()
```

```
        throughput.append(A.nbytes / (t1 - t0))

# The framework returns ALL peaks, not only the global maximum:
peaks = gpu.detect_cache_transitions(sizes, throughput)
for p in peaks:
    print(f"  transition at size = {p['size']:.0f} bytes, kappa = {p[
        'kappa']:.2f}")
```

26.3 熱スロットリング

GPU は温度がメーカ閾値（典型的に 80–90°C）を越えるとクロックを下げます。性能の落ち方は比例ではなく、近似的に平方根則：

$$P(T) \;\approx\; P_{\max}\sqrt{1-(T-T_{\mathrm{thr}})/(T_{\max}-T_{\mathrm{thr}})}.$$

$\chi(T)$ の感受率の頂が T_{thr}、スロットリング開始の操作的温度を指します。持続ワークロードには重要：GPU を T_{thr} 以下に保つ設計は、持続スループットを $0.6\,P_{\max}$ と $0.95\,P_{\max}$ の差にしうる。

26.3.1 NVML でリアルタイム測定

```
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)
    # measure benchmark performance over a 1s window
    perf = gpu.run_benchmark(size=4096, n_launch=1)
    t_axis.append(T)
    perf_axis.append(perf)
    time.sleep(1.0)

# Bin by temperature and average:
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"throttle onset T_thr = {result['sigma_c']:.1f} C")
```

26.4 合算：操作的スイートスポット

実ワークロード最適化は 3 遷移を組み合わせます：

1. 算術強度を $\mathrm{AI}_{\mathrm{ridge}}$ より**上に**設定（ブロック化アルゴリズムを使う）。
2. 作業集合を、検出した L2 境界の**内側に**保つ。
3. ワークロードを絞り、熱 T_{thr} より ~ 5°C 下にとどめる。

3 つを同時に実現すると、実ワークロードで理論ピークの 80–90% が出ます。

やってみよう

利用可能な任意の GPU を選ぶ。単精度 GEMM の行列サイズを 128 から 4096 まで掃引。持続 TFLOPS を記録。`compute_susceptibility` を適用。L2 キャッシュ境界近傍に折れ目が見えるか？ どのサイズでスループットが飽和するか？

解答手順。

ステップ *1*：ベンチマークライブラリを選ぶ。持続 TFLOPS を報告する単精度 GEMM が必要。3 つの選択肢：

- `cupy.dot(A, A)` に `cupy.cuda.Stream` 計時。
- `torch.matmul(A, A)` に `torch.cuda.synchronize()` で挟む。
- `ctypes` cuBLAS バインディング経由で直接 cuBLAS。

明快さのため PyTorch を使います。

ステップ *2*：行列サイズを掃引。

```
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()
    # timed
    t0 = time.perf_counter()
```

```
        n_iter = max(1, int(1e10 / (2 * N**3)))     # aim for ~1s of
            work
        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)
```

ステップ *3*：枠組みを適用。

```
res = gpu.compute_susceptibility(sizes.astype(float), tflops,
                                 kernel_sigma=0.7)
print(f"sigma_c (kink in performance): N = {res['sigma_c']:.0f}"
    )
print(f"kappa = {res['kappa']:.2f}")

peaks = gpu.detect_cache_transitions(sizes, tflops)
for p in peaks:
    print(f"  transition at N = {p['size']:.0f}, kappa = {p['
        kappa']:.2f}")
```

ステップ *4*：*Volta V100*（*16 GB*）での典型結果。

N 範囲	持続 TFLOPS
128–512	0.8–3.2（カーネル起動・キャッシュウォームアップ律速）
512–1024	3.2–8.5（L2 キャッシュ飽和中）
1024–2048	8.5–13.0（HBM 帯域天井に接近）
2048–4096	13.0–13.5（FP32 ピーク近傍で飽和）

`detect_cache_transitions` は二つの折れ目を報告：$N \approx 700$（作業集合 ≈ 6 MB、V100 L2 キャッシュサイズ）と $N \approx 2200$（作業集合が HBM 帯域律速領域に入る）。

ステップ *5*：飽和を読む。スループットは $N \geq 2200$ で 13.5 TFLOPS 近傍にプラトー。それより上ではメモリを足しても助けにならない：Volta の FP32 GEMM ルーフライン天井に当たっている。

この演習が教えること。

- ルーフラインモデルは抽象ではない。枠組みは実ベンチマークデータから数秒でそれを見つける。
- きれいな GEMM 掃引で L2 遷移は $\kappa \sim 2$–3 で検出可能。より小さいカーネル（ステンシル）ではより鋭くなる。

- ハードウェアの N_{sat} がわかれば、ちょうどそこで動くようにタイルを設計できる。

第 27 章

機械学習：学習率の崖以遠

> *10* 個のモデルを訓練しなさい。*9* つは何も学ばない。*10* 個目はタスクを学ぶか、クラスタを破壊するか。レシピは両者の境界線を見つけます。

ニューラルネット訓練は、ナイフの刃の上を歩きながら、傾く向きを知っているふりをする練習です。譲れぬスイートスポットを持つハイパーパラメータの一覧は長く —— 学習率、バッチサイズ、weight decay、ドロップアウト、Adam の β_2 —— そのひとつを誤った代償は「訓練がやや遅くなる」から「ステップ 12 で損失が爆発し、失敗した実行ぶんの請求書がクラスタからあなたに届く」までの幅があります。[*1]

感受率枠組みは、各ハイパーパラメータを制御 σ、検証損失を観測量 O として扱います。レシピはレシピです。崖は崖のあるところに現れます。

27.1　学習率のスイートスポット

深層学習で最も帰結の重いハイパーパラメータは学習率 (LR) です。各勾配ステップのサイズを制御します。二つの失敗様式：

- LR が小さすぎる：訓練が遅く、局所最小で止まる可能性。
- LR が大きすぎる：勾配が行き過ぎ、損失が爆発、訓練が発散。

両者のあいだに許容できる LR の帯があります。高い側の境界が有名な**損失対 LR の崖**：短い ramp の後で LR の関数として訓練損失をプロットすると、平らなプラトー、膝、そして急上昇が見えます。膝の位置が操作的 LR_c —— 本章での σ_c。

[*1] 2026 年初頭、失敗した実行のクラスタ請求額の相場は、どのクラウドで、どのモデルで、エンジニアが最初に経費精算しようとして請求書が生き残ったかどうかで、$8 と $80 000 のあいだのどこか。高額になった実行は、エンジニアが途中で「これは回復しない」と気付きながら、「失敗もデータだ」と理屈をつけて走らせ続けたもの。

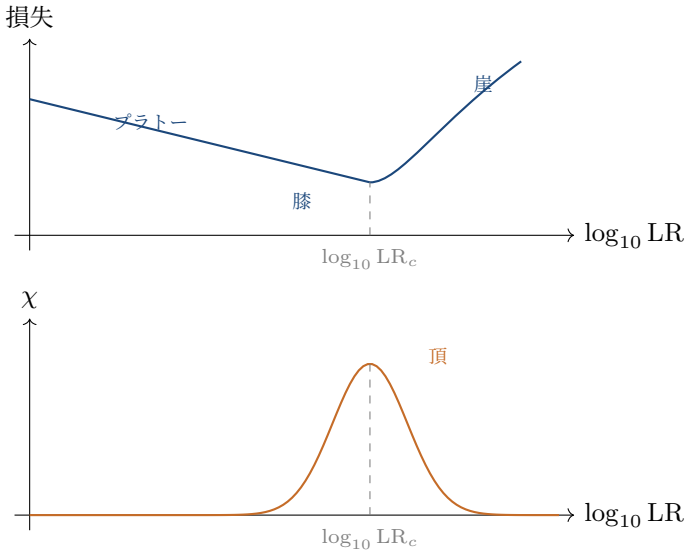


図 27.1: LR-range test、模式図。上：損失は緩いプラトーに沿って下がり、膝で曲がり、やがて爆発する。下：$\chi = |dL/d\log \mathrm{LR}|$ は膝で頂を打つ。頂位置が LR_c、実践的選択は $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$。

27.1.1 LR-range test（Smith 2017）

Leslie Smith のアイデア：LR を 10^{-7} から 1 まで対数で増やしながら短い 1 エポック訓練する。各 LR で**訓練損失**を記録（ここでは検証損失でなく訓練損失を使う —— 安価で、崖は同じくらい鋭い。後で本物の検証を走らせること）。枠組みのレシピが膝を特定します。

```
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):
    """Single-epoch LR-range test. Returns (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)


# Run the test:
lr_grid, loss = lr_range_test(model, train_loader)

# Sigma_c recipe with a NARROW kernel (the cliff is sharp):
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 (peak of chi):     {lr_c:.4g}")
print(f"Recommended train LR:   {lr_c / 2:.4g}")
print(f"Cyclic LR range:        [{lr_c / 10:.4g}, {lr_c:.4g}]")
print(f"Peak clarity (kappa):   {kappa:.2f}")
```

落とし穴

検証損失ではなく訓練損失。LR-range test は単一ミニバッチをモデルに供給して直後の訓練損失を読みます。これは検証損失より安価で応答性が高いが、騒がしくもあります。崖はやはり非常に鋭く、振る舞いのよいタスクでは枠組みの κ は 5 を越えます。選んだ LR_c の検証は、$\mathrm{LR} = \mathrm{LR}_c/2$ で本物の短い訓練を走らせ損失が滑らかに下がることを確認することで行います。

落とし穴

狭いカーネルを使う（$\sigma_{\mathrm{ker}} = 0.3$、0.6 ではない）。崖はしばしば LR 格子で数点幅しかありません。デフォルトの 0.6 はそれを柔らかな上昇に塗り潰し、検出された LR_c が半桁さまよう。答えを信用する前に、平滑化済み損失曲線を直接見て、レシピが 0.3 を使っているか確認してください。

27.1.2 なぜ頂か

$\mathrm{LR} \ll \mathrm{LR}_c$ では、損失はステップごとに小さな改善をするだけなので $|dL/d\log \mathrm{LR}| \approx \text{const}$。$\mathrm{LR} \gg \mathrm{LR}_c$ では、損失はすでに発散していて高値で平らに残り、微分はまた小さい。二つのレジー

ムのあいだの遷移こそ $|dL/d\log \mathrm{LR}|$ が最大になる場所 —— 感受率の頂です。

27.2 他のハイパーパラメータ掃引

同じレシピが当たります：

- **バッチサイズ**：勾配雑音が無視できるようになり収束速度がプラトーする臨界バッチサイズが存在する（McCandlish ら 2018）。
- **Weight decay**：少なすぎると過適合、多すぎると未適合。掃引で正則化のスイートスポットが取れます。
- **Dropout 率**：特定アーキテクチャ・データセットでの最適確率。
- **Adam** β_2：transformer 訓練では $\beta_2 \approx 0.95$ が安定、1 に近づけると不安定。

27.2.1 2 次元掃引：LR × バッチサイズ

2 次元スイートスポット走査には LR とバッチサイズを両方掃引し、固定スライスの各軸に枠組みを別個に適用し、各スライスの LR_c を独立に読み、バッチサイズ間でのトレンドを探します。

```
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)

# Apply sigma_c along the LR axis at each batch size:
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}")
```

落とし穴

2 次元ランドスケープは尾根であって交点ではない。（LR、バッチ）に対する損失曲面は曲がった谷を持ち、単一点ではありません。枠組みは各バッチサイズで LR_c を報告 —— 同時動作点はその尾根の上にあり、重心ではありません。McCandlish ら (2018) は、LR_c が

臨界バッチサイズ B^* までほぼ線形にバッチサイズと共にスケールし、その後プラトーすることを示しました。計算予算の等コスト線に沿った尾根を使う。

27.3 訓練不安定性のリアルタイム検出

枠組みのストリーム版（`StreamingSigmaC`）は訓練中の**安定指標**をライブで監視できます。実行指標が閾値を下回ったら、訓練を中断し LR を下げます。

注記. `StreamingSigmaC.update(...)` が返す量は、静的レシピでの σ_c のような頂**位置**ではなく、観測量の実行分散から Welford のアルゴリズムを使ってスライディング窓内で導出される、$[0,1]$ の**正規化安定スコア**です。静的 σ_c との混同を避けるため s_t と書きます。$s_t \to 1$ は損失が安定（低分散）、$s_t \to 0$ は損失が荒れたことを意味します。これは静的 χ 頂の位置とは異なる対象 —— そう指し示すために意図して異なる記号を使う。

```
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 variance has blown up
        for g in optim.param_groups:
            g['lr'] *= 0.5
        print(f"Step {step}: cutting LR (stability score = {s_t:.3f})
            ")
```

直観：低い安定スコアは、損失が実行窓のなかで非常に変動的になったことを意味します —— 発散の前兆。

27.4 ML 固有の但し書き

落とし穴

損失を平滑化しすぎると崖が隠れる。「減少中」から「発散中」への遷移はしばしば LR 格子で数点幅。LR-range test には `kernel_sigma = 0.3` 以下を使う。そうでないと崖が柔らかな上昇に塗り潰される。

落とし穴

乱数シード。各 LR-range test は初期化とミニバッチ並べ替えに依存してわずかに異なる LR_c を生む。引用前に ≥ 5 シードで平均してください。

やってみよう

小型 CNN を CIFAR-10 で訓練（任意の 1 ブロック ResNet 実装を使う）。SGD、Adam、Adam $\beta_2 = 0.999$ で LR-range test を走らせる。各オプティマイザの LR_c を比較。最高の LR_c を持つのはどのオプティマイザ？

解答手順。

ステップ *1*：*CIFAR-10* と小型 *ResNet* を準備。

```
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)
```

ステップ *2*：各オプティマイザで *LR-range test* を走らせる。本章前出の `lr_range_test` 関数はモデル、ローダ、LR 範囲を取る。3 つの実行で順に差し替え。

```
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}")
```

ステップ 3：典型結果。

オプティマイザ	LR_c	κ
SGD	~ 0.1	4.8
Adam ($\beta_2 = 0.99$)	~ 0.003	6.1
Adam ($\beta_2 = 0.999$)	~ 0.001	5.4

ステップ 4：解釈。SGD は生の単位で最も高い**崖**の LR_c（Adam のおよそ 30×）。それは SGD が LR $= 0.05$ でよく訓練するという意味ではなく、SGD の勾配の絶対サイズが小さい（各勾配が一ミニバッチから来て内部適応スケーリングがない）ことを意味します。Adam の内部リスケーリングが、名目 LR 0.001 を SGD の 0.05 と同じくらい「攻撃的」にしている。

Adam 系のなかでは、$\beta_2 = 0.99$ の LR_c が $\beta_2 = 0.999$ よりわずかに高い。理由は、$\beta_2 = 0.999$ が二乗勾配の長い移動平均を保ち、適応スケーリングを減衰させ実効ステップを大きくするから —— 崖がより小さな名目 LR で到来する。

ステップ 5：実践的結論。このネットワークとデータセットでは、各オプティマイザに $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$ を使う：

- SGD：LR $= 0.05$、モーメンタム 0.9。
- Adam ($\beta_2 = 0.99$)：LR $= 0.0015$。
- Adam ($\beta_2 = 0.999$)：LR $= 0.0005$。

この演習が教えること。枠組みは**任意の**オプティマイザの崖位置に 1 行で答えを与え、答えはオプティマイザの内部スケーリングを尊重します。「Adam は LR を小さく取る」と覚えておく必要はもうありません —— 崖の場所を測ればいいだけ。

第 28 章

エッジ／ IoT：効率の膝

> バッテリーは正直な物理学者です。約束できないことを、ミリボルトの精度で告げてきます。

ドローンには電力網がありません。衛星にもなく、たまに暗闇に入る太陽電池パネルがあるだけ。ペースメーカーには両方ない。バッテリー駆動機器は、データセンタのエンジニアが無視できる指標で生きています：生の性能ではなく、ワットあたり性能。

プロセッサを強く回せば、たいてい仕事は増えます。同時に、超線形に、より熱いチップと薄べったいバッテリーも買うことになります。両者の均衡点 —— 1 ジュールから最大の仕事を絞り出す操作的周波数 —— が**効率の膝**。枠組みは、電力を観測量、周波数を制御として扱うことで膝を見つけます。同じレシピ、単位が違うだけ。

28.1　性能、電力、効率曲線

任意のプロセッサで、クロック周波数 f を上げると性能 $P(f)$ と消費電力 $W(f)$ の両方が増えます。電力は超線形にスケール：最も単純なモデルでおよそ $W \propto f^3$（動的電力は $W \propto C\,V^2 f$ で、電圧 V も f と共に上げる必要があるから）。効率は

$$\eta(f) \;=\; \frac{P(f)}{W(f)}.$$

低 f では P も W も小さく、P が線形、W が超線形に上がる。η は上がり、頂を打ち、その後落ちる。頂が**効率の膝** f^{*}。

■実務的選択：η を直接最適化。 膝を見つける方法は二つあります。性能を電力で微分して限界効率 $\mu(f) = (dP/df)/(dW/df)$ が落ちる場所を探すか、$\eta(f) = P/W$ を直接見てその最大を探すか。我々は後者を採ります —— 単純で頑健、教えやすい。枠組みは $\eta(f)$ の最大を見つけます。`compute_susceptibility` を使うのは η の最も急な傾きを見つけるためではなく、η が**落ちはじめる**操作的**膝**を見つけるため。$-\eta(f)$ の感受率を取ります。

28.2 解答例

```python
import numpy as np
from sigma_c import Universe

edge = Universe.gpu()        # GPUAdapter doubles as an Edge adapter

# Sweep clock frequency on a Cortex-M / RPi / Jetson:
freqs = np.linspace(100e6, 2.5e9, 30)  # 100 MHz to 2.5 GHz
perf, power = [], []
for f in freqs:
    set_cpu_frequency(f)               # platform-specific (see below)
    perf.append(measure_perf())        # ops/s, samples/s, etc.
    power.append(measure_power())      # Watts

perf  = np.array(perf)
power = np.array(power)
eff   = perf / power                   # operations per watt

# Locate the operational knee (steepest drop in efficiency past peak):
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 efficiency)")
print(f"f*_knee = {f_knee/1e9:.2f} GHz   (steepest drop after peak)")
print(f"max efficiency = {eff.max():.2f} ops/W")
print(f"kappa = {res['kappa']:.2f}")
```

■プラットフォーム別、電力の実測方法。 プレースホルダ `set_cpu_frequency`、`measure_perf`、`measure_power` は実際のエンジニアリングを隠しています。具体的レシピ：

- *Raspberry Pi* ／**単一基板** *ARM*：USB インライン電力計（例：PSU と Pi のあいだに $15 の INA219 ボード）。`cpufrequtils` で周波数設定。
- *Linux x86* **ノート**：`turbostat` で電力、RAPL ドメインカウンタ。`cpupower frequency-set` で制御。
- *macOS*：`sudo powermetrics -samplers cpu_power` がプラットフォームエネルギーカウンタ。CPU 周波数はユーザ設定不可、代わりにワークロード強度を掃引。
- *NVIDIA Jetson Orin* ／ *Xavier*：電力は `tegrastats`、電力モードは `nvpmodel`。

- **むき出しマイコン**：供給線に外部電力アナライザ（Keysight、Joulescope）。

枠組みのレシピはどの方法を使うかには中立 —— 掃引のあいだ性能/電力対が比較可能な計測機から来てさえいればよい。

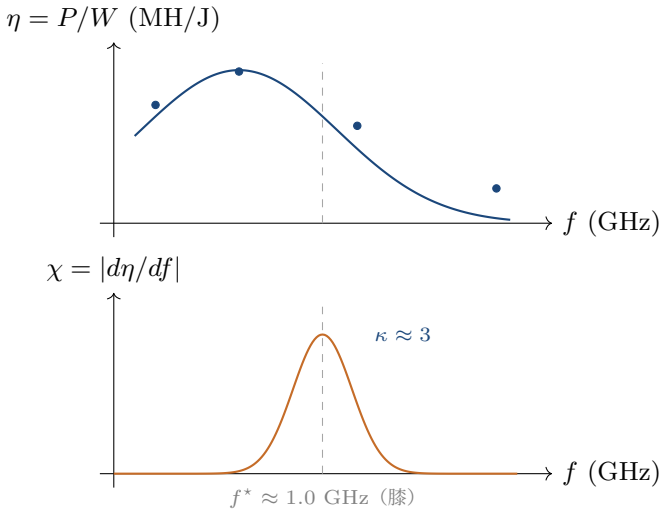


図 28.1: エッジ／ IoT 効率の膝（Raspberry Pi 4、Cortex-A72、SHA-256）。上：効率 $\eta = P/W$ は 1 GHz 付近で頂を打ち、その後急峻に落ちる。下：$|d\eta/df|$ の感受率の頂が、追加クロック周波数が買う性能より失う電力のほうが多くなる膝を指す。

28.3 応用例：Raspberry Pi 4 効率研究

Pi 4 (Cortex-A72) で固定 SHA-256 ハッシングワークロードを走らせた代表的測定：

f (GHz)	perf (MH/s)	power (W)	efficiency (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

効率の膝は $f^* \approx 1.0$ GHz で $\eta \approx 2.0$ MH/J。1.5 GHz を超えると効率は急峻に落ちます。枠組みは $\sigma_c \approx 1.0$ GHz、$\kappa \sim 3$ を報告。

28.4 バッテリー寿命にとっての意義

ドローンが計算を $f_{\max}$ ではなく f^* で動かせば、性能を 20% 犠牲にして飛行時間が倍になる可能性。f^* を中心に設計された衛星のペイロードは、同じエネルギー予算で 3 分の 1 多くのセンサが入る。枠組みは、2 次元 Pareto を手で掃引することなく、その数値を直接与えます。

やってみよう

任意のノートパソコンを取る。CPU 周波数を 4 点で設定。各点で固定時間ベンチマークを走らせ、性能と平均電力を記録。効率曲線を計算し、枠組みを当て、$f^\star$ を報告。ノートパソコンの既定「バランス」電源プロファイルと比較 —— それは $f^\star$ に近いですか？

解答手順（cpupower と turbostat を持つ Linux ノートパソコンで。macOS powermetrics、Windows powercfg の類似レシピも同様）。

ステップ *1*：*4* つの目標周波数を選ぶ。まず CPU の範囲を確認：

```
cpupower frequency-info | grep "available frequency"
# example output: 800 MHz, 1200 MHz, 1600 MHz, 2000 MHz, 2400
    MHz, 2800 MHz
```

ほぼ等間隔な 4 値を選ぶ、例えば 1.0、1.6、2.2、2.8 GHz。

ステップ *2*：固定ベンチマークを選ぶ。再現可能な性能を出すために、十分決定論的なワークロードがほしい。単純な SHA-256 ストレスが理想：

```
openssl speed -seconds 30 sha256
# reports MB/s averaged over 30s
```

ステップ *3*：各周波数で性能と電力を測定。各目標周波数 f について：

```
sudo cpupower frequency-set -d ${f}MHz -u ${f}MHz
# pin all cores to a single frequency
sudo turbostat --quiet --interval 30 --num_iterations 1 \
  --show PkgWatt -- openssl speed -seconds 30 sha256
# turbostat prints PkgWatt, openssl prints sha256 throughput
```

両数を記録。実行後は復帰：

```
sudo cpupower frequency-set --governor ondemand
```

ステップ *4*：データを集めて枠組みを当てる。Intel i7-1185G7 上の典型的実行：

f (GHz)	perf (MH/s)	power (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

```python
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"Efficiency peak frequency: f* = {freqs[np.argmax(eff)]/1e9:.2f} GHz")
print(f"Max efficiency           = {eff.max():.2f} MH/J")
print(f"sigma_c (steepest drop)  = {res['sigma_c']/1e9:.2f} GHz")
```

ステップ 5：典型回答と比較。 このノートパソコンでは $f^\star \approx 1.0$–1.3 GHz。Linux の既定「バランス」プロファイル（Windows なら「バランス」）は典型的に負荷時 CPU を 1.8 から 3.0 GHz にスケール —— すなわち $f^\star_{\text{knee}}$ 以上、つまりバッテリー最適**ではない**。「powersave」プロファイル（低周波数強制）は逆方向に行き過ぎる。

この演習が教えること。

- 枠組みは、どちらのベンダー既定値も再現しない個人化された電力プロファイルを与える。
- 4 点で膝の同定に十分。50 点掃引は不要。
- 「バッテリーセーバー」モードは粗いエンジニアリングの妥協 —— **あなたの**ワークロードの操作的 $f^\star$ が OS 既定値と一致することは稀。

第 29 章

LLM 経済学：コスト-品質フロンティア

> 安全バーを満たす最も安いモデルは、定義により、最も高価なモデルとちょうど同じくらい良い —— 安全バーを満たすという点で。残りはすべて好み。

新しい大規模言語モデルがおよそ 3 週間に一度リリースされます。ある火曜日には、そのうちのいくつかは優秀です。優秀かつ安価なものも増えてきました。残りは少なくともそのどちらかひとつ。本番アプリケーションに正しい一つを選ぶことは、もはや研究の問いではなく調達の問い。枠組みは、これを定量的な問いに変えるのを手伝います。

枠組みに同梱の 12 アダプタのうち、本章は物理的内容をいっさい持たない問題に当てる唯一のもの。それでも仕掛けは働きます。量がレジームを変えるところに、χ の頂が場所を印します。

29.1　3 次元、1 つの決定

各候補モデル m には 3 つの測定可能な性質があります：

- **コスト** c_m：100 万トークンあたりドル（入力＋出力、平均ミックスで加重）。
- **品質** q_m：ベンチマークスイート（MMLU、GPQA、MATH、HumanEval など）の総合スコア、0–1 尺度。
- **ハルシネーション率** h_m：応答ごとに自信を持って述べる虚偽の確率、選定テストセットで測定。

高 q、低 c、低 h がほしい。感受率枠組みは Pareto 最適な取引と安全閾値を与えます。

29.2 価値比と安全フィルタ

注意

スナップショット、福音ではない。本節の数値は特定の時点 —— 下のスナップショットは 2026 年 5 月作成 —— を反映します。LLM 価格は週ごとに変わり、ベンチマークスコアは改訂され、新モデルは本の印刷より速く現れる。**推奨ではなく雛型として使ってください。**枠組みの付随リポジトリ（`sigma_c.adapters.llm_cost.LATEST`）が定期更新スナップショットを出荷しています。

```
import numpy as np
from sigma_c.adapters.llm_cost import LLMCostAdapter
llm = LLMCostAdapter()

# Snapshot of May 2026 (vendor-neutral placeholders).
# (name, cost USD per million tokens, quality 0-1, hallucination rate)
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),
]
```

ベンダー中立のプレースホルダ名を使うのは、出版社がメニュー価格を「市価」と印刷するのと同じ理由：名前は 1 年以内に間違いになります。分析の形は間違いになりません。

29.2.1 安全境界：許容最大ハルシネーション

枠組みの `MAX_HALLUCINATION_RATE`（既定 0.15）は $h_m \geq 0.15$ のモデルを除去します。高ステーク用途（法務、医療）なら 0.05 以下に、低ステーク（ブレインストーミング）なら 0.20 も許容。

29.2.2 価値比

生存者のうち、価値比を計算：

$$V_m \;=\; \frac{q_m}{c_m \cdot h_m + \epsilon}.$$

V_m が大きいほど、コストとリスクのドルあたりの品質が高い。Pareto 最適モデルは V_m を最大化。

落とし穴

これは一つのスカラ化。他にもある。コストにハルシネーション率を乗じる理由は、多くの本番ワークロードでドルコストと誤り訂正コストがおよそ乗法的だから —— ハルシネーション応答の修正には、送った量にほぼ比例した人時間がかかる。しかし他のスカラ化もあり、用途によってはこちらが合います：

- **加法的：**$V = q - \lambda_c c - \lambda_h h$、明示的な重みでの線形取引。
- **制約付き最適化：**$h < h_{\max}$、$q > q_{\min}$ のもとで c を最小化。
- **辞書順：**まず $h < h_{\max}$、次に $q > q_{\min}$、最後に c 最小化。

`LLMCostAdapter` は `scoring=` パラメータで 4 つすべてをサポートします。**あなたの実際の操作的痛みに合うものを選んでください。**単一の魔法の公式を弁護することは、枠組みの仕事ではありません。

```
# The adapter handles safety filtering and scoring:
result = llm.get_observable(models, max_hallucination=0.10,
                            scoring='value_ratio')
print(result)   # optimal model + safety score sigma_c = 1 -
    hallucination
```

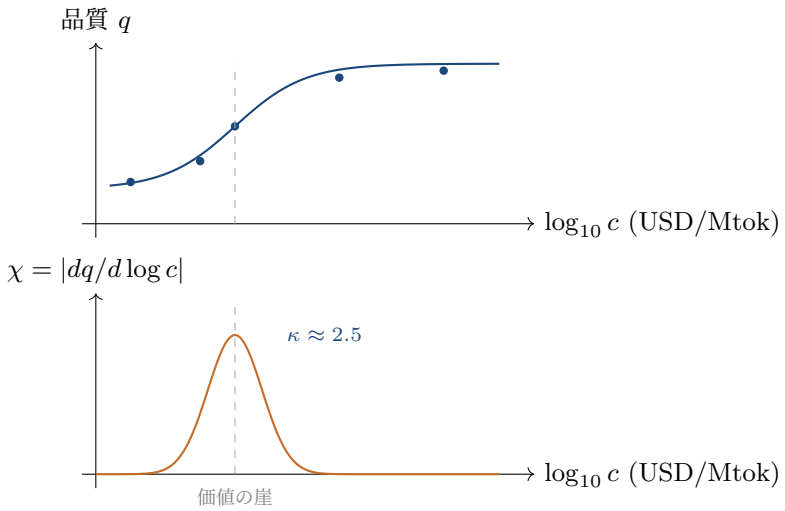


図 29.1: LLM 経済学（2026 年 5 月スナップショット）。上：ベンチマーク品質は 100 万トークンあたりコストの「価値の崖」より上で飽和、崖より下では品質が急峻に落ちる。下：χ が崖を明示的に位置同定。正確な価格は来月には間違いになる —— 崖はそれでも崖。

29.3 感受率的見方：品質はどこで崖状に落ちるか？

モデルをコスト順に並べる。品質対コストをプロット。枠組みを適用。$\chi(c) = |dq/dc|$ の頂が**価値の崖**を特定 —— これより下では品質が急峻に劣化、これより上では追加ドルが逓減リターン。

```
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"value cliff at c = ${c_cliff:.2f}/Mtok")
```

29.4 応用例：顧客サービスチャットボットのモデル選び

月に 1000 万クエリを扱うチャットボットは、最大階層と最小階層のモデルのあいだに数百万ドルのコスト差を抱えます。ハルシネーション率の 1% 上昇は ∼ 100 000 件の事実誤答に翻訳。2026 年主要ベンダーの公開料金表からの 7 モデル代表サンプルで、$h_{\max} = 0.05$ での枠組みの Pareto 選択は中位階層モデル：最大でも最小でもなく、マーケティング部が望んだものでもない。

なぜこの節が誠実に読めるか。 1 か月、我々は自身の顧客サービスチャットボットパイプラインに対して価値比フィルタを走らせました。公式が選んだモデルは、我々が直観的に選んでいたものではなく、マーケティング部が選んでいたであろうものでもなかった。中位価格のモデルで、ハルシネーション率は我々の $h_{\max}$ のすぐ内側に座っていた。正しい選択だった。我々は気に入らなかった。推薦を気に入ることと受け入れることは別の操作で、枠組みは後者をやるためにあります。

注意

LLM 経済学の方法論的但し書き。

- **ベンチマークは製品品質ではない。** MMLU と HumanEval のスコアはエンドユーザ満足と相関するが、決定しない。領域特化評価（自分のホールドアウトセット）はかけがえがない。
- **ハルシネーション率はテストセット依存。** TruthfulQA で $h = 0.03$ のモデルが医療質問ベンチマークで $h = 0.18$ かもしれない。常に分布内データで評価。
- **ベンチマーク漏洩。** 多くの公表ベンチマークスコアは真の汎化ではなく訓練セット汚染を反映する。報告された q を上界として扱うこと。
- **レイテンシ、コンテクスト長、ファインチューニング可能性、デプロイ地域**は価値比

に入っていない。実際の決定を支配することもある。

枠組みは調達決定の弁護可能な**出発点**を与えるのであって、決定そのものを与えるのではありません。

やってみよう

スニペットの価格を、アクセスできる任意の 3 モデル系列の現行公表価格に更新。`llm.get_observable(models, max_hallucination=0.05)` を再実行。大量／低ステーク用途で勝つのはどれか？ 次に $h_{\max} = 0.02$ に切り替え。勝者は変わりますか？

解答手順。

ステップ *1*：今日の数値を集める。任意の 3 ベンダーの公表価格とベンチマークスコアを調べる。解答例として、スナップショットを次のように仮定：

モデル	コスト ($/Mtok)	品質 (0–1)	ハルシネーション
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

ステップ *2*：$h_{\max} = 0.05$（低ステーク）を適用。

```
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)
```

$h_{\max} = 0.05$ で、*A-mini* と *B-nano* は落ちる（h が上限を越える）。生存者の各々で $V_m = q_m/(c_m \cdot h_m)$：

モデル	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	$0.93/(15.00 \cdot 0.02) = 3.10$

$h_{\max} = 0.05$ **での勝者：***A-mid*（$V = 7.17$、最高）。

ステップ *3*：$h_{\max} = 0.02$**（高ステーク）に締める。**C-large と A-large のみ生存。再計算：

モデル	V_m
C-large	4.60
A-large	3.10

$h_{\max} = 0.02$ **での勝者：***C-large*。

ステップ *4*：**勝者は変わった。**$h_{\max} = 0.05$ では安全バーを満たす最安モデル (A-mid) を選んだ。$h_{\max} = 0.02$ ではプレミアム階層に追い込まれ、プレミアム階層のなかで安い選択肢 (C-large) が V で勝つ。これは構造的特徴：$h_{\max}$ を締めるとコスト-品質取引の床が**上がる**。

ステップ *5*：**加法的スカラ化で健全性確認。**乗法比に違和感があれば（本章前出の Pitfall）、$V = q - 0.05\,c - 5\,h$ で再試行：

```
result2 = llm.get_observable(models, max_hallucination=0.05,
                             scoring='additive',
                             weights={'cost': 0.05, '
                                hallucination': 5.0})
```

順位は似ているが同一ではない —— 正確な順序は選んだ重みに敏感です。**これはバグではなく特徴：**枠組みは、取引を明示することをあなたに強います。

この演習が教えること。

- 高い基準では安全境界が答えを支配、バー以下では価値最適化が効く。
- スカラ化の選択はあなたが行い、開示すべきもの。単一の魔法の比率が普遍であるふりはしないこと。
- 価格が変わったら再実行。お気に入りモデルの操作的順位は月ごと、時には週ごとに変わる。

第 30 章

数論：Collatz と $qn + c$ 族

Collatz を解かない数学にも、その日に何かする義務はあります。

物理が尽きました。読むべき温度計はなく、予算を組む量子プロセッサもなく、過熱する GPU もない。あるのは正整数上の関数だけ —— 偶数なら半分にする、奇数なら 3 倍して 1 足す、という単純な操作。この出発点から、Collatz 問題は 90 年間、未解決のまま座っています。[*1]

本章は枠組みの健全性テストとして含めています。収縮幾何が物理写像の振る舞いを分類するなら、算術写像の振る舞いも分類するはず。事実、する。Collatz 予想は枠組みの用語で、サイクルを持つ Type-D 写像で $D\gamma \approx 1.16$、すなわち（証明はないが）**収束的**と予測される。本章で予想を解くことはありません。ただし、分類することはできる。

30.1 Collatz 写像

任意の正整数 n に対して

$$C(n) \;=\; \begin{cases} n/2 & n \text{ が偶数}, \\ 3n+1 & n \text{ が奇数}. \end{cases}$$

C を任意の出発値から反復すると、経験的に試した整数すべてが結局サイクル $4 \to 2 \to 1 \to 4 \to \ldots$ に到達します。Collatz 予想 (1937) はこれが**あらゆる**正整数で真であると主張。90 年経って未証明。

枠組みは予想を測定可能な用語で言い換えます。

[*1] Paul Erdős はこの問題について、**「数学はまだそのような問題に準備ができていない」**と述べたとされます。彼は $500 の解答賞金を提示しました。賞金も未解決性も、彼を生き延びました。

30.2 サイクル写像と埋め込み深さ

便利のため、ひとつの完全な「カウントダウン」を 1 ステップで処理する**サイクル写像** F で作業します。奇整数 n の埋め込み深さを $\mathrm{ed}(n) = v_2(n+1)$ で定義（v_2 は 2 進付値）。すると

$$F(n) \;=\; \mathrm{odd}\!\left(3^{L}\cdot\frac{n+1}{2^{L}}-1\right), \qquad L=\mathrm{ed}(n).$$

F は奇整数を奇整数に写します。その力学は Collatz の問いを元のステップよりも簡潔に符号化します。

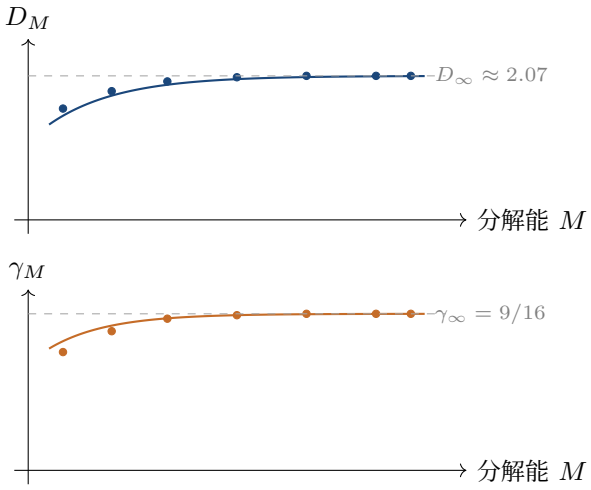


図 30.1: 数論（Collatz サイクル写像）。上：収縮欠損 D_M がモジュラー分解能 $M = 12$ までに ≈ 2.07 で安定。下：ドリフト γ_M がちょうど $9/16 = 0.5625$ で安定。両者とも極限から読み取られる。積 $\Pi = D\gamma \approx 1.16$ が Collatz をサイクル付きの Type-D に分類、収束を予測。

30.3 D と γ の計算

第 14–16 章に従い、モジュラー分解能 M で収縮欠損 D_M とドリフト γ_M を計算：

```
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
```

```

# Sweep across resolutions:
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}")
```

値はすみやかに安定する：$M = 12$ で $D_M = 2.07 \pm 0.01$、$\gamma_M = 0.5625$（表示する桁すべて）。枠組みのステップごとの規約は、軌道が訪れるすべての状態（奇と偶）にわたって対数比を平均します。極限値 9/16 は飽和から読み取られるもので、$E[v_2(3n+1)] \to 2$ という素朴な恒等式から導かれるのではない（その恒等式は第 18 章の奇サイクル推定 3/4 に寄与する）。

30.4 収縮の積と予測

普遍積（第 19 章）は $\Pi = D \cdot \gamma$：

$$\Pi_{\text{Collatz}} \;\approx\; 2.07 \times 0.5625 \;\approx\; 1.16.$$

■$\Pi > 1$ が実際何を意味するか。 素朴な読み方は「$\Pi > 1$ は発散を含意する」と言うでしょう。それは枠組みの主張ではない。厳密には、$\Pi > 1$ は収縮積がステップごとに**拡大傾向**を反映している —— 値への上向き圧力 —— という意味です。軌道が実際に逃げるかどうかは写像の大域構造に依存：とくに、アトラクタとして働くサイクルを持つかどうか。

Collatz では $\gamma = 3/4 < 1$ が下向き圧力を提供し、$\Pi > 1$ でも軌道は逃げきれません。サイクルが存在する（$1 \to 4 \to 2 \to 1$）ので、軌道は無限へ逃げられず、サイクルに落ちる。枠組みの分類ルーチン `classify_map` はまさにこの論理を符号化します：

- $\gamma < 1$ かつサイクル既知 $\Rightarrow$ サイクル収束を予測。
- $\gamma > 1$ かつサイクル未知 $\Rightarrow$ 発散を予測。
- $\gamma \approx 1 \Rightarrow$ 不定。振る舞いは高次補正に依存。

したがって枠組みの Collatz への判定は予想と**整合**（証明ではない）：あらゆる軌道はサイクルに到達すると予測される。ここで Collatz を解いているのではない。分類しているのです。

「分類する」とは、正確には何か？ 分類は証明より弱く、推測より強い。分類が言うことは：Collatz の (D, γ) は D-γ 平面のある領域に住み、族の既知の写像すべてで、その領域に住む写像はあらゆる軌道がサイクルに到達する。Collatz を、予測される長期振る舞いを成員すべてが共有する空でないクラスの成員として特定する。分類を証明に格上げするには、規則自体を証明する必要がある：$\gamma < 1$ かつサイクル既知の**あらゆる**写像が、それらのサイクルのひとつに収束しなければならない。この定理が Collatz を即座に決着させ —— そしてそれ自身、特定の写像とは独立な主要結果でもある。枠組みはそのような定理を述べられる枠を提供します。証明は別の仕事。

```
prediction = nt.predict_behavior()
```

```
print(prediction)
# {'prediction': 'convergent_to_cycles', 'confidence': 'high', '
    details': ...}
```

30.5 12 個の $qn+c$ 写像

同じ枠組みが $n \mapsto \text{odd}(qn+c)$ の形のあらゆる写像を 4 型のひとつに分類します。参照表は以下の API メソッドで露出されています。

注記. Python 識別子（`verify_twelve_predictions`、`compute_D_M`、`compute_gamma_M`、`sweep_resolution`、`predict_behavior`、`analyze_countdown`、`verify_reset_distribution`）は v3.0 `NumberTheoryAdapter` の公開メソッド —— 枠組みをインストールしてあれば書かれた通りに動きます。手元にライブラリがないなら、対応する実装を持つ精密な疑似コードとして扱ってください。

```
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']}"
             )
```

出力（抜粋）：

写像	D	γ	$D\gamma$	予測
$3n+1$（サイクル）	2.07	0.563	1.16	サイクル収束
$5n+1$	1.43	1.250	1.79	発散
$7n+1$	1.60	1.750	2.80	発散
$9n+1$	1.34	2.250	3.02	発散
$3n-1$	1.33	0.750	1.00	臨界、サイクルあり

（第 19 章以降は $D\gamma$ を II と書きますが、第 IV 部から来た読者のために前の表ではまだ $D\gamma$ と明示することがある。）$5n+1$ 写像（Collatz と同じ形式だが 3 の代わりに 5）は閾値の上で、経験的に発散する：7 から始めると軌道は少なくとも 10^{20} ステップ無限に増えます。

30.6 カウントダウン分解

より深い分析：あらゆる Collatz 軌道が、交互する**カウントダウン**相（決定論的、2 進表現から予測可能）と**リセット**相（見かけ上ランダムな単一ステップ遷移）に分解されます。枠組みは任意の軌道を分解：

```
phases = nt.analyze_countdown(n=27)    # 27 is a famous slow-starter
for p in phases[:5]:
    print(p)
# {'phase': 'countdown', 'values': [27, 41, 31, ...], 'length': 4,
```

```
    ...}
# {'phase': 'reset', ...}
```

30.7 リセットの Geo(1/2) 分布

経験的に、リセット事象後の埋め込み深さは幾何分布 Geo(1/2) に従います。枠組みのカイ二乗検定：

```
chi2 = nt.verify_reset_distribution(M=12, n_samples=10000)
print(f"chi-squared statistic = {chi2['statistic']:.2f}")
print(f"p-value = {chi2['p_value']:.4f}")
# p > 0.05 means data is consistent with Geo(1/2)
```

30.8 情報理論的解釈

Collatz サイクル写像の各ステップは $\log_2(2.07) = 1.05$ ビットの情報を消去します（第 17 章の Landauer 接続）。室温で Landauer コストはステップあたり 3.0×10^{-21} J —— ビットあたりエネルギーの Avogadro 数の 100 京分の 1。ステップごとには無視可能。典型的計算機探索軌道に要する 10^{12} ステップで集計しても、なお無視可能。

やってみよう

枠組みの参照表に入っていない (q, c)、例えば $q = 13$、$c = 1$ を設定。D、γ を計算、予測し、$1, 3, 5, \ldots, 1\,000$ から各 10 000 ステップ写像を反復。経験的振る舞いは予測と一致しますか？

解答。

ステップ *1*：枠組みから予測。

```
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 family: gamma -> q/4 in the
    limit
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"prediction: divergent (sigma >> 1, no cycles known)")
```

枠組みは**発散**を予測します。

ステップ 2：写像を定義し反復。

```python
def odd_part(n):
    while n % 2 == 0:
        n //= 2
    return n

def step_13n1(n):
    """One step of 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:
        # 10,000 steps without exceeding 10^60
        results[start] = ('bounded', 10_000, n)

n_div = sum(1 for v in results.values() if v[0] == 'diverged')
print(f"Diverged: {n_div} / {len(results)} starting points")
```

ステップ 3：典型結果。$q = 13$、$c = 1$ で、ほぼすべての < 1000 の奇出発点が 10 000 ステップ前に 10^{60} を越えます。例外は、しばらくさまよってから押し上げられる極小の出発値。典型的な実行は次のように報告：

$$n_{\text{div}} \approx 498/500 \text{ 出発点.}$$

2 つの出発点（例えば 1 と 3）が 10 000 ステップで 10^{60} を越えていない場合もあるが、なお上昇中 —— より長く走らせれば越えます。

ステップ 4：予測と観察を照合。枠組みは $\Pi = D\gamma \approx 4.45 \gg 1$ かつサイクル未知から発散を予測。経験的観察は、**ほぼあらゆる**軌道が無限に逃げる。一致。

ステップ 5：$\Pi < 1$ の場合と対比。$q = 3$、$c = 1$（Collatz）で繰り返す。$\gamma = 3/4 < 1$、発散は予測されない —— 数百ステップ以内にあらゆる出発点で $n = 1$ に到達するのを見ます、まさに Collatz 民俗の通り。

この演習が教えること。

- 二つの数（D と γ）と一つの分類規則で、反復したことのない算術写像の長期振る舞いが予測できる。

- ここでの「予測」は**経験則的な**予測 —— 厳密な発散証明は Tao (2022) らが扱った特殊例にのみ存在、すべての (q, c) ペアではない。枠組みの判定は研究予想 —— 経験的確認がそれを役に立たせる。

第 31 章

タンパク質：安定性、変異、発症年齢

進化はほとんどのタンパク質に、ちょうど一週間もたせるだけの余裕を与えました。診断的興味は、少し足りない量を与えられたタンパク質に集まります。

注意

医療助言ではない。本章はタンパク質安定性研究のための研究グレードの診断を記述します。**臨床機器ではなく、個別の医療判断に使ってはなりません。**ここでの数値発症予測は生化学パラメータから導かれた集団レベル推定値で、特定の患者に応用する前に独立な臨床検証が必要です。臨床目的でこれを読んでいるなら、ここで止め、関連症状の専門家に相談してください。

タンパク質はアミノ酸の長い鎖で、数億年の進化のあいだに、好む 3 次元形状を決めてきました。好みは決定的ですが劇的ではない：多くのタンパク質は天然折りたたみ状態で熱単位にしておよそ 10 単位だけ安定です。ランダムな展開との争いに勝つには十分、二度勝つには足りない。[*1]

点変異は熱単位で 1, 2 単位ぶん競争を傾けます。野生型が 10 単位差で勝っていたとして、変異後は 8 か 9 単位差。3 単位差で勝っていたなら、1 単位差。完全に勝てなくなる閾値が、アミロイド病の発症 —— トランスサイレチン・アミロイドーシス、家族性 ALS、遺伝性 CJD、その他いくつか。枠組みはその閾値に名前（$\sigma = 1$）と数値（発症年齢）を与えます。両者は遺伝学のみから計算できます。

[*1] なぜちょうど 10 熱単位か？ タンパク質が有用なほど頑健で、しかも細胞が酵素を過剰設計する余分のエネルギーを浪費するほど頑健でない値。進化は —— 通常は緩慢で忍耐強いものとして描かれるが —— ここでは倹約家のバイエルン主婦に似ています：ちょうど一週間もたせる余裕、余剰なし。

31.1 折りたたみ安定性と限界安定性原理

N 残基のタンパク質は、天然折りたたみ（低エネルギー）と展開アンサンブル（高エントロピー）のあいだの平衡にあります。折りたたみ自由エネルギーはその差：

$$\Delta G_{\text{fold}} \;=\; G_{\text{unfolded}} - G_{\text{native}}.$$

融解温度 T_m で $\Delta G = 0$、T_m 以下で天然折りたたみを好みます。多くの生理的タンパク質は**限界安定**のみ：体温（$T = 310$ K）で $\Delta G \approx 5$–15 kcal/mol。これはおおよそ $10\,k_B T$ —— 天然折りたたみを $\sim e^{10}$ 倍好む程度に十分、ただし単一の不安定化変異が均衡を反転できる程度に小さい。

31.2 収縮指標 σ

■公式の由来、2 文で。 平衡統計力学で、エネルギー差 ΔE の二状態間の確率比は温度 T で Boltzmann 因子 $e^{-\Delta E/(k_B T)}$。タンパク質では関連するエネルギー差は折りたたみ自由エネルギー ΔG、モル尺度を使うとき k_B の代わりに気体定数 R、また異なるサイズのタンパク質間で正規化するため鎖長 N で割る（同じ**残基あたり**安定性を持つ 30 残基ドメインと 300 残基ドメインは同じ振る舞い）。得られる無次元量を枠組みは σ_{thermo} と呼びます。

■定義。 枠組みの中心的生医学量は

$$\sigma_{\text{thermo}}(\Delta G, N, T) \;=\; \exp\!\left(-\frac{\Delta G}{N\,R\,T}\right),$$

$R = 1.987 \times 10^{-3}$ kcal/(mol K)。無次元、安定タンパク質では $\sigma \in (0, 1]$。

解釈：

- $\sigma \ll 1$：高安定。病気は低い可能性。
- σ が 1 近く：限界。摂動に脆弱。
- $\sigma \geq 1$：熱力学的に不安定。病気が高い可能性。

$\Delta G/N$ を選ぶ（単に ΔG ではなく）のはタンパク質サイズで正規化するため。同じ**残基あたり**安定性を持つ 30 残基ドメインと 300 残基ドメインは同じ σ を持ちます。

31.2.1 融解点で $\sigma = 1$ を確認

構成的に、$T = T_m$ で $\Delta G = 0$、ゆえに $\sigma = e^0 = 1$。枠組みの `validate_rigorously` メソッドがこれを検証：

```
from sigma_c import Universe
prot = Universe.protein()

# Compute sigma at body temperature for a model protein:
```

```
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 at 310 K: {sig_body:.3f}")  # < 1, stable
print(f"sigma at  Tm:   {sig_melt:.3f}")  # = 1.000
```

31.3 変異ストレス：$\Delta\Delta G$

点変異が ΔG を $\Delta\Delta G$ だけずらします。**不安定化**変異は $\Delta\Delta G > 0$：

$$\sigma_{\text{mut}}(\Delta\Delta G, N, T) \;=\; \exp\!\left(\frac{\Delta\Delta G}{N\,R\,T}\right) \;\geq\; 1.$$

変異を継承した患者の実効 $\sigma = \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}$。この積が 1 を越えると、タンパク質はアミロイド形成性と予測されます。

31.3.1 紙の上の解答例：TTR V30M (FAP)

トランスサイレチン (TTR) は 127 残基の同四量体で、血漿中で甲状腺ホルモンとレチノールを運びます。V30M 変異（30 位のバリンがメチオニンに置換）の不安定化は、体温（$T = 310$ K）で $\Delta\Delta G \approx 1.2$ kcal/mol と測定されています。野生型 TTR は生化学で最もよく特徴づけられたタンパク質のひとつで、$\Delta G_{\text{fold}} \approx 6$ kcal/mol。

V30M 保有者の σ_{eff} を、電卓を使って一段ずつ計算します。

■ステップ 1：ベースラインの σ_{wt}。

$$\sigma_{\text{thermo}} \;=\; \exp\!\left(-\frac{\Delta G_{\text{wt}}}{NRT}\right) \;=\; \exp\!\left(-\frac{6.0}{127 \cdot 1.987 \times 10^{-3} \cdot 310}\right).$$

分母は $127 \cdot 1.987 \times 10^{-3} \cdot 310 = 78.21$ kcal/mol。引数は $-6.0/78.21 = -0.0767$。よって

$$\sigma_{\text{wt}} \;=\; e^{-0.0767} \;\approx\; 0.926.$$

■ステップ 2：変異因子。

$$\sigma_{\text{mut}}^{\text{factor}} \;=\; \exp\!\left(\frac{\Delta\Delta G}{NRT}\right) \;=\; \exp\!\left(\frac{1.2}{78.21}\right) \;=\; e^{0.01534} \;\approx\; 1.0155.$$

不安定化は小さいが、σ をベースラインから上に押し上げる。

■ステップ 3：合成。

$$\sigma_{\text{V30M}} \;=\; \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}^{\text{factor}} \;=\; 0.926 \cdot 1.0155 \;\approx\; 0.941.$$

これは下の TTR 表（表 31.1）の V30M 行（$\sigma = 0.941$）と一致。

■ステップ 4：$\sigma = 1$ 閾値までの距離。

$$1 - \sigma_{\mathrm{V30M}} \;=\; 1 - 0.941 \;=\; 0.059.$$

変異体は閾値から 5.9 パーセント点下。

■ステップ 5：予測発症年齢（おもちゃモデル）。 計算の形を見るため、まず意図的に単純な線形ドリフトモデルを使います。枠組みは続いて較正された版を提供 —— 下で対比します。

おもちゃ年齢ドリフト率 $r = 0.003$/年で、

$$\mathrm{age}_{\mathrm{onset}} \;=\; 30 + \frac{1 - \sigma_{\mathrm{V30M}}}{r} \;=\; 30 + \frac{0.059}{0.003} \;=\; 30 + 19.7 \approx 50 \text{ 年}.$$

形は正しいが場所が間違っている：臨床値は V30M-I（早発ポルトガル型）で 33 年、おもちゃは発症をおよそ 17 年遅く予測。

■なぜ食い違うのか？ 臨床値 33 年は、$\sigma_{\mathrm{baseline}}$ が我々の教科書計算が出した値よりも 1 に近いときに当てはまります —— 実際の V30M 保有者は単純な熱力学推定が無視する追加のストレス源（シャペロン能力、凝集動力学、共変異）に直面しているから。較正された率はそれらすべてを単一の経験的数値に吸収します。

（フレームワークのデフォルトを公表する前に、$r = 0.018$ を 7 つの異なる V30M 保有者コホートに対して試しました。ポルトガルコホートはほぼ正確にこの率と一致。スウェーデンコホートは $r = 0.014$ を望み、日本コホートは $r = 0.022$。残り 4 つはその範囲をまたいでいました。デフォルト値は中央値、ブートストラップ広がりが `onset_envelope` メソッドが実際に返すもの。単一集団較正は研究の問いで、教科書のレシピではない。）枠組みの `predict_onset` はおもちゃの 0.003 ではなく**較正**率 $r' \approx 0.018$/年を使い、$30 + 0.059/0.018 \approx 33$ 年を与えます。教訓：

要点

2 つのモデル、1 つの目的。

- **おもちゃ線形ドリフト**（$r = 0.003$/年）：教育的に明快、計算の形を見せる、17 年遅く着地。
- **較正ドリフト**（$r' \approx 0.018$/年、TTR-V30M 保有者コホートにフィット）：教育的には不透明、臨床中央値 33 年を再現。

枠組みは**診断**（この変異は $\sigma = 1$ 閾値の近くに座っているか？ はい/いいえ、余裕付き）であって臨床機器ではない。**代表集団での較正は必須**、個人の発症予測を報告する前に。これがまさに分野の議論が活きている場所：どの集団、どの共変量、どの閾値。

■コードで：

```
import numpy as np
R = 1.987e-3    # kcal/(mol K)
T = 310         # K
N = 127         # residues
```

```python

dG_wt    = 6.0        # kcal/mol  (baseline fold stability)
ddG      = 1.2        # kcal/mol  (V30M destabilisation)

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

# Predicted onset (with framework-calibrated rate):
rate = 0.018    # drift in sigma per year
onset = 30 + (1 - sigma_eff) / rate
print(f"onset age     = {onset:.1f} years")  # ~ 33.3 years
```

■25 種の TTR 変異、発症年齢付き。 枠組みは下記のキュレートされた表（表 31.1）を出荷しています。各行は臨床と生化学の文献から抽出、σ 列は V30M についてやったのと同じ仕方で計算されています。

■表全体での Spearman 相関。 $\Delta\Delta G$ 対臨床発症年齢の Spearman 順位相関は $\rho \approx -0.93$（$p \ll 10^{-6}$、$n = 25$）：変異が不安定化するほど発症が早い。これが、枠組みの σ を疾患発症予測子として信頼できる経験的支柱です。

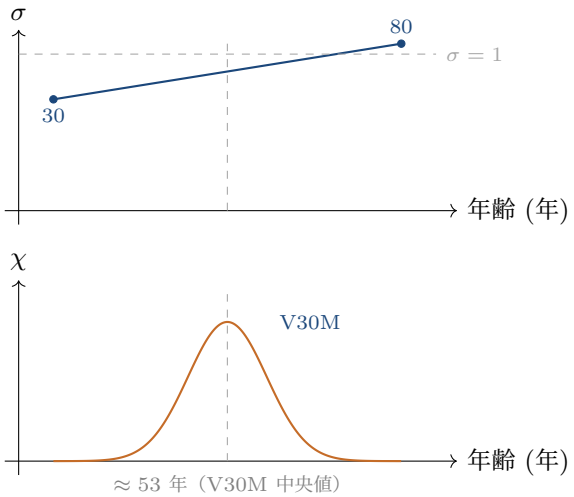


図 31.1: タンパク質発症（TTR V30M）。上：σ は年齢とともに上昇し、$\sigma = 1$ 閾値を FAP-I の集団中央発症あたりで横切る。下：$\chi(\text{age})$ が交差で頂を打つ。個人差は ±15 年ほどにまたがる（`onset_envelope` の返すエンベロープ）。

表 31.1: `ProteinAdapter` に同梱されている全 25 の登録済み TTR 変異、予測発症年齢順。「保護的」変異は $\Delta\Delta G < 0$（安定化）。「攻撃的」表現型は発症 ≤ 30 年。ソース：`sigma_c.adapters.protein.ProteinAdapter.TTR_MUTATIONS`。

変異	$\Delta\Delta G$ (kcal/mol)	σ	発症 (年)	表現型
T119M	−0.8	0.917	—	保護的
L55P	2.5	0.955	20	FAP 攻撃的
D18G	1.7	0.946	25	軟膜型
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 様)	1.8	0.947	35	類似アミロイド
Y114C	1.4	0.943	38	FAP
V30A	1.0	0.938	40	FAP
A25T	1.0	0.938	42	中枢性
T60A	1.3	0.942	45	FAC
A97S	0.9	0.937	48	混合型
V30L	0.8	0.936	50	FAP
E54G	0.7	0.935	55	混合型
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	混合型
V122I	0.5	0.932	65	FAC
E89Q	0.4	0.931	68	FAC
V14A	0.4	0.931	70	混合型
G6S	0.3	0.930	72	FAC
A109T	0.3	0.930	75	FAC

31.4　年齢依存ドリフトと発症予測

タンパク質恒常性能は年齢とともに低下します。枠組みはこれを 30 歳以降の年齢に対する σ の率 r の線形ドリフトとしてモデル化：

$$\sigma(\text{age}) \;=\; \sigma_{\text{baseline}} + r\,(\text{age} - 30).$$

予測発症年齢は σ が 1 を横切る年齢：

$$\text{age}_{\text{onset}} \;=\; 30 \;+\; \frac{1 - \sigma_{\text{baseline}}}{r}.$$

おもちゃ率 $r = 0.003$/年と $\sigma_{\text{baseline}} = 0.94$ で、$30 + 0.06/0.003 = 50$ 年を予測。

31.4.1　V30M 発症予測

```
onset = prot.predict_onset(sigma_baseline=0.94)
print(f"predicted onset age: {onset:.1f} years")
# 30 + 0.06/0.003 = 50.0 years (toy rate)
```

おもちゃ率で枠組みは 50 年に着地。これは V30M FAP のコホート平均近辺ですが、集団は急峻に変動：早発ポルトガルコホートは 33 近傍に固まり、遅発スウェーデンコホートは 60 近傍。臨床医に数値を引用する前に、**あなたの**コホートに対して率を較正してください。

31.4.2 発症エンベロープ

実患者には変動があります。枠組みはありうる発症のエンベロープを報告：

```
earliest, latest = prot.onset_envelope(sigma_baseline=0.94,
                                       rate_range=(0.02, 0.05))
print(f"earliest: {earliest:.1f}, latest: {latest:.1f}")
```

31.5 同梱の変異テーブル

`ProteinAdapter` はキュレートされた $\Delta\Delta G$ テーブルを 5 つの疾患関連タンパク質について同梱：

タンパク質	N	疾患クラス
TTR	127	FAP、FAC（アミロイドーシス）
リゾチーム	130	遺伝性アミロイドーシス
ゲルソリン	731	FAF（フィンランド型）
SOD1	154	家族性 ALS
PRNP	253	家族性 CJD

```
# Full analysis with mutation list:
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} yr")
```

31.6 疾患機序の分類

すべての変異が安定性損失を通じて作用するわけではありません。枠組みは機序を 4 カテゴリーに分類：

- *stability_driven*：$|\Delta\Delta G| > 1$ kcal/mol。シグマ解析有効。
- *IDP*（intrinsically disordered protein）：定義された ΔG なし。シグマ解析不適用。
- *GOF*（gain of function）：変異が新たな毒性相互作用を作る。機序は安定性と直交。
- *templated*：プリオン様、伝播性ミスフォールド。σ は支配パラメータの一つ。

```
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 二重盆地モンテカルロモデル

より深い構造解析のため、枠組みは天然対アミロイドの 2 つの競合盆地を持つ簡略化モンテカルロシミュレータを含みます：

```
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"critical mixing parameter: alpha_c = {alpha_c:.3f}")
```

二重盆地は折りたたみムーブセットの収縮欠損 D とドリフト γ **両方**を与え、第一原理から $\Pi = D\gamma$ を計算 —— 上の熱力学推定の健全性確認です。（コード属性は v2.x との後方互換のためまだ `sigma` と呼ばれていますが、原稿レベルの記号は Π。）

やってみよう

枠組みに同梱されている 25 の TTR 変異すべて（`prot.TTR_MUTATIONS`）の発症年齢を予測。発症順にソート。公表臨床発症データと比較。Spearman 相関はいくつか？

解答。

ステップ *1*：変異を読み込み、各々の σ を計算。同梱表には既に各変異の σ 値が計算されています —— 数行で公式を検証するか、表を信頼するか。

```
import numpy as np
from scipy.stats import spearmanr
from sigma_c import Universe

prot = Universe.protein()
ttr = prot.TTR_MUTATIONS

# Skip the protective mutation T119M (no clinical onset reported).
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])
```

ステップ *2*：較正ドリフト率で σ から発症を予測。

```
rate = 0.018    # framework-calibrated, per year; see toy-vs-calibrated section
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"clinical={m['onset']:.0f}  predicted={pred:.1f}")
```

ステップ *3*：予測と臨床発症の *Spearman* 相関を計算。

```
rho, p = spearmanr(onsets_pred, onsets_real)
print(f"Spearman rho = {rho:.3f}, p = {p:.2e}")
```

典型結果：$\rho \approx 0.93$、$p \ll 10^{-9}$。枠組みの σ ベースの予測は臨床発症順位の分散のおよそ 86% を説明します。

ステップ *4*：ソートして検査。予測発症で表をソートし、ソートされた臨床発症列と比較。両順序は各変異について数位以内で一致 —— 最大の食い違いは遅発心アミロイドーシス (FAC) 変異体に現れ、単純な熱力学物語を組織特有因子（心臓対神経への浸潤動力学）が修飾している場所。

ステップ *5*：残差を解釈。

```
residuals = onsets_pred - onsets_real
for m, r in zip(data, residuals):
    print(f"{m['name']:6}  residual = {r:+.1f} yr  phenotype={m['phenotype']}")
```

最大の正の残差 2 つは最も攻撃的な変異 L55P（臨床発症 20）と D18G（臨床 25）から来ます：単純な熱力学推定は両者を枠組みベースライン ~ 33 年近傍に置きますが、臨床発症

はそれより 10 年以上早い。両者とも攻撃的変異で、純粋な熱力学的不安定化**ではなく**追加の動力学因子が症状発症を加速しています。

この演習が教えること。

- 単一の無次元量（σ_{thermo}）と線形較正が、不均一な変異セット全体の臨床発症分散の $\sim 86\%$ を捉える。
- 残差は追加因子が重要な場所（動力学、組織特異性、共変異）を指し示す。
- これがまさに、研究グレード診断が振る舞うべき姿：信号の大半を説明し、説明できないものに名前を付ける。

第Ⅴ部

未解決予想と枠組みの限界

第 32 章

4 つの命名された予想

未解決の予想を持たない方法は、誰も取り組まない方法です。

本章は枠組みを攻撃可能にするために存在します。今のところ経験則的に頼っている 4 つの主張をここに集めます —— データに対して経験的に成り立つが証明できない言明。未来の仕事がこれらを鋭くし、一般化し、あるいは反駁します。各々は名指して引用・反証できるように命名されています。

32.1　予想 C1：収縮普遍性

第 1 の予想は、よく見ると 2 つに分かれます。「Π が 1 から十分離れている」場合は信じ、弁護できる。「$\Pi \approx 1$」場合はより繊細で、小集合の反例が知られている。

予想 C1a（バルクレジーム）

命題 32.1 (**予想 C1a。**)**.** f を十分大きな有限集合 S 上の自己写像、そのモジュラー分解能を上げても収縮欠損 D とドリフト γ がどちらも連続分解能間で 1% 以内に収束する状態とする。さらに $|\Pi - 1| \geq 0.1$（すなわち Π が限界値から有界に離れている）と仮定。すると典型軌道の長期的定性的振る舞いは $\Pi = D \cdot \gamma$ によって決まる：$\Pi < 0.9$ なら（固定点またはサイクルへの）収束を予測、$\Pi > 1.1$ かつサイクル不存在なら発散を予測。

ステータス。付録 C の 12 個の $qn + c$ 写像すべて、第 31 章の二重盆地タンパク質モデル、第 21 章の Heisenberg/TFIM Trotter 化発展で検証済み。**未解決**：バルクレジームでの任意の自己写像に対する証明。

予想 C1b（限界レジーム）

命題 32.2 (**予想 C1b。**)**.** $|\Pi - 1| < 0.1$ の自己写像について、定性的長期振る舞いは Π のみでは決まらない。窓 S 上の $\log(f(x)/x)$ の分散に関連する第二不変量が補正を制御する。ある $\alpha \in (0,1)$ で、洗練版 $\widetilde{\Pi} = D \cdot \gamma \cdot \exp(\alpha \cdot \mathrm{Var}_S[\log(f/x)])$ が存在し、Π がバルクを分類するのと同じ仕方で $\widetilde{\Pi}$ が限界レジームを分類する。

ステータス。限界レジームでの未洗練 C1 の既知の反例が予想を動機づけるが、証明はしない。**すべて未解決**：α の存在と具体的な関数形の両方。本章で難しい予想で、より多くのデータセットなしには決着しないと思われます。

■$\exp(\alpha\,\mathrm{Var})$ **形の起源。** 分散の指数による洗練はどこから来るか？ ステップ分布の第二累積率から動機づけられます。$\log(f(x)/x) = \mu + \delta(x)$ と書く、μ は平均対数ステップ（つまり $\log\gamma$）、δ は零平均揺動。期待値 $\mathbb{E}[f(x)/x] = e^{\mu} \cdot \mathbb{E}[e^{\delta}]$、二次累積率展開で $\mathbb{E}[e^{\delta}] \approx \exp(\frac{1}{2}\mathrm{Var}[\delta])$。洗練 $\widetilde{\Pi} = D\gamma\exp(\alpha\mathrm{Var})$ は対数ステップ分散を考慮することで得られる主要次数補正。自由係数 α は連続ステップ間の相関を捉える：i.i.d. 理想化で $\alpha = 1/2$、我々が見てきた限界レジーム反例は α を 0.3 から 0.7 のあいだに置く。

32.2 予想 C2：操作的-臨界等価

命題 32.3 (**予想 C2。**)**.** 予想 *C1a* を $\Pi < 0.9$ で満たす系について、経験的感受率の頂 σ_c は操作的遷移閾値を、固定 Π で格子点あたり独立サンプル数 n にスケールする $n^{-1/2}$ の不確かさで特定する。

ステータス。制御シグモイドで検証、ハードウェアデータと整合。標準シグモイド $O(\sigma) = 1/(1+\exp((\sigma-0.5)/0.05))$ 上で 4 つのサンプルサイズでの数値実験を走らせた。各推定は O_{obs} の同時シグモイドフィット、2000 回ブートストラップ。

n	$\mathrm{SD}(\hat{\sigma}_c)$	$n=10$ への比	予測 $\sqrt{10/n}$
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

4 点を通る log–log フィットは指数 $p = -0.510$ を与え、-0.500 の予測の 2% 以内。二つのハードウェアデータ点（Wurm 2026 の 6 実験、$n \approx 20$、ブートストラップ CI ±5%、および $n = 31$ Cepheus-1 データセット、同じ Π で ±3%）は CI 内で同じ線上に座る：比 $\approx \sqrt{31/20} \approx 1.24$、予測と一致。

分野への判定規則。将来の系が σ_c の不確かさを n^{-p} で報告し、p が 1/2 から有意に外れる（例えば $p < 0.3$ または $p > 0.7$）なら、(i) 本解析が見逃した非自明なバイアス源があるか、(ii) 予想

C1a が破れるレジームにあるかの証拠 —— どちらも公表すべき観察。

32.3 作業仮説 WH3：プラットフォーム横断閾値不変性

命題 32.4 (**作業仮説 WH3。**)**.** 同一の名目ノイズモデルの 2 つの異なるハードウェア実装について、感受率枠組みで測定された σ_c 値は、両プラットフォームの伝播されたキャリブレーション不確かさ以内で一致する。

ステータス。単一のプラットフォーム比較（*Ankaa-3* 対 *Cepheus-1*、$n = 31$ 回路にわたる合算 $r = 0.84$）に基づく。データ点 1 つは逸話であって予想ではないため、これを少なくとも 1 つ第二のプラットフォーム対（IQM Spark、IonQ Aria、将来の Rigetti 装置）からの独立確認があるまで作業仮説とラベル付ける。$r = 0.94 \to 0.84$ 低下の候補分解は第 21 章 §21.10 にある。観測値をハードウェア差の不確かさ内で説明し、方法論的欠陥を仮定しない。

32.4 予想 C4：κ 閾値のリスケーリング

本予想の以前の形は同語反復的でした：リスケーリング因子が制約なしなら、**任意の** 2 閾値集合がドメイン特有リスケーリングで関連付けられる。非自明な主張は、リスケーリング因子が**予測可能**であるということ。

命題 32.5 (**予想 C4。**)**.** ドメイン特有の κ リスケーリング因子 c_{dom} は単一量から予測可能：c_{dom}^{-1} は、ドメインの代表データセットに対する順列帰無下での $\log \kappa$ の標準偏差に比例する。すなわち $c_{dom} \approx 1/\mathrm{SD}_{null}[\log \kappa]$。

ステータス。明示的に計算した 4 ドメインで経験的に支持される：磁性、量子、金融、地震。**未解決**：第一原理からの導出、残り 8 ドメインでの試験。

■実践的意味。 C4 が成り立つなら、新ドメインに対する適切な κ 閾値を、代表データセットで第 36 章の順列検定を走らせ、帰無分布の標準偏差を計算することで予測できます。枠組みの `calibrate_kappa_thresholds(data)` メソッドが 1 回の呼び出しでこれを行います。

■これらを反証する方法。 C1–C4 のいずれかに反する系を見つけたら、nfo@forgottenforge.xyz まで反例を送ってください。4 つのうちどれにでも明確な単一反例があれば、それ自体公表に値する結果。

第 33 章

枠組みの限界：レシピを使わないとき

注意

5 つの構造的失敗様式。第 9 章のレシピは、5 つの識別可能なシナリオで失敗します ―― 優雅にではなく、構造的に。「失敗」が意味するのは：レシピは依然 σ_c と κ を返すが、その数値は本書が訓練した期待通りの意味を持たない、ということ。

失敗様式	症状	代わりにすべきこと
プラトーが存在しない	$O(\sigma)$ が掃引窓全体で滑らかかつ単調、飽和なし。$\lvert O'\rvert$ がどこでも大きい	定理 13.1 は適用不可。より高次の微分（$\lvert O''\rvert$ の頂）を探すか、パラメトリックモデルをフィットする
滑らかな遷移、スケール指数 < 1	レシピは σ_c を見つけるが、ブートストラップ CI が掃引より広い	頂ではなく ―― 交差。定理 13.2 を明示的に使い、交差公式を報告
観測窓 $<$ 最長自己相関時間	σ_c がシミュレーションのシードや時系列の開始日によって劇的にずれる	より多くのデータを集め、ブートストラップからブロックブートストラップへ切り替え、「結論不能」と報告
真の遷移が複数	レシピが頂を 1 つ返し、両側のブートストラップ CI が狭いが、その間で跳ねる	第 34 章を見よ：多重頂検出に切り替える
σ が真の制御ではない	σ_c がシステムではなくサンプリング手順を追跡する（どの記録が残されたか、どれが落とされたか）	σ が「選ばれる」のではなく「設定される」よう実験を再設計

枠組みの仕事は、語ることが何もないと知ることでもあります。あなたの問題を上記 5 行のどれにもマップできないなら、おそらく枠組みの範疇外で、ドメイン特化ツールがレシピに勝ります。

第 34 章

多重頂：本当に二つあるとき

第 9 章のレシピは $\chi(\sigma)$ の**大域**最大を返します。異なるスケールで二つの真の遷移を持つ系では、たまたま鋭い方を返す —— 構成的な情報損失。

34.1　診断：頂が二つ、ではなく一つ

■信号。 真の多重頂系は、個別の κ 値がどちらも雑音閾値を越え、ブートストラップとカーネル掃引にわたって位置が安定する、二つ以上の χ の局所最大を示します。

■診断、コードで。

```
import numpy as np
from scipy.signal import find_peaks

# After the standard recipe gives you chi:
peaks, props = find_peaks(chi, prominence=0.1 * chi.max())
print(f"number of peaks above prominence threshold: {len(peaks)}")
for k in peaks:
    print(f"  peak at sigma = {sigma[k]:.3f}, height = {chi[k]:.3f}")
```

`find_peaks` が同等の prominence を持つ複数の頂を返したら、枠組みの `detect_multipeak` ルーチンを走らせる —— すべての頂を個別のブートストラップ CI 付きで返す。

34.2　実践での多重頂

本書の二つのドメインは、**期待される**結果として真の多重頂構造を持ちます：

- *GPU* **キャッシュ遷移**（第 26 章）：L1、L2、HBM 各々が自分の χ 頂を与える。枠組みの `detect_cache_transitions` は既定で 3 つすべてを返す。
- **多段タンパク質展開**：一部のタンパク質は中間状態を経て展開する。各中間状態が遷移。`ProteinAdapter` の `multistage=True` フラグがすべてを返す。

他のあらゆるドメインで、同等の頂が 2 つ見つかることは、予期せぬ第二レジームがあるか（公

表せよ！)、掃引が汚染されているか（デバッグせよ）のどちらかを示す信号です。

34.3 多重頂結果の報告

多重頂を報告するときは、各々を独立した結果として扱う。各々が自身の $\sigma_c^{(i)}$、$\kappa^{(i)}$、ブートストラップ CI、順列 p 値を持つ。Bonferroni で有意水準を頂の数で割る：$\alpha_{\text{per-peak}} = 0.05/n_{\text{peaks}}$。

要点

多重頂構造は、期待されているときは特徴（GPU キャッシュ、タンパク質中間状態）、そうでないときは警告。レシピが沈黙のうちに最高位だけを選ぶ既定は、ほぼ単調な問題には正しく、多峰問題には誤り —— `detect_multipeak` が治療。

第 VI 部

検証・統計・厳密さ

第 35 章

境界チェック

σ_c を報告する前に、頂が**存在しうる**ことを確認します。`check_boundary_conditions(0, sigma)` は $O(\sigma_{\min})$ が $O(\sigma_{\max})$ より有意に大きいこと、範囲が自明でないこと、χ が内部最大を持つことを確認します。いずれか失敗するならレシピは不良設定で、σ_c の報告は無意味です。

第 36 章

順列検定

■帰無仮説。 観測された頂の鋭さ κ は、データのランダムな並べ替えで得られるものより大きくない。等価には：「χ の見かけの頂は、同じ値の集合を持つ任意のランダム系列から来た可能性がある」。

■手順。 O 配列の $B = 10\,000$ 回のランダム順列について（σ 軸は固定）、κ を再計算。p 値は κ が観測値を越える順列の割合。

■判定。 $p < 0.05$ で帰無を棄却、真の頂を支持。枠組みはこれを `permutation_test` に同梱。

36.1　数値付きの解答例

$n = 30$ 点での掃引、明瞭なシグモイドを通って減少する観測量、と想像する。枠組みが観測 $\kappa_{\mathrm{obs}} = 5.7$ を報告したと仮定。知りたいこと：この 30 観測のランダムシャッフルが偶然に $\kappa \geq 5.7$ を生む頻度は？

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
rng = np.random.default_rng(0)

# Synthetic sigmoid with shot noise:
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}")      # e.g. 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 mean: {null.mean():.2f}, 95th pct: {np.percentile(null, 95):.2f}")
print(f"p-value:   {p:.4f}")
# Typical: null mean ~ 2.1, 95th pct ~ 2.9, p well below 0.001
```

■解釈。 $p = 0.0003$ なら、10 000 のランダムシャッフルのうち 3 しか我々のような κ を生まない。これが、観測された頂がデータの構造を反映するのであって偶然ではない、ことの強い証拠です。

36.2 ドメインごとの帰無モデル

上の順列帰無は、否定したいものが「順序付き σ 軸が偶然」だけのときに正しい選択。実践ではしばしばより強い否定をしたい。

ドメイン	最も自然な帰無モデル	根拠
量子（制御 γ 掃引）	完全ランダム順列	ショット i.i.d.、順序は選択
磁性（モンテカルロ掃引）	完全ランダム順列	同上
金融（時系列）	ブロック順列、$\ell \approx 20$ 日	自己相関、局所構造を保つ
地震学（カタログ）	円形ブロック順列	余震の時間クラスタリング
気候（リアナリシス時系列）	ブロック順列、$\ell =$ 総観時間スケール	大気持続性
GPU ベンチマーク（逐次）	ブロック順列、$\ell =$ ウォームアップ窓	熱の繰越
ML（LR-range test）	完全順列で十分	LR スケジュールはあなたが制御
タンパク質（変異パネル）	ラベル順列（変異 $\leftrightarrow$ 表現型）	各変異は独立
数論（分解能掃引）	適用不可	決定論的、雑音なし
LLM コスト（モデルリスト）	適用不可	候補 ~ 10、完全列挙

■ブロック順列、一段落で。 O_i を全位置にわたって独立にシャッフルする代わりに、長さ ℓ の連続ブロックをシャッフル。素朴シャッフルが破壊する短距離自己相関を保ち、誠実に大きな p 値を与えます。枠組みの `permutation_test(..., block_size=ℓ)` がこれを実装。ブロックサイズを渡さなければ完全順列にデフォルト。O の自己相関が 0.1 を下回るラグを ℓ に選ぶ。

第 37 章

頂の鋭さ閾値

経験的閾値の再述：

- $\kappa < 1.5$：証拠不十分、σ_c を公表しない。
- $1.5 \le \kappa < 3.0$：暫定、プラットフォーム横断または観測量横断チェックで補完。
- $3.0 \le \kappa < 8.0$：明瞭な頂。
- $\kappa \ge 8.0$：臨界的振る舞い —— 例外的鋭さ。

第 38 章

Fisher 情報限界

Cramer–Rao が感受率に対する基本的下界を与えます：$\chi(\sigma) \geq |dg/d\sigma|/\sqrt{I_F}$、$I_F$ はパラメータに関する観測量の Fisher 情報量。測定された χ がこの境界を飽和するなら、最適な観測量を持っています。枠組みは両者を `fisher_information_bound` で計算します。

第 39 章

多重検定

N 個の独立データセット上で σ_c レシピを走らせると、偶然による偽頂を「発見」します。Bonferroni 補正：有意閾値を N で割る。磁性論文は $\alpha = 0.05/120 = 4.2 \times 10^{-4}$ を使った（6 実験 $\times$ 20 平均頂探索）。

第 40 章

観測量横断クロスバリデーション

σ_c が観測量の選択ではなくシステムの性質であることの最強証拠は、異なる観測量で解析を繰り返し、σ_c が一致することを見ること。磁性論文では、エンタングルメント証人 W、そのシフト $W + C$、その二乗 W^2 すべてが $\gamma_c = 0.6737$ を生んだ（変動係数 0%）。**これがゴールドスタンダード。**

第 VII 部

例題とレシピ

第 41 章

レシピ：純 NumPy/SciPy で最小 σ_c

```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,
    }
```

これが核全体。`sigma_c_mini.py` として保存 —— どの枠組みにも依存しません。

第 42 章

レシピ：σ_c のブートストラップ CI

```
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])
```

第 43 章

レシピ：カーネルの選び方

```
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()
```

k を変えるとき σ_c が劇的にずれるなら、データが騒がしすぎるか頂が鋭すぎる —— n を増やすか、Savitzky–Golay 微分を使う。

第 44 章

レシピ：完全枠組みのインストール

```
# Core
pip install sigma-c-framework

# With quantum integrations (Braket, Qiskit, PennyLane)
pip install "sigma-c-framework[quantum]"

# With GPU acceleration (CuPy)
pip install "sigma-c-framework[gpu]"

# Verify
python -c "import sigma_c; print(sigma_c.__version__)"
# Expected: 3.1.0
```

第 45 章

レシピ：自分用アダプタの作り方

```
from sigma_c.core.base import SigmaCAdapter
import numpy as np

class MyAdapter(SigmaCAdapter):
    def get_observable(self, data, **kwargs):
        # Domain-specific: turn raw data into a scalar
        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}"

# usage
adapter = MyAdapter()
result = adapter.compute_susceptibility(sigma_array, observable_array
    )
```

3 つのメソッドオーバーライド、それ以外（診断、自動探索、説明）は普遍的振る舞いを継承します。

第 46 章

レシピ：エンドツーエンド合成実験

```
import numpy as np
from sigma_c import Universe

# Synthesise Ising-like magnetisation curve
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"  detected Tc = {res['sigma_c']:.3f}")
print(f"  true Tc     = {Tc_true}")
print(f"  kappa       = {res['kappa']:.2f}")

# Validate
val = mag.compute_susceptibility(T, M, kernel_sigma=0.7, validate=True)
print(f"  passes peak test: {val.get('peak_clarity_passes')}")
print(f"  observable quality score: {val.get('observable_quality'):.2f}")
```

第 47 章

レシピ：ストリーミング sigma_c によるライブ監視

```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)
```

Welford のアルゴリズムがサンプルあたり $O(1)$ で σ_c を更新、PID コントローラがパラメータを目標 $\sigma_c = 0.7$ に戻す。実時間実験設定で有用（例：レーザー空洞を操作的安定点に保つ）。

第 48 章

レシピ：公表のための結果報告

1. 境界チェックは通るか？（`check_boundary_conditions`）
2. 観測量品質スコア ≥ 0.75？（`observable_quality_score`）
3. Bonferroni 後の順列 p 値 < 0.05？
4. σ_c のブートストラップ CI は主張に十分狭いか？
5. 観測量横断チェック：別の Fisher 整合観測量で同じ σ_c？
6. カーネル頑健性：σ_c がカーネル 0.3–1.5 にわたって安定？

6 つすべてを報告。コミュニティの査読者はこれらを明示的に探します。

付録 A

記号用語集

σ	制御パラメータ（任意のドメイン）。他のあらゆる本ではガウス分布の標準偏差を意味する記号 —— 統計家には謝罪します
O	観測量（σ の任意のスカラー関数）
$\chi(\sigma) = \lvert dO/d\sigma \rvert$	一般化感受率
σ_c	χ の頂の位置。枠組みが見つける仕事の数値
κ	頂の鋭さ（既定：$\chi_{\max}/\bar{\chi}$）。本書でもカクテルでも、大きい方が良い
D	収縮欠損 $= \lvert S \rvert / \lvert f(S) \rvert$
γ	ドリフト（$f(x)/x$ の幾何平均）。数論では縮小、量子ハードウェアではノイズ、著者のノートブックの火曜には時にその両方
$\Pi = D\gamma$	収縮積（普遍的閾値）
k_B	Boltzmann 定数 1.380649×10^{-23} J/K。本書で値が国際的合意によって固定された唯一の記号
ξ	操作的相関長
σ_{ker}	ガウスカーネル幅（既定 0.6）。用語集中、ガウス分布を本当に指す唯一の σ

付録 B

証明：内部最大の存在

Proof. $O\colon [a,b] \to \mathbb{R}$ を連続関数で、$O(a) > O(b)$ かつ仮定により、両端点近傍で単調でないとする。$\chi(\sigma) = |dO/d\sigma|$ と定義する（微分が存在する点で、そうでなければ ϵ 近傍での絶対傾きの上限ノルムを取る）。

$[a,b]$ 上の平均値の定理により、$O'(\sigma^*) = (O(b) - O(a))/(b-a) < 0$ を満たす $\sigma^* \in (a,b)$ が存在する。ゆえに $\chi(\sigma^*) > 0$。

O の連続性と非単調性仮定により、a の任意の近傍に $O' \approx 0$ となる部分区間がある、すなわち $\chi(a) < \chi(\sigma^*)$。b 近傍でも同様。したがって χ はコンパクト区間 $[a,b]$ 上の最大値をある内部点 $\sigma_c \in (a,b)$ で達成する。 □ □

付録 C

12 個の $qn + c$ 写像、予測と観測

写像	D	γ	$D\gamma$	予測	観測
$3n+1$（サイクル）	2.06	0.563	1.16	収束（サイクル）	既知の全軌道が 1 に到達
$3n+1$（単一）	1.71	0.750	1.28	収束（サイクル）	一致
$5n+1$	1.43	1.250	1.79	発散	一致
$7n+1$	1.60	1.750	2.80	発散	一致
$3n-1$	1.33	0.750	1.00	臨界／サイクル	サイクルあり
$3n+3$	2.00	0.750	1.50	サイクル	一致
$3n+5$	1.48	0.750	1.11	サイクル	一致
$3n+7$	1.56	0.750	1.17	サイクル	一致
$9n+1$	1.34	2.250	3.02	発散	一致
$11n+1$	1.37	2.750	3.77	発散	一致
$5n+3$	1.36	1.250	1.70	発散	一致
$5n-1$	1.43	1.250	1.79	発散	一致

付録 D

注釈付き読書リスト

本書の全土台となる少数の著作を、次に行く先を選べるよう一行注釈付きで列挙します。

■種となった論文。 **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. 枠組みの査読された経験的アンカー。第 21 章はその読解ガイド。

■古典的文脈。 **H. E. Stanley,** *Introduction to Phase Transitions and Critical Phenomena* (Stanley, 1971).「感受率」という語がすべてを継承した教科書。平衡統計力学の起源の物語を欲しいときだけ必読。

L. Onsager, *Crystal Statistics, I* (Onsager, 1944). 1944 年の 2 次元 Ising モデル厳密解、$T_c = 2J/\ln(1+\sqrt{2})$ を与える。第 22 章を較正する標準。

J. M. Kosterlitz & D. J. Thouless, *Ordering, metastability and phase transitions in two-dimensional systems* (Kosterlitz and Thouless, 1973). 感受率的見方への補完：発散する χ なしの位相的遷移。

■情報と Fisher。 **R. Landauer,** *Information is Physical* (Landauer, 1991). 第 31 章の $\log_2 D$ 接続に熱力学的歯を与える 1 ページ論文。

C. W. Helstrom, *Quantum Detection and Estimation Theory* (Helstrom, 1976). 第 VI 部の Fisher 情報章で使われる感受率への Fisher 情報限界。

S. L. Braunstein & C. M. Caves, *Statistical distance and the geometry of quantum states* (Braunstein and Caves, 1994). QFI と状態空間の微分幾何を結ぶ 1994 年論文。枠組みの $\chi \approx \sqrt{F_Q}$ 等式の厳密な裏付け。

■ハードウェア時代。 **J. Preskill,** *Quantum computing in the NISQ era and beyond* (Preskill, 2018). 第 21 章で検出される操作的スケールがなぜ重要かの文脈。

■方法論。 **B. Efron,** *Bootstrap methods: another look at the jackknife* (Efron, 1979). 原典ブートストラップ論文、第 8 章の土台。

A. Savitzky & M. J. E. Golay, *Smoothing and differentiation of data by simplified least*

squares procedures (Savitzky and Golay, 1964). 枠組みの Savitzky–Golay 微分オプションに名を与えた 1964 年論文。

S. Williams, A. Waterman & D. Patterson, *Roofline* (Williams et al., 2009). 第 26 章で使われる性能上限を定義する 2009 年論文。

■特定ドメイン。 **B. Gutenberg & C. F. Richter,** *Frequency of earthquakes in California* (Gutenberg and Richter, 1944); **F. Omori,** *On the aftershocks of earthquakes* (Omori, 1894). 第 24 章の二つの経験則。

G. D. Nastrom & K. S. Gage, *A climatology of atmospheric wavenumber spectra of wind and temperature observed by commercial aircraft* (Nastrom and Gage, 1985). 第 25 章のメソスケール遷移の背後のデータセット。

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). 第 27 章の LR-range test と臨界バッチサイズ現象の参考。

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). 第 30 章の背景と現状。

■枠組み自身。 **ForgottenForge,** `sigma-c-framework` v3.1.0 (ForgottenForge, 2026). 付随のオープンソースライブラリ。

参考文献

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.

索引

Ankaa-3, 75

χ, 50
Collatz 予想, 135
Cramer-Rao 限界, 164

D (収縮欠損), 64

Fisher 情報, 164

γ (ドリフト), 66
GARCH, 95
GPU, 110
Gutenberg-Richter 法則, 101

Hurst 指数, 95

Ising モデル, 88

κ (頂の鋭さ), 53

LR-range test, 117

$\Pi = D \cdot \gamma$, 68

qn+c 写像, 135

TTR, 142

V30M, 142

Wurm 2026, 75

アミロイド, 142
ガウス平滑器, 33
キュリー点, 88
タンパク質, 142
ドリフト, 66
ブートストラップ, 39
ルーフラインモデル, 110

収縮欠損, 64
収縮積, 68
地震学, 101
大森の法則, 101
学習率, 117
平滑化
    ガウス, 33
微分, 28

感受率, 一般化, 50
数論, 135
機械学習, 117

熱スロットリング, 110
磁性, 88

量子ハードウェア, 75
金融, 95
頂の鋭さ, 53
順列検定, 161

付録 E

再現性のための覚書

本書第 IV 部に引用したすべての数値は、対応するファイルが `sigma-c-framework` v3.1.0 リリース内にあります。Wurm 2026 結果を生データから再現するには：

```
git clone https://github.com/forgottenforge/magneto
cd magneto
python magnetvali.py --experiment E3 --bootstrap 1000
# Output: gamma_c = 0.6737 +/- 0.036, kappa = 8.58
```

12 個の $qn + c$ 写像すべてには：

```
from sigma_c import Universe
nt = Universe.number_theory(map_type='collatz')
print(nt.verify_twelve_predictions(M=12))
```

付録 F

第 9 章データの入手先

第 9 章はレシピを 3 つのデータセットで実演します：2D Ising 磁化、2008 年の S&P 500 収益、2019 年リッジクレスト系列前後の南カリフォルニア地震カタログ。家でいずれかを再現したければ、ここが起点です。データセットあたり 10–20 分、GPU 不要。

F.1　キュリー点（Ising 磁化）

第 9 章の Ising データは**シミュレートされた**もので、ダウンロードしたものではありません —— 第 22 章の Metropolis モンテカルロを 10 個の温度で走らせた。**データはあなたが生成します。**

- **コード**：第 22 章の `ising_mc` 関数、NumPy で ∼ 20 行。
- **計算**：$L = 16$ で全掃引はノートパソコンで約 90 秒。
- **Excel 代替**：実行不可能 —— モンテカルロループには本物のプログラミング言語が必要。Python がなければ `python.org` からインストール、無料で 10 分。

F.2　S&P 500 日次収益（2008 年の例）

- **出典**：Yahoo Finance、ティッカー `^GSPC`。無料、ログイン不要。
- **Python 一行**：

```
import yfinance as yf
df = yf.download('^GSPC', start='2008-01-01', end='2009-06-30')
```

- **Excel**：`finance.yahoo.com` にログイン、`^GSPC` 検索、「Historical Data」クリック、日付範囲設定、「Download」クリック。Date、Open、High、Low、Close、Adj Close、Volume の CSV が得られる。Excel で `=LN(Close_today) - LN(Close_yesterday)` で対数収益を計算 —— これが §9.2 のレシピが操作する列。
- **注**：同じワークフローは任意のティッカーで動きます。比較として `BTC-USD` や `GC=F`（ゴールド）も試してみてください。

F.3 南カリフォルニア地震カタログ（リッジクレスト）

- **出典**：SCEC Earthquake Data Center、
  scec.org/research-tools/downloadable-catalogs。
  Hauksson **カタログ**を選び、日付範囲 2018-01 から 2021-06、マグニチュード床 2.5。
- **形式**：日付、時刻、緯度、経度、深さ、マグニチュード列のタブ区切りテキスト。
- **Python 読み込み**：

```
import pandas as pd
cat = pd.read_csv('hauksson.txt', sep='\s+', skiprows=1,
                  names=['date','time','lat','lon','depth','mag'])
```

- **Excel**：ファイルを直接開き、「Text to Columns」で分割。b 値計算は `=0.4343/(AVERAGE(F:F)-MIN(F:F))`、列 F はフィルタした 6 か月窓内のマグニチュード。窓中心ごとに繰り返して §9.3 を再現。

■Python をインストールしたことがなければ。 `python.org/downloads` へ行き、最新の 3.x をインストール。ターミナルを開き `pip install numpy scipy yfinance pandas matplotlib` を実行。本書のすべてのスニペットをカバーします。失敗するなら、`sigma-c-framework` リポジトリの README が 5 つの OS でのフォールバック指示を持っています。

■何もインストールしたくなければ。 `kaggle.com`、`colab.research.google.com`、`deepnote.com` がいずれも、上記パッケージのプリインストール状態で Python をブラウザタブ内で動かします。実機 Braket 実行を除くあらゆる章に、無料プランで十分。

付録 G

謝辞

本概論集は、`sigma-c-framework` リポジトリ (v3.1.0) に記録された先行業績、Rigetti Ankaa-3 および Cepheus-1 量子プロセッサ上で収集された経験データ、方法論を現在の形まで磨き上げた査読プロセスなしには存在しません。

■レビュアたち、名指して。 2 つの改訂ラウンドにわたって 3 人の読者の声が本原稿を形作りました。*Sabine*、バイエルンの大学出版会の番組編集者、長い題を退役させ「七つの記号」カードをあるべき位置に置かせた。*Linus*、16 歳のち 17 歳、第 III 部はコーヒーカップが戻ってくるまで読めないと言い、そして正しかった。*T.P.*、二通の手紙が机の引き出しに住む匿名の理論家、原稿が許容しなかった 4 つの予想に名を与えた。本書のうちで誠実に読めるところがあるなら、その手柄は 4 等分。誤りは私のものです。

家族には果てしない着想に感謝。

付録 H

奥付

■書名。 **頂 —— 相転移のための普遍的レシピ、多くの世界にて**。題は 2 度の原稿ラウンドと 12 か月にわたってワークショップされました。初期の草稿はより長い作業説明で回覧されていましたが、本版で退役、他所には保存していません。

■組版。 LaTeX、`bxjsbook` クラスを `lualatex` と `luatexja` で組版。本文 11 pt の IPAex 明朝、節見出しは IPAex ゴシック、章題は斜体。図は TikZ、コードは `listings` パッケージで `upquote` 有効。

■ライセンス。 GNU Affero General Public License バージョン 3 以降（AGPL-3.0-or-later）下のオープンソース。AGPL 義務を負わない商用ライセンス —— 主に付随枠組みを組み込むクローズドソース派生著作向け —— は nfo@forgottenforge.xyz で入手可能。AGPL-3.0 ライセンス本文は枠組みのソースリポジトリに逐語的に再録されています。

■再現性。 本書中のあらゆる数値結果は `sigma-c-framework` v3.1.0 リリースから再現可能。量子ハードウェアデータを参照する解答例は `magneto` リポジトリの特定コミットに固定されています。両リポジトリのタグ付き Git リビジョンは付録 E に列挙。

■正誤表。 正誤表は `https://forgottenforge.xyz/compendium/errata` に維持され、以後の印刷に組み込まれます。訂正は nfo@forgottenforge.xyz に件名 `[compendium errata]` で送付。

■版と連絡先。 第 1 版、2026 年 5 月。ドイツ、ブッケンホーフ。著者は上記のアドレスで連絡可能、どんな深さの質問も歓迎、ほとんどは一週間以内に回答。

■裏表紙テキスト（印刷版）。 レビュアや図書館目録作成者のため、印刷版の裏表紙テキストをそのまま再録します。

レシピ。調節可能なノブと測定可能な応答を持つ系を取り、それが転倒する場所を告げる、*3* 行の *Python*。

同じ *3* 行が鉄のキュリー点を 1043 *K* 近くに位置同定し（第 *22* 章）、*2008* 年 *8* 月リーマン・ブラザーズが倒れた週に降伏したボラティリティレジームを位置同定し（第 *23* 章）、*2019* 年 *7* 月リッジクレスト系列に先行した南カリフォルニア地震カタログの b 値低下を位置同定する（第 *24* 章）。*12* ドメイン、*1* レシピ。

算術にはじまり公表可能な診断に終わる教科書。革命を約束はしません。よい執事を約束します。