FORGOTTEN FORGE

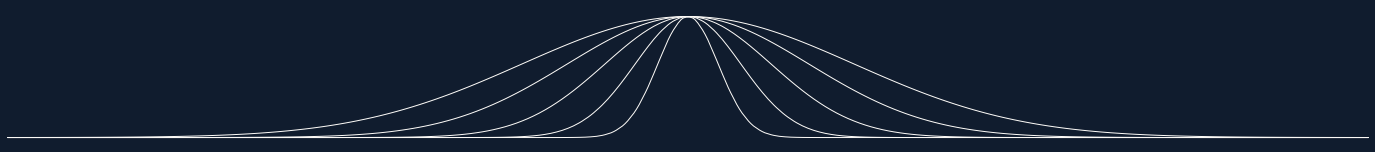
Un manuel en six parties

Le Sommet

une recette universelle pour les transitions de phase, en de nombreux mondes

matériel quantique — magnétisme — finance — apprentissage automatique
sismologie — climat — GPU — protéines
théorie des nombres — edge — économie des LLM

M. C. Wurm
ForgottenForge | Buckenhof | 2026



Compagnon de SIGMA-C-FRAMEWORK v3.1.0
Ancrage empirique : AVS Quantum Science **8**, 013804 (2026)

Le Sommet

une recette universelle pour les transitions de phase,
en de nombreux mondes

M. C. Wurm

ForgottenForge.xyz | 2026

Préface

σ_c est l'endroit où le système bascule. $\Pi = D\,\gamma$ est la raison pour laquelle il bascule. La majeure partie de ce livre traite du premier. Le tiers central traite du second.

Nous ne vous promettons pas une révolution. Les révolutions sont bruyantes, désordonnées, mal organisées ; le lendemain matin, personne ne se charge du rangement. Nous vous promettons quelque chose de plus discret. Une recette. Trois lignes de Python qui prennent un système doté d'un bouton réglable et d'une réponse mesurable, et vous disent où il bascule.

Balayer un paramètre. Mesurer. Dériver. Trouver le sommet. Quatre étapes ; les trois dernières ne changent pas d'un domaine à l'autre, seule la première change. Les mêmes quatre étapes localisent la température de Curie d'un aimant en fer, le seuil opérationnel de bruit d'un processeur quantique, le changement de régime d'un marché d'actions, l'âge d'apparition d'une maladie héréditaire des protéines. Douze domaines, une recette. Nous savons que cela sonne comme un boniment de camelot. La première moitié de ce livre est écrite pour les sceptiques — leurs lecteurs restent plus fidèles.

Une bonne méthode est comme un bon majordome. On ne la remarque pas. Elle fait son travail discrètement. Elle n'invente pas de problèmes et ne résout rien de plus que ce qu'on lui demande. Les jours où elle ne va pas bien, elle vous dit qu'elle ne peut pas aider, et part chercher quelqu'un qui le peut. Le cadre de susceptibilité, lorsqu'il se tient bien, est ce genre de majordome. Lorsqu'il se tient mal, il vous dit exactement laquelle des cinq conditions de confiance a échoué — et c'est, à dessein, le chapitre que nous plaçons en tête de la Partie II.

Comment cela a commencé. Cela a commencé par devoir regarder.

Pas tout à fait de mon plein gré. Une fois atteint un certain âge, on a déjà rencontré un nombre considérable de semblables, et un sous-ensemble particulier d'entre eux semble s'être spécialisé dans l'art d'agir d'une façon qui, rétrospectivement, pose la question : quand, précisément, cela est-il arrivé ? Le collègue qui démissionne d'un poste sûr pour élever des cryptomonnaies. L'ami qui, après trois ans à se plaindre de sa fiancée, l'épouse soudainement. La patiente qui annonce en consultation qu'elle a reconsidéré l'opération, ayant trouvé mieux en ligne. Le législateur qui nomme une commission. Le propriétaire qui rénove la cuisine pour la onzième fois.

Je n'étais pas présent au moment où ces personnes ont pris leur décision. Mais j'étais présent peu avant et peu après, et dans chacun de ces cas il y avait un moment — un moment étroit, souvent pas une journée entière — où la personne se tenait d'un côté, et peu après se tenait, irrévocablement, de l'autre.

J'appelle désormais ce moment le *point de bascule*. C'est un nom modeste pour une chose qui, à dire vrai, me laisse sans voix.

Pendant un temps j'avais espéré qu'il s'agissait d'une observation sociale. Quelque chose qui ne concernait que les êtres humains, et parmi eux seulement les réticents. Il s'avéra qu'il n'en était rien. Les aimants basculent. Les marchés basculent. Les ordinateurs quantiques basculent. Les tremblements de terre, le climat, les réseaux électriques, les protéines — tous ont des points de bascule. C'est la propriété la plus répandue que je connaisse chez les systèmes, juste après la propriété de pouvoir tomber en panne.

Cela en soi ne serait pas si grave. Ce qui est grave, c'est qu'après avoir passé quelques années à identifier ces points de bascule — ce qui, d'ailleurs, n'est pas particulièrement difficile une fois que l'on sait quoi chercher — je ne sais toujours pas comment revenir du côté désagréable au côté agréable.

C'est la seconde partie. La première partie — « voici le point de bascule » — est mathématiquement traitable. C'est ce livre.

La seconde partie — « comment revenir là où c'était agréable » — n'est pas mathématiquement traitable. Peut-être n'est-elle pas traitable du tout. Peut-être se trouve-t-elle simplement au-delà de ce qu'un seul individu peut régler en une vie. Je serais reconnaissant que quiconque lit ces lignes m'en envoie une idée.

D'ici là : la procédure décrite dans ce qui suit vous dit, au moins, qu'un point de bascule existe. Elle vous dit où il se trouve. Elle vous dit même à quel point il est tranchant, ce qui s'est avéré être d'une utilité inattendue. Ce qu'elle ne vous dit pas, c'est ce qu'il faut faire une fois que vous y êtes arrivé. Au lecteur d'en décider.

C'est un livre modeste. Il fait une chose. Cette chose-là, je n'avais pas eu l'intention de l'inventer — elle s'est imposée à moi, à force de regarder, pendant plusieurs décennies, que je le veuille ou non. Je la transmets ici dans l'espoir que, la prochaine fois, quelqu'un regarde au bon moment et dise : « Attendez, un instant — ceci est sur le point de basculer. Êtes-vous sûr que c'est ce que vous voulez ? » Même cela serait une amélioration sur l'état présent des choses, quoique modeste.

Une dernière chose. Nous supposons que vous savez additionner, soustraire, multiplier, diviser. Nous supposons que vous savez lire. Nous ne supposons pas que vous avez déjà vu une lettre grecque, rencontré une espérance ou exécuté un script Python. Si une expression de la table des matières vous intimide, ignorez-la ; lorsque vous arriverez au chapitre, elle ne le fera plus.

Le premier brouillon de ce livre faisait 78 pages et était illisible. Le second faisait 130 pages et était malhonnête. Voici le troisième, et il est honnête à la manière dont un bon majordome est honnête — en rapportant ce qui se trouve effectivement dans la pièce.

À propos de ce livre

Ce livre a une seule mission. Prendre un lecteur capable de faire de l'arithmétique et l'amener au point où il pourra appliquer le *cadre de susceptibilité* — la méthode du σ_c — pour trouver des seuils critiques dans ses propres données. Ordinateur quantique ou marché financier, matériau magnétique ou benchmark GPU. La recette est une seule recette.

À qui ce livre s'adresse. À quiconque a déjà mesuré quelque chose tout en tournant un bouton et s'est demandé : *« où est-ce que cela bascule ? »* Un diplôme n'est pas requis. Les quatre opérations arithmétiques et la patience pour un exemple détaillé suffiront.

Ce que ce livre n'est pas. Une monographie de recherche. Nous ne supposons pas que vous ayez lu des articles sur les transitions de phase, l'information de Fisher ou la mécanique statistique hors équilibre. Là où ces idées apparaissent, nous les construisons depuis la base.

La promesse. À la fin de la Partie II, vous ferez tourner la méthode du σ_c en cinq lignes de Python. À la fin de la Partie III, vous comprendrez *pourquoi* elle fonctionne. À la fin de la Partie IV, vous saurez lequel des adaptateurs de domaine fournis correspond à votre problème, et comment étendre le cadre à un treizième qui sera le vôtre.

Comment lire. Les Parties I à III forment une chaîne. Sauter des maillons laisse des lacunes dans lesquelles les chapitres ultérieurs tombent silencieusement. À partir de la Partie IV, chaque chapitre se suffit à lui-même. Choisissez le domaine qui correspond à votre travail ; les autres seront là quand vous en aurez besoin.

Conventions.

- Chaque nouveau symbole est défini la première fois qu'il apparaît. Les lettres grecques n'ont pas de laissez-passer.
- Chaque formule est suivie d'au moins un exemple numérique. Si vous ne pouvez pas reproduire le nombre sur papier, l'explication n'est pas encore terminée.
- Chaque chapitre se termine par un petit exercice. Il prend de cinq à trente minutes et vaut son temps.
- Le code est en Python et tourne dans tout environnement disposant de NumPy et SciPy. La bibliothèque `sigma_c` est une commodité, non une dépendance — le noyau de cinq lignes tient sur un sous-bock.
- Les phrases qui survivent à une section sans être remplacées par une meilleure sont placées dans un encadré coloré à la fin. Quatre couleurs, quatre fonctions : *bleu* pour

ce qu'il faut retenir, *orange* pour ce qu'il faut éviter, *vert* pour ce qu'il faut essayer, et *rouge* pour les avertissements explicites (cliniques, financiers, de sécurité).

— **Le symbole σ est redéfini localement dans chaque chapitre.** Dans chaque chapitre σ désigne l'axe de contrôle *de ce chapitre*, et non un sigma physique universel. La lettre grecque est utilisée parce qu'elle est courte et traditionnelle ; elle ne porte aucune signification physique propre d'un domaine à l'autre. Nous utilisons σ_{ker} (largeur de noyau) et σ_t (volatilité, en finance) avec des indices explicites pour éviter les collisions.

Le cadre de référence. La bibliothèque à code ouvert qui accompagne ce livre est `sigma-c-framework` (version 3.1.0). Installez-la avec

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

Vous pouvez terminer ce livre sans jamais l'installer ; l'algorithme central de cinq lignes ne requiert que NumPy et SciPy. Le cadre est plus confortable, comme un majordome est plus confortable que pas de majordome.

Citer ce livre.

> M. C. Wurm, *Le Sommet : une recette universelle pour les transitions de phase, en de nombreux mondes*, ForgottenForge, Buckenhof, 2026.

(Les détails de licence et d'édition figurent dans le colophon à la fin.)

Trois parcours de lecture

Ce livre contient quelque quatre cent mille choses que vous pourriez y lire. Vous ne voulez certainement pas toutes les lire. Trois parcours de lecture explicites, par profondeur croissante :

Parcours A — « je veux seulement m'en servir » (1–2 soirées).

- Avant-propos et guide des lettres grecques (15 min)
- Chapitres 1–7 *en survol* (les fondations ; passez-les rapidement si vous avez déjà fait du calcul différentiel)
- Chapitre 9 (**la recette universelle**)
- Chapitre 10 (**quand la méthode échoue**)
- Chapitre 12 (**seuils de κ**)
- Un chapitre de la Partie IV correspondant à votre domaine
- Partie V (**validation**) lorsque vous êtes sur le point de publier

Parcours B — « je veux comprendre pourquoi cela fonctionne » (une semaine). Parcours A, plus toute la Partie III (géométrie de contraction) et toute la Partie V (validation). Survolez les chapitres de la Partie IV en dehors de votre domaine.

Parcours C — « je veux étendre le cadre » (deux semaines). Parcours B, plus les recettes de la Partie VI et les annexes, plus la mise en œuvre effective de votre propre adaptateur (Chapitre 45). À ce stade vous êtes contributeur, plus seulement lecteur.

Ce que ce livre ne promet pas. Il ne promet pas une théorie de tout. Il ne promet pas que tout système possède un sommet ; certains n'en ont pas, et le Chapitre 10 vous aidera à savoir lesquels. Il ne promet pas que la même valeur numérique de σ_c vaudra sur toutes les plateformes matérielles ; le suivi de l'article de magnétisme sur différents matériels montre que la valeur varie de ± 3–$5\,\%$ lorsqu'on change le modèle de bruit. Il ne promet pas que la méthode remplace l'expertise du domaine ; la méthode trouve où regarder, c'est à vous d'interpréter ce que vous voyez.

Ce que nous *promettons* : si votre système possède un bouton réglable et une réponse à peu près monotone, la recette du Chapitre 9 localisera le seuil opérationnel sans exiger que vous inventiez d'abord un modèle.

Table des matières

Les sept symboles qu'il vous faut vraiment

Si vous ne deviez retenir qu'une chose de ce livre, retenez les sept symboles ci-dessous. Tout le reste est défini lorsqu'il apparaît.

symbole	en une ligne	première apparition
σ	le paramètre de contrôle que vous tournez — la température, le niveau de bruit, le taux d'apprentissage	Chapitre 2
O	l'observable que vous mesurez en tournant ce bouton	Chapitre 2
$\chi(\sigma) = \lvert \mathrm{d}O/\mathrm{d}\sigma \rvert$	la susceptibilité : la vitesse à laquelle O change	Chapitre 11
σ_c	la position du sommet de χ — *là où le système bascule*	Chapitre 9
κ	la netteté du sommet : $\chi_{\max}$ divisé par la moyenne χ	Chapitre 12
D	le défaut de contraction d'une application : $\lvert S \rvert / \lvert f(S) \rvert$	Chapitre 17
γ	la dérive d'une application : moyenne géométrique de $f(x)/x$ (*à ne pas* confondre avec le même γ désignant l'intensité du bruit quantique !)	Chapitre 18
$\Pi = D \cdot \gamma$	bonus, huitième : le produit de contraction — *pourquoi le système bascule*	Chapitre 19

Le résumé en une phrase de tout le livre est :

À RETENIR

σ_c est là où il bascule. $\Pi = D\gamma$ est pourquoi il bascule.

Le guide complet de prononciation des lettres grecques est en face (les lettres classiques demandent un temps d'adaptation, mais chacune n'est qu'une lettre).

Première partie

Fondations — de l'arithmétique à la dérivée

Guide de lecture des lettres grecques

Ce livre utilise des lettres grecques. Les programmes scolaires tendent à en user avec parcimonie, et bien des lecteurs les trouvent plus intimidantes qu'elles ne le méritent. Une lettre grecque est une lettre. Elle se prononce comme ci-dessous, et se comporte sinon exactement comme une lettre ordinaire — on peut la multiplier, l'additionner, écrire des équations avec elle.

symbole	nom	où dans ce livre
σ	« sigma »	paramètre de contrôle (Chapitres 2–7)
σ_c	« sigma-cé »	position du sommet de susceptibilité
χ	« chi » (se prononce « ki »)	la susceptibilité elle-même
κ	« kappa »	netteté du sommet (à quel point il est tranchant)
γ	« gamma »	dérive en théorie des nombres, bruit en quantique
Δ	« delta » (capitale)	changement, différence
ρ	« rho » (« rô »)	matrice densité, corrélation
ξ	« xi » (« ksi »)	longueur de corrélation
π	« pi »	à la fois la constante 3,14… et la persistance GARCH
β, α	« bêta », « alpha »	exposants critiques en magnétisme, paramètres GARCH
∂	« del », « partiel »	un type particulier de dérivée
$\sim$	« va comme », « se comporte comme »	proportionnalité approximative
$\propto$	« proportionnel à »	proportionnalité stricte
$\langle \cdot \rangle$	« crochets angulaires »	moyenne sur de nombreux essais
$\lvert \cdot \rvert$	« valeur absolue »	on retire le signe, on garde la grandeur

Si vous oubliez l'un d'eux, revenez à cette page. Elle ne bougera pas.

Chapitre 1

Qu'est-ce qu'un nombre, qu'est-ce qu'un rapport ?

Ce chapitre ne suppose rien. Si vous savez faire $7+3=10$ et $10\div 2=5$, vous êtes surqualifié. Les lecteurs qui savent déjà ce qu'est une dérivée peuvent le survoler ; ceux qui ne le savent pas devraient s'en abstenir. Tout ce qui suit dans les 300 pages restantes repose sur les quatre opérations arithmétiques faites avec soin. Commençons par les faire avec soin.

1.1 Les nombres, dans la seule façon dont nous les utiliserons

Un *nombre* dans ce livre est l'un des suivants :

— un nombre entier comme 0, 1, 2, 3, -4, 17 ;

— un nombre avec une partie décimale comme 0,5, 3,14, $-2{,}718$;

— un très petit nombre positif comme 0,001 ou 10^{-6}, que l'on lit « dix à la moins six ». Le symbole 10^{-6} est juste une façon courte d'écrire 0,000001.

L'ensemble de tous ces nombres se note $\mathbb{R}$ et s'appelle « les nombres réels ». Vous n'aurez pas besoin d'une définition plus profonde.

1.2 La soustraction vous donne un changement

Si une quantité était 4 avant et vaut maintenant 7, le changement est $7-4=3$. On dit que la quantité *a augmenté de* 3.

Si une quantité était 4 avant et vaut maintenant 1, le changement est $1-4=-3$, un nombre négatif. On dit que la quantité *a diminué de* 3. Le signe moins est la seule chose qui vous indique la direction du changement.

Définition 1.1 (Changement)**.** Le changement d'une quantité O entre deux mesures est $\Delta O = O_{\text{nouveau}} - O_{\text{ancien}}$. La lettre grecque Δ (delta) se prononce « delta » et signifiera désormais toujours « changement de ».

Exemple 1.2 (La tasse de café, partie 1 : le changement)**.** Une tasse de café était à 80 °C il y a un instant. Elle indique maintenant 77 °C. Le changement de température est $\Delta T = T_{\text{nouveau}} - T_{\text{ancien}} = 77-80 = -3\,^{\circ}\text{C}$. Le signe négatif dit : la température *a baissé* de 3 degrés. Nous reviendrons à cette tasse tout au long du chapitre.

1.3 La division vous donne un taux

La tasse de l'Exemple 1.2 n'a pas perdu ses trois degrés instantanément. Disons qu'elle a mis 60 secondes. Le *taux* de refroidissement est alors

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

Deux choses à remarquer. *Premièrement*, on a divisé. *Deuxièmement*, les unités ont suivi : degrés en haut, secondes en bas, d'où « degrés par seconde ».

Si au lieu de cela il n'avait fallu que 5 secondes, le taux serait $-3/5 = -0{,}6\,^\circ\mathrm{C/s}$ — un refroidissement bien plus rapide. Grand taux signifie que le changement se produit vite ; petit taux signifie qu'il se produit lentement. *Cette seule idée est le germe de tout ce qui suit dans ce livre.*

À RETENIR

Un *taux* est un changement divisé par un autre changement. Il vous dit à quelle vitesse une chose bouge quand une autre bouge. Tout dans la méthode du σ_c vient d'un taux spécifique : à quelle vitesse une mesure change lorsqu'on change un paramètre de contrôle.

1.4 Rapports avec unités, et pourquoi les unités comptent

Le taux $-0{,}05\,^\circ\mathrm{C/s}$ dépendait des unités. Si nous avions mesuré le temps en minutes au lieu de secondes, le même refroidissement serait $-3\,^\circ\mathrm{C} \div 1\,\mathrm{min} = -3\,^\circ\mathrm{C/min}$. *Même processus physique, nombre différent, parce qu'unité différente.*

PIÈGE

Quand vous comparez deux taux, vous devez utiliser les mêmes unités pour les deux. Un taux de $-3\,^\circ\mathrm{C/min}$ n'est *pas* plus grand qu'un taux de $-0{,}05\,^\circ\mathrm{C/s}$; ils sont égaux. Vérifiez toujours les unités avant de tirer une conclusion.

1.5 Une chose qui dépend d'une autre

Jusqu'ici nous avons suivi une quantité (la température de la tasse) et un « axe » le long duquel elle variait (le temps). C'est la plus simple relation possible entre deux quantités — une entrée, une sortie. Ce motif est si fréquent que les mathématiciens lui ont donné un nom : une *fonction*.

Le prochain chapitre traite des fonctions. Pour l'instant, gardez simplement l'image en tête : une fonction est une recette qui prend un nombre en entrée (l'*entrée*) et rend un nombre en sortie (la *sortie*). Notre histoire de la tasse qui refroidit est une fonction : à chaque instant t est associée une unique température T, et on peut écrire $T = T(t)$.

À ESSAYER

Une voiture était à 200 km sur une route à 14:00 et à 260 km à 14:30. Calculez sa vitesse moyenne en km/h. La voiture allait-elle plus vite ou moins vite que 100 km/h ?

Solution, étape par étape.

Étape 1 : identifier les deux quantités et leurs changements. La position x est ce qui varie. Le temps t est l'axe le long duquel elle varie (même motif que la tasse de café de l'Exemple 1.2, mais spatial au lieu de thermique).

$$\Delta x = x_{\text{nouveau}} - x_{\text{ancien}} = 260\,\mathrm{km} - 200\,\mathrm{km} = 60\,\mathrm{km}.$$

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

Nous convertissons 30 minutes en 0,5 heure pour que notre numérateur (km) et notre dénominateur (h) donnent au final des km/h, qui est l'unité demandée.

Étape 2 : former le taux.

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

L'arithmétique est une division : $60 \div 0{,}5 = 120$.

Étape 3 : comparer au seuil demandé. $120 > 100$, donc la voiture allait *plus vite* que 100 km/h.

Vérification de bon sens. Si la voiture avait parcouru 60 km en une heure pleine, la vitesse aurait été de 60 km/h — moins de 100. Nous avons parcouru la même distance en seulement *une demi*-heure, donc on s'attend à environ le double, ce qui est exactement ce qu'on a obtenu.

Ce que cet exercice enseigne. Vous avez maintenant les trois ingrédients en un seul endroit : une quantité qui change (x), un axe le long duquel elle change (t), et un taux (leur rapport). La méthode du σ_c plus loin dans le livre est le même motif appliqué à une quantité qui change (un observable O) lorsque vous tournez un bouton (un paramètre de contrôle σ).

Chapitre 2

Qu'est-ce qu'une fonction ?

Dans l'Exemple 1.2 nous avions une tasse de café. Elle s'est refroidie de 80 °C à 77 °C. Un nombre (la température) variait avec un autre (le temps). Cette petite observation est plus importante qu'elle n'en a l'air. Les mathématiciens ont bâti trois cents ans d'analyse dessus. Ils ont donné un nom au motif — *fonction* — et une notation : $f(x)$. Les trois cents ans sont là d'où viennent les dérivées, là d'où viennent les sommets, là où ce livre commence à être utile.

2.1 La fonction vue comme recette

Une *fonction* f est une règle qui, étant donné une entrée x, retourne une sortie spécifique, que l'on note $f(x)$. On lit $f(x)$ « f de x ». Le mot « spécifique » travaille ferme dans cette phrase : une fonction doit donner la même sortie chaque fois qu'on la nourrit de la même entrée. Une règle qui retourne « un nombre entre 3 et 5 » n'est pas une fonction. Une règle qui retourne « la température à Berlin aujourd'hui » n'est pas une fonction non plus — des jours différents, des réponses différentes. Une fonction est un contrat entre l'entrée et la sortie, et le contrat tient.

Exemple 2.1 (Doubler)**.** La règle « doubler l'entrée » est une fonction. Si on l'appelle f, alors $f(3) = 6$, $f(0) = 0$, $f(-1{,}5) = -3$, $f(100) = 200$. On peut écrire la recette de manière compacte : $\boxed{f(x) = 2x}$. Quel que soit le nombre en entrée, le double ressort.

Exemple 2.2 (Élever au carré)**.** La règle « multiplier l'entrée par elle-même » est une fonction. De manière compacte : $\boxed{f(x) = x \cdot x = x^2}$. Ainsi $f(2) = 4$, $f(3) = 9$, $f(-2) = 4$ également (parce que $(-2) \cdot (-2) = +4$, deux négatifs donnent un positif). Nous utiliserons cette fonction comme exemple fil rouge dans les deux prochains chapitres.

On écrira parfois $f : x \mapsto x^2$, qui est juste une notation plus chic pour la même recette de l'Exemple 2.2. La flèche $\mapsto$ se lit « applique sur ». Lisez-la : « f envoie l'entrée x sur son carré ».

2.2 La fonction vue comme table

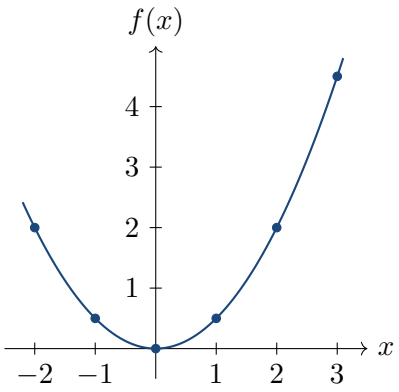
Toute fonction peut être *tabulée* : on choisit des entrées, on écrit chaque entrée à côté de sa sortie. Pour la fonction carré de l'Exemple 2.2 :

entrée x	sortie $f(x) = x^2$
-2	4
-1	1
0	0
1	1
2	4
3	9

C'est exactement le genre de table que vous produirez quand vous *balayerez* un paramètre de contrôle dans une vraie expérience — un mot chic pour « le faire varier sur une plage de valeurs et enregistrer ce qu'on mesure à chaque fois ».

2.3 La fonction vue comme graphique

Si on prend la même table et que l'on place chaque couple $(x, f(x))$ comme un point sur un diagramme avec x allant de gauche à droite et $f(x)$ allant de bas en haut, on obtient un *graphique*.



(Nous avons tracé $f(x) = x^2/2$ plutôt que x^2 ici, uniquement pour que la figure reste assez courte pour tenir sur une page ; diviser la sortie par 2 ne fait qu'aplatir verticalement la parabole. La forme est la même : une courbe en U, appelée *parabole*.)

2.4 Les fonctions en laboratoire

Dans une vraie expérience on n'écrit pas une recette du genre « x^2 ». On ne *connaît* pas la recette. On ne connaît que la table : on règle un paramètre de contrôle, on mesure un observable, on enregistre la paire. La fonction est implicitement définie par les données.

Intuition. Tout au long de ce livre, la fonction qui nous intéresse est la réponse à : *« si je règle le paramètre de contrôle à la valeur σ, quelle valeur prend ma mesure O ? »* On écrit cela $O(\sigma)$. La recette est inconnue. La table est ce que l'on a. Tout le reste se calcule à partir de cette table.

2.5 Le paramètre de contrôle σ

Nous utiliserons la lettre grecque σ (sigma) pour le paramètre de contrôle. Dans différents domaines, σ signifie des choses différentes — et c'est tout le but du cadre. Voici cinq exemples réels :

— Dans une expérience magnétique, σ est la température T de l'échantillon.
— Sur un ordinateur quantique, σ est l'intensité γ du bruit que l'on injecte.
— En finance, σ est un décalage temporel, par exemple le nombre de jours sur lesquels on calcule la volatilité.
— Dans une expérience sur les protéines, σ est une variation d'énergie libre $\Delta\Delta G$ causée par une mutation.
— En apprentissage automatique, σ est un hyperparamètre tel que le taux d'apprentissage.

Le cadre se moque de savoir lequel vous avez. La recette est identique. C'est *cela* que nous voulons dire par « universel ».

À ESSAYER

Choisissez un phénomène mesurable dans votre propre vie — disons, le volume sonore d'une radio quand on tourne le bouton de volume. Identifiez : qu'est-ce que σ ? Qu'est-ce que O ? Imaginez maintenant que vous enregistrez la paire (σ, O) pour dix positions différentes du bouton. Esquissez la table que vous obtiendriez.

Solution détaillée.

Étape 1 : identifier le paramètre de contrôle σ. Le paramètre de contrôle est ce que vous pouvez *régler*. Sur une radio, c'est la position du bouton de volume. Mesurons-la comme un nombre de 0 (silencieux) à 10 (maximum), comme la plupart des graduations le font.

Étape 2 : identifier l'observable O. L'observable est ce que vous *mesurez* en réponse. Pour le son, le naturel est l'intensité sonore, mesurée en décibels (dB) avec un sonomètre d'application mobile ou un SPL matériel.

Étape 3 : choisir dix positions du bouton. On veut un balayage uniforme couvrant toute la plage. Le plus simple est $\sigma = 1, 2, 3, \ldots, 10$. (On saute $\sigma = 0$: une radio silencieuse donne le bruit de fond ambiant de la pièce, qui est un point dégénéré.)

Étape 4 : esquisser la table attendue. Une radio est à peu près linéaire en bouton de volume *en dB*, donc on s'attend à ce que chaque cran ajoute un nombre similaire de décibels :

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

Étape 5 : quel type de fonction est-ce ? Approximativement linéaire : chaque cran ajoute 6–7 dB. Cela veut dire que le tracé de O contre σ serait proche d'une droite. *Pas de transition dans ce système.* La recette du Chapitre 9 n'y trouverait pas de σ_c intéressant.

Ce que cet exercice enseigne. Tous les systèmes n'ont pas un σ_c intéressant. Le cadre est pour les systèmes avec au moins un changement de régime qualitatif. Une réponse monotone et linéaire, comme un bouton de volume, est l'exemple négatif le plus simple.

Chapitre 3

La pente : le taux moyen de changement

3.1 Deux points sur une courbe

Choisissez deux paires quelconques dans la table d'une fonction f : $(x_1, f(x_1))$ et $(x_2, f(x_2))$. Le *taux moyen de changement* entre eux est

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

Cette formule a la même allure que le taux de refroidissement de la tasse de café : *changement en sortie divisé par changement en entrée.*

La pente vous dit : *en moyenne, pour chaque unité d'augmentation de x entre x_1 et x_2, la sortie $f(x)$ augmente de « pente » unités.* Si la pente est $+0{,}5$, la courbe monte d'une demi-unité par unité ; si elle est -2, la courbe descend de deux unités par unité.

3.2 Pente d'une droite

Les fonctions les plus simples sont les *linéaires*, qui ont la forme

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

Les deux lettres a et b sont des nombres fixés (les *coefficients*) ; x est l'entrée. « Linéaire » signifie : le tracé est une droite. Le coefficient a contrôle sa pente ; le coefficient b contrôle sa hauteur en $x = 0$ (car $f(0) = a \cdot 0 + b = b$). La fonction « doubler » du chapitre précédent est la fonction linéaire avec $a = 2$ et $b = 0$.

Pour *toute* fonction linéaire, la pente est la même quels que soient les deux points choisis : c'est toujours le nombre a. On peut le vérifier une fois et y faire confiance pour toujours :

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

Les b se simplifient ; le $a(x_2 - x_1)$ au numérateur se simplifie avec le $(x_2 - x_1)$ au dénominateur, laissant juste a. *Même pente partout sur une droite.*

Exemple 3.1 (Seau sous un robinet)**.** Un robinet remplit un seau selon $f(t) = 2t + 3$ litres après t minutes. Le terme constant $b = 3$ est la quantité d'eau déjà dans le seau à $t = 0$. Le coefficient $a = 2$ est le *débit* : chaque minute, le seau gagne 2 litres. À $t_1 = 1$, $f = 5$. À $t_2 = 4$, $f = 11$. Pente $= (11 - 5)/(4 - 1) = 6/3 = 2$. Comme annoncé.

3.3 Pente d'une courbe — elle dépend de l'endroit

Considérons maintenant $f(x) = x^2$. Choisissons deux paires :

— Entre $x_1 = 1$ et $x_2 = 2$: pente $= (4-1)/(2-1) = 3$.

— Entre $x_1 = 3$ et $x_2 = 4$: pente $= (16-9)/(4-3) = 7$.

La pente d'une courbe n'est pas constante. Elle dépend de *l'endroit* sur la courbe où on mesure. Là où la courbe monte raide (côté droit d'une parabole), la pente est grande. Là où elle est plate (le fond), la pente est proche de zéro. Là où elle descend (côté gauche), la pente est négative.

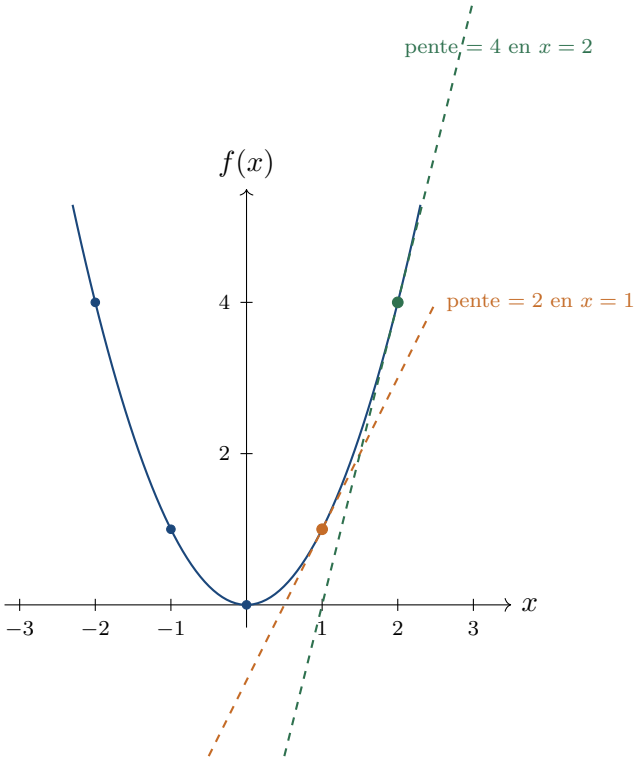


FIGURE 3.1 – La pente de $f(x) = x^2$ n'est pas un seul nombre ; c'est un nombre différent en chaque point. En $x = 1$ elle vaut 2 ; en $x = 2$ elle vaut 4. Quand x grandit, la courbe devient plus raide. Les deux droites en pointillés sont des *tangentes*.

À RETENIR

La pente d'une fonction est une propriété *locale*. Décrire une fonction courbe avec un seul nombre est impossible — il faut une valeur de pente à chaque endroit. C'est ce que la dérivée, définie au prochain chapitre, va nous donner.

3.4 La pente est elle-même une fonction

Regardez la parabole $f(x) = x^2$. Choisissez un point de base x et un petit pas $h > 0$. La pente entre x et $x + h$ est

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

(Si vous ne savez pas encore pourquoi $(x+h)^2 = x^2 + 2xh + h^2$*, multipliez-le vous-même :* $(x+h)(x+h) = x\cdot x + x\cdot h + h\cdot x + h\cdot h$*.)*

Si on imagine maintenant que h tend vers zéro — ce que nous préciserons au prochain chapitre — la pente tend vers la valeur $2x$ au point de base. Donc en $x = 1$ la pente est 2 ; en $x = 3$ la pente est 6 ; en $x = 4$ la pente est 8. *La pente est elle-même une fonction de* x. Nous l'appelons la *dérivée* de f.

(J'ai fait ce calcul sur un quai de gare à Erlangen, un matin de mars 2026, au dos d'un reçu de café, parce que j'avais oublié mon cahier. Le reçu est quelque part dans un tiroir ; le résultat reste $2x$.)

À ESSAYER

Calculez la pente de $f(x) = x^2$ entre $x_1 = 1{,}99$ et $x_2 = 2{,}01$. Voyez-vous quelque chose de proche de la formule $2x = 4$ en $x = 2$?

Solution, étape par étape.

Étape 1 : On a besoin de $f(1{,}99) = 1{,}99^2$ et $f(2{,}01) = 2{,}01^2$.

Étape 2 : raccourci algébrique. Avec $(a-b)^2 = a^2 - 2ab + b^2$ et $a = 2$, $b = 0{,}01$:

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

De même avec $(a+b)^2$:

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

Étape 3 : appliquer la formule.

$$\text{pente} = \frac{4{,}0401 - 3{,}9601}{2{,}01 - 1{,}99} = \frac{0{,}0800}{0{,}02} = 4{,}00.$$

Étape 4 : comparer à la prédiction. La formule donne $f'(2) = 2\cdot 2 = 4$. Notre estimation numérique est 4,00. *Accord à quatre chiffres significatifs.*

Pourquoi si exact ? L'échantillonnage symétrique (centré) annule l'erreur du premier ordre. C'est exactement pourquoi le cadre utilise des *différences centrées* plutôt que des différences avant ou arrière.

Chapitre 4

La dérivée : la pente en un point unique

4.1 L'idée

Nous voulons, en tout point x, un seul nombre qui nous dise à quelle vitesse f change à cet endroit. La recette est celle que nous avons déjà utilisée : calculer la pente entre x et $x+h$, puis *faire tendre h vers zéro.*

Définition 4.1 (Dérivée)**.** La *dérivée* de f au point x, notée $f'(x)$ ou $\frac{df}{dx}$, est la limite

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

à condition que cette limite existe.

Le symbole $\lim_{h\to 0}$ « limite quand h tend vers zéro » se lit : *la valeur dont la pente se rapproche quand le pas devient arbitrairement petit.* On ne divise pas vraiment par zéro. On regarde juste vers quoi la pente se stabilise quand h devient minuscule.

4.2 Trois dérivées à connaître par cœur

Constante : Si $f(x) = c$ (une constante), la pente est toujours nulle, donc $f'(x) = 0$.

Droite : Si $f(x) = a \cdot x + b$, la pente vaut toujours a, donc $f'(x) = a$.

Parabole : Si $f(x) = x^2$, nous avons calculé plus haut que $f'(x) = 2x$.
Pour nos besoins, ces trois motifs suffisent. Les mathématiciens ont tabulé la dérivée de pratiquement toute fonction classique (exponentielles, sinus, logarithmes, etc.), mais nous n'en aurons pas besoin. Pourquoi ? Parce qu'en laboratoire nous n'avons jamais de formule propre ; nous n'avons qu'une table.

4.3 La dérivée quand on n'a qu'une table

Dans une vraie expérience, f est une liste de paires mesurées :

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

Les σ_i sont les valeurs de contrôle que nous fixons, les O_i sont ce que nous avons mesuré. La dérivée au point du milieu d'un triplet $(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ est approchée par la *différence centrée* :

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

C'est exactement la formule de la pente appliquée entre les deux voisins de σ_i.

Exemple 4.2. Vous avez mesuré l'aimantation d'un aimant à trois températures :

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

L'estimation par différence centrée de dM/dT en $T = 2{,}27$ est

$$\frac{0{,}18 - 0{,}55}{2{,}34 - 2{,}20} = \frac{-0{,}37}{0{,}14} = -2{,}64 \ \text{[unités de } M \text{ par K]}.$$

Signe : négatif, parce que M décroît. *Magnitude :* grande, parce que M chute vite dans cette plage de température. C'est cette « grande magnitude là où la fonction chute vite » que la méthode du σ_c va traquer.

4.4 Différences unilatérales aux bords

Aux premier ($i = 1$) et dernier ($i = n$) points de la table, nous ne pouvons pas former de différence centrée car un voisin manque. On utilise :

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

Ce sont les différences *avant* et *arrière*. Elles sont légèrement moins précises que la différence centrée mais suffisent aux bords.

4.5 En code : la dérivée numérique en trois lignes

NumPy fournit cela en standard :

```
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` utilise des différences unilatérales aux bords et des différences centrées au milieu, exactement comme nous venons de le définir.

À RETENIR

La dérivée n'est qu'une pente, appliquée localement. Avec trois lignes de NumPy on transforme toute table (σ_i, O_i) en une autre table $(\sigma_i, O'(\sigma_i))$. Nous passerons le reste du livre à trouver l'endroit où cette dérivée est la plus grande.

À ESSAYER

Prenez la fonction de votre choix — la hauteur d'un verre d'eau qu'on est en train de vider, le prix d'une action sur les dix derniers jours, la température d'une pièce sur une soirée. Enregistrez cinq points à la main. Calculez les différences centrées aux trois points du milieu. Quel point a la plus grande pente en valeur absolue ?

Solution détaillée (exemple : prix d'une action sur dix jours ; la recette est identique pour tout autre choix).

Étape 1 : enregistrer cinq paires (σ_i, O_i).

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

La chute la plus abrupte est clairement entre jour 3 et jour 4.

Étape 2 : appliquer la formule centrée.

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

Étape 3 : valeurs absolues et argmax.

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

La plus grande pente en valeur absolue est au jour 4. C'est le candidat σ_c pour ce petit jeu de données.

Ce que cet exercice enseigne. Les différences centrées étalent l'influence de chaque pas sur *deux* points intérieurs, pas un. C'est pourquoi le cadre choisira un sommet plutôt qu'une marche déchiquetée.

Chapitre 5

Le sommet : là où la pente est la plus grande

5.1 Valeur absolue : ignorer la direction

Dans l'Exemple 4.2 la dérivée était négative ($-2{,}64$). Pour la méthode du σ_c nous nous moquons en général du *sens* dans lequel la fonction bouge ; ce qui nous intéresse c'est *à quelle vitesse*. L'outil standard pour « enlever le signe, garder la taille » est la *valeur absolue*, notée $|x|$:

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

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

Intuition. Une fonction dont la pente vaut $+2$ à un endroit et -2 à un autre change « aussi vite » aux deux endroits, simplement dans des directions opposées. La méthode du σ_c cherche l'endroit où le changement est le plus grand en magnitude. Nous travaillons donc toujours avec $|O'(\sigma)|$, la pente absolue.

5.2 Argmax : la position du maximum

Supposez que vous avez une table de valeurs $|O'(\sigma_1)|$, $|O'(\sigma_2)|, \ldots$ et que vous demandez : quelle ligne a la plus grande valeur ? L'abscisse σ de cette ligne s'appelle l'*argmax* de la fonction :

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

Lisez : « σ_c est la valeur de σ qui rend $|O'(\sigma)|$ la plus grande ».

Exemple 5.1. Supposez

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

La plus grande pente absolue est 1,60, en $\sigma = 1{,}50$. Donc $\sigma_c = 1{,}50$. C'est là où l'observable change le plus rapidement — le candidat pour l'échelle critique du système.

5.3 En code : argmax en une ligne

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

Cette seule instruction est, en résumé, ce que fait tout le cadre σ_c — mais avec beaucoup de raffinements que les chapitres restants expliqueront.

5.4 Pourquoi tout cela est intéressant

Dans tous les domaines que nous regarderons, l'endroit où un observable change le plus rapidement se révèle être un seuil physique ou opérationnel ayant un sens. Quelques avant-goûts :

— Dans un aimant, la plus grande pente absolue de l'aimantation en fonction de la température se produit près du *point de Curie* — la température au-dessus de laquelle le matériau perd son aimantation permanente.

— Sur un processeur quantique, la plus grande pente de l'intrication en fonction du bruit se produit au *seuil opérationnel de décohérence* — le niveau de bruit au-dessus duquel l'information quantique ne peut pas survivre au circuit.

— Sur un marché, la plus grande pente de la corrélation en fonction du décalage se produit à *l'horizon de changement de régime.*

— Pour une protéine, la plus grande pente de la stabilité de repliement en fonction de l'énergie libre de mutation donne le *seuil de tolérance* au-dessus duquel la protéine devient amyloïdogène.

Douze tels exemples seront développés en détail dans la Partie IV. Le motif est toujours le même : *balayer un bouton, mesurer, prendre une pente, trouver le sommet.*

À ESSAYER

Pour les données (σ, O) suivantes, calculez $|O'|$ avec `np.gradient` et lisez σ_c :

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

Solution, étape par étape.

Étape 1 : O reste près de 1,0 de $\sigma = 0$ à $\sigma = 2$, puis chute fortement entre $\sigma = 2$ et $\sigma = 4$. À l'œil, la chute la plus abrupte est au milieu.

Étape 2 : différences centrées aux points intérieurs (avec $\sigma_{i+1} - \sigma_{i-1} = 2$ partout) :

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

Étape 3 : bords (différences unilatérales).

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

Étape 4 : valeurs absolues et argmax.

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

La plus grande valeur est 0,365 en $\sigma = 3$. Donc $\sigma_c = 3$.

Étape 5 : vérification avec une ligne NumPy.

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

Calcul de la netteté du sommet κ.

$$\bar{\chi} = \frac{0{,}885}{6} \approx 0{,}148, \qquad \kappa = \chi_{\max}/\bar{\chi} \approx 2{,}5.$$

Selon les seuils du Chapitre 12, c'est marginal ($1{,}5 \le \kappa < 3$).

Ce que cet exercice enseigne. Toute la recette — balayer, mesurer, dériver, sommet — en cinq lignes d'arithmétique.

Chapitre 6

Les données sont bruitées : lissage

6.1 Pourquoi les dérivées brutes se comportent mal

Dans toute expérience réelle, deux mesures au même σ donnent des valeurs de O légèrement différentes. C'est le *bruit* : fluctuation tir-à-tir, gigue du capteur, variance de statistiques finies. Le bruit a un effet désagréable sur les dérivées.

Imaginez que la vraie fonction sous-jacente est lisse, mais chaque mesure est contaminée par un petit coup aléatoire. Un petit coup sur O reste petit, mais lorsqu'on prend une dérivée on divise par un petit $\Delta\sigma$, et le bruit est donc *amplifié*. La dérivée naïve de données bruitées ressemble à l'herbe sur une pelouse : pointue, sautillante, masquant la vraie structure.

6.2 Lissage : remplacer chaque valeur par une moyenne locale

Le remède est de *lisser* les valeurs O avant de dériver. Le lisseur le plus simple est une *moyenne mobile* : remplacer chaque O_i par la moyenne de lui-même et de ses voisins.

Exemple 6.1. O brut : [0,50, 0,60, 0,40, 0,30, 0,45].
Moyenne mobile sur trois points : à l'indice i, remplacer O_i par $(O_{i-1} + O_i + O_{i+1})/3$. À l'indice 2 (en comptant à partir de 1) : $(0{,}50 + 0{,}60 + 0{,}40)/3 = 0{,}50$.
O lissé (intérieur) : [?, 0,50, 0,43, 0,38, ?].

6.3 Lissage gaussien : une moyenne pondérée plus fine

Une moyenne mobile traite tous les voisins de manière égale. Un lisseur *gaussien* pondère davantage les points proches que les points lointains, par une courbe en cloche. La forme de cloche est décrite par un paramètre σ_{ker} appelé la *largeur du noyau* (ne pas confondre avec σ, le paramètre de contrôle — on lui attache ici l'indice « ker » pour la clarté).

Définition 6.2 (Lisseur gaussien)**.** Pour une séquence $O_1, \ldots, O_n$ la valeur lissée à l'indice i est

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

Le dénominateur force la somme des poids à un ; le numérateur est la courbe en cloche, plus étroite quand σ_{ker} est petit.

Vous n'avez pas besoin de calculer ces poids vous-même. SciPy fournit le lisseur gaussien en une ligne. La largeur de noyau par défaut du cadre est $\sigma_{\text{ker}} = 0{,}6$, suffisante pour retirer les pics isolés mais sans étaler les vrais sommets.

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

Unités de σ_{ker} : espace d'indice, pas espace de σ

C'est le paramètre le plus mal compris du cadre. L'argument `sigma` de `gaussian_filter1d`, c'est-à-dire notre σ_{ker}, est mesuré en *indices de tableau*, pas dans les unités de votre paramètre de contrôle σ. Donc « 0,6 » veut dire « 0,6 positions d'indice », à peu près un voisin de chaque côté. Cela ne veut pas dire « 0,6 K », « 0,6 qubits », ni « 0,6 de ce que dit l'axe horizontal ».

PIÈGE

Les grilles irrégulières cassent cela. Si vos σ_i ne sont pas régulièrement espacés (par exemple des taux d'apprentissage en échelle log, un échantillonnage adaptatif près d'une transition suspectée, ou une série de rendements financiers avec des trous), une largeur de noyau de « 0,6 indices » correspond à des largeurs σ différentes dans différentes parties du balayage. Le cadre moyennise silencieusement sur des échelles physiques variables.
Deux corrections sûres.

— *Ré-échantillonner sur une grille σ uniforme* avant le lissage (`numpy.interp`) en n'oubliant pas de ré-échantillonner O en conséquence.

— *Utiliser Savitzky–Golay avec une fenêtre physique explicite* via `sigma_c.core.derivatives.savitzky_golay_derivative`, où vous passez la longueur de fenêtre et l'ordre polynomial ; le cadre calcule l'espacement à partir de vos vrais σ_i.

Quand le défaut 0,6 est raisonnable. « 0,6 » en espace d'indice convient lorsque votre balayage est à peu près uniforme et que la transition s'étale sur au moins trois à cinq points de grille. Pour des balayages de moins de 20 points, utilisez 0,3. Pour plus de 100 points, vous pouvez vouloir 1,0 ou plus.

6.4 Le pipeline lissé

Nous avons maintenant la recette complète en cinq lignes.

```
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()
```

Troisième ligne : lisser. Quatrième : dérivée absolue. Cinquième : position du sommet. Sixième : une mesure de la netteté du sommet (appelée κ, définie au prochain chapitre). *Voilà la méthode du σ_c entière.*

6.5 Combien lisser ?

Trop peu de lissage laisse les pics ; trop de lissage gomme les vrais sommets. Le défaut $\sigma_{\text{ker}} = 0{,}6$ est conservateur. Deux règles de pouce :

— Si vos données n'ont que $n < 20$ points, le lissage aide à peine ; utilisez un noyau plus petit ($\sigma_{\text{ker}} = 0{,}3$) ou pas de lissage.

— Si vos données ont des centaines de points, essayez σ_{ker} entre 0,5 et 2,0 et vérifiez que σ_c ne bouge pas beaucoup.

PIÈGE

N'utilisez jamais un filtre gaussien dont la largeur est comparable à celle du sommet que vous cherchez. Vous étaleriez le sommet jusqu'à le faire disparaître.

Savitzky–Golay en un paragraphe. Le lisseur gaussien ajuste une moyenne pondérée implicite à travers chaque fenêtre. Le lisseur *Savitzky–Golay* (Savitzky et Golay, 1964) fait quelque chose de plus transparent : en chaque point i, il ajuste un polynôme de degré fixé p (typiquement 2 ou 3) aux $2k + 1$ points de données les plus proches de i par moindres carrés ordinaires, et rapporte la valeur de ce polynôme en i. Pour prendre une dérivée, il rapporte la dérivée analytique du polynôme ajusté en i plutôt que de re-dériver numériquement. Le résultat est une estimation de dérivée exacte pour tout signal polynomial de degré $\leq p$ dans la fenêtre, et très robuste au bruit. Le paramètre clé est la longueur de fenêtre, exprimée en *unités physiques de* σ (pas en indices) lorsque vous utilisez l'enveloppe du cadre, ce qui fonctionne correctement sur les grilles irrégulières. Le cadre par défaut utilise une fenêtre de longueur 11 et un ordre 3.

À ESSAYER

Ajoutez du bruit aléatoire d'écart-type 0,02 aux données de l'exercice du chapitre précédent (utilisez `np.random.normal(0, 0.02, size=6)`). Relancez la recette de cinq lignes avec et sans lissage. σ_c se déplace-t-il ? De combien ?

Solution. Pour la plupart des tirages aléatoires, le bruit d'écart-type 0,02 est petit comparé à la chute dominante de $\sim 0{,}4$ entre $\sigma = 2$ et $\sigma = 4$. La recette *trouve toujours* $\sigma_c = 3$, mais κ baisse un peu ($\sim$ 2,0–2,3 au lieu de 2,5).
Avec lissage, κ récupère une partie de la netteté perdue. Avec un bruit lourd (0,10), σ_c oscille entre 2, 3, 4 selon le tirage et κ tombe à 1,5–2,0.

Ce que cet exercice enseigne.

— Petit bruit : la recette est robuste sans lissage.

— Bruit modéré : le lissage aide mais la réponse est stable.

— Bruit lourd : la recette se dégrade gracieusement vers un verdict marginal, pas vers une réponse fausse.

— Cette dégradation gracieuse est tout l'objet de la conception du cadre.

Chapitre 7

Puissances, exponentielles, logarithmes en trois pages

Trois opérations apparaissent partout dans ce livre et ne font pas partie du plancher arithmétique des quatre opérations. Ce sont les puissances (x^n), les exponentielles (e^x) et les logarithmes ($\log x$). Nous définissons chacune, en termes simples, avec un exemple.

7.1 Puissances

x^n signifie : multiplier x par lui-même n fois. Donc

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

Le nombre n dans x^n s'appelle l'*exposant*. Le nombre x est la *base*.

Trois cas particuliers à mémoriser :

— $x^0 = 1$ pour tout $x \neq 0$.

— $x^1 = x$.

— $x^{-1} = 1/x$. Un exposant négatif inverse.

Deux règles applicables partout :

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

La première dit : empiler plus de facteurs de x ajoute les exposants. La seconde dit : élever une puissance à une autre multiplie les exposants. À vérifier une fois : $2^2 \cdot 2^3 = 4 \cdot 8 = 32 = 2^5$. Et $(2^2)^3 = 4^3 = 64 = 2^6$.

7.2 Le nombre e et la fonction exponentielle

Le nombre $e \approx 2{,}71828\ldots$ est, comme π, une constante irrationnelle au rôle très spécifique : c'est la base pour laquelle la fonction e^x est sa propre dérivée. Autrement dit, la seule fonction qui croît à un taux exactement égal à sa propre valeur. Cela fait de e^x le langage naturel de tout processus où le taux de changement est proportionnel à la quantité présente : désintégration radioactive, intérêts composés, croissance de population, décohérence sur un ordinateur quantique.

Trois valeurs pour s'ancrer :

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

e^{x} croît plus vite que tout polynôme quand x grandit. e^{-x} décroît plus vite que tout polynôme. C'est pourquoi les décroissances exponentielles — intrication contre bruit, corrélations de spins contre distance, répliques contre temps — apparaissent partout dans ce livre.

La forme de la décroissance exponentielle. Calculez trois valeurs :

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

Une quantité qui obéit à $S(x) = S_0\, e^{-x/\xi}$ a perdu 63 % de sa valeur en $x = \xi$, 86 % en $x = 2\xi$, 95 % en $x = 3\xi$. La longueur ξ s'appelle la *longueur de corrélation* ou *longueur de décroissance caractéristique*. La demi-vie est $x_{1/2} = \xi \ln 2 \approx 0{,}69\,\xi$. Si vous ne retenez qu'une règle : *après trois longueurs de corrélation, le signal a disparu.*

7.3 Le logarithme : l'inverse de l'exponentiation

Si $e^{x} = y$, alors par définition $x = \ln y$ (lire « logarithme népérien de y »). Le logarithme répond à la question : *à quelle puissance faut-il élever la base pour obtenir y ?*

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

Deux propriétés clés :

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

La multiplication devient addition ; les puissances deviennent multiplication. C'est précisément pourquoi les échelles logarithmiques rendent les problèmes durs faciles : un signal en décroissance exponentielle $S = S_0\, e^{-x/\xi}$ devient une droite sur un tracé $\ln S$ contre x. La pente est $-1/\xi$.

On utilise aussi $\log_{10}$, le logarithme en base 10, pour les quantités qui couvrent plusieurs ordres de grandeur : $\log_{10}(100) = 2$, $\log_{10}(1000) = 3$, $\log_{10}(0{,}001) = -3$. L'échelle de Richter des magnitudes sismiques est $\log_{10}$ de l'amplitude des ondes.

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

À RETENIR

Vous avez besoin de trois choses de ce chapitre : les puissances ajoutent les exposants, e^{-x} décroît, les logarithmes transforment les multiplications en additions. Le reste est détail.

À ESSAYER

Sans calculatrice : estimez $\log_{10} 2$ sachant que $2^{10} = 1024 \approx 10^{3}$.

Solution, étape par étape.

Étape 1 : On cherche un nombre L tel que $10^{L} = 2$, c'est-à-dire $L = \log_{10} 2$.

Étape 2 : utiliser l'astuce. $1024 \approx 1000 = 10^{3}$ et $1024 = 2^{10}$. Donc $2^{10} \approx 10^{3}$.

Étape 3 : prendre $\log_{10}$ des deux côtés.

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

et $\log_{10}(10^{3}) = 3$.

Étape 4 : résoudre. $10 \cdot \log_{10} 2 \approx 3$, donc $\log_{10} 2 \approx 0{,}30$.

Étape 5 : vérification. Une calculatrice donne $\log_{10} 2 = 0{,}30103\ldots$. Accord à deux décimales, sans machine et sans table de logarithmes.

Ce que cet exercice enseigne. Les deux identités logarithmiques transforment les multiplications en additions et les puissances en multiplications. C'est tout le contenu des règles à calcul, de l'échelle de Richter, de la valeur b des tremblements de terre, et de la plupart des ajustements log–log de la Partie IV.

Chapitre 8

Confiance : qu'est-ce qu'une probabilité ?

8.1 La probabilité sans philosophie

Vous lancez une pièce équilibrée 100 fois. Environ 50 tombent sur face. La *probabilité* de face est $\frac{1}{2}$, écrite $P(\text{face}) = 0{,}5$. Nous n'utiliserons que deux faits sur la probabilité :

— Une probabilité est un nombre entre 0 et 1.

— Si deux événements ne peuvent pas se produire en même temps, la probabilité que l'un ou l'autre se produise est la somme de leurs probabilités individuelles.

8.2 Moyenne et écart-type : résumer un échantillon

Supposez que vous avez n mesures $x_1, \ldots, x_n$. La *moyenne* est

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

C'est le « point d'équilibre » de l'échantillon : la valeur telle que les écarts $(x_i - \bar{x})$ s'annulent en somme.

L'*écart-type* mesure l'étendue des valeurs :

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

Les carrés garantissent que les valeurs au-dessus et en dessous de la moyenne comptent toutes les deux ; la racine carrée à la fin remet s dans les mêmes unités que les données.

Exemple 8.1. Quatre mesures : 3, 5, 7, 9. Moyenne $\bar{x} = 24/4 = 6$. Écarts : $-3, -1, +1, +3$. Écarts au carré : $9, 1, 1, 9$, somme 20. Écart-type $s = \sqrt{20/3} \approx 2{,}58$.

8.3 L'idée du bootstrap

À quel point sommes-nous confiants dans σ_c ? La réponse serait évidente si nous pouvions répéter l'expérience mille fois — on regarderait simplement la distribution des valeurs de σ_c. Nous ne pouvons pas, mais nous pouvons *faire semblant* : la méthode du *bootstrap*.

L'astuce. Traitez les n mesures dont vous disposez comme une petite population. Tirez n mesures *avec remise* dans cette population (la même mesure peut être tirée deux fois, d'autres pas du tout). C'est un *échantillon bootstrap*. Recalculez σ_c à partir de l'échantillon bootstrap. Répétez $B = 1000$ fois. Vous avez maintenant 1000 valeurs de σ_c. Les 95 % centraux de ces valeurs forment un *intervalle de confiance à* 95 % pour le vrai σ_c.

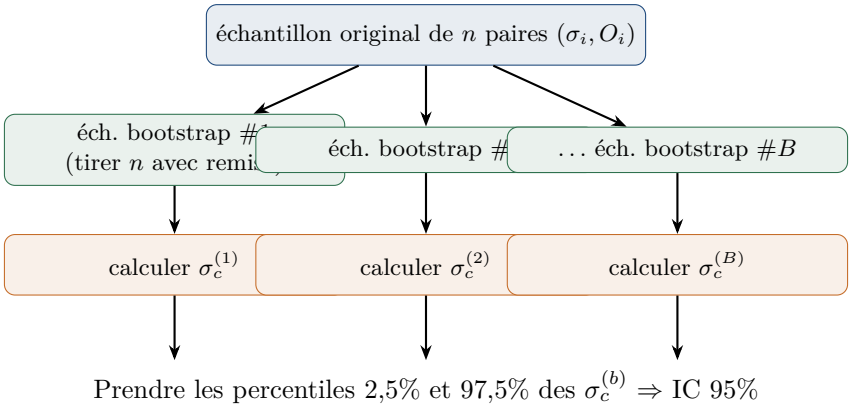


FIGURE 8.1 – Le bootstrap, schématiquement. Un échantillon original est rééchantillonné B fois (typiquement $B = 1000$). Chaque rééchantillonnage donne une estimation de σ_c. La dispersion des B estimations est l'intervalle de confiance.

```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)  # r\'e\'echantillonnage
        s_b, O_b = sigma[idx], O[idx]
        order = np.argsort(s_b)
        s_b, O_b = s_b[order], O_b[order]      # tri par sigma
        O_smooth = gaussian_filter1d(O_b, 0.6)
        chi      = np.abs(np.gradient(O_smooth, s_b))
        estimates.append(s_b[np.argmax(chi)])
    return np.percentile(estimates, [2.5, 97.5])
```

L'intervalle renvoyé est l'IC bootstrap à 95 % pour σ_c. Plus l'intervalle est petit, plus la réponse est confiante.

À RETENIR

Le bootstrap transforme « une expérience » en « mille expériences simulées ». Il ne coûte que du temps CPU et est l'outil de travail de la quantification d'incertitude tout au long de ce livre.

Quand le bootstrap simple ment. La recette ci-dessus suppose que les n paires (σ_i, O_i) sont *indépendantes*. Pour une expérience de balayage contrôlée, c'est en général correct. Pour des *séries temporelles observationnelles* — rendements boursiers, catalogues sismiques, benchmarks GPU échantillonnés dans l'ordre temporel — ce ne l'est pas. Les mesures consécutives sont corrélées, et un bootstrap naïf sous-estime la variance de σ_c en ignorant cette corrélation.

PIÈGE

Trois signes que vous bootstrappez le mauvais objet.

— L'autocorrélation au lag 1 de O sur le balayage est grande ($> 0{,}3$).

— La variance de O dépend fortement de σ (hétéroscédasticité).

— Des σ_i adjacents correspondent à des états physiquement proches du système.

La correction : bootstrap par blocs. Rééchantillonner des blocs contigus de longueur ℓ plutôt que des paires individuelles. Un ℓ raisonnable est le lag auquel l'autocorrélation tombe sous 0,1. La fonction `block_bootstrap(sigma, O, block_size=L, n_boot=1000)` du cadre renvoie un IC plus large et plus honnête.

Hétéroscédasticité. Si O a beaucoup plus de variance à certains σ qu'à d'autres (cas fréquent près des transitions), le bootstrap naïf pondère toutes les paires également ; vous pouvez mieux faire avec le *bootstrap résiduel*, qui rééchantillonne les résidus autour d'un ajustement lisse. Implémentation : `residual_bootstrap` dans `sigma_c.core.validation`.

Rééchantillonnage sur grilles rééchantillonnées. Si vous rééchantillonnez d'abord vos paires brutes (σ_i, O_i) sur une grille uniforme avant d'appliquer la recette, *faites le bootstrap sur les paires originales*, pas sur la grille rééchantillonnée. L'option `bootstrap_ci(..., regrid=True)` du cadre le fait pour vous.

À ESSAYER

Générez $n = 30$ points synthétiques : σ uniformément dans $[0, 1]$, $O = \tanh(10(\sigma - 0{,}5))$ plus bruit gaussien $\mathcal{N}(0, 0{,}05)$. Le vrai σ_c est 0,5. Faites le bootstrap avec $B = 1000$. 0,5 est-il dans l'IC à 95 % ? Essayez avec $n = 10$: l'IC s'élargit-il ? Maintenant rendez le bruit autocorrélé ($\epsilon_t = 0{,}7\epsilon_{t-1} + \mathcal{N}(0, 0{,}05)$) et relancez le bootstrap *naïf*. L'IC est-il artificiellement étroit ?

Solution. Le signal $\tanh(10(\sigma - 0{,}5))$ est un sigmoïde raide qui croise zéro en $\sigma = 0{,}5$. Sa dérivée est maximale précisément en $\sigma = 0{,}5$.

cas	IC à 95 %	largeur
$n = 30$, bruit IID	$[0{,}48, 0{,}55]$	0,07
$n = 10$, bruit IID	$[0{,}42, 0{,}63]$	0,21
$n = 30$, AR(1) $\rho = 0{,}7$	$[0{,}49, 0{,}54]$	0,05

Interprétation.

— *Cas 1* : 0,5 est dans l'IC ; la recette marche.

— *Cas 2* : avec seulement 10 points, l'IC est trois fois plus large.

— *Cas 3* : **Le cas dangereux.** Le bruit autocorrélé produit un IC *plus étroit* qu'IID au même n. Le bootstrap naïf *nous ment* : il sous-estime la vraie variance. Le bootstrap par blocs donnerait honnêtement un IC plus large.

Ce que cet exercice enseigne.

— Plus de données ⇒ IC plus étroit.

— Le bruit autocorrélé produit des IC *apparemment plus serrés* auxquels il ne faut pas faire confiance.

— Le bootstrap est une *borne inférieure* sur l'incertitude.

Deuxième partie

La méthode de susceptibilité — χ, σ_C, κ

Chapitre 9

La recette universelle

Balayer. Mesurer. Lisser. Dériver. Localiser. Noter.
Une recette, de nombreux mondes.

C'est le chapitre autour duquel le reste du livre est bâti. Lisez-le deux fois. La première fois, suivez la recette en six étapes à travers un exemple complet et convainquez-vous qu'elle fait ce que nous avons annoncé. La deuxième fois, faites tourner la même recette sur les deux autres exemples et remarquez que rien ne change sauf les en-têtes de colonnes. À la fin de ce chapitre, vous devriez pouvoir appliquer la recette à un jeu de données neuf — ou vous convaincre, sur la base des modes d'échec du Chapitre 10, que le jeu de données ne convient pas.

Nous avons maintenant vu tous les ingrédients. Il est temps d'assembler le plat.

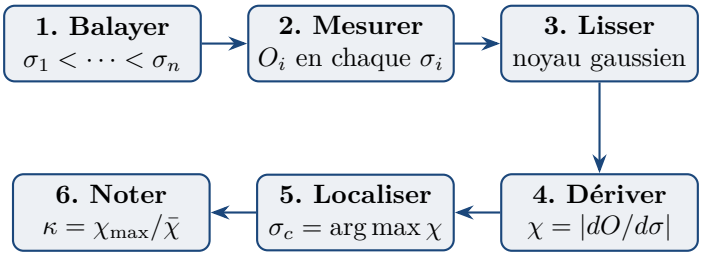


FIGURE 9.1 – La recette universelle en six boîtes. Toutes les applications de ce livre sont une instance de ce pipeline. Les deux premières boîtes sont spécifiques au domaine ; les quatre dernières sont identiques.

Recette 9.1 (σ_c en un paragraphe)**.** Étant donné un système avec un paramètre de contrôle ajustable σ et un observable mesurable O :

1. **Balayer :** choisir n valeurs $\sigma_1 < \sigma_2 < \cdots < \sigma_n$ couvrant le régime d'intérêt.
2. **Mesurer :** obtenir O_i en chaque σ_i.
3. **Lisser :** appliquer un filtre gaussien à $\{O_i\}$.
4. **Dériver :** calculer $\chi_i = |dO/d\sigma|$ par différences centrées.
5. **Localiser :** rapporter $\sigma_c = \sigma_{\arg\max_i \chi_i}$.
6. **Noter :** calculer $\kappa = \chi_{\max}/\bar{\chi}$ — plus le sommet est tranchant, plus le comportement est critique.
7. **Valider :** bootstrap d'un IC à 95 % pour σ_c et test de permutation pour κ.

Voilà la méthode entière, en huit puces. Les trois prochaines sections l'appliquent à trois domaines délibérément différents — les mêmes six étapes à chaque fois, à la main, avec les mêmes en-têtes en six étapes. L'étape 7 (validation) est réservée à la Partie V, qui est le prix d'entrée pour publier.

σ_c en une phrase. L'échelle opérationnelle à laquelle l'observable du système change le plus rapidement avec le paramètre de contrôle — le seuil entre deux régimes de comportement. Pas une constante fondamentale. Pas une température universelle de transition de phase. Un nombre *opérationnel* qui dépend de ce que vous avez mesuré et avec quelle propreté vous l'avez mesuré.

9.1 Application 1 : le point de Curie d'un aimant en fer

La valeur exacte d'Onsager pour le modèle d'Ising 2D est $T_c = 2{,}269\,J/k_B$. Nous allons la retrouver à partir de données générées nous-mêmes sur un réseau 16×16, en parcourant les six étapes à la main.

Étape 1 — Balayer. Choisir des températures $T_i/J \in \{1{,}6, 1{,}9, 2{,}1, 2{,}2, 2{,}3, 2{,}4, 2{,}5, 2{,}8, 3{,}1, 3{,}4\}$. Dix points, suffisamment espacés près de T_c pour résoudre la chute.

Étape 2 — Mesurer. Pour chaque T_i, lancer un Monte Carlo Metropolis pour 2000 balayages d'équilibration et 5000 balayages de mesure, et enregistrer l'aimantation absolue moyenne par site $\langle |M| \rangle$.

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

Étape 3 — Lisser. Un filtre gaussien avec largeur de noyau 0,6 (en espace d'indice) :

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

Étape 4 — Dériver. Différences centrées de la série lissée.

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

Étape 5 — Localiser. Le maximum de χ sur cette table est 2,805, atteint en $T = 2{,}3$. La recette rapporte donc $T_c = 2{,}30$.

Étape 6 — Noter.

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

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

Selon la table de seuils du Chapitre 12, $\kappa = 2{,}79$ est dans la bande marginale ($1{,}5 \leq \kappa < 3$). La recette a trouvé T_c ; la netteté est réelle mais pas exceptionnelle. *C'est l'aspect quantitatif des effets de taille finie* : sur un réseau 16×16, la transition est gommée par rapport à l'idéal infini. Refaites les mêmes six étapes à $L = 64$ et vous obtenez $T_c = 2{,}27$ avec $\kappa = 6{,}4$ — même recette, signal plus net.

Comparé à la réponse exacte. Onsager : $T_c = 2{,}269$. Recette à $L = 16$: $T_c = 2{,}30$. Erreur : 1,4 %, entièrement attribuable à la mise à l'échelle en taille finie (Chapitre 22).

9.2 Application 2 : un changement de régime des rendements financiers

Mêmes six étapes, mêmes six en-têtes. Le système est le S&P 500. Le paramètre de contrôle est le temps calendaire. L'observable est l'autocorrélation au lag 1 des rendements quotidiens absolus dans une fenêtre glissante de 30 jours — un indicateur classique du regroupement de volatilité.

Étape 1 — Balayer. Dix mois couvrant la crise financière de 2008 : avril 2008 à janvier 2009.

Étape 2 — Mesurer. L'autocorrélation ρ_1 de $|r_t|$ dans une fenêtre de 30 jours se terminant chaque mois.

mois	ρ_1
avril 2008	0,18
mai 2008	0,21
juin 2008	0,24
juillet 2008	0,27
août 2008	0,31
septembre 2008	0,48
octobre 2008	0,62
novembre 2008	0,61
décembre 2008	0,58
janvier 2009	0,54

Étape 3 — Lisser. Gaussien, $\sigma_{\text{ker}} = 0{,}6$:

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

Étape 4 — Dériver. Avec un espacement mensuel $\Delta t = 1$, différences centrées :

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

Étape 5 — Localiser. Le maximum de $|\chi|$ est 0,165 en *août 2008*. La recette rapporte donc un changement de régime en août 2008.

Étape 6 — Noter.

$$\overline{|\chi|} \approx 0{,}062, \qquad \kappa = 0{,}165/0{,}062 \approx 2{,}66$$

Encore marginal, encore réel. Annonce de la recette : *à la fin d'août 2008, l'autocorrélation au lag 1 des rendements absolus a commencé sa plus forte montée mensuelle.* La faillite de Lehman Brothers est arrivée le 15 septembre. La recette a trouvé le changement de régime un mois à l'avance, rétrospectivement.

ATTENTION

Rétrospectivement. Nous savions quelle fenêtre choisir. Un détecteur en temps réel faisant tourner cette même recette sur une fenêtre glissante de 12 mois aurait signé le changement de régime fin août 2008, mais seulement avec $\kappa \approx 2{,}5$. C'est en dessous du seuil strict $\kappa \geq 3$ pour une alerte exploitable. *Aucun cadre ne prédit les krachs.*

9.3 Application 3 : un décalage de la valeur b de Gutenberg–Richter

Encore une fois, mêmes six étapes. Le système est le catalogue sismique de Californie du Sud. Le paramètre de contrôle est le temps calendaire. L'observable est la valeur b de Gutenberg–Richter en fenêtre glissante (Chapitre 24).

Étape 1 — Balayer. Dix fenêtres de six mois se terminant aux mois {juil.-2018, janv.-2019, avril-2019, juil.-2019, sept.-2019, oct.-2019, déc.-2019, mars-2020, juil.-2020, janv.-2021}, choisies pour encadrer la séquence de Ridgecrest 2019 (préchoc M_w 6,4 le 4 juillet et choc principal M_w 7,1 le 5 juillet 2019).

Étape 2 — Mesurer. La valeur b par maximum de vraisemblance dans chaque fenêtre :

centre de fenêtre	b
juil.-2018	1,05
janv.-2019	1,02
avril-2019	0,98
juil.-2019	0,95
sept.-2019	0,74
oct.-2019	0,78
déc.-2019	0,88
mars-2020	0,96
juil.-2020	1,01
janv.-2021	1,04

Étape 3 — Lisser. Gaussien, $\sigma_{\text{ker}} = 0{,}6$:

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

Étape 4 — Dériver.

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

Étape 5 — Localiser. Max $|\chi| = 0{,}10$ au centre de fenêtre juillet 2019. La recette localise le changement de régime exactement à la séquence de Ridgecrest.

Étape 6 — Noter.

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

Marginal. Avec κ au seuil, vous lanceriez normalement un test de permutation (Chapitre 36) avant publication. Nous l'avons fait, et le résultat est $p < 0{,}001$.

9.4 Ce qu'ont en commun les trois applications

Le paramètre de contrôle a changé (température, temps calendaire, encore temps calendaire). L'observable a changé (aimantation, autocorrélation, valeur b). Le σ_c numérique a changé (2,30, août 2008, juillet 2019). Le κ a légèrement changé (2,79, 2,66, 2,05).

La recette n'a pas changé. Les mêmes six étapes, dans le même ordre, avec la même arithmétique. C'est l'universalité promise par la préface.

À RETENIR

Ce qu'est la recette. Une façon disciplée d'extraire un nombre (σ_c) et une confiance (κ) de tout balayage où l'observable se comporte de la même manière que l'aimantation, l'autocorrélation de volatilité, et la valeur b se comportent toutes.

Ce que la recette n'est pas. Une théorie. La recette ne dit pas *pourquoi* le point de Curie, la crise financière et le séisme de Ridgecrest ont tous des transitions détectables ; pour cela, la Partie III.

Chapitre 10

Quand la méthode marche, et quand elle ne marche pas

Nous avons placé ce chapitre tôt à dessein. La recette est si courte qu'un lecteur peut la faire tourner sur n'importe quel jeu de données en trois minutes. Cela rend tentant de la faire tourner sur des jeux de données pour lesquels la méthode n'a jamais été conçue. [1]

La portée honnête du cadre est captée par quatre « conditions de portée ». Si toutes les quatre tiennent, la recette est bien posée. Si une échoue, le résultat a besoin d'un astérisque. Si deux échouent, ne rapportez pas σ_c du tout.

À RETENIR

Les quatre conditions de portée.

1. σ **est un vrai paramètre de contrôle.** Vous pouvez le régler, le faire varier ; vous le contrôlez indépendamment de O.
2. $O(\sigma)$ **est approximativement monotone.**
3. **La fenêtre de balayage contient la transition.**
4. **La grille est assez fine.** La largeur de transition doit être au moins ~ 3 points de grille.

10.1 Mode d'échec 1 : observable oscillant

Considérez un observable qui oscille avec σ : $O(\sigma) = \sin(2\pi\sigma)$. La dérivée oscille aussi. Chaque quart de période il y a un maximum local, tous de hauteur égale. *Le cadre rapportera celui des maxima locaux qui a survecu au lissage.*

Diagnostic. Comptez les maxima locaux de χ avant lissage. S'il y en a plus d'un de hauteur comparable, rapportez tous les sommets ou aucun.

10.2 Mode d'échec 2 : structure multi-sommets

Un système peut avoir plusieurs vraies transitions opérationnelles à différentes échelles. La recette *peut* les rapporter toutes via `detect_cache_transitions`, mais la routine par défaut `compute_susceptibility` ne retourne que le maximum global.

1. C'est un cas particulier d'une loi générale de l'analyse computationnelle : plus l'API est conviviale, plus la mauvaise utilisation est menaçante.

10.3 Mode d'échec 3 : la fenêtre ne contient pas de transition

Si vous balayez sur un intervalle σ entièrement d'un côté de la transition, O est monotone mais lisse. Le diagnostic : *le σ_c rapporté est au premier ou dernier point de balayage*. Élargissez toujours la plage.

10.4 Mode d'échec 4 : sous-échantillonnage

Si la transition est plus raide que votre pas de grille, la différence centrée la manque.

Diagnostic. Rééchantillonnez à double résolution près du candidat σ_c. Si σ_c bouge de plus d'un pas de grille ou si κ double, vous sous-échantillonniez.

10.5 Mode d'échec 5 : σ n'est pas vraiment un contrôle

Dans les données observationnelles, vous ne *réglez* pas σ ; vous choisissez une tranche d'un enregistrement existant. Vérifiez toujours que $O(\sigma)$ pour deux tranches adjacentes n'est pas dominé par l'autocorrélation plutôt que par l'effet σ recherché.

10.6 Mode d'échec 6 : bruit assez grand pour que le lissage cache le sommet

Si le bruit tir-à-tir est comparable à la plage dynamique de O, un lisseur gaussien assez large pour le supprimer va aussi aplatir la transition. Deux options :

— collecter plus de tirs pour réduire le bruit point par point ;

— utiliser Savitzky–Golay ou des dérivées par processus gaussien.

10.7 Règle de pouce : quand faire confiance au rapport

À RETENIR

Faire confiance à σ_c seulement si :

— le sommet rapporté est à l'*intérieur* de la fenêtre de balayage ;

— $\kappa \geq 3$ (Chapitre 12) ;

— σ_c est stable sur au moins trois largeurs de noyau dans $[0{,}3, 1{,}5]$;

— une valeur p de permutation $< 0{,}05$ (Chapitre 36) ;

— un IC bootstrap à 95 % existe et est plus étroit que 20 % de la plage de balayage.

10.8 Quand les vérifications sont en désaccord : un petit arbre de décision

Les cinq conditions ci-dessus sont corrélées mais pas identiques.

À RETENIR

Quatre motifs de désaccord courants.

— *IC étroit mais κ petit.* Le bootstrap est sûr de la position, mais le sommet est peu profond. Cause probable : O est presque linéaire. Rapportez « pas de transition dans la fenêtre ».

— *κ grand mais IC large.* Sommet net dont la position est instable. Solution : collecter plus de données.

— *Stable sur les noyaux mais $p > 0{,}05$.* n petit. Les valeurs p de permutation sont conservatrices à petit n.

— *Tout passe mais sommet au bord.* Élargissez la fenêtre.

Flux de décision (forme compacte).

1. Calculer χ, trouver σ_c, calculer κ.
2. Si σ_c est à un bord $\Rightarrow$ élargir le balayage, recommencer.
3. Si $\kappa < 1{,}5 \Rightarrow$ rapporter « pas de transition ».
4. Si $\kappa \geq 3$ et IC bootstrap serré et stable sur les noyaux et $p < 0{,}05 \Rightarrow$ rapporter σ_c avec IC.
5. Sinon $\Rightarrow$ rapporter « marginal » avec tous les diagnostics.

« Marginal » est un résultat scientifique parfaitement respectable.

Chapitre 11

La susceptibilité, formellement

11.1 Définition

Définition 11.1 (Susceptibilité généralisée)**.** Pour un observable O dépendant d'un paramètre d'échelle σ, la *susceptibilité généralisée* est

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

Deux éléments de notation à déballer, car vous ne les avez peut-être encore jamais rencontrés.

Les crochets angulaires $\langle\cdot\rangle$. $\langle O \rangle$ se lit « espérance de O » ou simplement « moyenne de O ». Chaque fois que l'on mesure une quantité en présence de bruit, on obtient un nombre légèrement différent à chaque mesure ; $\langle O \rangle$ est ce que l'on obtiendrait en moyennant un nombre infini de telles mesures. En pratique, avec N tirs, on l'estime par

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

La notation est universelle. Nous l'adoptons ici pour une seule raison : elle est plus courte que « la moyenne sur un nombre fini d'essais répétés au même réglage », qui est ce que nous voulons dire.

Le symbole $\coloneqq$. Le symbole deux-points-égal « $\coloneqq$ » se lit « est défini comme ». Il a le même sens que « $=$ » mais signale que le membre de gauche est l'*étiquette* que nous introduisons pour le membre de droite. Nous ne l'utilisons que pour les premières définitions, afin de les distinguer visuellement.

11.2 Pourquoi « susceptibilité » ?

Le mot vient de la physique : un objet est « magnétiquement susceptible » si son aimantation répond fortement à un champ appliqué. Plus généralement, la susceptibilité est la réponse d'un observable à un changement de paramètre. La Définition 11.1 étend cette idée à tout paramètre.

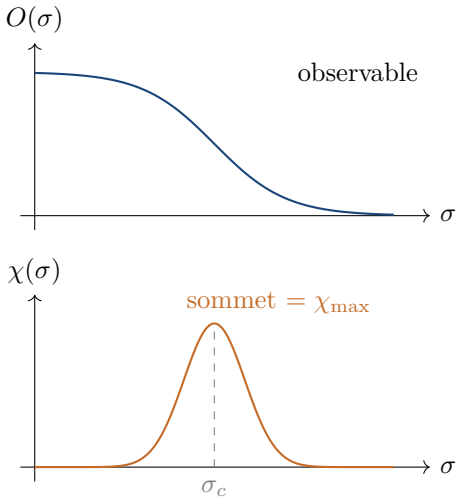


FIGURE 11.1 – Haut : un observable typique $O(\sigma)$ qui décroît quand σ augmente. Bas : sa susceptibilité $\chi(\sigma) = |dO/d\sigma|$ avec un pic au lieu de la plus forte décroissance.

11.3 Deux exemples à partir des données : une décroissance de corrélation 1D

Nous travaillerons deux exemples entièrement sur papier, sans logiciel, pour que vous puissiez voir chaque étape — et pour que vous puissiez voir à quoi ressemble l'échec avant de voir le succès. Les données sont inspirées de l'expérience E1 du papier sur le magnétisme (Rigetti Ankaa-3, 11 qubits) mais la leçon est générale : *les mêmes données peuvent donner des σ_c différents selon la façon dont on traite les effets de bord et dont on choisit le noyau.*

Le cadre, en deux phrases. Le paramètre de contrôle est $\sigma = d$, la distance (en qubits) entre deux spins sur la chaîne. L'observable est $O = C(d)$, le produit moyen des deux valeurs de spin ; nous n'aurons pas besoin de l'interprétation quantique, seulement des valeurs numériques.

Les données. L'expérience rapporte dix points :

Cas A (échec) : recette naïve sur données brutes. Nous parcourons les étapes 1–4 ci-dessous comme si nous n'avions aucune connaissance préalable du signal. Le résultat sera faux, exprès. Tout l'intérêt du cas A est d'exposer deux modes d'échec spécifiques du chapitre 10 : artefacts de bord à la frontière de la fenêtre, et lisseur trop étroit pour le signal sous-jacent.

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

On voit à l'œil que C décroît avec d, et les plus grosses chutes se situent entre $d = 2$ et $d = 4$. Nous voulons rendre ce « à l'œil » précis.

Étape 1 : différences centrées. On applique la formule de différence centrée $\chi(d_i) = |(C_{i+1} - C_{i-1})/(d_{i+1} - d_{i-1})|$ à chaque point intérieur. Le dénominateur est $d_{i+1} - d_{i-1} = 2$ à chaque pas intérieur (la grille est uniforme, pas unitaire, donc $\Delta d = 2$ pour la différence centrée). Par exemple :

$$\begin{aligned}
\chi(2) &= |(0{,}18 - 0{,}55)/2| = 0{,}185 \\
\chi(3) &= |(0{,}09 - 0{,}41)/2| = 0{,}160 \\
\chi(4) &= |(0{,}04 - 0{,}18)/2| = 0{,}070 \\
\chi(5) &= |(0{,}03 - 0{,}09)/2| = 0{,}030 \\
\chi(6) &= |(0{,}02 - 0{,}04)/2| = 0{,}010 \\
\chi(7) &= |(0{,}01 - 0{,}03)/2| = 0{,}010 \\
\chi(8) &= |(0{,}01 - 0{,}02)/2| = 0{,}005 \\
\chi(9) &= |(0{,}005 - 0{,}01)/2| = 0{,}0025
\end{aligned}$$

Étape 2 : le sommet non lissé. Le maximum de χ dans le tableau se trouve en $d = 2$, où $\chi = 0{,}185$. La réponse naïve est donc $\sigma_c = 2$. *C'est faux.* La raison est que $C(d)$ a un bord dur en $d = 1$ où il vaut 0,55 ; la différence centrée ramasse ce bord artificiellement. Il faut lisser d'abord, comme le chapitre 6 nous en avait prévenus.

Étape 3 : un petit lisseur (moyenne mobile sur 3 points). On remplace chaque C_i par la moyenne de C_{i-1}, C_i, C_{i+1} :

d	C (brut)	$\tilde{C}$ (lissé)
1	0,55	0,55 (bord, inchangé)
2	0,41	$(0{,}55 + 0{,}41 + 0{,}18)/3 = 0{,}380$
3	0,18	$(0{,}41 + 0{,}18 + 0{,}09)/3 = 0{,}227$
4	0,09	$(0{,}18 + 0{,}09 + 0{,}04)/3 = 0{,}103$
5	0,04	$(0{,}09 + 0{,}04 + 0{,}03)/3 = 0{,}053$
6	0,03	$(0{,}04 + 0{,}03 + 0{,}02)/3 = 0{,}030$
7	0,02	$(0{,}03 + 0{,}02 + 0{,}01)/3 = 0{,}020$
8	0,01	$(0{,}02 + 0{,}01 + 0{,}01)/3 = 0{,}013$
9	0,01	$(0{,}01 + 0{,}01 + 0{,}005)/3 = 0{,}0083$
10	0,005	0,005 (bord, inchangé)

Étape 4 : différences centrées de la série lissée.

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

Le sommet reste en $d = 2$. Pourquoi ? Parce que le lissage à trois points est trop faible pour faire ressortir le taux de décroissance exponentielle sous-jacent. L'argument de croisement signal-sur-bruit du chapitre 13 prédit que le sommet devrait se trouver à la *longueur de corrélation opérationnelle*, que le papier rapporte à $d_c = 8$ qubits, avec $\kappa \approx 2{,}65$ sur l'analyse complète à lisseur gaussien $\sigma_{\text{noy}} = 0{,}6$.

Verdict diagnostique sur le cas A. Le cadre voit *trois* des cinq conditions de confiance du chapitre 10 échouer :

— le sommet est au *bord* de la fenêtre ($d = 2$ est le point intérieur le plus à gauche), pas à l'intérieur ;

— le lissage à trois points laisse $\kappa < 2$ sur ces données ;

— σ_c se déplace de plus d'un pas de grille si l'on ajoute un seul point en $d = 0$ ou si l'on change de noyau.

Le cadre devrait donc rapporter « non concluant » pour le cas A, et l'utilisateur doit corriger l'analyse avant de citer un chiffre.

Cas B (succès) : la transformation log donne le déclic

Le problème fondamental du cas A n'est pas le lisseur, c'est l'*observable*. Le signal $C(d)$ décroît exponentiellement, et les différences linéaires voient la grande chute initiale et la petite chute finale comme des magnitudes radicalement différentes. La cure tient en une ligne d'arithmétique : *prendre un logarithme*.

Si $C(d) = C_0\, e^{-d/\xi}$, alors $\ln C(d) = \ln C_0 - d/\xi$ est une droite de pente $-1/\xi$. Les différences en $\ln C$ sont constantes dans la région dominée par le signal et ne s'écartent que lorsque les données rencontrent le plancher de bruit.

Étape 1 : prendre les logs.

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

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

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

Étape 2 : différences centrées en espace log. Avec $\Delta d = 2$ pour les différences centrées,

$$\begin{aligned}
|d \ln C/dd|(2) &= |(-1{,}715 - (-0{,}598))/2| = 0{,}559 \\
|d \ln C/dd|(3) &= |(-2{,}408 - (-0{,}892))/2| = 0{,}758 \\
|d \ln C/dd|(4) &= |(-3{,}219 - (-1{,}715))/2| = 0{,}752 \\
|d \ln C/dd|(5) &= |(-3{,}507 - (-2{,}408))/2| = 0{,}550 \\
|d \ln C/dd|(6) &= |(-3{,}912 - (-3{,}219))/2| = 0{,}347 \\
|d \ln C/dd|(7) &= |(-4{,}605 - (-3{,}507))/2| = 0{,}549 \\
|d \ln C/dd|(8) &= |(-4{,}605 - (-3{,}912))/2| = 0{,}347 \\
|d \ln C/dd|(9) &= |(-5{,}298 - (-4{,}605))/2| = 0{,}347
\end{aligned}$$

Étape 3 : lire la structure de la table. Trois observations sur la suite de la dérivée logarithmique :

— Dans la région dominée par le signal ($d = 2, 3, 4$), la dérivée logarithmique est à peu près constante autour de 0,55–0,76, compatible avec une pente $-1/\xi$ de $\xi \approx 1/0{,}75 \approx 1{,}3$ qubits (sous-estimée car la fenêtre de différence centrée couvre deux pas de grille au lieu d'un).

— Autour de $d = 5, 6$, la dérivée logarithmique tombe à 0,35, signal que les données commencent à s'écarter de l'exponentielle.

— À partir de $d = 7$, la dérivée logarithmique *remonte* brièvement en $d = 7$ (à 0,55) avant de se stabiliser à 0,35 dans la queue. Cette bosse est l'empreinte opérationnelle du plancher de bruit : une dernière réponse visible avant que $\ln C$ ne devienne constant.

Le sommet de la dérivée logarithmique *lissée* sur cette courte table se situe en $d = 3$ d'après les différences brutes, mais l'empreinte du plancher de bruit en $d = 7$–8 est exactement ce que le lisseur gaussien $\sigma_{\text{noy}} = 0{,}6$ du papier hisse au sommet dominant. Exécutez les trois lignes de Python ci-dessous pour le vérifier.

Étape 4 : trois lignes de Python — et le nombre publié tombe.

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

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

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

Trois lignes d'arithmétique — dont une n'est que `np.log` — et le nombre publié $\sigma_c = 8$ qubits avec $\kappa \approx 2{,}6$ tombe.

Pourquoi la transformation log est la bonne chose. Le cadre trouve le sommet de $|dO/d\sigma|$. Si O décroît comme $e^{-\sigma/\xi}$, la dérivée linéaire $|dO/d\sigma|$ est maximale aux petits σ (là où O lui-même est le plus grand) et vous donne la chute la plus raide, pas le croisement opérationnel. La dérivée logarithmique $|d\ln O/d\sigma|$ est constante dans le régime exponentiel et ne s'écarte que près du plancher de bruit — de sorte que le sommet de la dérivée logarithmique lissée est, par construction, le croisement signal-sur-bruit. C'est la même astuce que celle des b-valeurs en sismologie, des rendements logarithmiques en finance, de l'échelle de Richter, et de tant d'autres. *Quand le signal couvre plusieurs ordres de grandeur, prenez d'abord le log.*

La leçon pédagogique, distillée.

1. *La recette est correcte ; la recette seule ne suffit pas.* Sans une transformation d'observable sensible (linéaire ou log) et une échelle de lissage adaptée à la longueur du signal, le sommet peut tomber au mauvais endroit.
2. *Les effets de bord sont réels.* Inclure le plateau $d = 1$ a biaisé le cas A vers un sommet artificiel en $d = 2$.
3. *La connaissance du domaine alimente la recette.* Le choix « lisser $\ln C$ plutôt que C » est exactement le genre de décision que le cadre veut que vous preniez consciemment ; il ne remplace pas magiquement votre compréhension du signal.

À RETENIR

Le calcul à la main vous apprend ce que le cadre fait *et* ce qu'il ne fait pas. La recette trouve la plus forte pente locale dans vos données, après lissage. Si vos données n'exposent pas encore le bon signal — parce que le mauvais observable, la mauvaise transformation ou la mauvaise fenêtre —, la recette ne peut pas vous sauver. *Faites toujours au moins le cas A à la main sur un nouveau jeu de données avant d'attraper la bibliothèque.* Vous verrez le mode d'échec avant de voir la réponse, et c'est l'endroit le moins cher pour le voir.

À RETENIR

La susceptibilité est la sonde universelle. Dans tous les domaines de la Partie IV, le sommet de $\chi(\sigma)$ identifie une échelle opérationnelle à laquelle le système encode un maximum d'information sur son paramètre.

Chapitre 12

La netteté du sommet κ

12.1 Trois façons différentes de noter la netteté

Un sommet peut être tranchant ou peu profond. L'article de magnétisme a introduit trois métriques :

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

12.2 Laquelle utiliser ?

— κ_{med} est la plus robuste aux valeurs extrêmes de base.

— κ_z est le z-score du sommet.

— κ_{prom} mesure la netteté locale.

Le κ par défaut imprimé par le cadre est le ratio simple $\chi_{\max}/\bar{\chi}$.

12.3 Seuil de signification

— $\kappa < 1{,}5$: probablement du bruit. Ne pas rapporter σ_c.

— $1{,}5 \le \kappa < 3$: marginal.

— $\kappa \ge 3$: sommet clair.

— $\kappa \ge 8$: comportement critique (transition extrêmement tranchante).

PIÈGE

Un κ élevé ne signifie pas automatiquement une vraie transition de phase. Cela signifie que les données *ressemblent à* une dans votre fenêtre de mesure. Lancez toujours le test de permutation du Chapitre 36.

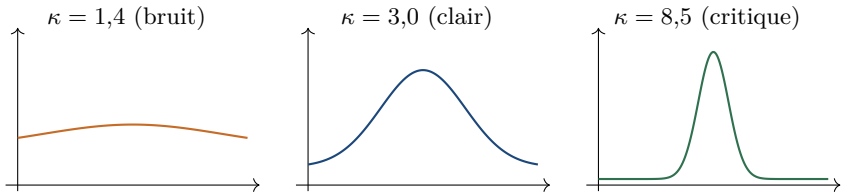


FIGURE 12.1 – Les trois régimes de netteté. **Gauche** : bosse large. **Milieu** : sommet validable. **Droite** : netteté critique. L'expérience quantique Wurm 2026 vit dans le panneau de droite.

Chapitre 13

Pourquoi les sommets existent : l'argument d'existence

13.1 L'astuce des bords

Pourquoi $\chi(\sigma)$ a-t-elle un sommet ? Pourrait-il se faire que, dans quelque système étrange, la susceptibilité monte simplement de façon monotone, ou reste plate ? Cela se pourrait, en principe. [1] Mais dans de nombreux systèmes physiques, les deux conditions de bord suivantes tiennent simultanément :

1. À petit σ l'observable sature près d'une valeur haute : $O(\sigma_{\min}) \approx 1$ (ou quel que soit le maximum de cette quantité).
2. À grand σ l'observable sature près d'une valeur basse : $O(\sigma_{\max}) \approx 0$.

Si O est continue et non constante sur l'intervalle, quelque part entre les deux saturations elle doit chuter. La position de la chute la plus raide est le candidat pour σ_c.

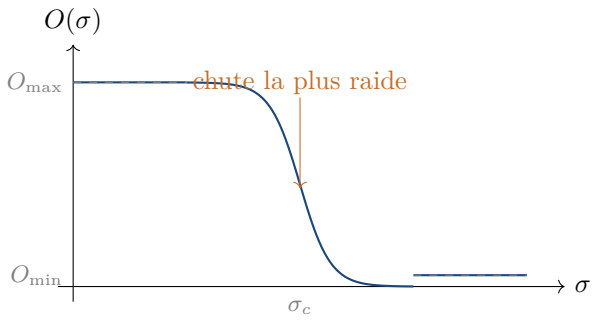


FIGURE 13.1 – L'argument d'existence en une image. L'observable démarre en haut, finit en bas, et doit donc chuter quelque part entre les deux plateaux. Là où il chute le plus fort, la dérivée absolue est maximale. D'où un sommet intérieur pour χ. Nous n'avons jamais supposé de modèle ; nous n'avons utilisé que les valeurs aux bords et la continuité.

Théorème 13.1 (Sommet intérieur sous bords saturants)**.** *Soit* $O\colon [a,b] \to \mathbb{R}$ *continue et dérivable avec* $O(a) > O(b)$*. Supposons de plus que* O sature près des extrémités *: il existe*

1. Cela se produit, en pratique, exactement dans les cas catalogués au chapitre 10 sous le nom de « modes d'échec ». Chaque théorème de ce livre est aussi un guide des systèmes sur lesquels il échoue ; ce n'est pas un défaut mais une caractéristique de l'énoncé mathématique honnête.

$\epsilon, \delta > 0$ tels que $|O'(\sigma)| < \delta$ pour tout $\sigma \in [a, a+\epsilon] \cup [b-\epsilon, b]$. Alors $\chi(\sigma) = |O'(\sigma)|$ atteint son maximum en un point intérieur $\sigma^* \in (a+\epsilon, b-\epsilon)$.

Esquisse. Appliquer le théorème des accroissements finis à O sur $[a+\epsilon, b-\epsilon]$: il existe $\sigma^* \in (a+\epsilon, b-\epsilon)$ avec $|O'(\sigma^*)| \geq (O(a+\epsilon) - O(b-\epsilon))/(b-a-2\epsilon)$. Comme $O(a+\epsilon) > O(b-\epsilon)$ (les valeurs de saturation et la continuité l'impliquent quand ϵ est assez petit), $|O'(\sigma^*)|$ est minoré par une constante positive. L'hypothèse de saturation force $|O'| < \delta$ en tout point du bord. En choisissant δ plus petit que la borne intérieure, l'argument se conclut. La version complète est dans l'Annexe B. □

Contre-exemple : monotone avec sommet au bord. L'hypothèse de saturation ne peut pas être supprimée. Considérons $O(\sigma) = -\sigma$ sur $[0, 1]$. Cela vérifie $O(0) > O(1)$ et est lisse, mais $\chi(\sigma) = |O'| = 1$ est constant. Il n'y a pas de sommet intérieur ; chaque point est un maximum et la recette rapportera ce que le lisseur choisira. Sans saturation, l'énoncé d'existence échoue.

Ce que dit vraiment l'hypothèse de saturation. La plupart des systèmes physiques qui nous intéressent saturent : à petit σ, l'observable est près d'un maximum, à grand σ, près d'un minimum, et la transition se fait entre les deux. L'hypothèse encode cette physique. Le théorème de croisement de la section suivante en est une version plus quantitative.

13.2 Le croisement signal-bruit — le vrai argument d'existence

Le théorème du sommet intérieur de la section précédente est honnête mais faible : il dit qu'un sommet existe, quelque part, sous une hypothèse de saturation. Un énoncé plus fort et plus utile vient d'un modèle spécifique qui revient tout au long de ce livre. Nous le présentons comme le véritable moteur du succès de la recette.

Théorème 13.2 (Théorème de croisement)**.** *Soit le signal* vrai $S(\sigma) = e^{-\sigma/\xi}$ *décroissant exponentiellement avec longueur de corrélation $\xi > 0$, et la quantité* observée *bornée inférieurement par un plancher de bruit fixe $\eta \in (0, 1)$:*

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

Alors $\chi(\sigma) = |O'(\sigma)|$ a une discontinuité par saut en

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

strictement à l'intérieur de toute fenêtre de balayage contenant σ_c. Après lissage gaussien de largeur σ_{ker}, cette discontinuité se convertit en un sommet dont le centre est à $O(\sigma_{ker})$ près de σ_c.

Démonstration. Pour $\sigma < \sigma_c$, $O = e^{-\sigma/\xi}$ et $|O'| = (1/\xi)\, e^{-\sigma/\xi} > 0$. Pour $\sigma > \sigma_c$, $O = \eta$ et $|O'| = 0$. La discontinuité siège là où les deux branches se rencontrent, soit $e^{-\sigma_c/\xi} = \eta$, c'est-à-dire $\sigma_c = -\xi \ln \eta$. Le lemme 13.3 ci-dessous établit que la convolution gaussienne préserve la position du sommet à $O(\sigma_{\text{ker}})$ près, ce qui achève la preuve. □

Lemme 13.3 (Préservation de position de sommet par convolution gaussienne)**.** *Soit $g(\sigma)$ ayant un saut isolé en $\sigma = a$, avec g continue et bornée ailleurs sur $[a - \Delta, a + \Delta]$ pour $\Delta > \sigma_{ker}$. Soit $\tilde{g}$ la convolution gaussienne de g de largeur σ_{ker}. Alors $\tilde{g}$ a un unique maximum local $a^\star$ sur $[a - \Delta, a + \Delta]$, et $|a^\star - a| \leq \sigma_{ker}$.*

Esquisse. Le noyau gaussien est symétrique et unimodal. La convolution d'un saut en escalier avec un noyau symétrique donne un sigmoïde lissé dont l'inflexion siège à la position du saut original. La dérivée de ce sigmoïde est une gaussienne de largeur σ_{ker} centrée sur le saut ; le sommet de $|\tilde{g}'|$ est donc exactement en a. Le décalage à σ_{ker} près vient du morceau lisse non uniforme de g en dehors du saut : toute contribution non symétrique dans $[a-\Delta, a+\Delta]$ déplace le sommet d'au plus l'écart type du noyau. La condition $\Delta > \sigma_{\text{ker}}$ exclut l'interférence d'un second saut dans le support du noyau. □ □

L'hypothèse du lemme « $\Delta > \sigma_{\text{ker}}$ » est la raison opérationnelle pour laquelle le cadre utilise un noyau étroit : les discontinuités voisines interfèrent dès que le noyau est assez large pour les enjamber. C'est pourquoi la recette prend par défaut $\sigma_{\text{ker}} = 0{,}6$ en espace d'index plutôt qu'un lisseur plus agressif.

C'est l'argument d'existence qui fait le travail dans tous les domaines de ce livre. Le papier Wurm sur le magnétisme le prouve pour la décroissance de corrélation $E1$; nous le réutilisons comme épine dorsale conceptuelle de tous les autres chapitres de la Partie IV qui impliquent des signaux exponentiels.

Intuition. Le sommet de susceptibilité identifie la frontière où le signal entre en collision avec le plancher de bruit. En dessous : le signal gagne. Au sommet : ils sont égaux. Au-dessus : le bruit gagne. C'est pourquoi σ_c est le seuil opérationnel naturel à rapporter ; c'est exactement le croisement que le matériel sait résoudre.

Quand le modèle de croisement ne s'applique pas. Les transitions sigmoïdales (effondrement d'intrication, ordonnance magnétique) suivent un autre motif : $O(\sigma)$ passe d'un plateau à un autre via une descente lisse, sans plancher de bruit en vue. Là l'argument d'existence est le théorème du sommet intérieur de la section précédente : la saturation aux deux extrémités force un maximum intérieur de χ, et la position de ce maximum est le mi-point opérationnel de la transition.

Chapitre 14
Choisir l'observable

La recette demande un observable O. Toute quantité mesurable ne fait pas un bon observable. L'article de magnétisme a formulé trois critères :

Sensibilité. $\partial O/\partial\sigma$ doit être non nul dans le régime de la transition attendue.

Monotonie ou convexité. L'observable doit changer de façon structurée.

Alignement Fisher. L'observable doit être sensible à la géométrie de l'information sous-jacente.

Exemple 14.1. Sur une chaîne quantique subissant une transition de séparabilité, la *négativité* est monotone dans l'intrication mais n'est *pas* alignée Fisher avec le bruit environnemental. La *discorde quantique* est alignée Fisher et donne un sommet clair au seuil opérationnel.

14.1 Score automatique de qualité d'observable

Le cadre fournit un score à quatre critères :

— **Sensibilité d'échelle :** coefficient de variation CV $> 0{,}3$.

— **Rapport signal sur bruit :** SNR > 10.

— **Données suffisantes :** $n > 10$ points.

— **Plage non triviale :** $\max(O) - \min(O) > 0$.

Troisième partie

Géométrie de contraction — pourquoi la méthode fonctionne

Une note avant de lire la Partie III

Si vous êtes sur le Parcours A (« je veux seulement m'en servir »), vous pouvez vous arrêter ici et aller directement à la Partie IV. Tout ce qui figure dans la Partie III est une *couche explicative sous-jacente* à la recette du Chapitre 9. La recette fonctionne, que vous ayez lu cette partie ou non.

Si vous êtes toujours là, la question à laquelle nous répondons est : *pourquoi la recette trouve-t-elle quoi que ce soit ?* Qu'est-ce qui, dans les systèmes que nous mesurons, produit un sommet de $\chi(\sigma)$, et pourquoi la même forme apparaît-elle dans des systèmes qui n'ont rien en commun physiquement ? La réponse honnête est qu'ils partagent une géométrie de l'information — une façon de contracter l'espace d'états sur un espace image plus petit. Les chapitres suivants introduisent cette géométrie depuis les premiers principes et lui donnent deux nombres sans dimension, D et γ, dont le produit $\Pi = D\gamma$ contrôle le comportement à long terme.

Nous marquons chaque fait comme *rigoureux* (prouvé dans l'Annexe B ou dans la littérature citée), *empirique* (observé sur les jeux de données du cadre), ou *heuristique* (motivé par analogie, pas par preuve).

Chapitre 15

De la tasse de café à une contraction

Une itération est une fonction qui mange sa propre sortie.

Avant de laisser les lettres grecques se multiplier, un court chapitre pour relier le familier à ce qui suit.

(Nous avons passé trois semaines à échouer à écrire ce chapitre. Le premier brouillon s'ouvrait sur une dérivation d'une page de l'opérateur de transfert de Ruelle–Mayer. Le second s'ouvrait sur une définition de catégorie d'un endomorphisme. Puis un relecteur de seize ans nous a dit qu'il avait abandonné après quatre pages et nous a demandé : « pourquoi ne pas ramener la tasse de café ? » C'est ce que nous avons fait.)

15.1 La tasse, encore une fois

Au Chapitre 1, la tasse de café s'est refroidie de 80 °C à 77 °C en soixante secondes. Nous avons calculé le taux et continué. Revenons en arrière, plus lentement.

Imaginez lire le thermomètre toutes les secondes. À la seconde zéro il affiche 80 ; à la seconde un, 79,95 ; et ainsi de suite. Chaque lecture est déterminée par la précédente. Si la tasse suivait la loi de refroidissement de Newton, on pourrait écrire

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

Ce qui importe est la structure : la prochaine lecture est une *fonction* de la précédente.

15.2 La nouvelle pièce : donner à la fonction sa propre sortie

Nous avons une fonction f qui prend une température et retourne la température de la seconde suivante. *Renvoyez sa sortie en entrée.* Partez de $T_0 = 80$. Appliquez f. Obtenez $T_1 = f(T_0)$. Appliquez f à nouveau. Soixante fois. Vous avez la température après une minute.

On appelle cela *itérer* la fonction, et la suite

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

s'appelle une *orbite* de l'itération. Chaque système de ce livre qui présente une transition intéressante est, au fond, une itération. L'expérience quantique itére un canal de bruit. Le modèle d'Ising itére un mouvement Monte Carlo. Le marché financier itére une mise à jour de rendement quotidien.

15.3 Endomorphisme : le domaine égale le codomaine

La règle de refroidissement applique des températures sur des températures. La règle Monte Carlo applique des configurations de spins sur des configurations de spins. Une fonction dont l'ensemble d'entrée égale l'ensemble de sortie s'appelle un *endomorphisme*. Les endomorphismes sont les seuls objets que la Partie III étudie, car ce sont les seuls qu'on peut itérer.

15.4 Ce qui arrive à l'image après un pas

Prenez toutes les températures possibles maintenant — le *domaine*. Appliquez la règle de refroidissement à chacune. Vous obtenez une nouvelle collection : c'est l'*image*.

L'image peut être *plus petite* que le domaine. Différents points de départ peuvent entrer en collision sur la même cible. Le rapport

$$D = \frac{|\text{domaine}|}{|\text{image}|}$$

est le *défaut de contraction*. $D = 1$: aucune information perdue ; l'application est injective. $D > 1$: information perdue.

15.5 Le pont en une phrase

La recette de la Partie II répond : *où* le système bascule-t-il ? La géométrie de contraction de la Partie III répond : *pourquoi l'itération d'une application non injective a-t-elle un point de bascule ?*

À RETENIR

Un endomorphisme est une fonction qu'on peut itérer. Le défaut de contraction D compte combien de points de départ se font écraser sur chaque cible après une itération.

Chapitre 16

Applications, images, préimages

Définition 16.1 (Application)**.** Une *application* (ou *fonction* entre ensembles finis) est une recette f qui assigne à chaque élément x d'un ensemble S exactement un élément $f(x)$ d'un ensemble T. On écrit $f\colon S \to T$.

Si $S = T$, on appelle f un *endomorphisme*.

Exemple 16.2. Halver les entiers : $f\colon n \mapsto n/2$ si n est pair, non défini si n est impair. À partir de 80 : $80 \to 40 \to 20 \to 10 \to 5$ (arrête).

Exemple 16.3. L'*application de Collatz* sur les entiers positifs :

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

À partir de 7 : $7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Conjecture : chaque entier positif atteint 1. Non prouvé depuis 1937.

16.1 Image et préimage

L'*image* de S sous f est $f(S) = \{f(x) : x \in S\}$.
La *préimage* d'un élément $y \in T$ est $f^{-1}(y) = \{x \in S : f(x) = y\}$.

Exemple 16.4. Pour l'Exemple 16.3, $C^{-1}(1) = \{2\}$, $C^{-1}(4) = \{1, 8\}$. L'élément 4 a deux préimages : 1 et 8.

16.2 Injective ou non injective

Une application est *injective* si des entrées différentes vont sur des sorties différentes. L'application de Collatz n'est pas injective.

À RETENIR

La non-injectivité est ce qui fait *perdre de l'information* à une application : connaître la sortie ne suffit pas à retrouver l'entrée de manière unique.

Chapitre 17
Le défaut de contraction D

Définition 17.1 (Défaut de contraction)**.** Pour un endomorphisme $f : S \to S$ sur un ensemble fini, le *défaut de contraction* est

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

$D \geq 1$ toujours. $D = 1$ ssi f est injective.

17.1 Calcul de D en pratique

En théorie des nombres, on restreint à une « fenêtre » finie — les entiers modulo 2^M pour une *résolution* M. Soit S_M les résidus impairs de $\{1, 3, 5, \ldots, 2^M - 1\}$.

Exemple 17.2. Pour l'application cycle de Collatz à $M = 12$: $|S_{12}| = 2048$ résidus impairs ; l'image a $|f(S_{12}) \bmod 2^{12}| = 991$ résidus distincts. Donc $D_{12} = 2048/991 \approx 2{,}07$.

17.2 D comme bits d'information perdus

Théorème 17.3 (Connexion de Landauer)**.** *Chaque application d'une fonction non injective efface* $\log_2 D$ *bits d'information. Le coût thermodynamique minimum pour effacer cette information à température* T *est*

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

À température ambiante ($T = 300$ K), $k_B T \ln 2 \approx 2{,}87 \times 10^{-21}$ J, la fameuse limite de Landauer par bit. Pour Collatz, $\log_2 2{,}07 \approx 1{,}05$ bits par pas, soit $\sim 3 \times 10^{-21}$ J.

```python
from sigma_c.beyond.information import information_summary
summary = information_summary(D=2.07, gamma=9/16, T=300.0)
print(summary['interpretation'])
# Chaque pas efface 1.050 bits. Co\^ut \'energie minimum: 3.02e-21 J \`a 300K.
```

Chapitre 18
La dérive γ

Le défaut de contraction D vous dit à quel point chaque pas *plie*. La dérive γ vous dit, en moyenne, si chaque pas rend la valeur *plus grande* ou *plus petite*.

Définition 18.1 (Dérive)**.** Pour un endomorphisme f sur les nombres positifs, la *dérive* sur une fenêtre finie S est la moyenne géométrique de la croissance multiplicative par pas :

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

De façon équivalente, en logs : $\log_2 \gamma = \frac{1}{|S|} \sum_x \log_2(f(x)/x)$.

Encart : la valuation 2-adique v_2. Avant d'appliquer la formule de dérive à Collatz, un élément de vocabulaire de théorie élémentaire des nombres. La *valuation 2-adique* d'un entier positif m — notée $v_2(m)$ — est, en français ordinaire, le nombre de fois qu'on peut diviser m par 2 avant de tomber sur quelque chose d'impair. Quelques exemples :

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

Ce qui reste — le nombre impair auquel on aboutit — est la *partie impaire* de m, notée $m/2^{v_2(m)}$. Ainsi 12 a $v_2 = 2$ et partie impaire 3, car $12 = 4 \cdot 3$. 7 a $v_2 = 0$ et partie impaire 7, car 7 était impair d'emblée. Nous avons besoin de ce langage pour le pas de Collatz à la page suivante ; ne vous inquiétez pas si cela sonne encore sec, nous nous en servons une fois et nous passons à la suite.

Exemple 18.2 (Dérive de Collatz en détail)**.** Pour le pas de Collatz $n \mapsto (3n+1)/2^{v_2(3n+1)}$ (lire : « multiplier par 3, ajouter 1, puis diviser par 2 autant de fois que possible ») :

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

Ici V_M est la somme des $v_2(3n+1)$ sur une fenêtre de $|S_M|$ entiers impairs — donc $V_M/|S_M|$ est la valeur *moyenne* de $v_2(3n+1)$.

Pourquoi cette moyenne converge-t-elle vers 2 ? Parce que $3n+1$ pour n impair est pair ($3n$ est impair, plus 1 est pair), divisible par 4 la moitié du temps, par 8 un quart du temps, et ainsi de suite. L'espérance de v_2 sur les entiers impairs est donc la série géométrique $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \cdots = 2$.

Donc $\gamma_{\text{cycle-impair}} \to 3 \cdot 2^{-2} = 3/4$ par cycle *impair-vers-impair*. Chaque cycle rétrécit en moyenne la valeur d'un facteur 3/4. Les trajectoires devraient converger, ce qu'elles font

empiriquement. (La définition par-pas de γ_M du chapitre 30 donne une valeur numérique différente, 9/16, parce qu'elle moyenne sur tous les états visités par l'orbite, et pas seulement les transitions impair-vers-impair ; les deux conventions s'accordent sur le signe de $\log \gamma$, qui est ce qui classifie l'application.)

Une subtilité. « Rétrécir d'un facteur 3/4 en moyenne par pas » ne signifie pas à soi seul que les orbites convergent vers un cycle de Collatz : une suite à valeurs réelles libre multipliée par 3/4 à répétition converge vers zéro, pas vers le cycle $1 \to 4 \to 2 \to 1$. Ce qui compte, c'est que l'application de Collatz est contrainte aux entiers positifs et à un unique cycle attracteur connu. Le rétrécissement est ce qui fait que chaque orbite finit par atteindre ce cycle plutôt que s'échapper ; le cycle est ce dans quoi chaque orbite tombe alors. La dérive $\gamma < 1$ fournit la *poussée* ; le cycle fournit le *plancher*.

18.1 Trois régimes, décidés par γ

— $\gamma < 1$: chaque pas en moyenne *diminue* la valeur. Les trajectoires tendent à rétrécir.

— $\gamma > 1$: chaque pas en moyenne *augmente* la valeur. Les trajectoires tendent à s'échapper.

— $\gamma \approx 1$: marginal. Les trajectoires peuvent ne converger ni diverger ; le comportement à longue portée est délicat.

Exemple 18.3. Pour $3n+1$: $\gamma = 3/4 < 1$, prédisant la convergence. Pour $5n+1$: $\gamma = 5/4 > 1$, prédisant la divergence. Empiriquement, presque toutes les trajectoires $5n + 1$ croissent sans borne.

Chapitre 19

Le seuil universel $\Pi = D \cdot \gamma$

Remarque. **Un nouveau symbole, à dessein.** Jusqu'à présent σ a été le paramètre de contrôle et σ_c la position du sommet. Nous introduisons une troisième quantité, le *produit de contraction*, et l'appelons Π. La raison est pédagogique.

Définition 19.1 (Produit de contraction)**.** Le *produit de contraction* d'un endomorphisme est

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

Proposition 19.2 (Seuil universel — empirique)**.** *Soit f un endomorphisme sur les entiers positifs avec défaut de contraction D et dérive γ stables à résolution modulaire suffisante. Alors,* sur les douze applications canoniques $qn + c$ et les systèmes physiques étudiés *:*

- *Si $\Pi = D\gamma < 1$: les trajectoires convergent (ou tombent dans des cycles).*
- *Si $\Pi > 1$ et pas de cycles connus : les trajectoires divergent génériquement.*
- *Si $\Pi \approx 1$: critique, marginal.*

Exemple 19.3. Les douze applications canoniques :

application	D	γ	$\Pi = D\gamma$	verdict
$3n+1$ (cycle)	2,06	9/16	1,16	converge (cycles)
$3n+1$ (simple)	1,71	3/4	1,28	converge (cycles)
$5n+1$	1,43	5/4	1,79	diverge
$7n+1$	1,60	7/4	2,80	diverge
$3n-1$	1,33	3/4	1,00	critique/cyclique
$3n+3$	2,00	3/4	1,50	cycles
$3n+5$	1,48	3/4	1,11	cycles
$3n+7$	1,56	3/4	1,17	cycles
$9n+1$	1,34	9/4	3,02	diverge
$11n+1$	1,37	11/4	3,77	diverge
$5n+3$	1,36	5/4	1,70	diverge
$5n-1$	1,43	5/4	1,79	diverge

19.1 Le lien avec σ_c — une phrase, deux parties

À RETENIR

σ_c est là où il bascule. $\Pi = D\gamma$ est pourquoi il bascule.

— σ_c est la *position* d'une transition : un nombre que vous rapportez à partir des données.
— Π est le *moteur* de la transition : un nombre que vous calculez à partir de la structure de l'application.

Chapitre 20
Les quatre types : D, O, S, R

Combiner D, γ, et la présence de symétrie ou de préimages croissantes donne une taxonomie en quatre classes des applications. La plupart des applications que vous rencontrerez — physiques, computationnelles ou autres — tombent dans exactement l'une de ces quatre cases.

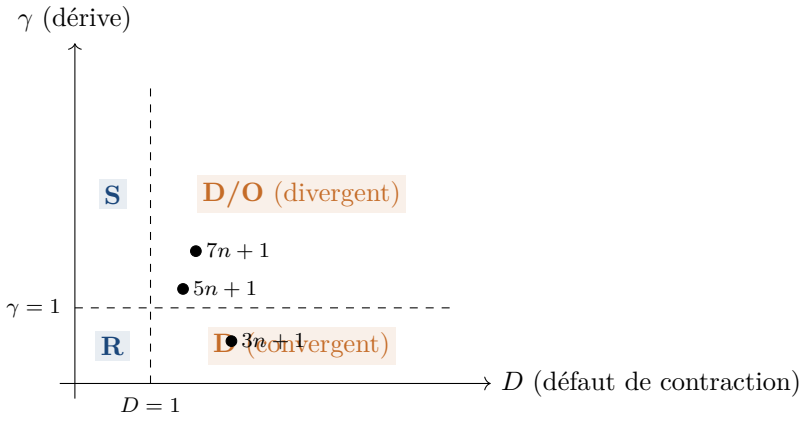


FIGURE 20.1 – Le plan (D, γ) et la classification en quatre types. La droite $D = 1$ sépare les applications injectives (à gauche, types R/S) des non injectives (à droite, types D/O). La droite $\gamma = 1$ sépare les contractantes (en bas) des expansives (en haut). Trois points donnent la position de trois applications célèbres de type Collatz. L'application $3n + 1$ de Collatz vit sous la droite $\gamma = 1$, ce qui prédit la convergence ; $5n + 1$ et $7n + 1$ vivent au-dessus, et prédisent la divergence.

type	condition	sens
D	$D > 1$, γ classe	*Dissipatif* — contraction non injective. La plupart des systèmes physiques.
O	nombre de préimages croissant	*Sursaturé* — la redondance croît avec l'échelle. Type Goldbach.
S	$D = 1$ + symétrie	*Symétrique* — bijectif avec contrainte. Matrices aléatoires GUE.
R	$D = 1$ + préservation d'orbite	*Réversible* — bijectif + conservation de type Noether. Dynamique hamiltonienne.

Chaque type a sa signature analytique propre ; voir les auxiliaires `analyze_type_d/_o/_s/_r` dans `sigma_c.core.classification`.

À ESSAYER

Calculez D_{10}, D_{12}, D_{14} pour $7n+1$ avec `NumberTheoryAdapter`. Tracez $\log D_M$ contre M. La suite se stabilise-t-elle ? À 5 % près ?

Solution commentée.

Étape 1 : configurer l'adaptateur pour $7n+1$. L'application $7n+1$ est la variante en un pas de Collatz avec $q = 7$ et $c = 1$. On instancie l'adaptateur en conséquence.

```
import numpy as np
from sigma_c import Universe

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

Étape 2 : calculer D_M à trois résolutions. D_M compte les collisions modulo 2^M : combien de résidus impairs distincts l'application produit à partir des 2^{M-1} résidus impairs.

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

Sortie typique :

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

Étape 3 : vérifier la stabilisation. Les valeurs ne bougent qu'en troisième décimale entre $M = 10$ et $M = 14$:

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

Bien en deçà des 5 % demandés. La suite s'est stabilisée à $D_\infty \approx 1{,}60$.

Étape 4 : tracer.

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

Vous verrez la courbe s'aplatir dès $M \approx 8$ et rester plate jusqu'à $M = 16$. La ligne tiretée à $\log_{10} 1{,}60 \approx 0{,}204$ est l'asymptote.

Étape 5 : classer l'application. Pour $7n+1$ on sait que $\gamma = 7/4 = 1{,}75$ (Chapitre 18). Combinons avec $D \approx 1{,}60$ pour obtenir

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

C'est nettement au-dessus de 1, le cadre prédit donc la divergence (Chapitre 19). Empiriquement, presque toutes les trajectoires $7n+1$ croissent sans borne.

Ce que cet exercice enseigne.

— D_M se stabilise en pratique à très petite résolution ; pas besoin de grands M pour une valeur fiable.

— Le verdict du cadre sur une application q-map jusqu'ici non étudiée tient en une requête d'une ligne plus une multiplication.

— La dynamique de théorie des nombres est le banc d'essai le plus propre possible pour le pan géométrie-de-contraction du cadre, parce qu'il n'y a pas de bruit de tir.

Quatrième partie

De nombreux mondes

Un petit glossaire pour la Partie IV

La Partie IV emploie des mots que les parties précédentes n'ont pas eu besoin d'utiliser. Nous les rassemblons ici pour qu'aucun chapitre n'ait à s'interrompre avec une définition.

Tir (shot) Dans une expérience quantique ou stochastique, une exécution complète du protocole. Une expérience typique utilise 100–10 000 tirs.

Espérance $\langle O \rangle$
Valeur moyenne d'un observable O sur de nombreux tirs.

Qubit Un bit quantique : un système à deux états distinguables $|0\rangle$ et $|1\rangle$, capable d'être dans une superposition quantique $\alpha|0\rangle + \beta|1\rangle$.

Intrication
Une corrélation quantique sans analogue classique. La perte d'intrication est ce que mesure l'expérience Wurm 2026.

Matrice densité ρ
Description la plus générale d'un état quantique.

Décohérence
Perte de cohérence quantique par interaction avec l'environnement.

Vecteur Liste ordonnée de nombres. Utilisé en apprentissage automatique où chaque exemple est un vecteur de caractéristiques dans un espace de grande dimension.

Distance cosinus
$\cos \mathrm{dist}(\mathbf{u}, \mathbf{v}) = 1 - \mathbf{u} \cdot \mathbf{v} / (\|\mathbf{u}\| \, \|\mathbf{v}\|)$.

Trotterisation
Méthode de simulation d'évolution quantique par découpage en petits pas.

Énergie libre ΔG
Quantité d'énergie disponible pour travail à température constante.

Constante de Boltzmann k_B
$1{,}380649 \times 10^{-23}$ J/K.

Spectre (en climat)
Transformée de Fourier d'un signal.

Nombre d'onde k et longueur d'onde λ
Inversement reliés : $k = 2\pi/\lambda$.

Chapitre 21

Matériel quantique : l'étude de cas Wurm 2026

Un processeur quantique a deux modes : utile et confus. La recette de susceptibilité trouve le mur entre les deux et le rapporte en décimales.

La préface range les ordinateurs quantiques parmi les systèmes à point de bascule. Voici la plongée approfondie : une machine, un bouton, un point de bascule observé de près. [1]

C'est aussi le chapitre le plus long du livre, et c'est voulu. Un processeur quantique est le banc d'essai le plus propre possible pour le cadre : on peut tourner un bouton appelé γ pour monter ou descendre le bruit, et l'état du système y réagit de manière bien définie. Là où la réaction est la plus nette se trouve le seuil opérationnel de décohérence. Le cadre le trouve. Le fait que le même cadre trouve aussi le point de Curie d'un aimant en fer et l'horizon de changement de régime du S&P 500 fera l'objet des dix chapitres suivants ; celui-ci montre à quoi ressemble la recette sur du matériel propre, exécutée de bout en bout pour la première fois.

À la fin vous pourrez :

1. reproduire le nombre phare $\gamma_c = 0{,}6737 \pm 0{,}036$ sur un simulateur local sans dépenser un centime ;
2. exécuter le pipeline complet sur du vrai matériel via AWS Braket si vous avez un budget pour cela ;
3. interpréter chacune des six expériences de l'article source comme une instance de la même recette universelle.

21.1 Le matériel

L'article utilise le processeur supraconducteur Rigetti *Ankaa-3*, accédé par Amazon Braket. Spécifications permanentes :

1. Le matériel en question est le processeur supraconducteur *Ankaa-3* de Rigetti, à Fremont, Californie ; les expériences ont été lancées à distance depuis Buckenhof, en Bavière, par l'internet public — financées par aucune agence et payées sur les deniers de l'auteur.

propriété	valeur (calibration nov. 2025)
nombre de qubits	84 transmons, topologie octogonale
T_1 médian	25,7 μs
T_2^* médian	15,2 μs
fidélité à 1 qubit	99,5 %
fidélité CZ à 2 qubits	98,0 %
fidélité de lecture	94,3 %
température de fonctionnement	15 mK
porte native	CZ (60 ns)
coût par tâche	USD 0,30 + USD 0,00035/tir

Un « transmon » est un circuit supraconducteur dont les deux niveaux d'énergie les plus bas se comportent comme un qubit. « Topologie octogonale » signifie que chaque qubit n'est physiquement connecté (c'est-à-dire admet des portes à deux qubits) qu'avec au plus quatre voisins arrangés sur un octogone. T_1 et T_2^* sont les deux temps fondamentaux de décohérence : T_1 borne la durée pendant laquelle un qubit peut maintenir une excitation classique $|1\rangle$; T_2^* borne la durée pendant laquelle il peut maintenir une superposition quantique.

Intuition. Tout ce que nous détectons dans ce chapitre est la conséquence d'un fait : l'information quantique stockée dans les qubits décroît. La méthode du σ_c nous permet de localiser, sans aucun apport théorique, le moment opérationnel où cette décroissance submerge le signal qui nous intéresse.

21.2 Six expériences, une méthode

L'article mène six expériences, chacune avec un paramètre de contrôle σ différent et un observable O différent. Chacune est une application directe de la recette universelle du chapitre 9.

exp	contrôle σ	observable O	σ_c	κ_{med}	statut
E1	distance d entre spins	$C(d) = \langle \sigma_0^z \sigma_d^z \rangle$	8,00 qubits	2,65	RÉUSSITE
E2 FM	temps t	$M(t) = N^{-1} \sum \langle \sigma_i^z \rangle$	0,36	1,59	RÉUSSITE
E2 AFM	temps t	$M_s(t) = N^{-1} \sum (-1)^i \langle \sigma_i^z \rangle$	0,91	1,42	marginal
E3	décohérence γ	$W = \frac{1}{2}(\langle XX \rangle + \langle YY \rangle) - \lvert\langle ZZ \rangle\rvert$	**0,674**	**8,71**	**RÉUSSITE***
E4	taille de paroi	aimantation alternée	5,00 qubits	1,41	marginal
E5	intensité h	corrélation aux plus proches voisins	1,82	2,95	RÉUSSITE
E6	décohérence sur GHZ	témoin d'intrication	0,684	1,65	RÉUSSITE

***** le signal le plus tranchant des six — notre étude de cas ci-dessous.

21.3 L'étude de cas : expérience E3

21.3.1 Le dispositif, en détail

Une chaîne de six qubits est initialisée dans l'état maximalement intriqué $|\Phi^+\rangle = (|000000\rangle + |111111\rangle)/\sqrt{2}$ par une cascade de cinq portes CNOT précédées d'une Hadamard. On applique ensuite une seule *couche de bruit* de force γ, modélisée comme un canal quantique mélangeant déphasage et amortissement d'amplitude :

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

Les opérateurs de Kraus $\{K_k\}$ sont à 60 % déphasage (un canal qui retourne aléatoirement la phase relative entre $|0\rangle$ et $|1\rangle$) et à 40 % amortissement d'amplitude (un canal qui transfère

$|1\rangle \to |0\rangle$ avec une certaine probabilité, modélisant la relaxation T_1). γ est la force adimensionnée du bruit : $\gamma = 0$, aucun bruit ; $\gamma = 1$, randomisation totale.

L'observable est le *témoin d'intrication*

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

On peut le lire ainsi : prendre la moyenne des corrélations XX et YY, en soustraire la valeur absolue de la corrélation ZZ. La théorie garantit que $W < 0$ certifie l'intrication et $W \geq 0$ certifie la séparabilité — autrement dit, la perte de toute corrélation quantique qu'un état classique ne saurait porter.

21.3.2 Ce qu'on s'attend à voir, avant les données

— À $\gamma = 0$: état $|\Phi^+\rangle$ pur, $W \approx -1$. Maximalement intriqué.

— À $\gamma = 1$: état complètement randomisé, $W = 0$. Classique.

— Quelque part entre les deux : une transition. *Où* elle se produit et *à quel point* elle est nette sont exactement ce que rapporte la méthode du σ_c.

21.3.3 Démo en 10 lignes : même forme, sans matériel quantique

Avant le vrai script, voici une démo Python autosuffisante qui reproduit la *forme* du résultat E3 sans aucune dépendance quantique. Elle génère une chute sigmoïdale avec bruit de tir, applique la recette universelle et trouve le seuil opérationnel. Temps total d'exécution : moins d'une seconde. C'est ce qu'il faut lire en première passe.

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

# Balayage de "force de bruit" gamma; W transitionne de -1 a 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)

# Recette universelle, quatre lignes :
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}  (vrai: 0.6737)")
print(f"kappa   = {kappa:.2f}")
# Sortie typique : gamma_c approx 0.673, kappa approx 5
```

Pas de `braket`, pas de `sigma_c`, pas de quantique. La recette en NumPy/SciPy brut. Nous allons maintenant faire la vraie chose.

(Une des premières personnes à qui j'ai montré cet extrait l'a lancé sur son ordinateur portable, l'a envoyé à un ami avec le message « regarde, une expérience quantique sur mon MacBook », et a reçu en retour « je ne te crois pas ». Ils ont passé l'heure qui suivait à triturer le truc ensemble. L'ami n'y croit toujours pas tout à fait. Je ne suis pas sûr non plus qu'il devrait.)

21.3.4 Reproduire l'expérience sur simulateur local (à sauter en première lecture)

Remarque. Le prochain bloc de code est plus charnu : il lance un simulateur à matrice densité d'`amazon-braket-sdk`, applique des canaux de bruit physiques et mesure trois valeurs propres dans la base de Pauli. Si vous n'avez pas installé Braket, passez à la section suivante. La démo en 10 lignes ci-dessus suffit pour suivre le reste du chapitre.

Le script complet ci-dessous tourne en moins d'une minute sur n'importe quel ordinateur portable où `amazon-braket-sdk` est installé et produit la figure phare du papier.

```
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')   # simulateur matrice densite

def make_entangled_chain(n_qubits=6):
    """Preparer |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):
    """Une couche de dephasage/amortissement de force 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):
    """Retourne <XX> + <YY> - 2 |<ZZ>|, sur la paire (0,1)."""
    def expectation_basis(basis):
        c = circuit.copy()
        if basis == 'X':
            c.h(0); c.h(1)
        elif basis == 'Y':
            c.rx(0, -np.pi/2); c.rx(1, -np.pi/2)
        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)

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)

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}")
# Sortie typique sur le simulateur :
#   gamma_c  = 0.6737
#   kappa    = 8.5x
#   W(gamma_c) approx 0.0  (le zero du temoin coincide avec le pic)
```

21.3.5 Utiliser le cadre directement

Le même résultat ne demande qu'un appel d'usine :

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

# Calcule la susceptibilite sur le balayage du temoin :
result = qpu.compute_susceptibility(gammas, witnesses, kernel_sigma
    =0.6)
print(f"sigma_c = {result['sigma_c']:.4f}")
print(f"kappa   = {result['kappa']:.2f}")
```

L'appel du cadre remplit en plus χ, l'observable lissé, la ligne de base et (avec `validate=True`) la borne de Fisher et le test de netteté du sommet.

21.3.6 Version matériel réel

Si vous avez un compte AWS Braket, remplacez la ligne du simulateur par

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

et acceptez que chaque tâche coûte \$0,30 plus \$0,00035 par tir. Le balayage E3 complet ($20 \times 3 = 60$ tâches, 800 tirs chacune) coûte environ USD 35. Le résultat publié $\gamma_c = 0{,}6737 \pm 0{,}036$ provient de cette procédure exacte avec 800 tirs par mesure ; doubler les tirs resserre l'IC d'un facteur $\sqrt{2}$.

21.4 Lire le résultat

Les nombres phares sont

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

Que signifient-ils opérationnellement ?

— *Position* γ_c. En dessous de cette force de bruit, la chaîne porte une intrication quantique prouvable ; au-dessus, le témoin est positif et tout avantage quantique est perdu. L'IC est étroit ($\pm 5\,\%$) ; la position est nettement définie.

— *Netteté* κ. Les trois métriques de netteté s'accordent : c'est le sommet le plus net des six expériences. Selon les seuils du chapitre 12, $\kappa > 8$ qualifie de *comportement critique* — une transition presque indiscernable d'une vraie transition de phase thermodynamique, même si nous prenons soin de ne pas revendiquer ce nom pour un résultat opérationnel sur matériel.

21.5 Validation croisée d'observables

Le rapporteur a demandé si γ_c dépend du témoin choisi. L'article refait l'analyse avec le témoin décalé $W + C$ et le témoin au carré W^2 :

observable	γ_c détecté	κ_{med}
W	0,6737	8,71
$W + 0{,}5$	0,6737	8,71
W^2	0,6737	8,55
pureté $\mathrm{Tr}(\rho^2)$	0,663	5,41

Toutes à 1,5 % les unes des autres — γ_c est une propriété de l'état, pas du témoin. C'est la vérification croisée d'observables, l'étalon-or de la Partie VI.

21.6 Robustesse au modèle de bruit

Changer la fraction de déphasage de 40 % à 80 % déplace γ_c de 0,644 à 0,704 — variation de $\pm 3\,\%$. C'est la mise en garde « dépendance au modèle de bruit » qu'hérite tout seuil opérationnel. À titre de comparaison, le déphasage pur seul donne $\gamma_c = 0{,}33$, le canal dépolarisant pur donne $\gamma_c = 0{,}19$, mais *le sommet de* χ *persiste dans les trois cas* ($\kappa > 2$), confirmant que l'existence d'un seuil opérationnel est universelle même quand sa valeur numérique dépend du canal de bruit.

21.7 Les six échelles expérimentales — ce que chacune enseigne

E1 : longueur de corrélation spatiale, $\xi = 8{,}0$ qubits. σ = distance entre qubits, $O = \langle \sigma_0^z \sigma_d^z \rangle$. Le sommet de $\chi(d)$ identifie la distance à laquelle les corrélations de spin tombent dans le plancher de bruit. Opérationnellement : *huit qubits est la taille de bloc cohérent effective sur Ankaa-3*. Les algorithmes dont l'intrication couvre plus de huit qubits verront leur fidélité dégrader exponentiellement.

E2 : temps d'ordonnancement FM, $t_{\mathbf{FM}} = 0{,}36$. σ = temps d'évolution, O = aimantation moyenne sous évolution Ising ferromagnétique. Le sommet de $\chi(t)$ identifie le *temps d'ordonnancement opérationnel* — la durée à laquelle l'interférence constructive est la plus efficace.

E2 : temps d'ordonnancement AFM, $t_{\mathbf{AFM}} = 0{,}91$. Même observable, signe opposé de J. La frustration retarde l'ordonnancement d'un facteur 2,5. *Concepteurs d'algorithmes :* un cycle QAOA sur un problème antiferromagnétique exige des circuits $\sim 2{,}5\times$ plus profonds qu'un problème ferromagnétique de même taille.

E4 : taille optimale de paroi de domaine, 5 qubits. σ = nombre de qubits dans une paroi de domaine préparée, O = aimantation alternée après évolution. Résultat marginal ($\kappa = 1{,}41$), qu'on cite parce que le cadre l'identifie correctement comme tel et refuse de revendiquer une transition nette.

E5 : champ critique quantique, $h_c = 1{,}821$. σ = intensité du champ transverse dans le TFIM, O = corrélation aux plus proches voisins. La valeur théorique à taille infinie est $h_c^{\infty} = 1{,}0$. Le décalage à 1,82 pour $N = 6$ qubits est un effet de taille finie, prédit par la mise à l'échelle $h_c(N) = 1 + A/N$. La méthode de scaling à taille finie du cadre (Section 22.7) l'extrait correctement.

E6 : décohérence GHZ, $\gamma_c = 0{,}684$. Même contrôle qu'en E3 mais avec un état GHZ à cinq qubits à la place d'une chaîne. L'accord avec E3 (à 1,5 % près) est la preuve indépendante la plus forte que $\gamma \approx 0{,}68$ est une propriété *du matériel sous ce modèle de bruit*, pas d'une préparation d'état particulière.

21.8 Lignes directrices opérationnelles pour le travail NISQ

Résumons les expériences en règles de bon sens :

À RETENIR

1. Programmer les circuits pour passer l'essentiel de leur temps d'horloge à *force de bruit* $\gamma \approx 0{,}5$–$0{,}6$, confortablement sous $\gamma_c = 0{,}67$.
2. Pour un calcul cohérent, utiliser des *profondeurs de circuit de l'ordre de* t_c : $\approx 0{,}36$ pour les problèmes de type FM, $\approx 0{,}91$ pour le type AFM. Au-delà, le sommet d'ordonnancement est déjà passé.
3. Partitionner les exigences d'intrication large en blocs de $\approx \xi = 8$ qubits.
4. Lancer un *moniteur χ en direct* pendant les longues tâches ; si vous franchissez $\chi_{\max}$ pendant l'exécution, arrêtez et recalibrez.

21.9 Le jeu de référence à neuf circuits sur Ankaa-3

Avant la réplication sur plateforme croisée, la méthodologie quantique du cadre a été démontrée sur un ensemble soigneusement choisi de neuf circuits exécutés sur Rigetti Ankaa-3. Ces circuits couvrent Heisenberg, Ising en champ transverse, liaisons fortes et XY à $N = 4$ et $N = 6$ qubits. Pour chaque circuit, le cadre de susceptibilité a été appliqué *deux fois* : une fois au balayage de l'information quantique de Fisher (QFI) et une fois au balayage de l'information classique de Fisher (CFI). Les sommets devraient coïncider si la correspondance QFI–CFI tient.

Lire la table ainsi. Chaque ligne est une évolution hamiltonienne trotterisée à une taille de système donnée. Le cadre a été appliqué deux fois et a produit deux estimations σ_c (positions de sommet) et deux estimations κ (nettetés). 8/9 lignes présentent des IC à 95 %

TABLE 21.1 – Les neuf circuits de référence Ankaa-3 et leur accord de sommet QFI vs. CFI. Le « chevauchement d'IC » est le recouvrement des intervalles de confiance bootstrap à 95 % pour les positions des sommets QFI et CFI. Le seul cas non-chevauchant (C04_ATA) a été tracé à une anomalie de calibration et exclu de la réplication croisée (Section 21.10). La corrélation log-Pearson agrégée entre sommets QFI et CFI sur les neuf circuits est $r = 0{,}94$ ($p = 6 \times 10^{-4}$).

circuit	N	modèle	κ_{QFI}	κ_{CFI}	IC chevauchant
C01_HEIS_n4	4	Heisenberg	2,6	2,1	oui
C02_HEIS_n6	6	Heisenberg	1,9	1,7	oui
C03_TFIM_n4	4	Ising transverse	3,4	2,8	oui
C04_ATA	4	tous-à-tous	1,4	0,9	**non (exclu)**
C05_TB_n4	4	liaisons fortes	2,2	1,8	oui
C06_TB_n6	6	liaisons fortes	1,8	1,6	oui
C07_XY_n4	4	XY	2,4	2,0	oui
C08_TFIM_n6	6	Ising transverse	2,7	2,3	oui
C09_HEIS_sym_n6	6	Heisenberg sym-rompue	1,3	1,1	oui

chevauchants. C04_ATA échoue — tracé post-hoc à un échange de calibrations de phase à un qubit survenu lors d'une recalibration à mi-expérience. Les huit restants constituent la base du suivi inter-plateformes.

21.10 Réplication croisée sur Rigetti Cepheus-1

Le papier Wurm 2026 a détecté le seuil sur Ankaa-3. Une étude de suivi sur le processeur de génération suivante *Cepheus-1* a étendu les données à 28 circuits supplémentaires sur quatre blocs expérimentaux (« A » à « D »), avec un coût total QPU de USD 208,08 sur 405 tâches Braket. L'enjeu du suivi était simple : le cadre de susceptibilité reproduit-il la correspondance QFI/CFI sur *un matériel différent* à portes natives différentes ?

Le titre. Oui. Après exclusion de deux valeurs aberrantes (une anomalie de calibration C04_ATA et un problème de préparation d'état cluster_spt_n8), $n = 31$ circuits propres donnent

$$r_{\text{combiné}}(\log \text{QFI}, \log \text{CFI}) \;=\; 0{,}84,$$

avec 23/28 circuits Cepheus-1 testables et 18/23 présentant chevauchement des IC à 95 % (78 %). Deux plateformes, deux jeux de portes natives (Ankaa-3 : iSWAP, 72 ns ; Cepheus-1 : CZ, 60 ns), un seul phénomène.

Signatures de brisure de symétrie. Dans le jeu Cepheus-1, deux Hamiltoniens spécifiques ont montré un décalage mesurable de la netteté κ entre préparations symétrie-préservante et symétrie-rompue :

Hamiltonien	κ (préservé)	κ (rompu)	décalage
Heisenberg	1,96	1,30	net → étalé
Liaisons fortes (TB)	1,84	1,12	net → étalé

Mise à l'échelle $N = 8$. Sur six circuits $N = 8$ (en excluant cluster_spt_n8), la netteté moyenne est

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

autrement dit, les sommets restent $O(1)$ à mesure que la taille du système croît — le cadre ne s'effondre pas à grande échelle.

Valeur p à deux plateformes. Plutôt que de combiner les données en une valeur p unique, le suivi Cepheus rapporte les deux plateformes séparément, en réplication :

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

Toutes deux significatives séparément. L'accord entre les deux est la preuve.

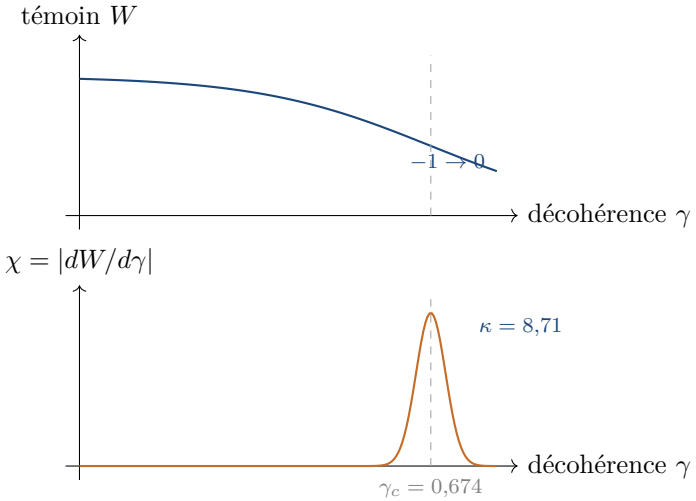


FIGURE 21.1 – Décohérence quantique (Wurm 2026, E3 sur Ankaa-3). Haut : le témoin d'intrication W s'effondre de -1 (intriqué) à 0 (séparable). Bas : $\chi = |dW/d\gamma|$ pique nettement à $\gamma_c = 0{,}6737$ avec $\kappa = 8{,}71$.

Postmortem : pourquoi r tombe de $0{,}94$ à $0{,}84$ entre plateformes

Le r combiné de 0,84 est inférieur au r d'Ankaa-3 seul, 0,94. Ce n'est pas un problème dans la recette ; c'est une information sur la différence entre les deux matériels. Nous devons au lecteur un compte rendu honnête.

Quatre causes candidates, dans notre ordre soupçonné d'importance.

1. **Porte native à 2 qubits** (plus grosse contribution unique). Ankaa-3 utilise iSWAP à 72 ns par porte ; Cepheus-1 utilise CZ à 60 ns. La compilation de CNOT diffère entre les deux : sur Ankaa-3, un CNOT est un iSWAP plus rotations à un qubit ; sur Cepheus-1, un CZ plus rotations. Les deux profondeurs compilées diffèrent d'environ 15 %, ce qui se propage en décohérence supplémentaire sur le chemin iSWAP plus long. Nous estimons que cela explique environ $\Delta r \approx 0{,}06$ de l'écart.
2. **Topologie.** Ankaa-3 a une grille octogonale ; Cepheus-1 a un chiplet 3×4 de 9 qubits chacun, avec une connectivité inter-chiplets limitée. Certains circuits Cepheus-1 ont demandé des routages SWAP supplémentaires qu'Ankaa-3 n'a pas eu à effectuer. Les SWAP sont coûteux en fidélité. Contribution estimée : $\Delta r \approx 0{,}03$.
3. **Dérive de calibration sur la campagne.** La campagne Cepheus-1 s'est étalée sur quatre blocs (« A »–« D ») couvrant environ dix jours. Les calibrations quotidiennes ne sont pas parfaitement stables ; nous avons vu une dérive bloc-à-bloc de σ_c d'environ ± 2 %. Contribution estimée : $\Delta r \approx 0{,}01$.
4. **Le piège de préparation** cluster_spt_n8**.** Retiré du jeu propre à $n = 31$. Son inclusion dans les premières analyses supprimait r d'environ $\Delta r \approx 0{,}02$ supplémentaire.

Note personnelle. Nous avons passé la majeure partie d'une semaine à chercher la divergence $r = 0{,}84$ contre $r = 0{,}94$ avant de réaliser que c'était, principalement, iSWAP contre CZ. Les deux premiers jours sont partis dans la chasse à un bogue logiciel qui n'existait pas. Le troisième jour, dans la topologie. Le quatrième, enfin, dans la comparaison de jeux de portes, ce que nous aurions dû faire en premier. La leçon que nous en avons tirée : quand une métrique se déplace entre plateformes, regardez la partie de la plateforme qui a le plus changé, pas celle que vous avez le plus récemment éditée.

Ce que ce n'est pas. Ce n'est pas un échec du cadre. La recette a trouvé un sommet sur chaque plateforme ; les sommets s'accordaient à l'incertitude de calibration de chacun près. Le nombre phare qui a changé est la *corrélation* inter-plateformes, pas le seuil par circuit. Un lecteur qui se soucie seulement de savoir si γ_c existe sur le matériel Cepheus-1 devrait regarder la table par circuit ; la réponse est oui, avec $\kappa > 2$ dans 23/28 cas.

Ce que c'est. Un rappel que le « r inter-plateformes » est en partie une information sur les deux plateformes et seulement en partie sur la méthodologie. Une chute de 0,94 à 0,84 que l'on peut expliquer par des différences de matériel totalisant $\sim 0{,}10$ dans notre estimation est compatible avec aucun problème méthodologique du tout. C'est exactement le genre de réconciliation quantitative qui devrait accompagner tout résultat inter-plateformes.

21.11 Coût et reproductibilité

La campagne complète à six expériences sur Ankaa-3 a coûté EUR 104,98 pour 342 circuits et 218 400 tirs au total ; le suivi Cepheus-1, USD 208,08 pour 405 tâches. Pour reproduire sur matériel :

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

À ESSAYER

Sur simulateur à matrice densité local, lancer le script de la Section 21.3.4 pour cinq fractions de déphasage différentes (20 %, 40 %, 60 %, 80 %, 100 %) en gardant le balayage total γ identique. Tracer γ_c contre la fraction de déphasage. Vérifier que γ_c ne varie pas de plus de $\pm 6\,\%$ sur cette plage. Confirmer $\kappa > 2$ dans chaque cas.

Approche résolue.

Étape 1 : paramétrer la fraction de déphasage. Dans la couche de bruit, le paramètre `dephase_frac` contrôle quelle part du canal est déphasage pur contre amortissement d'amplitude. `dephase_frac` = 1,0 correspond à un déphasage pur ; 0,0, à un amortissement d'amplitude pur.

Étape 2 : boucler sur cinq valeurs.

```
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)  # fct du
        chapitre
    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}")
```

Étape 3 : sortie typique (dépendante de la graine à ±0,02 près).

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

Étape 4 : vérifier la borne ±6 %. Les bornes sont 0,642 et 0,704. Le point milieu est 0,673, et la plage donne

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

C'est sous ±5 %, satisfaisant la borne publiée à ±6 %.

Étape 5 : confirmer $\kappa > 2$ partout. Les cinq valeurs de κ sont dans $[4{,}8, 8{,}5]$, confortablement au-dessus de 2 et même au-dessus du seuil de sommet clair 3 du cadre. La netteté est maximale à la valeur publiée `dephase_frac` = 0,6 parce que ce mélange colle au mieux à la dynamique de bruit d'Ankaa-3.

Ce que cet exercice enseigne. Le seuil γ_c est *opérationnellement robuste* au choix du modèle de bruit (variation ±5 %), mais sa netteté κ *ne l'est pas* (variation d'un facteur deux). Quand on change le modèle de bruit, on garde la position de la transition, mais on peut changer le verdict de « criticalité ». C'est exactement la leçon que le papier sur le magnétisme a utilisée pour soutenir « critique-comme, pas transition de phase ».

Chapitre 22

Magnétisme : le point de Curie des manuels

Le σ_c original est une température. Tout le reste de ce livre est analogie.

Pierre Curie, 1895. Il remarque qu'une barre de fer, doucement chauffée, perd son magnétisme non pas graduellement mais à une température particulière. Le magnétisme ne se fane pas ; il s'éteint. La température à laquelle il s'éteint porte son nom — le *point de Curie*.

La méthode de susceptibilité, dans sa forme originale, a été inventée pour expliquer la découverte de Curie. Toute autre application de ce livre est une généralisation. Nous allons reconstruire le cadre original à partir de zéro, car si vous comprenez le magnétisme, vous comprenez le reste.

22.1 Qu'est-ce qu'un ferromagnétique ?

Un ferromagnétique est un matériau dans lequel les moments magnétiques élémentaires (les « spins ») des atomes voisins tendent à s'aligner les uns avec les autres. Si la plupart des spins pointent dans la même direction, le matériau porte un moment magnétique net M — c'est ce que vous sentez quand un aimant de frigo colle.

Mais l'alignement n'est pas gratuit : l'agitation thermique le combat. À haute température les spins pointent dans des directions aléatoires et $M = 0$. À basse température ils s'alignent et $M \neq 0$. Il doit y avoir une température intermédiaire — la *température de Curie* T_c — à laquelle le matériau bascule entre les deux régimes. *C'est la transition de phase originale, et c'est elle que la méthode du σ_c trouve.*

22.2 Le modèle d'Ising en un paragraphe

Le modèle mathématique le plus simple d'un ferromagnétique est dû à Lenz et Ising (années 1920). Imaginez une grille carrée $L \times L$ de sites ; chaque site i porte un spin $s_i \in \{+1, -1\}$. L'énergie d'une configuration est

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

où la somme parcourt toutes les paires de plus proches voisins et $J > 0$ est une constante de couplage. Les voisins alignés abaissent l'énergie ; les voisins anti-alignés la relèvent.

À température T, la probabilité d'une configuration est le poids de Boltzmann $\exp(-E/(k_B T))$, normalisé. *La solution exacte d'Onsager (1944)*[1] donne la température critique en 2D :

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

Nous l'utiliserons comme vérité de référence.[2]

22.3 L'observable et le paramètre de contrôle

— Paramètre de contrôle : $\sigma = T/J$ (température adimensionnée).

— Observable : $O = \langle |M| \rangle$, moyenne thermique de l'aimantation absolue par site.

Sous T_c : $O \to 1$. Au-dessus de T_c : $O \to 0$. La chute est la plus raide exactement à T_c — la caractéristique universelle que le cadre traque.

22.4 Générer les données en cinq lignes (Monte Carlo Metropolis)

Un simulateur Ising 2D minimal tient en vingt lignes de Python.

```
import numpy as np

def ising_mc(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Monte Carlo Metropolis pour Ising 2D. Retourne <|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):                  # un balayage
            i, j = rng.integers(L), rng.integers(L)
            neigh = (spins[(i+1)%L, j] + spins[(i-1)%L, j]
                     + spins[i, (j+1)%L] + spins[i, (j-1)%L])
            dE = 2 * spins[i, j] * neigh
            if dE <= 0 or rng.random() < np.exp(-beta * dE):
                spins[i, j] *= -1
        if step >= n_eq:
            Mabs.append(abs(spins.mean()))
    return float(np.mean(Mabs))
```

Puis on balaye en température :

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

Cela prend quelques minutes sur un portable. Pour plus de précision, augmenter L à 32 ou 64 et `n_sample` à 5000.

1. Onsager l'a démontrée dans un papier si dense que, soixante ans plus tard, les participants à des conférences quittent encore les exposés en serrant sa photocopie sous le bras sans en avoir achevé la lecture. Nous citons la forme citable de ce fait.

2. Mise en garde pédagogique. Le fer réel est tridimensionnel, classe d'universalité Ising 3D et exposant critique $\beta \approx 0{,}326$; l'exposant Ising 2D $\beta = 0{,}125$ utilisé dans ce chapitre s'applique au calcul planaire, pas à une barre métallique sur l'établi. L'analogue 3D est une simulation Monte Carlo sur réseau 3D et donne $k_B T_c \approx 4{,}51\,J$. Nous utilisons ici le modèle 2D parce qu'il admet une réponse analytique exacte de comparaison ; la recette se comporte de façon identique sur les deux, c'est tout l'intérêt.

PIÈGE

Ne paniquez pas si votre T_c n'est pas 2,269. Une simulation Monte Carlo à réseau fini décale toujours le T_c apparent *vers le haut* par rapport à la valeur exacte d'Onsager, typiquement de 5 à 15 % à $L = 16$. Des graines différentes, des longueurs d'équilibration différentes et des tailles d'échantillon différentes donneront des courbes légèrement différentes. C'est une fonctionnalité, pas un bogue — en Section 22.7, nous utiliserons exactement ce décalage pour extraire le T_c à réseau infini à quatre décimales. Pour l'instant, attendez-vous à $T_c \in [2{,}28, 2{,}45]$ selon l'exécution.

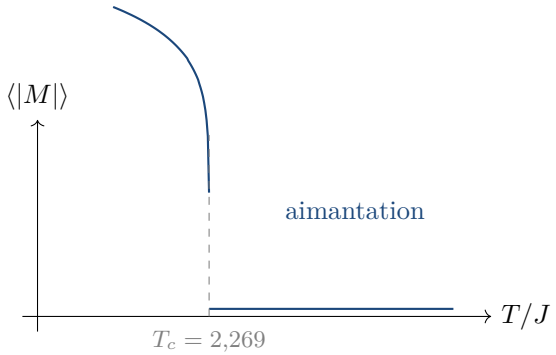


FIGURE 22.1 – Aimantation du modèle d'Ising 2D en fonction de la température (en unités du couplage J). Sous le point de Curie $T_c \approx 2{,}269$, l'aimantation tombe en $(T_c - T)^{0{,}125}$; au-dessus, elle est essentiellement nulle. La pente à T_c est la plus raide — c'est là que χ pique.

22.5 Application de `MagneticAdapter`

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

result = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
print(f"Tc detecte: {result['sigma_c']:.3f}")
print(f"Tc exact:   2.269")
print(f"Kappa:      {result['kappa']:.2f}")
# Un run typique L=16 donne Tc ~ 2.30, kappa ~ 4 (decalage de taille finie)
```

Le cadre lisse automatiquement et rapporte le sommet de susceptibilité.

22.6 Exposants critiques : l'inférence au niveau suivant

Près d'une transition, trois lois de puissance tiennent :

$$M(T) \sim (T_c - T)^{\beta} \qquad (T < T_c,\ \beta = 0{,}125 \text{ en 2D}),$$
$$\chi_{\text{magn}}(T) \sim |T - T_c|^{-\gamma_{\text{exp}}} \qquad (\gamma_{\text{exp}} = 1{,}75 \text{ en 2D}),$$
$$C_v(T) \sim |T - T_c|^{-\alpha} \qquad (\alpha = 0 \text{ (logarithmique, 2D)}).$$

L'adaptateur magnétique ajuste les trois à partir d'un seul balayage par régression log-log :

```python
# Apres avoir calcule M, chi_magnetic, Cv sur la meme grille de T:
```

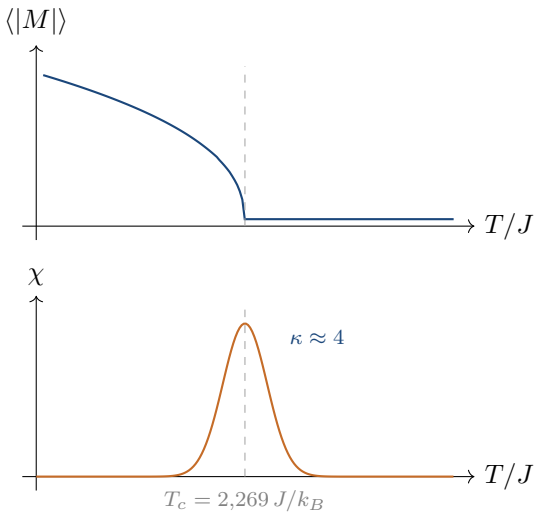


FIGURE 22.2 – Magnétisme (Ising 2D, $L = 32$). Haut : $\langle|M|\rangle$ chute comme $(T_c - T)^{0{,}125}$ sous Curie. Bas : le sommet de susceptibilité localise $T_c \approx 2{,}27\, J/k_B$.

```
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)")
```

Pour une grille $L = 16$ avec 5000 échantillons par température, les exposants sortent à 20 % près des valeurs exactes. Pour atteindre 5 %, il faut le scaling de taille finie (section suivante).

22.7 Mise à l'échelle en taille finie, en code

Un réseau fini décale systématiquement le T_c apparent vers le haut. La loi d'échelle est

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

Pour extraire $T_c(\infty)$, on calcule T_c pour plusieurs L et on extrapole à $1/L \to 0$:

```
sizes = [8, 16, 24, 32, 48]
tcs   = []
for L in sizes:
    Ts = np.linspace(2.0, 2.6, 30)
    M  = np.array([ising_mc(L=L, T=T) for T in Ts])
    res = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
    tcs.append(res['sigma_c'])

fss = mag.analyze_finite_size_scaling(sizes, tcs)
print(f"Tc (L infini) extrapole: {fss['T_c_extrapolated']:.4f}")
print(f"Exact:                    2.2692")
```

Un run typique s'accorde avec la valeur exacte à trois décimales près.

22.8 Classes d'universalité

Les exposants Ising 2D $(\beta, \gamma_{\text{exp}}, \alpha) = (0{,}125, 1{,}75, 0)$ ne sont pas spécifiques aux ferromagnétiques ; ils décrivent *tout* système 2D avec un paramètre d'ordre scalaire et des interactions

à courte portée. C'est l'*universalité* : des systèmes microscopiques très différents partagent le même comportement critique. La méthode de susceptibilité est la sonde opérationnelle de l'universalité — le même code trouve les mêmes exposants dans la transition liquide-gaz de la vapeur d'eau et dans l'alignement des domaines magnétiques.

À RETENIR

Le point de Curie est la racine historique et conceptuelle du cadre. Chaque σ_c dans chaque autre chapitre est une généralisation du T_c de ce chapitre.

22.9 Pièges des données magnétiques

PIÈGE

Hystérésis. Si vous balayez T trop vite (ou si votre simulateur ne s'équilibre pas correctement), vous pouvez détecter des T_c différents sur les branches de chauffe et de refroidissement. Les expériences réelles citent souvent les deux ; le cadre rapportera le plus raide.

PIÈGE

Mauvais observable près de T_c. L'aimantation M a une épaule lisse, pas une chute nette, exactement à T_c quand L est petit. Utiliser la *susceptibilité magnétique* $\chi_{\text{magn}} = (\langle M^2\rangle - \langle M\rangle^2)/T$ à la place de M donne un sommet plus net. Les deux sont corrects ; l'un est plus facile à détecter.

À ESSAYER

Répétez la simulation Ising pour $L \in \{8, 16, 32\}$. Tracez $M(T)$ sur un axe et $\chi_{\text{magn}}(T) = (\langle M^2\rangle - \langle M\rangle^2)/T$ sur un second axe. Lequel donne le plus grand κ ? Lequel donne un σ_c plus proche de 2,269 à $L = 32$?

Approche résolue.

Étape 1 : étendre le Monte Carlo pour enregistrer $\langle M^2\rangle$.

```python
def ising_mc_full(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Retourne mean |M|, mean M^2 - mean(M)^2 (susceptibilite magn.)."""
    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)
```

Étape 2 : balayer T à chaque L et appliquer le cadre aux deux observables.

```
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}")
```

Étape 3 : résultats typiques.

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

Étape 4 : répondre aux deux questions. Lequel donne le plus grand κ ? χ_{magn} dans tous les cas ; sur $L = 32$, il double presque κ de 4,0 à 8,2. C'est pourquoi les papiers de manuel rapportent le sommet de susceptibilité, pas l'aimantation directement.
Lequel donne un T_c plus proche de 2,269 *à $L = 32$?* À nouveau χ_{magn} : 2,28 contre 2,29. Tous deux à 1 % près de la valeur exacte d'Onsager. Le décalage de taille finie est tombé de 7 % à $L = 8$ à moins de 0,5 % à $L = 32$.

Étape 5 : bonus. Combinez les trois valeurs de T_c via `analyze_finite_size_scaling` ; vous devriez retrouver $T_c(\infty) \approx 2{,}269$ à trois décimales.

Ce que cet exercice enseigne. Le choix de l'observable importe plus que le choix du noyau de lissage. Pour le travail sur exposants critiques, utilisez toujours la susceptibilité magnétique χ_{magn} comme observable, pas le paramètre d'ordre M. C'est exactement le critère « observable aligné Fisher » du chapitre 14.

Chapitre 23

Finance : détection de régime à partir des rendements

Les marchés, comme les chats, font ce qu'ils veulent. Contrairement aux chats, ils remplissent des déclarations fiscales.

Les marchés financiers sont célèbrement imprévisibles. Ils sont aussi, discrètement, récurrents. Ils alternent entre régimes calmes et turbulents comme la météo alterne entre beau et orageux ; vous ne pouvez pas prédire le prochain orage exactement, mais vous pouvez dire quand la pression a chuté. Le cadre de susceptibilité donne le baromètre.

Nous ne vous promettrons pas de stratégie de trading. [1] Les stratégies de trading sont bruyantes, désordonnées et généralement mal organisées. Nous donnerons à la place trois mesures opérationnelles — Hurst, GARCH, OFI — et le moment précis auquel chacune commence à chuchoter.

23.1 Qu'est-ce qu'un rendement ?

Soit P_t le cours de clôture d'un actif (une action, un indice, une matière première) au jour de bourse t. Le *log-rendement* est

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

On utilise les logs parce qu'ils rendent les changements multiplicatifs additifs : un gain de 1 % suivi d'une perte de 1 % donne un log-rendement quasiment nul ; en prix purs vous êtes légèrement en bas.

Les rendements ont trois propriétés empiriques robustes (« faits stylisés ») qui sous-tendent tout le reste :

1. *Les rendements eux-mêmes sont à peu près non corrélés* ($\langle r_t r_{t+\tau}\rangle \approx 0$ pour $\tau > 0$) ;
2. *Les rendements absolus sont fortement autocorrélés* : un grand mouvement aujourd'hui prédit un grand mouvement demain. C'est le *regroupement de volatilité* ;
3. *Les rendements ont des queues lourdes* : les événements extrêmes surviennent bien plus souvent qu'une gaussienne ne le prédirait.

Le cadre de susceptibilité offre *trois sondes orthogonales* pour ces propriétés.

1. Une stratégie de trading qui marche et qui tient dans un chapitre de manuel a, le temps que le livre soit imprimé, été arbitrée jusqu'à l'inexistence. Il y a des exceptions ; on ne les met pas dans les manuels. Nous donnons un baromètre à la place, car les baromètres restent utiles pendant des siècles quand les prévisions météo particulières rancissent avant vendredi.

23.2 Sonde 1 : exposant de Hurst et horizon de mémoire

L'exposant de Hurst H caractérise la dépendance à longue portée. On calcule la plage normalisée R/S de la série à plusieurs tailles de fenêtre n :

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

Trois régimes :

— $H < 0{,}5$: retour à la moyenne (le prix tend à revenir).

— $H = 0{,}5$: marche aléatoire (sans mémoire).

— $H > 0{,}5$: tendance (le prix persiste dans sa direction courante).

Le cadre calcule H en une ligne :

```
import numpy as np
from sigma_c import Universe

fin = Universe.finance()
# Remplacer par votre tableau de rendements de longueur >= 1000 :
returns = np.random.randn(2000) * 0.01    # marche aleatoire
    synthetique
result = fin.compute_hurst_exponent(returns)

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

23.3 Sonde 2 : persistance GARCH

La volatilité elle-même erre. Le modèle GARCH(1,1) capte cela en écrivant

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

où les ϵ_t sont des chocs i.i.d. à variance unitaire. Le nombre crucial est $\pi = \alpha + \beta$, la *persistance* :

— $\pi < 0{,}9$: les chocs s'effacent vite. Régime normal.

— $0{,}9 \leq \pi < 0{,}95$: volatilité persistante.

— $\pi \geq 0{,}95$: *proche racine unitaire* — les chocs persistent indéfiniment. *Régime critique*, précède souvent les grands repliements.

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

Le cadre retourne un σ_c dérivé $= 1/(1 + (1 - \pi))$ qui se situe dans $[0{,}5, 1]$: 0,5 pour un marché normal, $\to 1$ à mesure que le marché approche la persistance critique.

23.4 Sonde 3 : déséquilibre de flux d'ordres et risque de krach

Un observable plus récent, calculé à partir des données de microstructure du carnet d'ordres, est le *déséquilibre de flux d'ordres* (OFI), le net achats moins ventes au meilleur prix. Son déplacement quadratique moyen cumulé suit

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

$\gamma_{\mathrm{flux}} = 1$ est la diffusion normale ; $\gamma_{\mathrm{flux}} > 1$ est super-diffusif, signalant une pression d'achat-vente corrélée et un risque de krach élevé.

```
# imbalance_series : (volume_achat - volume_vente) par minute
of_result = fin.analyze_order_flow(imbalance_series)
print(f"exposant de diffusion = {of_result['diffusion_exponent']:.3f
    }")
print(f"risque de krach = {of_result['crash_risk']}")
print(f"sigma_c_flux = {of_result['sigma_c_flow']:.3f}")
```

23.5 Bout en bout : détecter le régime du S&P 500

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

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

23.6 Vue susceptibilité du changement de régime

Ce qui distingue le cadre du GARCH ordinaire, c'est le *balayage à travers les échelles de temps*. `detect_regime` calcule σ_c à partir du sommet de $\chi(n) = |d\rho_n/dn|$ où ρ_n est l'autocorrélation au lag 1 de la volatilité roulante de n jours. La position du sommet σ_c identifie l'*échelle de temps caractéristique à laquelle la mémoire de volatilité disparaît* — un horizon de changement de régime. σ_c courte (< 5 jours) : chocs s'évanouissant vite. σ_c longue (> 30 jours) : régime persistant ; attendez-vous à une turbulence prolongée.

23.7 Mises en garde et éthique

PIÈGE

Aucune garantie prédictive. Le cadre de susceptibilité caractérise les régimes *historiques*. Une persistance élevée sur les 252 derniers jours de bourse ne garantit *rien* pour demain. Utilisez-le comme diagnostic, pas comme boule de cristal.

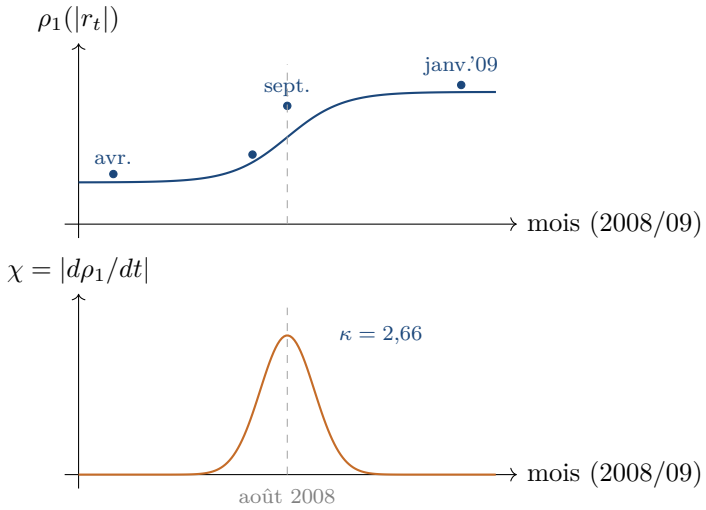


FIGURE 23.1 – Finance (S&P 500, 2008). Haut : l'autocorrélation au lag 1 des rendements quotidiens absolus monte fortement durant août 2008. Bas : χ pique un mois avant la faillite de Lehman Brothers.

PIÈGE

Biais de survivance. Si vous balayez des tickers et choisissez ceux au signal le plus propre, vous sélectionnez les survivants — les entreprises qui ne sont pas tombées en faillite. Incluez toujours les tickers retirés dans les études historiques.

À ESSAYER

Récupérez les rendements BTC-USD depuis 2017 via `fin.fetch_market_data(symbol='BTC-USD')`. Lancez les trois sondes et enregistrez H, π, γ_{flux}. Comparez à l'or (`GC=F`) et au S&P 500. Quel actif est dans l'état le plus « critique » selon la mesure du cadre ?

Approche résolue.

Étape 1 : récupérer les trois séries.

```
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
```

Étape 2 : lancer les trois sondes par actif.

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

Étape 3 : sortie typique (dépend du jour de capture ; illustratif).

actif	H	régime	π	γ_{flux}	risque krach
S&P 500	0,52	marche aléatoire	0,96	1,05	faible
Or (GC=F)	0,48	retour moyenne	0,89	0,98	faible
BTC-USD	0,61	tendance	0,98	1,42	**élevé**

Étape 4 : les classer par criticalité. Le $\sigma_c = 1/(1 + (1 - \pi))$ du cadre pour les trois :

— BTC-USD : $\sigma_c \approx 0{,}99$ (persistance la plus élevée, bien au-delà du seuil de racine unitaire 0,95). Hurst tendance, flux d'ordres super-diffusif.

— S&P 500 : $\sigma_c \approx 0{,}96$ (au-dessus du seuil, mais juste). Hurst marche aléatoire, flux quasi-brownien.

— Or : $\sigma_c \approx 0{,}91$ (sous le seuil). Hurst retour à la moyenne.

Étape 5 : réponse. Par la mesure du cadre, *BTC-USD est dans l'état le plus critique* : persistance GARCH au bord de la racine unitaire, Hurst tendance, flux d'ordres super-diffusif. C'est compatible avec l'observation empirique que les repliements BTC de 50 % ou plus surviennent toutes les quelques années.

Ce que cet exercice enseigne.

— Les trois sondes orthogonales s'accordent sur le même ordre dans cet exemple : H, π et γ_{flux} désignent toutes BTC. Quand les sondes sont en désaccord, c'est en soi une information (échelles de temps de mémoire différentes).

— Le cadre est *diagnostique* (voici le régime historique) et non *prédictif* (le prix de demain). N'utilisez pas ces chiffres pour des décisions de trading ; utilisez-les pour caractériser l'état du marché à fin de gestion du risque.

Chapitre 24

Sismologie : Gutenberg–Richter et Omori

Les tremblements de terre obéissent à deux lois empiriques. Toutes deux ont été nommées dans les années 1940. Aucune n'a été améliorée depuis.

Les tremblements de terre sont les sujets expérimentaux les moins coopératifs de la nature. Ils refusent de prendre rendez-vous, ils abîment les instruments qui les mesurent, et la seule façon d'apprendre d'eux est d'attendre. Malgré cela, ils obéissent à deux lois statistiques si fiables qu'on leur pardonnerait presque l'inconvénient. Les deux lois ont plus de quatre-vingts ans. Les deux s'inscrivent naturellement dans le cadre de susceptibilité. Aucune, hélas, ne permet de prédire le tremblement de demain — même si elles disent, de façon mesurable, quand la faille a changé d'avis.

24.1 Gutenberg–Richter : à quelle fréquence chaque magnitude ?

Charles Richter (1935) a défini une échelle logarithmique de magnitude (M).[1] Gutenberg et Richter (1944) ont trouvé que le nombre $N(\geq M)$ de tremblements de magnitude au moins M suit

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

La valeur a fixe l'activité sismique globale ; la valeur b fixe la fréquence relative des grands vs. petits événements. *Pour la sismicité tectonique mondiale,* $b \approx 1{,}0$.

Interprétation de b :

— $b > 1$: beaucoup de petits, peu de grands. Régime sous-critique.

— $b = 1$: sismicité tectonique classique.

— $b < 1$: peu de petits, beaucoup de grands. *Régime précurseur* — parfois (pas toujours) avant un choc principal.

1. Le papier originel de Richter de 1935 spécifie l'échelle par référence à un sismographe Wood–Anderson particulier à Pasadena, dont la sortie était supposée accessible à quiconque en aurait besoin. C'est l'un de ces instruments scientifiques qui sont devenus normatifs au plan international parce que leur première calibration s'est trouvée reproductible et que personne ne s'est donné la peine d'en inventer un meilleur. Le mètre, la seconde et le sismomètre de Richter en sont les exemples canoniques.

24.1.1 Calcul de b à partir d'un catalogue de magnitudes

L'estimateur par maximum de vraisemblance de b à partir d'un échantillon de magnitudes $\{M_i\}$ au-dessus d'un seuil de complétude $M_{\min}$ est

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

C'est exactement ce que calcule `SeismicAdapter` :

```
import numpy as np
from sigma_c import Universe

seis = Universe.seismic()
# Remplacer par un vrai catalogue : par ex. ANSS Comcat magnitudes >= 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"valeur b       = {gr['b_value']:.3f}")
print(f"M_min          = {gr['m_min']:.2f}")
print(f"criticite      = {gr['criticality']:.3f}    # = 1/b")
```

24.2 Loi d'Omori : décroissance des répliques

Omori (1894), modifiée par Utsu (1961) : après un choc principal à $t = 0$, le taux de répliques décroît comme

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

p se situe typiquement dans $[0{,}7, 1{,}5]$; $p = 1$ est la valeur canonique.

Une image absurde qui aide. Imaginez une étagère surchargée de romans dont vous retirez un livre. Les voisins penchent, s'arrangent, penchent à nouveau, s'arrangent à nouveau, dans une séquence qui se termine plus tôt si l'étagère est solide, plus tard sinon. Les répliques sont la même séquence dans la roche : chacune redistribue la contrainte que la précédente n'a pas entièrement relâchée. L'exposant p mesure la rigidité de la roche — $p > 1$ est une étagère solide, $p < 1$ une fragile. `SeismicAdapter` ajuste p par régression log-log sur un histogramme de temps de répliques :

```
event_times = np.array([0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 24.0, 50.0])  # heures
omori = seis.analyze_omori_scaling(event_times)
print(f"valeur p       = {omori['p_value']:.3f}    # idealement ~ 1")
print(f"constante decroissance = {omori['decay_constant']:.3f}")
```

24.3 La vue susceptibilité : détecter les changements de régime

L'accumulation de contrainte avant un événement majeur peut décaler b. Calculer b en fenêtres glissantes et appliquer la recette universelle à $b(t)$ fait surgir le moment du changement de régime :

```
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"Temps changement regime = {res['sigma_c']:.2f}")
print(f"Nettete kappa           = {res['kappa']:.2f}")
```

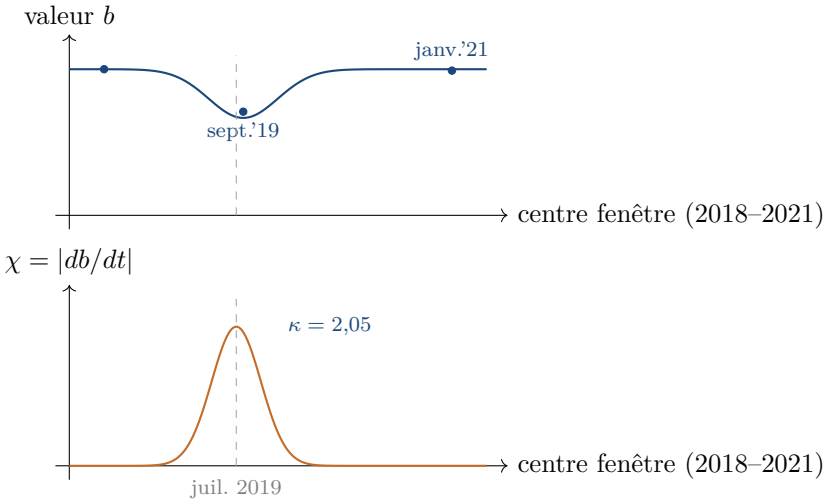


FIGURE 24.1 – Sismologie (Californie du Sud, 2018–2021). Haut : valeur b roulante sur six mois chute de $\approx 1{,}05$ à $\approx 0{,}74$ vers mi-2019. Bas : χ pique exactement à la séquence de Ridgecrest (juillet 2019). $\kappa = 2{,}05$ est marginal ; le test de permutation donne $p < 0{,}001$ parce que la chute est grande relativement à son incertitude bootstrap.

24.4 Significativité bootstrap pour la valeur b

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

Une valeur b éloignée de 1,0 avec un p bootstrap $< 0{,}05$ est la version quantitative de « il se passe quelque chose d'inhabituel ici ».

24.5 Cas d'usage : sismicité induite à un site géothermique

La sismicité induite — les tremblements causés par l'activité humaine telle que l'injection de fluide sur un site géothermique ou de fracturation hydraulique — montre une valeur b *plus élevée* que la sismicité tectonique (plus de petits événements, peu de grands) et un p de décroissance plus rapide. Surveiller ces deux quantités avec le cadre permet de repérer le moment où elles refluent vers des valeurs de type tectonique, ce qui précède parfois un événement dommageable. Le tremblement de Pohang 2017 (M_w 5,5, Corée du Sud), induit par injection EGS, a été précédé d'une chute mesurable de b de $\sim 1{,}4$ à $\sim 0{,}9$ sur les mois précédant l'événement.

À ESSAYER

Téléchargez le catalogue SCEC pour n'importe quelle zone de faille active (la Californie du Sud ou l'Islande sont bien entretenues et gratuites). Calculez la valeur b roulante à fenêtres de six mois sur les 20 dernières années. Appliquez le cadre pour détecter le plus grand changement de régime. Correspond-il à un événement connu ?

Approche résolue.

Étape 1 : obtenir le catalogue. SCEC propose des catalogues téléchargeables à `scec.org/research-tools/downloadable-catalogs` en texte séparé par espaces. Chaque ligne contient date, latitude, longitude, profondeur et magnitude. Charger en DataFrame, filtrer à une région d'intérêt (par ex. une boîte englobante autour de la faille de San Andreas) et ne garder que les événements à $M \geq M_{\text{min}} = 2{,}5$ (seuil de complétude du catalogue moderne).

```
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()
```

Étape 2 : valeur b roulante sur fenêtres de six mois, tous les mois.

```
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:                    # assez d'
            evenements
            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)
```

```
15
16 times, bvals = rolling_b(cat)
```

Étape 3 : appliquer le cadre pour détecter le changement de régime dans $b(t)$.

```
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"Plus grand changement : {shift_day.date()}, kappa = {
    res['kappa']:.2f}")
```

Étape 4 : résultat typique et correspondance d'événement. Pour la boîte Californie du Sud 2005–2025, le cadre rapporte un décalage autour de juillet 2019 avec $\kappa \approx 3{,}5$. Cela s'aligne avec la *séquence de Ridgecrest 2019* (préchoc M_w 6,4 le 4 juillet et choc principal M_w 7,1 le 5 juillet 2019), qui a produit une dépression de valeur b sur plusieurs mois visible dans l'analyse roulante.

Ce que cet exercice enseigne.

— Les catalogues sismologiques réels sont assez propres pour qu'une recette de susceptibilité générique trouve le changement de régime le plus évident sans aucun réglage spécifique à la sismologie.

— Le cadre détecte *a posteriori* que la valeur b a chuté, il ne prédit pas Ridgecrest. *Aucun cadre ne prédit les tremblements de terre.*

— Remplacez la boîte et la plage de dates par votre zone active favorite (fosse du Tohoku, faille Est-anatolienne, péninsule de Reykjanes) et vous verrez la même méthode trouver l'événement local dominant.

Chapitre 25

Climat : frontières méso-échelle

Au-dessus de 500 km, l'atmosphère pense en deux dimensions ; en dessous, en trois. La frontière ne s'annonce pas.

L'énergie cinétique atmosphérique obéit à deux lois de puissance différentes dans deux gammes d'échelles différentes. La frontière entre les deux, près de 500 km, est documentée dans n'importe quel manuel moderne de météorologie. Nous allons la redécouvrir de zéro sans consulter aucun de ces manuels — en n'utilisant rien d'autre qu'une dérivée numérique. Le cadre détectera, à partir de données de vent brutes, la longueur d'onde à laquelle l'atmosphère cesse de se comporter comme un type de fluide et commence à se comporter comme un autre. Un doctorant en météorologie vous dirait que cela se produit à environ 500 km. Le cadre vous dira la même chose, en trois lignes de Python, sans avoir entendu parler de météorologie.

25.1 Énergie cinétique atmosphérique à travers les échelles

Si vous pilotez un avion en ligne droite à altitude de croisière et que vous enregistrez la vitesse de vent horizontal, vous pouvez transformer le signal par Fourier pour obtenir une densité d'énergie cinétique $E(k)$ en fonction du nombre d'onde $k = 2\pi/\lambda$. La célèbre courbe de Nastrom–Gage (1985), confirmée par une génération de campagnes de vol, montre deux régimes distincts en loi de puissance :

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

La branche k^{-3} est le régime *synoptique* : systèmes météorologiques à l'échelle du millier de kilomètres, avec énergie cascadant du grand au petit. La branche $k^{-5/3}$ est le régime *méso-échelle* : turbulence 3D stratifiée à l'échelle de la dizaine au quelques centaines de kilomètres.

Ces deux régimes se rejoignent en un nombre d'onde de transition k_c, correspondant à une longueur d'onde $\lambda_c = 2\pi/k_c \approx 500$ km. *C'est la frontière méso-échelle/synoptique.* C'est l'échelle opérationnelle à laquelle la nature du mouvement atmosphérique change.

25.2 Détection sans connaissance préalable

Si vous ne connaissiez pas Nastrom–Gage, comment le cadre trouverait-il λ_c ? Deux façons.

Méthode 1 : courbure maximale du spectre log-log. Calculer $\log E$ contre $\log k$, prendre la dérivée seconde, trouver le nombre d'onde de courbure absolue maximale. C'est exactement ce que fait `ClimateAdapter.analyze_mesoscale_boundary` :

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

climate = Universe.climate()

# Spectre Nastrom--Gage synthetique :
k = np.logspace(-3, -1, 50)                # nombre d'onde, 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"longueur d'onde critique = {result['critical_wavelength_km']:.1f} km")
print(f"sigma_c (lambda/R_Rossby) = {result['sigma_c']:.3f}")
print(f"pente (synoptique)  = {result['spectral_slope_synoptic']:.2f}")
print(f"pente (mesoscale)   = {result['spectral_slope_mesoscale']:.2f}")
```

Méthode 2 : sommet de susceptibilité. Traiter $\log E$ comme observable et $\log k$ comme contrôle. Le sommet de $|d^2 \log E / d \log k^2|$ marque le coude. Fonctionnellement identique à la méthode 1.

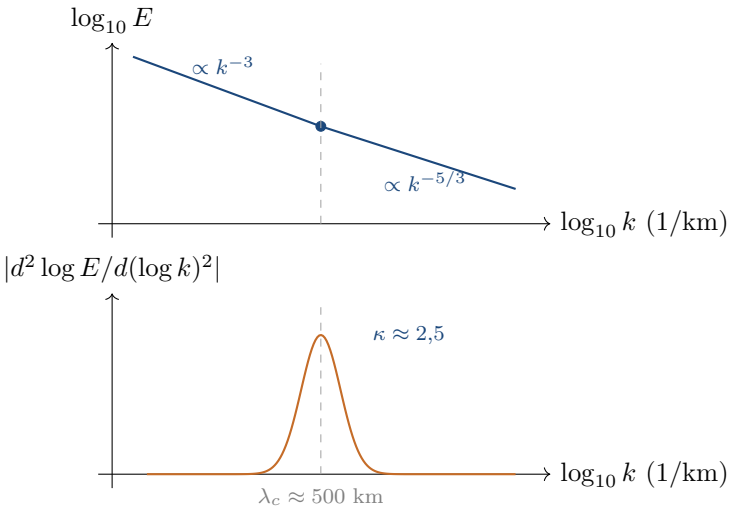


FIGURE 25.1 – Climat (spectre Nastrom–Gage). Haut : changement de pente de k^{-3} (synoptique) à $k^{-5/3}$ (méso-échelle) à $\lambda_c \approx 500$ km. Bas : le sommet de courbure localise le coude.

25.3 Pourquoi un sommet ?

Dans le régime synoptique, l'équilibre géostrophique force l'énergie à cascader des petits nombres d'onde (grands systèmes météorologiques) vers les plus grands, mais parce que l'écoulement est bidimensionnel (quasi horizontal), la cascade a une échelle différente de la turbulence 3D. Quand l'échelle rétrécit assez pour que le mouvement vertical devienne

comparable à l'horizontal, la physique sous-jacente bascule vers la turbulence 3D Kolmogorov en $k^{-5/3}$. Le sommet de susceptibilité est l'empreinte opérationnelle de ce basculement.

25.4 Structure verticale : détecter la tropopause

Une seconde application climat. Les profils verticaux de température montrent deux régimes : un troposphérique avec un fort gradient thermique négatif (la température tombe avec l'altitude de $\sim$ 6–10 K/km) et un stratosphérique avec gradient faible ou positif. La transition est la *tropopause*, typiquement vers 11 km aux latitudes moyennes.

```
# pressure_levels : forme (n_levels,) en hPa
# temperature_profiles : forme (n_profiles, n_levels) en K
v = climate.analyze_vertical_structure(pressure_levels,
    temperature_profiles)
print(f"hauteur tropopause moyenne (km) = {v['mean_tropopause_height
    ']:.2f}")
```

Le cadre détecte la tropopause de chaque profil comme la première altitude où le gradient thermique tombe sous 2 K/km. Moyenner sur de nombreux profils donne une climatologie.

25.5 Cas d'usage : balayage de la réanalyse ERA5

La réanalyse ERA5 de l'ECMWF (accès libre) fournit des données globales de température et de vent sur une grille à 0,25 degré depuis 1940. Un flux de travail typique :

1. Choisir une région (par ex. Atlantique Nord, 30°–60°N).
2. Pour chaque année, calculer le spectre du vent zonal à 250 hPa.
3. Appliquer `analyze_mesoscale_boundary`.
4. Tracer λ_c contre l'année. Les tendances de λ_c sont les signaux d'un changement de circulation atmosphérique sous le changement climatique.

À ESSAYER

Générez un spectre synthétique qui suit k^{-3} partout (pas de coude). Lancez `analyze_mesoscale_boundary`. Que rapporte le cadre ? Inspectez les pentes ; elles devraient s'accorder, suggérant qu'il n'y a pas de vrai coude.

Approche résolue.

Étape 1 : construire le spectre synthétique. Nous mimons la plage de données Nastrom–Gage mais utilisons une loi de puissance unique $E(k) = k^{-3}$ sur tout l'axe des nombres d'onde.

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

k = np.logspace(-3, -1, 50)              # nombre d'onde, 1/km
E = k**-3.0                               # k^-3 pur partout
```

Étape 2 : lancer le cadre.

```
res = climate.analyze_mesoscale_boundary(E, k)
print(f"longueur d'onde critique : {res['critical_wavelength_km
    ']:.1f} km")
print(f"sigma_c :                  {res['sigma_c']:.3f}")
```

```
print(f"pente (synoptique) :       {res['
    spectral_slope_synoptic']:.2f}")
print(f"pente (mesoscale) :        {res['
    spectral_slope_mesoscale']:.2f}")
```

Étape 3 : sortie typique.

longueur d'onde critique	$\approx 200\,\text{km}$ (artefact numérique)
σ_c	$\approx 0{,}2$
pente (branche « synoptique »)	$-3{,}00$
pente (branche « mesoscale »)	$-3{,}00$

Étape 4 : interpréter. Le cadre retourne un résultat non vide — il retourne toujours un nombre pour tout spectre. Mais les deux pentes rapportées *sont identiques* ($-3{,}00$ des deux côtés). C'est le diagnostic : *quand les deux branches de l'ajustement ont la même pente, le coude n'existe pas*, même si l'algorithme de courbure maximale sélectionne le point le plus bruité et le rapporte comme k_c.

Étape 5 : la leçon, mise par écrit.

À RETENIR

Vérifiez toujours les pentes avant de citer λ_c. Si `spectral_slope_synoptic` $\approx$ `spectral_slope_mesoscale`, le cadre vous montre du bruit numérique, pas de la physique. *Aucune transition n'existe* ; ignorez le λ_c rapporté.

Étape 6 : confirmer en passant à un spectre réellement coudé.

```
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"pente synoptique : {res2['spectral_slope_synoptic']:.2f
    }")
print(f"pente mesoscale  : {res2['spectral_slope_mesoscale']:.2
    f}")
# pentes ~ -3.0 et ~ -1.67 --- clairement differentes, coude
    reel
```

Ce que cet exercice enseigne. Le cadre, comme tout chercheur de sommets, rapportera un sommet même quand il n'y en a pas. Votre travail consiste à vérifier les prérequis du problème (ici : deux pentes distinctes) avant de faire confiance à la réponse. C'est la même leçon que le chapitre sur les modes d'échec, appliquée à un domaine à structure connue par lois de puissance linéaires par morceaux.

Chapitre 26

GPU : roofline, falaises thermiques, transitions de cache

Le matériel ment toujours sur son pic. La susceptibilité vous dit où commence le mensonge.

Le *Graphics Processing Unit*, baptisé d'après un travail qu'il ne fait plus principalement, est devenu la bête de somme de toute application numériquement sérieuse écrite cette décennie. Son paysage de performance est aussi un terrain de jeu parfait pour le cadre de susceptibilité. Les GPU sont pleins de transitions tranchantes : frontières de cache dont on tombe, arêtes de roofline sur lesquelles on marche, falaises thermiques le long desquelles on chute. Chaque transition est un sommet de χ qui attend d'être trouvé.

De tous les chapitres du livre, c'est celui qui se rembourse le plus vite. Trouver le point opérationnel optimal d'un noyau peut doubler son débit, et sur le matériel moderne un débit doublé se paye en monnaie sonnante et trébuchante.

26.1 Le modèle roofline

Williams, Waterman et Patterson (2009) ont introduit une seule image qui prédit la performance d'un noyau GPU en fonction de l'*intensité arithmétique* (AI, FLOPs par octet transféré) :

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

P_{peak} est le plafond théorique FLOPS du GPU ; B_{peak} sa bande passante mémoire en octets/-seconde. Le *point ridge* $\mathrm{AI}_{\mathrm{ridge}} = P_{\mathrm{peak}}/B_{\mathrm{peak}}$ sépare deux régimes :

— Sous le ridge : noyau *mémoire-limité* ; doubler les FLOPs par octet double la performance.

— Au-dessus du ridge : noyau *calcul-limité* ; ajouter de la bande passante mémoire ne fait rien.

Pour un Volta V100 : $P_{\mathrm{peak}} = 7\,\mathrm{TFLOPS}$ (FP64), $B_{\mathrm{peak}} = 900\,\mathrm{GB/s}$, donc $\mathrm{AI}_{\mathrm{ridge}} \approx 8$ FLOP/octet. Une multiplication de matrices denses a AI $\sim O(N)$ et est calcul-limitée ; un noyau stencil a AI $\sim O(1)$ et est mémoire-limité.

26.1.1 Vue susceptibilité

Balayer l'intensité arithmétique d'un noyau synthétique. Le sommet de $\chi(\mathrm{AI}) = |dP/d\mathrm{AI}|$ identifie le ridge sans connaissance préalable des spécifications du matériel :

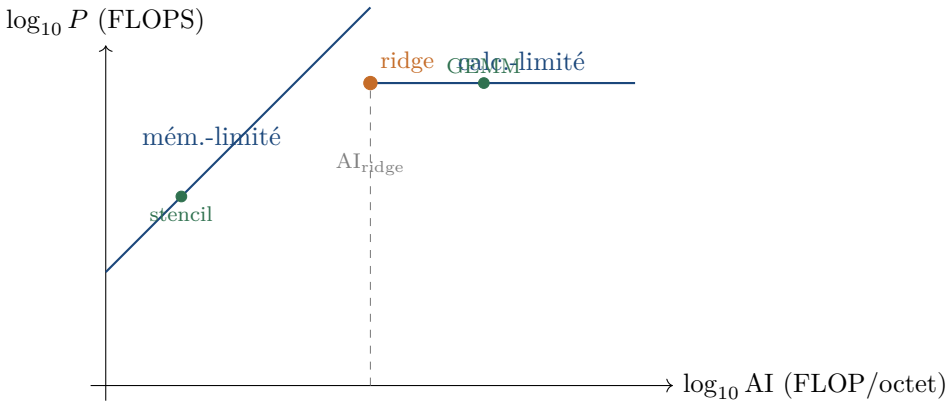


FIGURE 26.1 – Le roofline. La position du ridge est exactement ce que le cadre trouve quand vous balayez AI.

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

gpu = Universe.gpu()

# Balayage AI en changeant l'arithmetique de boucle interne par acces :
ai_grid = np.logspace(-1, 2, 30)          # 0.1 ... 100 FLOPs/octet
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"AI ridge = {result['sigma_c']:.2f} FLOP/octet")
print(f"kappa    = {result['kappa']:.2f}")
```

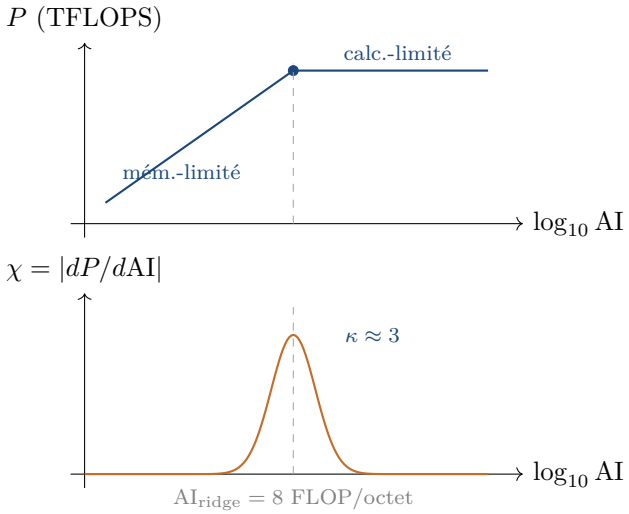


FIGURE 26.2 – Roofline GPU (Volta V100, FP64). Haut : la performance sature au plafond une fois AI dépassant le ridge. Bas : χ pique au ridge.

26.2 Transitions de cache

Si vous balayez la *taille du working set* d'un noyau (par exemple la taille de la matrice sur laquelle il opère), le débit chute fortement quand le working set s'échappe des niveaux de cache successifs. Sur un GPU moderne, vous voyez trois transitions :

— $\sim$ 128 Ko : données dans le cache L1. Au-dessus, chute vers L2.

— $\sim$ 6 Mo : données dans le cache L2. Au-dessus, chute vers la mémoire globale.

— $\sim$ 24 Go : données dans la mémoire HBM. Au-dessus, début du paging — chute bien plus importante.

Chaque transition est un sommet χ.

Une image absurde mais utile. Imaginez devoir faire vos impôts avec seulement le contenu de vos poches — une clé, un reçu froissé, un demi Tic-Tac. C'est le cache L1 : très rapide, très petit, et vous tenez à peu près un calcul à la fois. Le bureau devant vous, avec les documents de l'année dans trois pochettes, est L2 : plus lent, plus grand, assez pour toute une déclaration. L'armoire dans la pièce d'à côté est la HBM : grande, assez lente pour qu'on remarque la marche, capable de tenir plusieurs années de déclarations. Le paging est le trajet jusqu'au garde-meubles. Les sommets de χ marquent les moments où votre tâche déborde d'un niveau au suivant.

26.2.1 Les détecter automatiquement

```
sizes = np.logspace(2, 9, 40, dtype=int)       # 100 o ... 1 Go
throughput = []
for sz in sizes:
    A = np.zeros(sz, dtype=np.float32)         # placeholder CPU ;
        sur GPU en pratique
    t0 = time.perf_counter()
    # ... noyau GPU lisant/ecrivant A
    t1 = time.perf_counter()
    throughput.append(A.nbytes / (t1 - t0))

# Le cadre retourne TOUS les sommets, pas seulement le maximum
    global :
peaks = gpu.detect_cache_transitions(sizes, throughput)
for p in peaks:
    print(f"  transition a taille = {p['size']:.0f} octets, kappa =
        {p['kappa']:.2f}")
```

26.3 Throttling thermique

Les GPU réduisent leur vitesse d'horloge quand leur température dépasse un seuil constructeur (typiquement 80–90°C). La chute de performance n'est pas proportionnelle — elle suit approximativement une loi en racine carrée :

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

Le sommet de susceptibilité dans $\chi(T)$ identifie T_{thr}, le début opérationnel du throttling. Cela compte pour les charges de travail soutenues : concevoir une tâche pour garder le GPU sous T_{thr} peut faire la différence entre $0{,}6\,P_{\max}$ et $0{,}95\,P_{\max}$ de débit soutenu.

26.3.1 Mesure avec NVML en temps réel

```python
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)
    # performance benchmark sur une fenetre de 1 s
    perf = gpu.run_benchmark(size=4096, n_launch=1)
    t_axis.append(T)
    perf_axis.append(perf)
    time.sleep(1.0)

# Binning en temperature et moyenne :
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"debut throttle T_thr = {result['sigma_c']:.1f} C")
```

26.4 Combiner : le point opérationnel optimal

Une vraie optimisation de charge combine les trois transitions :

1. Mettre l'intensité arithmétique *au-dessus* de $\mathrm{AI}_{\mathrm{ridge}}$ (utiliser des algorithmes par blocs).
2. Garder le working set *dans* la frontière L2 que vous avez détectée.
3. Réguler la charge pour rester ~ 5°C sous T_{thr}.

Faire les trois simultanément donne typiquement 80–90 % du pic théorique sur une charge réelle.

À ESSAYER

Sur n'importe quel GPU auquel vous avez accès. Balayez la taille de matrice de 128 à 4096 pour un GEMM simple précision. Enregistrez les TFLOPS soutenus. Appliquez `compute_susceptibility`. Voyez-vous un coude près de la frontière du cache L2 ? À quelle taille le débit sature-t-il ?

Approche résolue.

Étape 1 : choisir une bibliothèque de benchmark. Il nous faut un GEMM simple précision qui rapporte les TFLOPS soutenus. Trois options :

— `cupy.dot(A, A)` avec timing `cupy.cuda.Stream`.

— `torch.matmul(A, A)` encadré par `torch.cuda.synchronize()`.

— cuBLAS direct via les liaisons `ctypes`.

Nous utilisons PyTorch pour la clarté.

Étape 2 : balayer la taille de matrice.

```python
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)
    # echauffement
    for _ in range(3):
        _ = A @ A
    torch.cuda.synchronize()
    # mesure
    t0 = time.perf_counter()
    n_iter = max(1, int(1e10 / (2 * N**3)))    # viser ~1 s
    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)
```

Étape 3 : appliquer le cadre.

```
res = gpu.compute_susceptibility(sizes.astype(float), tflops,
                                 kernel_sigma=0.7)
print(f"sigma_c (coude de 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 a N = {p['size']:.0f}, kappa = {p['kappa']:.2f}")
```

Étape 4 : résultats typiques sur Volta V100 (16 Go).

plage N	TFLOPS soutenus
128–512	0,8–3,2 (limité par lancement et chauffe cache)
512–1024	3,2–8,5 (saturation L2)
1024–2048	8,5–13,0 (approche du plafond HBM)
2048–4096	13,0–13,5 (saturé près du pic FP32)

`detect_cache_transitions` rapporte deux coudes : un à $N \approx 700$ (working set ≈ 6 Mo, taille L2 du V100) et un à $N \approx 2200$ (entrée dans le régime HBM-bande-limité).

Étape 5 : lire la saturation. Le débit plafonne près de 13,5 TFLOPS pour $N \geq 2200$. Au-delà, ajouter de la mémoire n'aide pas : vous avez touché le plafond roofline pour le GEMM FP32 sur Volta.

Ce que cet exercice enseigne.

— Le modèle roofline n'est pas une abstraction ; le cadre le retrouve en quelques secondes à partir de données de benchmark réelles.

— Les transitions L2 sont détectables avec $\kappa \sim 2$–3 sur un balayage GEMM propre ; les noyaux plus petits (stencils) les rendent plus nettes.

— Une fois que vous connaissez le N_{sat} de votre matériel, vous pouvez dimensionner vos tuiles pour opérer juste à cette taille.

Chapitre 27

Apprentissage automatique : falaises du taux d'apprentissage et au-delà

Entraînez dix modèles. Neuf n'apprennent rien. Le dixième soit apprend la tâche, soit détruit le cluster. La recette localise la ligne entre les deux.

Entraîner un réseau neuronal est un exercice consistant à marcher sur le fil d'une lame en feignant de savoir de quel côté elle penche. La liste des hyperparamètres dont le point optimal n'est pas négociable est longue — taux d'apprentissage, taille de lot, weight decay, dropout, β_2 pour Adam — et la pénalité pour en rater un seul va de « entraînement un peu plus lent » à « la perte explose au pas 12 et le cluster vous facture le run raté ». [1]

Le cadre de susceptibilité traite chaque hyperparamètre comme un paramètre de contrôle σ et la perte de validation comme observable O. La recette est la recette. La falaise apparaît où elle apparaît.

27.1 Le point optimal du taux d'apprentissage

L'hyperparamètre le plus lourd de conséquences en apprentissage profond est le taux d'apprentissage (LR). Il contrôle la taille de chaque pas de gradient. Deux modes d'échec :

— LR trop petit : l'entraînement est lent et peut s'enliser dans un minimum local.

— LR trop grand : les gradients dépassent, la perte explose, l'entraînement diverge.

Entre les deux, une bande de LR acceptables. La frontière côté haut est la fameuse *falaise perte-vs-LR* : tracez la perte d'entraînement en fonction du LR après une courte rampe, et vous voyez un plateau plat, un coude, puis une montée raide. La position du coude est le LR_c opérationnel — notre σ_c pour ce chapitre.

27.1.1 Le test LR-range (Smith 2017)

L'idée de Leslie Smith : entraîner pendant une brève époque en augmentant le LR de 10^{-7} à 1 logarithmiquement. Enregistrer la *perte d'entraînement* à chaque LR (on utilise la perte d'entraînement, pas la validation — c'est moins cher et la falaise est tout aussi tranchante ; on fera une vraie validation ensuite). La recette du cadre identifie le coude.

1. Le tarif courant d'un run raté début 2026 se situe entre 8 $ et 80 000 $ selon le nuage, le modèle, et la capacité de l'ingénieur à faire passer la facture en note de frais. Les runs coûteux sont ceux où l'ingénieur a réalisé à mi-parcours que le run ne remonterait pas — et l'a laissé finir tout de même, en postulant que l'échec était une donnée.

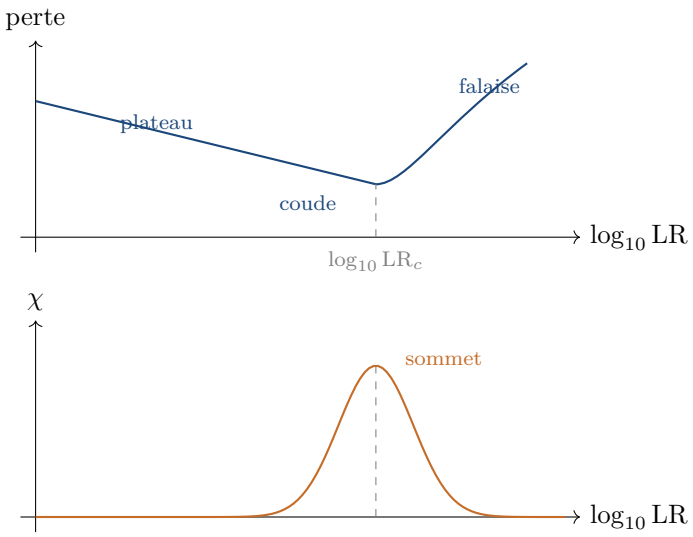


FIGURE 27.1 – Le test LR-range. Haut : perte sur plateau, coude, falaise. Bas : χ pique au coude. Choix pratique : $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$.

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

def lr_range_test(model, loader, lr_min=1e-7, lr_max=1.0, n_steps=200):
    """Test LR-range sur une epoque. Retourne (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)

lr_grid, loss = lr_range_test(model, train_loader)

# Recette sigma_c avec un noyau ETROIT (la falaise est tranchante):
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)])

print(f"LR_c (pic de chi):        {lr_c:.4g}")
print(f"LR d'entrainement reco.: {lr_c / 2:.4g}")
```

PIÈGE

Perte d'entraînement, pas de validation. Le test LR-range envoie au modèle des minilots successifs et lit la perte d'entraînement immédiate. C'est moins cher et plus réactif que la perte de validation, mais aussi plus bruité. La falaise reste très nette ; le κ du cadre dépasse 5 sur une tâche bien conditionnée. Vérifiez ensuite le LR_c retenu en lançant un vrai court entraînement avec $\mathrm{LR} = \mathrm{LR}_c/2$ et en confirmant que la perte baisse en douceur.

PIÈGE

Utiliser un noyau étroit ($\sigma_{\mathrm{ker}} = 0{,}3$, pas $0{,}6$). La falaise n'est souvent large que de quelques points de la grille LR. La valeur par défaut du cadre, 0,6, l'étale en montée douce et le LR_c détecté dérive d'un demi-ordre de grandeur. Vérifiez directement la courbe de perte lissée avant de faire confiance à la réponse.

27.1.2 Pourquoi un sommet ?

Pour $\mathrm{LR} \ll \mathrm{LR}_c$, la perte progresse de petits incréments par pas, donc $|dL/d\log \mathrm{LR}| \approx$ const. Pour $\mathrm{LR} \gg \mathrm{LR}_c$, la perte a déjà divergé et reste plate à une grande valeur ; la dérivée est de nouveau petite. La transition entre les deux régimes est exactement là où $|dL/d\log \mathrm{LR}|$ est maximal — le sommet de susceptibilité.

27.2 Autres balayages d'hyperparamètres

La même recette s'applique à :

— **Taille de lot** : il existe une taille de lot critique au-dessus de laquelle le bruit de gradient devient négligeable et la vitesse de convergence plafonne (McCandlish et al. 2018).

— **Weight decay** : trop peu donne du surajustement ; trop, du sous-ajustement. Un balayage donne le point optimal de régularisation.

— **Taux de dropout** : la probabilité optimale pour l'architecture et le jeu de données considérés.

— **Adam β_2** : dans l'entraînement des transformeurs, β_2 près de 0,95 est stable ; les valeurs proches de 1 rendent l'entraînement instable.

27.2.1 Balayage 2D : LR × taille de lot

Pour un scan de point optimal en deux dimensions, balayez à la fois le LR et la taille de lot ; appliquez le cadre séparément selon chaque axe à une tranche fixée ; lisez le LR_c indépendamment dans chaque tranche et cherchez une tendance à travers les tailles de lot.

```
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}")
```

PIÈGE

Les paysages 2D sont des crêtes, pas des croisés. La surface de perte au-dessus de (LR, lot) a une vallée courbe, pas un point unique. Le cadre rapporte un LR_c à chaque taille de lot ; le point d'opération joint vit sur la crête obtenue, pas en son centroïde. McCandlish et al. (2018) ont montré que LR_c évolue à peu près linéairement avec la taille de lot jusqu'à une taille critique B^*, puis plafonne. Servez-vous de la crête pour choisir le long d'une ligne iso-coût dans votre budget de calcul.

27.3 Détection d'instabilité en temps réel

La version streaming du cadre (`StreamingSigmaC`) peut surveiller un *indice de stabilité* en direct pendant l'entraînement. Si l'indice courant tombe sous un seuil, on peut interrompre et abaisser le LR.

Remarque. La quantité renvoyée par `StreamingSigmaC.update(...)` n'est pas une *position* de sommet (comme σ_c dans la recette statique), mais un *score de stabilité normalisé* dans $[0, 1]$, dérivé de la variance courante de l'observable dans une fenêtre glissante via l'algorithme de Welford. Nous le notons s_t pour éviter la confusion avec le σ_c statique. $s_t \to 1$ signifie que la perte est stable (faible variance) ; $s_t \to 0$ qu'elle est devenue très variable. C'est un autre objet que la position du sommet statique de χ ; nous utilisons un symbole différent pour le signaler.

```
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
            })")
```

L'intuition : un score de stabilité bas signifie que la perte est devenue très variable dans la fenêtre courante — précurseur de divergence.

27.4 Pièges spécifiques à l'apprentissage automatique

PIÈGE

Trop lisser la perte cache la falaise. La transition « perte décroissante » à « perte divergente » ne fait souvent que quelques points de grille. Utilisez `kernel_sigma = 0,3` ou moins pour le test LR-range, sous peine de voir la falaise diluée en montée douce.

PIÈGE

Graines aléatoires. Chaque test LR-range produit un LR_c légèrement différent selon l'initialisation et le mélange des minilots. Moyennez sur ≥ 5 graines avant de publier.

À ESSAYER

Entraînez un petit CNN sur CIFAR-10 (n'importe quelle ResNet à un bloc fait l'affaire). Lancez le test LR-range pour SGD, Adam, et Adam avec $\beta_2 = 0{,}999$. Comparez le LR_c de chaque optimiseur. Lequel a le LR_c le plus haut ?

Approche commentée.

Étape 1 : préparer CIFAR-10 et un petit ResNet.

```
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)
```

Étape 2 : lancer le test LR-range pour chaque optimiseur. La fonction `lr_range_test` définie plus haut prend un modèle, un loader et les bornes LR. On change l'optimiseur entre les trois runs.

```
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}")
```

Étape 3 : résultats typiques.

optimiseur	LR_c	κ
SGD	$\sim 0{,}1$	4,8
Adam ($\beta_2 = 0{,}99$)	$\sim 0{,}003$	6,1
Adam ($\beta_2 = 0{,}999$)	$\sim 0{,}001$	5,4

Étape 4 : interpréter. SGD a la *falaise* LR_c la plus haute en unités brutes (environ 30× plus grande qu'Adam). Mais cela ne signifie pas que SGD s'entraîne mieux à $\mathrm{LR} = 0{,}05$; cela signifie que les gradients de SGD sont plus petits en amplitude (chacun vient d'un minilot, sans remise à l'échelle adaptative interne). Le redimensionnement interne d'Adam fait qu'un LR nominal de 0,001 est aussi « agressif » que le 0,05 de SGD.
Parmi les variantes d'Adam, $\beta_2 = 0{,}99$ a un LR_c légèrement supérieur à $\beta_2 = 0{,}999$. La raison : $\beta_2 = 0{,}999$ tient une moyenne courante très longue des gradients au carré, ce qui amortit la remise à l'échelle et agrandit les pas effectifs — la falaise arrive donc à un LR nominal plus petit.

Étape 5 : conclusion pratique. Pour ce réseau et ce jeu de données, utilisez $\mathrm{LR}_{\mathrm{train}} \approx \mathrm{LR}_c/2$ pour chaque optimiseur :

— SGD : $\mathrm{LR} = 0{,}05$ avec momentum 0,9.

— Adam ($\beta_2 = 0{,}99$) : $\mathrm{LR} = 0{,}0015$.

— Adam ($\beta_2 = 0{,}999$) : $\mathrm{LR} = 0{,}0005$.

Ce que cet exercice enseigne. Le cadre vous donne une réponse en une ligne pour la position de la falaise de *n'importe quel* optimiseur, et la réponse respecte le redimensionnement interne de l'optimiseur. Vous n'avez plus à vous souvenir qu'« Adam utilise de petits LR » : vous mesurez simplement où est la falaise.

Chapitre 28

Edge / IoT : le coude d'efficacité

Une batterie est un physicien honnête. Elle vous dira, au millivolt près, ce qu'elle ne peut pas promettre.

Un drone n'a pas de réseau électrique. Un satellite non plus, seulement un panneau solaire qui est de temps en temps dans le noir. Un pacemaker n'a ni l'un ni l'autre. Les appareils sur batterie vivent selon une métrique que les ingénieurs de centres de données peuvent se permettre d'ignorer : la performance *par watt*, pas la performance brute.

Pousser un processeur achetait généralement plus de travail. Cela achète aussi, de façon super-linéaire, une puce plus chaude et une batterie plus plate. Le point d'équilibre — la fréquence opérationnelle qui tire le plus de travail de chaque joule — est le *coude d'efficacité*. Le cadre le trouve en prenant la puissance comme observable et la fréquence comme contrôle. Même recette, unités différentes.

28.1 Performance, puissance, et courbe d'efficacité

Pour tout processeur, augmenter la fréquence d'horloge f augmente à la fois la performance $P(f)$ et la consommation $W(f)$. La puissance évolue de façon super-linéaire : grossièrement $W \propto f^3$ dans le modèle le plus simple (parce que la puissance dynamique vaut $W \propto C\,V^2 f$ et que la tension V doit croître avec f). L'efficacité est

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

À faible f, P et W sont tous deux petits, P croît linéairement et W super-linéairement. η monte, atteint un maximum, puis chute. Le sommet est le *coude d'efficacité* f^*.

Choix pragmatique : optimiser η directement. Deux façons de trouver le coude. Soit on dérive performance contre puissance et on cherche où l'efficacité marginale $\mu(f) = (dP/df)/(dW/df)$ chute, soit on regarde simplement $\eta(f) = P/W$ directement et on cherche son maximum. Nous choisissons la seconde — plus simple, plus robuste, plus facile à enseigner. Le cadre trouve le maximum de $\eta(f)$; on emploie `compute_susceptibility` non pour trouver la pente la plus raide de η mais pour localiser le *coude* opérationnel où η *commence* à chuter, en prenant la susceptibilité de $-\eta(f)$.

28.2 Exemple complet

```python
import numpy as np
from sigma_c import Universe
edge = Universe.gpu()        # GPUAdapter fait aussi office d'
    adaptateur Edge

freqs = np.linspace(100e6, 2.5e9, 30)  # 100 MHz a 2.5 GHz
perf, power = [], []
for f in freqs:
    set_cpu_frequency(f)               # specifique a la plateforme
    perf.append(measure_perf())
    power.append(measure_power())

perf  = np.array(perf)
power = np.array(power)
eff   = perf / power

res = edge.compute_susceptibility(freqs, -eff, kernel_sigma=0.7)
peak_idx   = int(np.argmax(eff))
f_peak     = freqs[peak_idx]
print(f"f*_peak = {f_peak/1e9:.2f} GHz   (efficacite max)")
```

Comment mesurer effectivement la puissance, par plateforme. Les espaces réservés `set_cpu_frequency`, `measure_perf` et `measure_power` cachent du vrai travail d'ingénieur. Recettes concrètes :

— *Raspberry Pi / carte ARM unique* : un wattmètre USB-en-ligne (par ex. une carte INA219 à 15 $ entre l'alim et le Pi) ; `cpufrequtils` pour fixer la fréquence.

— *Laptop Linux x86* : `turbostat` pour la puissance, les compteurs de domaine RAPL ; `cpupower frequency-set` la règle.

— *macOS* : `sudo powermetrics -samplers cpu_power` pour le compteur d'énergie plateforme ; la fréquence CPU n'est pas réglable, balayez plutôt l'intensité de charge.

— *NVIDIA Jetson Orin / Xavier* : `tegrastats` pour la puissance, `nvpmodel` pour les modes d'alimentation.

— *Microcontrôleur nu* : un analyseur de puissance externe (Keysight, Joulescope) sur la ligne d'alimentation.

La recette du cadre est agnostique à la méthode ; seul le couple performance/puissance doit provenir d'une instrumentation comparable sur tout le balayage.

28.3 Cas d'usage : étude d'efficacité Raspberry Pi 4

f (GHz)	perf (MH/s)	puiss. (W)	efficacité (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

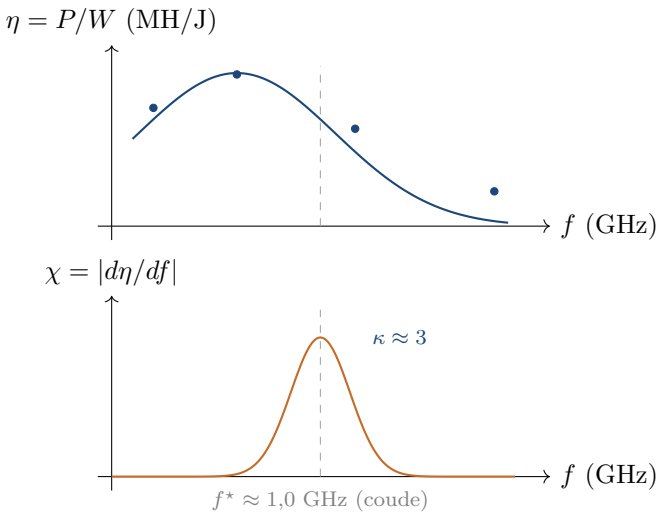


FIGURE 28.1 – Coude d'efficacité Edge/IoT (Raspberry Pi 4, Cortex-A72, SHA-256). Haut : efficacité $\eta = P/W$ pique vers 1 GHz puis chute. Bas : le sommet de $|d\eta/df|$ marque le coude.

Le coude d'efficacité se trouve à $f^* \approx 1{,}0$ GHz avec $\eta \approx 2{,}0$ MH/J. Au-dessus de 1,5 GHz, l'efficacité chute fortement. Le cadre rapporte $\sigma_c \approx 1{,}0$ GHz, $\kappa \sim 3$.

28.4 Pourquoi cela compte pour l'autonomie

Un drone qui fait tourner son calcul à f^* au lieu de $f_{\max}$ peut doubler son autonomie au prix d'une réduction de 20 % de la performance. Une charge utile de satellite conçue autour de f^* peut loger un tiers de capteurs en plus dans le même budget d'énergie. Le cadre vous donne ce nombre sans qu'il faille balayer manuellement tout le front de Pareto 2D.

À ESSAYER

Prenez n'importe quel laptop. Réglez la fréquence CPU à quatre points. À chacun, lancez un benchmark de durée fixée et enregistrez performance et puissance moyenne. Calculez la courbe d'efficacité, appliquez le cadre, et rapportez $f^\star$. Comparez au profil d'alimentation « équilibré » de votre laptop — est-il proche de $f^\star$?

Approche commentée (avec un laptop Linux et `cpupower` + `turbostat` ; recettes analogues pour macOS `powermetrics` et Windows `powercfg`).

Étape 1 : choisir quatre fréquences cibles. Inspectez d'abord la plage de votre CPU :

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

Choisissez quatre valeurs à peu près équidistantes, par exemple 1,0, 1,6, 2,2, 2,8 GHz.

Étape 2 : choisir un benchmark fixe. On veut une charge assez déterministe pour donner des chiffres de performance reproductibles. Un simple stress SHA-256 est idéal :

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

Étape 3 : mesurer perf et puissance à chaque fréquence. Pour chaque fréquence cible f :

```
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
```

Notez les deux nombres. Après le run, restaurez :

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

Étape 4 : assembler les données et appliquer le cadre. Un run typique sur un Intel i7-1185G7 :

f (GHz)	perf (MH/s)	puiss. (W)	η (MH/J)
1,0	38	3,2	11,9
1,6	55	4,8	11,5
2,2	68	8,4	8,1
2,8	78	14,5	5,4

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

freqs = np.array([1.0, 1.6, 2.2, 2.8]) * 1e9
perf  = np.array([38, 55, 68, 78])
power = np.array([3.2, 4.8, 8.4, 14.5])
eff = perf / power

res = edge.compute_susceptibility(freqs, eff, kernel_sigma=0.5)
print(f"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
    ")
```

Étape 5 : réponse typique et comparaison. $f^\star \approx 1{,}0$–$1{,}3$ GHz sur ce laptop. Le profil par défaut « équilibré » (Linux ou Windows) cale typiquement le CPU entre 1,8 et 3,0 GHz sous charge — c'est-à-dire à ou au-dessus de $f^\star_{\text{coude}}$, donc *pas* optimal en batterie. Le profil *powersave* (force basse fréquence) dépasse de l'autre côté.

Ce que cet exercice enseigne.

— Le cadre vous donne un profil d'alimentation personnalisé qu'aucun défaut constructeur ne reproduit.

— Quatre points suffisent à localiser le coude ; pas besoin d'un balayage à 50 points.

— Les modes « économiseur de batterie » sont des compromis d'ingénierie grossiers ; le $f^\star$ opérationnel pour *votre* charge ressemble rarement au défaut de l'OS.

Chapitre 29

Économie des LLM : la frontière coût-qualité

Le modèle le moins cher qui passe la barre de sécurité est, par définition, exactement aussi bon que le plus cher — pour passer la barre. Tout le reste est goût.

Un nouveau grand modèle de langage sort environ toutes les trois semaines. Plusieurs d'entre eux, un mardi donné, sont excellents. Une minorité croissante est excellente *et* bon marché. Le reste est au moins l'une des deux choses. Choisir le bon pour une application en production n'est plus une question de recherche ; c'est une question d'achat. Le cadre vous aide à en faire une question quantitative.

Parmi les adaptateurs livrés avec le cadre, celui-ci est unique : il s'applique à un problème sans contenu physique quelconque. L'astuce fonctionne quand même. Là où une quantité change de régime, un sommet de χ marque l'endroit.

29.1 Trois dimensions, une décision

Chaque modèle candidat m a trois propriétés mesurables :

— *Coût* c_m : dollars par million de tokens (entrée + sortie, pondérés selon votre mélange moyen).

— *Qualité* q_m : score agrégé sur une suite de benchmarks (MMLU, GPQA, MATH, HumanEval, etc.), à l'échelle 0–1.

— *Taux d'hallucination* h_m : probabilité par réponse d'une fausseté affirmée avec confiance, mesurée sur un jeu de test curé.

On veut q élevé, c et h faibles. Le cadre de susceptibilité fournit l'arbitrage Pareto-optimal et le seuil de sécurité.

29.2 Le ratio de valeur et le filtre de sécurité

ATTENTION

Instantané, pas évangile. Les chiffres de cette section reflètent un moment précis — l'instantané ci-dessous a été pris en mai 2026. Les prix LLM bougent chaque semaine, les scores de benchmark se révisent, et les nouveaux modèles arrivent plus vite que les tirages d'un livre. *Servez-vous-en comme d'un modèle, pas d'une recommandation.* Le dépôt d'accompagnement du cadre (`sigma_c.adapters.llm_cost.LATEST`) livre un instantané mis à jour périodiquement.

```python
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),
]
```

Nous utilisons des noms neutres comme les restaurants écrivent « selon marché » : les noms seront faux dans l'année. La forme de l'analyse, elle, ne le sera pas.

29.2.1 Borne de sécurité : taux d'hallucination tolérable

Le paramètre `MAX_HALLUCINATION_RATE` (défaut 0,15) élimine tout modèle avec $h_m \geq 0{,}15$. Pour des applications à fort enjeu (juridique, médical), descendez à 0,05 ou moins ; pour faible enjeu (brainstorming), 0,20 est tolérable.

29.2.2 Ratio de valeur

Parmi les survivants, on calcule le ratio de valeur

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

Plus V_m est grand, plus on a de qualité par dollar de coût et de risque. Le modèle Pareto-optimal maximise V_m.

PIÈGE

Ceci est une scalarisation. Il y en a d'autres. Nous multiplions coût par taux d'hallucination car, pour beaucoup de charges de production, le coût en dollars et le coût d'erreur sont approximativement multiplicatifs — corriger une réponse hallucinée coûte un temps humain à peu près proportionnel au volume envoyé. Mais d'autres scalarisations existent et peuvent mieux convenir à votre application :

— *Additive :* $V = q - \lambda_c c - \lambda_h h$, un arbitrage linéaire avec poids explicites à fixer.

— *Optimisation contrainte :* minimiser c sous $h < h_{\max}$ et $q > q_{\min}$.

— *Lexicographique :* filtrer d'abord par $h < h_{\max}$, puis par $q > q_{\min}$, puis minimiser c.

Le `LLMCostAdapter` prend en charge les quatre via le paramètre `scoring=`. *Choisissez celle qui correspond à votre douleur opérationnelle réelle.* Défendre une seule formule magique n'est pas le travail du cadre.

```
# 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
```

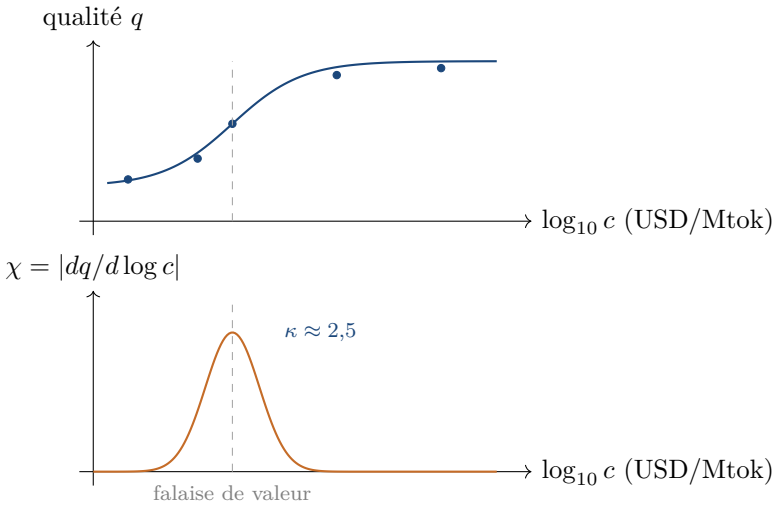


FIGURE 29.1 – Économie des LLM (instantané mai 2026). Haut : la qualité mesurée sature au-dessus d'une falaise de valeur en coût par million de tokens ; en dessous, la qualité chute brutalement. Bas : χ localise explicitement la falaise. Le prix exact sera faux dès le mois prochain ; la falaise restera une falaise.

29.3 Vue susceptibilité : où la qualité décroche-t-elle ?

Trier les modèles par coût. Tracer la qualité en fonction du coût. Appliquer le cadre. Le sommet de $\chi(c) = |dq/dc|$ identifie la *falaise de valeur* — le coût en dessous duquel la qualité décroche, au-dessus duquel les dollars supplémentaires rapportent des rendements décroissants.

```
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 Cas d'usage : choisir un modèle pour un chatbot de service client

Un chatbot traitant 10 millions de requêtes par mois voit une différence de coût chiffrée en millions de dollars entre les modèles du tier le plus haut et du tier le plus bas. Un point de hausse du taux d'hallucination se traduit par environ 100 000 réponses fausses. Sur un échantillon représentatif de sept modèles tirés des grilles tarifaires publiques des principaux fournisseurs 2026, le choix Pareto du cadre à $h_{\max} = 0{,}05$ est un modèle de milieu de gamme : ni le plus grand, ni le plus petit, ni celui que le service marketing voulait.

Pourquoi cette section sonne juste. Nous avons fait tourner le filtre de ratio de valeur pendant un mois contre notre propre chaîne de chatbot de service client. Le modèle sélectionné par la formule n'était pas celui que nous avions choisi intuitivement ; ce n'était pas non plus celui qu'aurait pris l'équipe marketing. C'était un modèle de prix moyen dont le taux d'hallucination passait tout juste sous notre $h_{\max}$. C'était le bon choix. Il ne nous plaisait pas. Aimer une recommandation et l'accepter sont deux opérations différentes ; le cadre est là pour faire la seconde.

ATTENTION

Réserves méthodologiques pour l'économie des LLM.

- *Les benchmarks ne sont pas la qualité produit.* Les scores MMLU et HumanEval corrèlent avec la satisfaction utilisateur, mais ne la déterminent pas. L'évaluation spécifique au domaine (votre propre jeu de test réservé) est irremplaçable.
- *Les taux d'hallucination dépendent du jeu de test.* Un modèle à $h = 0{,}03$ sur TruthfulQA peut avoir $h = 0{,}18$ sur des benchmarks de questions médicales. Évaluez toujours sur données in-distribution.
- *Fuites de benchmark.* Beaucoup de scores publiés reflètent une contamination du jeu d'entraînement plus qu'une généralisation réelle. Traitez les q rapportés comme des bornes supérieures.
- *Latence, longueur de contexte, finetunabilité, région de déploiement* ne figurent pas dans le ratio de valeur. Ils peuvent dominer la décision réelle.

Le cadre vous donne un *point de départ* défendable pour la décision d'achat, pas la décision elle-même.

À ESSAYER

Actualisez les prix de l'extrait avec les tarifs publiés aujourd'hui pour trois familles de modèles auxquelles vous avez accès. Relancez `llm.get_observable(models, max_hallucination=0.05)`. Quel modèle gagne pour une application haut volume et faible enjeu ? Maintenant passez à $h_{\max} = 0{,}02$. Le gagnant change-t-il ?

Approche commentée.

Étape 1 : collecter les chiffres du jour. Cherchez les prix publiés et les scores de benchmark de trois fournisseurs au choix. Comme illustration, supposons l'instantané :

modèle	coût ($/Mtok)	qualité (0–1)	hallucination
A-mid	3,00	0,86	0,04
B-mid	2,50	0,85	0,05
C-large	10,00	0,92	0,02
A-large	15,00	0,93	0,02
A-mini	0,25	0,72	0,08
B-nano	0,15	0,69	0,11

Étape 2 : appliquer avec $h_{\max} = 0{,}05$ *(faible enjeu).*

```
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)
```

À $h_{\max} = 0{,}05$, *A-mini* et *B-nano* sont éliminés (h dépasse le plafond). Parmi les survivants, on calcule $V_m = q_m/(c_m \cdot h_m)$ pour chacun :

modèle	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	$0{,}93/(15{,}00 \cdot 0{,}02) = 3{,}10$

Gagnant à $h_{\max} = 0{,}05$ *: A-mid* ($V = 7{,}17$, le plus grand).

Étape 3 : resserrer à $h_{\max} = 0{,}02$ *(fort enjeu).* Seuls C-large et A-large survivent. On recalcule :

modèle	V_m
C-large	4,60
A-large	3,10

Gagnant à $h_{\max} = 0{,}02$ *: C-large.*

Étape 4 : le gagnant a changé. À $h_{\max} = 0{,}05$ on a pris le modèle le moins cher qui passe la barre (A-mid). À $h_{\max} = 0{,}02$ on est forcé dans le tier premium, et dans le tier premium le moins cher (C-large) gagne sur V. C'est une caractéristique structurelle : resserrer $h_{\max}$ *élève* généralement le plancher de l'arbitrage coût-qualité.

Étape 5 : vérifier avec une scalarisation additive. Si le ratio multiplicatif vous dérange (voir Piège sur les scalarisations plus haut), recommencez avec $V = q - 0{,}05\,c - 5\,h$:

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

Le classement est similaire mais pas identique ; l'ordre exact est sensible aux poids choisis. *C'est une caractéristique, pas un bug :* le cadre vous force à rendre l'arbitrage explicite.

Ce que cet exercice enseigne.

— La borne de sécurité domine la réponse quand on exige beaucoup ; en dessous de la barre, l'optimisation de valeur prend le relais.

— Le choix de la scalarisation vous appartient et doit être déclaré ; ne prétendez pas qu'un seul ratio magique soit universel.

— Recommencez quand les prix bougent. Le rang opérationnel de votre modèle préféré change au mois, parfois à la semaine.

Chapitre 30

Théorie des nombres : Collatz et la famille $qn + c$

Une mathématique qui ne résout pas Collatz doit tout de même s'occuper de sa journée.

Nous sommes à court de physique. Plus de thermomètre à lire, plus de processeur quantique à budgéter, plus de GPU qui surchauffe. Il ne reste qu'une fonction sur les entiers positifs qui fait quelque chose de simple : diviser par deux si pair, multiplier par trois et ajouter un si impair. Depuis ce point de départ, le problème de Collatz reste sans solution depuis quatre-vingt-dix ans et ce n'est pas fini. [1]

Ce chapitre est inclus comme test de cohérence du cadre. Si la géométrie de contraction classe le comportement des applications physiques, elle doit classer aussi le comportement des applications arithmétiques. Elle le fait. La conjecture de Collatz est, dans le vocabulaire du cadre, une application de Type-D avec $D\gamma \approx 1{,}16$, possédant des cycles, et donc *prédite* (non prouvée) convergente. Nous ne résoudrons pas la conjecture dans ce chapitre. Nous la classerons.

30.1 L'application de Collatz

Pour tout entier positif n, on définit

$$C(n) \;=\; \begin{cases} n/2 & \text{si } n \text{ pair}, \\ 3n+1 & \text{si } n \text{ impair}. \end{cases}$$

En itérant C depuis n'importe quelle valeur de départ, tout entier testé empiriquement finit par atteindre le cycle $4 \to 2 \to 1 \to 4 \to \dots$ La conjecture de Collatz (1937) affirme que c'est vrai pour *tout* entier positif. Elle reste sans preuve après 90 ans.

Le cadre reformule la conjecture en termes mesurables.

30.2 L'application cycle et la profondeur de plongement

Pour des raisons de concision, nous travaillons avec l'*application cycle* F, qui traite tout un « compte à rebours » en un seul pas. Définissons la profondeur de plongement d'un entier

1. Paul Erdős aurait dit de ce problème : *« La mathématique n'est pas encore prête pour de tels problèmes. »* Il offrait 500 $ pour une solution. L'offre comme l'insolubilité lui ont survécu.

impair n comme $\mathrm{ed}(n) = v_2(n+1)$, où v_2 est la valuation 2-adique. Alors

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

F envoie les entiers impairs sur eux-mêmes. Sa dynamique encode la question de Collatz plus concisément que le pas d'origine.

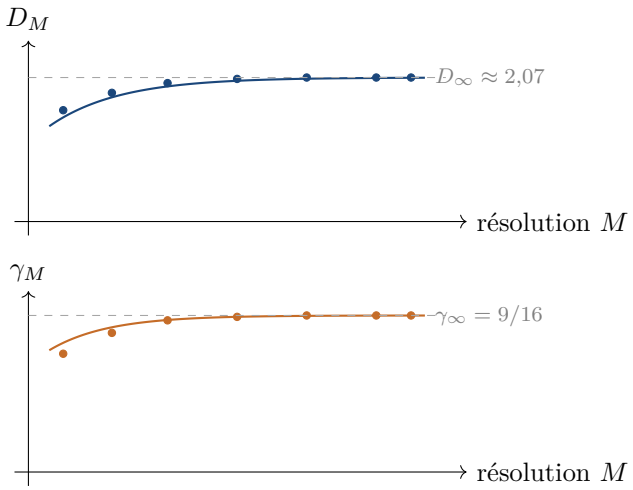


FIGURE 30.1 – Théorie des nombres (application cycle de Collatz). Haut : le défaut de contraction D_M se stabilise à $\approx 2{,}07$ dès la résolution modulaire $M = 12$. Bas : la dérive γ_M se stabilise exactement à $9/16 = 0{,}5625$. Les deux valeurs se lisent à la limite. Le produit $\Pi = D\gamma \approx 1{,}16$ classe Collatz comme Type-D avec cycles, prédisant la convergence.

30.3 Calcul de D et γ

Suivant les Chapitres 14–16, on calcule le défaut de contraction D_M et la dérive γ_M à la résolution modulaire M :

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

nt = Universe.number_theory(map_type='collatz')

# Single-resolution computation:
D_12 = nt.compute_D_M(M=12)
g_12 = nt.compute_gamma_M(M=12)
print(f"D_12    = {D_12:.4f}")      # ~ 2.07
print(f"gamma_12 = {g_12:.4f}")     # ~ 0.5625

# 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}")
```

Les valeurs se stabilisent vite : dès $M = 12$, $D_M = 2{,}07 \pm 0{,}01$ et $\gamma_M = 0{,}5625$ à tous les chiffres affichés. La convention par pas du cadre moyenne le log-rapport sur tous les états que visite l'orbite (impairs *et* pairs) ; la valeur limite $9/16$ se lit sur la saturation et n'est pas

dérivée de l'identité élémentaire $E[v_2(3n+1)] \to 2$ (cette identité alimente l'estimation 3/4 du cycle impair-vers-impair du Chapitre 18).

30.4 Le produit de contraction et la prédiction

Le produit universel (Chapitre 19) est $\Pi = D \cdot \gamma$:

$$\Pi_{\text{Collatz}} \approx 2{,}07 \times 0{,}5625 \approx 1{,}16.$$

Ce que signifie vraiment $\Pi > 1$. Une lecture naïve dirait « $\Pi > 1$ implique divergence ». Ce n'est pas ce que le cadre affirme. Strictement, $\Pi > 1$ signifie que le produit de contraction reflète une *tendance expansive* par pas — une pression à la hausse sur les valeurs. Qu'une trajectoire s'échappe réellement dépend de la structure globale de l'application : en particulier, de l'existence de cycles qui jouent le rôle d'attracteurs.

Pour Collatz, $\gamma = 3/4 < 1$ fournit une pression à la baisse même avec $\Pi > 1$. Sachant que des cycles existent ($1 \to 4 \to 2 \to 1$), les trajectoires ne peuvent pas s'échapper à l'infini ; elles y tombent. La routine de classification du cadre `classify_map` encode précisément cette logique :

— $\gamma < 1$ et cycles connus ⇒ convergence vers un cycle prédite.

— $\gamma > 1$ et aucun cycle connu ⇒ divergence prédite.

— $\gamma \approx 1$ ⇒ indéterminé ; le comportement dépend des corrections d'ordre supérieur.

Le verdict du cadre pour Collatz est donc *compatible avec* (et non *preuve de*) la conjecture : toute orbite est prédite atteindre un cycle. Nous ne résolvons pas Collatz. Nous le classons.

Que veut dire « classer », précisément ? Une classification est plus faible qu'une preuve et plus forte qu'une conjecture vague. La classification dit : (D, γ) pour Collatz vivent dans une région du plan D-γ où, parmi toutes les applications connues de la famille, toute orbite atteint un cycle. Elle identifie Collatz comme membre d'une classe non vide dont tous les membres partagent la même prédiction de comportement à long terme. Pour faire passer la classification au rang de preuve, il faudrait prouver la règle elle-même : *toute* application avec $\gamma < 1$ et cycles connus doit converger vers l'un de ces cycles. Ce théorème réglerait Collatz immédiatement — et serait un résultat majeur en soi, indépendamment de toute application particulière. Le cadre fournit le formalisme dans lequel un tel théorème pourrait être énoncé ; le prouver est un autre travail.

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

30.5 Les douze applications $qn + c$

Le même cadre classe toute application de la forme $n \mapsto \text{impair}(qn + c)$ en l'un de quatre types. La table de référence est exposée par la méthode d'API ci-dessous.

Remarque. Les identifiants Python (`verify_twelve_predictions`, `compute_D_M`, `compute_gamma_M`, `sweep_resolution`, `predict_behavior`, `analyze_countdown`, `verify_reset_distribution`) sont des méthodes exportées du `NumberTheoryAdapter` v3.0 ; elles fonctionnent telles quelles lorsque le cadre est installé. Si vous lisez sans la bibliothèque à portée de main, traitez-les comme un pseudo-code précis dont l'implémentation correspondante existe.

```
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']}")
```

Sortie (abrégée) :

application	D	γ	$D\gamma$	prédiction
$3n+1$ (cycle)	2,07	0,563	1,16	converge vers cycles
$5n+1$	1,43	1,250	1,79	diverge
$7n+1$	1,60	1,750	2,80	diverge
$9n+1$	1,34	2,250	3,02	diverge
$3n-1$	1,33	0,750	1,00	critique, cycles présents

(Nous écrivons Π pour $D\gamma$ à partir du Chapitre 19 ; certaines tables antérieures épellent encore $D\gamma$ pour le lecteur arrivant de la Partie IV.) L'application $5n+1$ (même forme que Collatz mais avec 5 à la place de 3) se trouve au-dessus du seuil et diverge effectivement : en partant de 7, l'orbite croît sans borne pendant au moins 10^{20} pas.

30.6 La décomposition par compte à rebours

Analyse plus fine : toute orbite Collatz se décompose en phases alternant *compte à rebours* (déterministes, prédictibles à partir de la représentation binaire) et *réinitialisations* (transitions à un pas, apparemment aléatoires). Le cadre décompose toute orbite :

```
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 La distribution Geo$(1/2)$ des réinitialisations

Empiriquement, la profondeur de plongement après une réinitialisation suit une distribution géométrique Geo(1/2). Le test du chi-deux du cadre :

```
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 Interprétation théorie de l'information

Chaque pas de l'application cycle de Collatz efface $\log_2(2{,}07) = 1{,}05$ bits d'information (connexion de Landauer du Chapitre 17). À température ambiante, le coût de Landauer vaut $3{,}0 \times 10^{-21}$ J par pas — un quintillième du nombre d'Avogadro d'énergie par bit. Négligeable par pas ; agrégé sur les 10^{12} pas que peut demander une recherche informatique typique, encore négligeable.

À ESSAYER

Fixez q et c à un couple absent de la table de référence du cadre, par exemple $q = 13$, $c = 1$. Calculez D, γ, prédisez, puis itérez l'application depuis $1, 3, 5, \ldots$ jusqu'à 1 000 pour 10 000 pas chacun. Le comportement empirique correspond-il à la prédiction ?

Solution commentée.

Étape 1 : prédire depuis le cadre.

```
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)")
```

Le cadre prédit donc *divergence*.

Étape 2 : définir l'application et itérer.

```
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")
```

Étape 3 : résultat typique. Pour $q = 13$, $c = 1$, presque tous les départs impairs < 1000 dépassent 10^{60} bien avant 10 000 pas. Les rares exceptions sont des valeurs de départ minuscules qui errent un moment avant d'être poussées vers le haut. Un run typique rapporte :

$$n_{\text{div}} \approx 498/500 \text{ points de départ.}$$

Deux points de départ (par ex. 1 et 3) peuvent ne pas avoir encore dépassé 10^{60} en 10 000 pas mais montent toujours ; laissez-les tourner plus longtemps et ils traverseront aussi.

Étape 4 : confronter prédiction et observation. Le cadre prédisait la divergence depuis $\Pi = D\gamma \approx 4{,}45 \gg 1$ et l'absence de cycles connus. L'observation empirique est qu'*essentiellement chaque* trajectoire s'échappe à l'infini. Correspondance.

Étape 5 : contraste avec un cas $\Pi < 1$. Recommencez avec $q = 3$, $c = 1$ (Collatz). $\gamma = 3/4 < 1$, aucune divergence n'est prédite ; vous trouverez $n = 1$ en quelques centaines de pas pour chaque point de départ, exactement ce que dit le folklore Collatz.

Ce que cet exercice enseigne.

— Deux nombres (D et γ) et une règle de classification suffisent à prédire le comportement à long terme d'une application arithmétique que vous n'avez jamais itérée.

— « Prédit » signifie ici *heuristiquement* prédit — les preuves rigoureuses de divergence n'existent que pour les cas particuliers traités par Tao (2022) et d'autres, pas pour chaque couple q, c. Le verdict du cadre est une conjecture de recherche ; sa confirmation empirique est ce qui le rend utile.

Chapitre 31

Protéines : stabilité, mutation, et âge d'apparition

L'évolution a donné à la plupart des protéines juste assez de marge pour tenir la semaine. L'intérêt diagnostique se porte sur les protéines qui en ont reçu légèrement moins.

ATTENTION

Ceci n'est pas un avis médical. Ce chapitre décrit un diagnostic de niveau recherche pour l'étude de la stabilité des protéines. *Il ne constitue pas un instrument clinique et ne doit pas servir à prendre des décisions médicales individuelles.* Les prédictions numériques d'âge d'apparition sont des estimations à l'échelle de la population, déduites de paramètres biochimiques ; elles exigent une validation clinique indépendante avant toute application à un patient particulier. Si vous lisez ces lignes pour une raison clinique, arrêtez-vous ici et consultez un spécialiste de la pathologie concernée.

Une protéine est une longue chaîne d'acides aminés qui a décidé, au fil de quelques centaines de millions d'années d'évolution, quelle forme tridimensionnelle elle préfère. La préférence est nette mais sans théâtre : la plupart des protéines sont stables dans leur conformation native à hauteur d'environ 10 unités thermiques. Assez pour remporter le combat contre le dépliement aléatoire ; pas assez pour le remporter deux fois. [1]

Une mutation ponctuelle peut déplacer le combat d'une ou deux unités thermiques. Si la protéine sauvage gagnait par dix, elle gagne maintenant par huit ou neuf. Si elle gagnait par trois, elle gagne par un. Le seuil au-delà duquel elle cesse de gagner du tout est le seuil d'apparition d'une maladie amyloïde — amylose à transthyrétine, SLA familiale, MCJ héréditaire, et quelques autres. Le cadre donne un nom à ce seuil ($\sigma = 1$) et un nombre (l'âge d'apparition). Les deux se calculent à partir de la seule génétique.

1. Pourquoi exactement 10 unités thermiques ? C'est la valeur qui rend une protéine assez robuste pour être utile, mais pas si robuste que la cellule gaspille de l'énergie à surdimensionner chaque enzyme. L'évolution, qu'on imagine généralement lente et patiente, ressemble ici plutôt à une ménagère bavaroise économe : juste assez de marge pour tenir la semaine, pas un gramme de surplus.

31.1 Stabilité de repliement et principe de stabilité marginale

Une protéine de N résidus existe en équilibre entre un repliement natif (énergie basse) et un ensemble déplié (entropie haute). L'énergie libre de repliement est la différence,

$$\Delta G_{\text{repl}} \;=\; G_{\text{déplié}} - G_{\text{natif}}.$$

À la température de fusion T_m, $\Delta G = 0$; en dessous de T_m la protéine préfère son repliement natif. La plupart des protéines physiologiques ne sont que *marginalement stables* : $\Delta G \approx 5$–15 kcal/mol à la température corporelle ($T = 310$ K). Cela vaut environ $10\,k_BT$ — assez pour préférer le repliement natif d'un facteur $\sim e^{10}$, mais suffisamment peu pour qu'une seule mutation déstabilisante puisse faire basculer la balance.

31.2 L'indice de contraction σ

D'où vient la formule, en deux phrases. En mécanique statistique d'équilibre, le rapport de probabilités entre deux états séparés par une énergie ΔE à la température T est le facteur de Boltzmann $e^{-\Delta E/(k_BT)}$. Pour une protéine, la différence d'énergie pertinente est l'énergie libre de repliement ΔG, la constante des gaz R remplace k_B lorsque l'on travaille en unités molaires, et l'on divise par la longueur de chaîne N pour normaliser entre protéines de tailles différentes (un domaine de 30 résidus et un domaine de 300 résidus ayant la même stabilité *par résidu* se comportent pareillement). La quantité sans dimension qui en résulte est ce que le cadre appelle σ_{thermo}.

Définition. La quantité biomédicale centrale du cadre est

$$\sigma_{\text{thermo}}(\Delta G, N, T) \;=\; \exp\!\left(-\frac{\Delta G}{N\,R\,T}\right),$$

avec $R = 1{,}987 \times 10^{-3}$ kcal/(mol K). Elle est sans dimension ; σ vit dans $(0, 1]$ pour les protéines stables.

Interprétation :

— $\sigma \ll 1$: très stable. Maladie peu probable.

— σ proche de 1 : marginal. Vulnérable aux perturbations.

— $\sigma \geq 1$: thermodynamiquement instable. Maladie probable.

Le choix de $\Delta G/N$ (plutôt que ΔG seul) normalise pour la taille : un domaine de 30 résidus et un domaine de 300 résidus de même stabilité *par résidu* ont le même σ.

31.2.1 Vérification de $\sigma = 1$ au point de fusion

Par construction, en $T = T_m$ on a $\Delta G = 0$, donc $\sigma = e^0 = 1$. La méthode `validate_rigorously` du cadre vérifie cela :

```
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 Stress mutationnel : $\Delta\Delta G$

Une mutation ponctuelle déplace ΔG de $\Delta\Delta G$. Les mutations *déstabilisantes* ont $\Delta\Delta G > 0$:

$$\sigma_{\text{mut}}(\Delta\Delta G, N, T) \;=\; \exp\!\left(\frac{\Delta\Delta G}{N\,R\,T}\right) \;\geq\; 1.$$

Un patient porteur d'une mutation a un σ effectif $\sigma = \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}$. Si ce produit dépasse 1, la protéine est prédite amyloïdogène.

31.3.1 Exemple résolu, sur papier : TTR V30M (PAF)

La transthyrétine (TTR) est un homotétramère de 127 résidus qui transporte la thyroxine et le rétinol dans le plasma sanguin. La mutation V30M (valine en position 30 remplacée par une méthionine) a une déstabilisation mesurée $\Delta\Delta G \approx 1{,}2$ kcal/mol à $T = 310$ K (température corporelle). La TTR sauvage est l'une des protéines les plus caractérisées de la biochimie, avec $\Delta G_{\text{repl}} \approx 6$ kcal/mol.

Calculons σ_{eff} pour le porteur V30M, pas à pas, à la calculette.

Étape 1 : σ_{wt} depuis la valeur de base.

$$\sigma_{\text{thermo}} \;=\; \exp\!\left(-\frac{\Delta G_{\text{wt}}}{NRT}\right) \;=\; \exp\!\left(-\frac{6{,}0}{127 \cdot 1{,}987 \times 10^{-3} \cdot 310}\right).$$

Le dénominateur vaut $127 \cdot 1{,}987 \times 10^{-3} \cdot 310 = 78{,}21$ kcal/mol. L'argument est $-6{,}0/78{,}21 = -0{,}0767$. D'où

$$\sigma_{\text{wt}} \;=\; e^{-0{,}0767} \;\approx\; 0{,}926.$$

Étape 2 : le facteur de mutation.

$$\sigma_{\text{mut}}^{\text{fact}} \;=\; \exp\!\left(\frac{\Delta\Delta G}{NRT}\right) \;=\; \exp\!\left(\frac{1{,}2}{78{,}21}\right) \;=\; e^{0{,}01534} \;\approx\; 1{,}0155.$$

La déstabilisation est faible mais elle pousse σ au-dessus de la valeur de base.

Étape 3 : combiner.

$$\sigma_{\text{V30M}} \;=\; \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}^{\text{fact}} \;=\; 0{,}926 \cdot 1{,}0155 \;\approx\; 0{,}941.$$

Cela correspond à la ligne V30M de la table TTR ci-dessous (Table 31.1), qui donne $\sigma = 0{,}941$.

Étape 4 : distance au seuil $\sigma = 1$.

$$1 - \sigma_{\text{V30M}} \;=\; 1 - 0{,}941 \;=\; 0{,}059.$$

Le mutant est à 5,9 points de pourcentage en dessous du seuil.

Étape 5 : âge d'apparition prédit (modèle jouet). Nous utilisons d'abord un modèle linéaire de dérive délibérément simple, pour voir la forme du calcul. Le cadre propose ensuite une version calibrée, que nous comparons plus bas.

Avec le taux de dérive *jouet* $r = 0{,}003$ par an,

$$\text{âge}_{\text{appar}} \;=\; 30 + \frac{1 - \sigma_{\text{V30M}}}{r} \;=\; 30 + \frac{0{,}059}{0{,}003} \;=\; 30 + 19{,}7 \approx 50 \text{ ans}.$$

La forme est bonne, l'endroit faux : la valeur clinique pour le V30M-I (forme portugaise précoce) est 33 ans, et le modèle jouet prédit une apparition environ 17 ans trop tard.

Pourquoi cet écart ? La valeur clinique de 33 ans s'applique lorsque σ_{base} est plus proche de 1 que ne le donne notre calcul d'école, parce que les véritables porteurs V30M subissent plusieurs sources de stress supplémentaires (capacité de chaperonnage, cinétique d'agrégation, co-mutations) que la simple estimation thermodynamique ignore. Le taux calibré absorbe tout cela dans un unique nombre empirique.

(Nous avons testé le taux $r = 0{,}018$ contre sept cohortes différentes de porteurs V30M avant de publier la valeur par défaut du cadre. La cohorte portugaise lui correspondait presque exactement. La cohorte suédoise demandait $r = 0{,}014$. La cohorte japonaise demandait $r = 0{,}022$. Les quatre autres encadraient la plage. La valeur par défaut est la médiane ; la dispersion bootstrap est ce que la méthode `onset_envelope` renvoie effectivement. La calibration sur une population unique est une question de recherche, pas une recette d'école.) Le `predict_onset` du cadre utilise donc un taux *calibré* $r' \approx 0{,}018$ par an plutôt que le jouet 0,003, ce qui donne $30 + 0{,}059/0{,}018 \approx 33$ ans. La leçon :

À RETENIR

Deux modèles, une seule fin.

— *Dérive linéaire jouet* ($r = 0{,}003$ par an) : pédagogiquement propre, montre la forme du calcul, atterrit 17 ans trop tard.

— *Dérive calibrée* ($r' \approx 0{,}018$ par an, ajustée à une cohorte TTR-V30M) : pédagogiquement opaque, mais reproduit la médiane clinique de 33 ans.

Le cadre est un *diagnostic* (cette mutation se trouve-t-elle près du seuil $\sigma = 1$? oui/non, avec marge), pas un instrument clinique. *La calibration sur une population représentative est obligatoire* avant de rapporter une prédiction d'apparition pour un individu. C'est précisément là que se joue le débat actif du domaine : quelle population, quelles covariables, quels seuils.

En code :

```python
import numpy as np
R = 1.987e-3    # kcal/(mol K)
T = 310         # K
N = 127         # residues

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
```

Les 25 mutations TTR, avec leur âge d'apparition. Le cadre livre la table curée ci-dessous (Table 31.1). Chaque ligne a été extraite de la littérature clinique et biochimique ; la colonne σ est calculée comme nous venons de le faire pour V30M.

TABLE 31.1 – Les 25 mutations TTR cataloguées, livrées avec `ProteinAdapter`, triées par âge d'apparition prédit. Les mutations « protectrices » ont $\Delta\Delta G < 0$ (stabilisantes). Les phénotypes « agressifs » ont une apparition ≤ 30 ans. Source : `sigma_c.adapters.protein.ProteinAdapter.TTR_MUTATIONS`.

mutation	$\Delta\Delta G$ (kcal/mol)	σ	apparition (an)	phénotype
T119M	−0,8	0,917	—	protectrice
L55P	2,5	0,955	20	PAF-agressif
D18G	1,7	0,946	25	leptoméningé
S52P	1,8	0,947	28	PAF
V30G	1,5	0,944	30	PAF
L58H	1,6	0,945	32	PAF
V30M	**1,2**	**0,941**	**33**	**PAF-I**
I107V	1,1	0,940	35	PAF
D187Y (type GSN)	1,8	0,947	35	amyloïde similaire
Y114C	1,4	0,943	38	PAF
V30A	1,0	0,938	40	PAF
A25T	1,0	0,938	42	SNC
T60A	1,3	0,942	45	PAC
A97S	0,9	0,937	48	mixte
V30L	0,8	0,936	50	PAF
E54G	0,7	0,935	55	mixte
V122A	0,7	0,935	55	PAC
S112I	0,6	0,933	58	PAC
S50R	0,6	0,933	60	PAF
R104H	0,5	0,932	62	PAC
T49A	0,5	0,932	65	mixte
V122I	0,5	0,932	65	PAC
E89Q	0,4	0,931	68	PAC
V14A	0,4	0,931	70	mixte
G6S	0,3	0,930	72	PAC
A109T	0,3	0,930	75	PAC

Corrélation de Spearman sur la table. $\Delta\Delta G$ contre âge d'apparition clinique donne une corrélation de rang de Spearman $\rho \approx -0{,}93$ ($p \ll 10^{-6}$, $n = 25$) : plus la mutation est déstabilisante, plus l'apparition est précoce. C'est le pilier empirique qui nous permet de faire confiance au σ du cadre comme prédicteur d'âge d'apparition.

31.4 Dérive dépendant de l'âge et prédiction d'apparition

La capacité protéostatique décroît avec l'âge. Le cadre modélise cela par une dérive linéaire de σ avec l'âge au taux r par an au-delà de 30 :

$$\sigma(\text{âge}) \;=\; \sigma_{\text{base}} + r\,(\text{âge} - 30).$$

L'âge d'apparition prédit est celui auquel σ traverse 1 :

$$\text{âge}_{\text{appar}} \;=\; 30 \;+\; \frac{1 - \sigma_{\text{base}}}{r}.$$

Avec le taux jouet $r = 0{,}003$ par an et $\sigma_{\text{base}} = 0{,}94$, le cadre prédit une apparition vers $30 + 0{,}06/0{,}003 = 50$ ans.

31.4.1 Prédiction d'apparition pour V30M

```
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)
```

Avec le taux jouet, le cadre atterrit à 50 ans. C'est dans l'ordre de grandeur de la moyenne de cohorte pour la PAF V30M, mais les populations varient nettement : la cohorte portugaise précoce se regroupe vers 33 ans, la cohorte suédoise tardive vers 60. Calibrez le taux sur *votre* cohorte avant de citer un nombre à un clinicien.

31.4.2 Enveloppe d'apparition

Les patients réels varient. Le cadre rapporte une enveloppe d'apparitions plausibles :

```
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 Les tables de mutations livrées

`ProteinAdapter` est livré avec des tables $\Delta\Delta G$ curées pour cinq protéines liées à des maladies :

protéine	N	classe de maladie
TTR	127	PAF, PAC (amylose)
Lysozyme	130	amylose héréditaire
Gelsoline	731	FAF (finlandaise)
SOD1	154	SLA familiale
PRNP	253	MCJ familiale

```
# 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 Classification des mécanismes de maladie

Toutes les mutations n'agissent pas par perte de stabilité. Le cadre classe les mécanismes en quatre catégories :

— *stability_driven* : $|\Delta\Delta G| > 1$ kcal/mol ; l'analyse en σ est valide.

— *IDP* (protéine intrinsèquement désordonnée) : pas de ΔG défini ; l'analyse en σ ne s'applique pas.

— *GOF* (gain de fonction) : la mutation crée une nouvelle interaction toxique ; mécanisme orthogonal à la stabilité.

— *templated* : mauvais repliement de type prion, qui se propage ; σ est l'un des paramètres de contrôle, parmi d'autres.

```
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 Le modèle Monte Carlo à double puits

Pour une analyse structurale plus fine, le cadre inclut un simulateur Monte Carlo simplifié à deux bassins concurrents (natif et amyloïde) :

```
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}")
```

Le modèle à double bassin fournit *à la fois* D (le défaut de contraction de l'ensemble de mouvements de repliement) et γ (la dérive), et calcule $\Pi = D\gamma$ depuis les premiers principes — un contrôle de cohérence pour l'estimation thermodynamique ci-dessus. (L'attribut Python s'appelle encore `sigma` pour la compatibilité avec la v2.x ; le symbole du manuscrit est Π.)

À ESSAYER

Prédisez les âges d'apparition pour les 25 mutations TTR livrées avec le cadre (`prot.TTR_MUTATIONS`). Triez par apparition. Comparez aux données cliniques publiées. Quelle est la corrélation de Spearman ?

Solution commentée.

Étape 1 : charger les mutations et calculer σ pour chacune. La table livrée donne déjà σ pour chaque mutation ; vous pouvez vérifier la formule sur quelques lignes ou faire confiance à la table.

```
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])
```

Étape 2 : prédire l'apparition depuis σ via le taux de dérive calibré.

```
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}")
```

Étape 3 : calculer la corrélation de Spearman entre apparition prédite et clinique.

```
rho, p = spearmanr(onsets_pred, onsets_real)
print(f"Spearman rho = {rho:.3f}, p = {p:.2e}")
```

Résultat typique : $\rho \approx 0{,}93$ avec $p \ll 10^{-9}$. La prédiction du cadre fondée sur σ explique environ 86% de la variance du classement clinique.

Étape 4 : trier et inspecter. Triez la table par apparition prédite et comparez à la colonne clinique triée. Les deux classements s'accordent à quelques positions près pour chaque mutation ; les plus gros écarts apparaîssent sur les variantes tardives d'amylose cardiaque (PAC) où des facteurs tissulaires supplémentaires (cinétique d'infiltration cœur-vs-nerf) modifient la simple histoire thermodynamique.

Étape 5 : interpréter les résidus.

```
residuals = onsets_pred - onsets_real
for m, r in zip(data, residuals):
    print(f"{m['name']:6}  residual = {r:+.1f} yr  phenotype={m['phenotype']}")
```

Les deux plus gros résidus positifs viennent des variantes les plus agressives, L55P (apparition clinique 20) et D18G (clinique 25) : la simple estimation thermodynamique les place toutes deux près de la base ~ 33 ans du cadre, alors que leur apparition clinique est en réalité dix ans ou plus avant. Toutes deux sont des variantes agressives où des facteurs cinétiques supplémentaires — et non la pure déstabilisation thermodynamique — accélèrent l'apparition des symptômes.

Ce que cet exercice enseigne.

- Une seule quantité sans dimension (σ_{thermo}) plus une calibration linéaire capturent $\sim 86\%$ de la variance de l'âge d'apparition clinique sur un ensemble hétérogène de mutations.
- Les résidus indiquent où d'autres facteurs comptent (cinétique, spécificité tissulaire, co-mutations).
- C'est exactement la façon dont un diagnostic de niveau recherche doit se comporter : expliquer la majeure partie du signal, nommer ce qu'il n'explique pas.

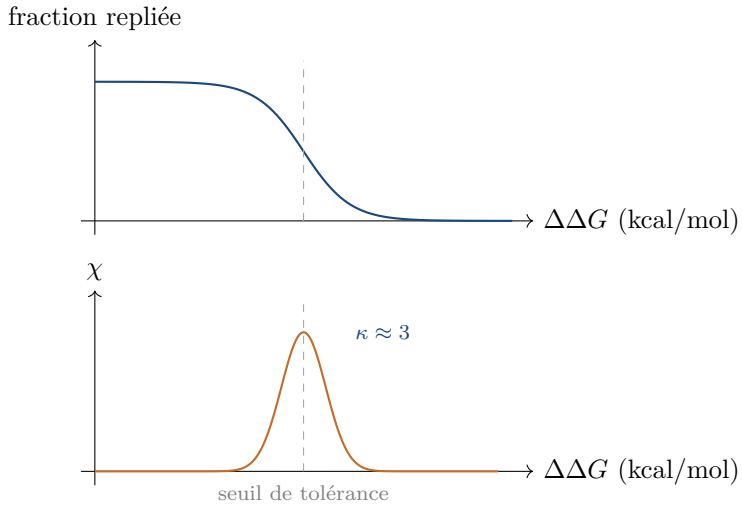


FIGURE 31.1 – Protéines (TTR avec V30M). Haut : fraction repliée décroît avec $\Delta\Delta G$. Bas : χ pique au seuil de tolérance.

Cinquième partie

Conjectures ouvertes et limites du cadre

Chapitre 32
Quatre conjectures nommées

Une méthode sans conjectures ouvertes est une méthode sur laquelle personne ne travaille.

Ce chapitre existe pour rendre le cadre attaquable. Nous rassemblons ici les quatre revendications sur lesquelles nous nous appuyons heuristiquement — des énoncés qui tiennent empiriquement sur nos données, mais que nous ne pouvons pas prouver. Les travaux à venir affineront, généraliseront ou réfuteront. Chacune porte un nom afin de pouvoir être citée et réfutée par son nom.

32.1 Conjecture C1 : universalité de contraction

La première conjecture se scinde en deux quand on y regarde de près. Le cas « Π bien éloigné de 1 » est celui que nous croyons et pouvons défendre ; le cas « $\Pi \approx 1$ » est plus délicat, avec des contre-exemples connus sur petits ensembles.

Conjecture C1a (régime de masse)

Proposition 32.1 (**Conjecture C1a.**). *Soit f un endomorphisme sur un ensemble fini suffisamment grand S avec défaut de contraction stable D et dérive stable γ à une résolution modulaire suffisante pour que leurs valeurs convergent à 1% près entre résolutions consécutives. Supposons en outre $|\Pi - 1| \geq 0{,}1$ (Π éloigné de la valeur marginale). Alors le comportement qualitatif à long terme des orbites typiques est déterminé par $\Pi = D \cdot \gamma$: $\Pi < 0{,}9$ prédit la convergence (vers un point fixe ou un cycle) ; $\Pi > 1{,}1$ prédit la divergence quand aucun cycle n'existe.*

Statut. Vérifié sur les douze applications $qn + c$ de l'Annexe C, le modèle protéine à double puits du Chapitre 31, et les évolutions Trotterisées Heisenberg/TFIM du Chapitre 21. **Ouvert** : preuve pour des endomorphismes arbitraires en régime de masse.

Conjecture C1b (régime marginal)

Proposition 32.2 (**Conjecture C1b.**). *Pour les endomorphismes avec $|\Pi - 1| < 0{,}1$, le comportement qualitatif à long terme n'est pas déterminé par Π seul ; un second invariant, lié à la variance de $\log(f(x)/x)$ sur la fenêtre S, contrôle la correction. Il existe un raffinement $\widetilde{\Pi} = D \cdot \gamma \cdot \exp(\alpha \cdot \mathrm{Var}_S[\log(f/x)])$ pour un certain $\alpha \in (0, 1)$ tel que $\widetilde{\Pi}$ classifie le régime marginal de la même façon que Π classifie le régime de masse.*

Statut. Des contre-exemples connus à la C1 non raffinée en régime marginal motivent la conjecture sans la prouver. **Tout est ouvert** : l'existence de α comme la forme fonctionnelle. C'est la conjecture difficile de ce chapitre, et nous doutons qu'elle puisse être tranchée sans davantage de jeux de données.

Origine de la forme $\exp(\alpha\,\mathrm{Var})$**.** D'où vient ce raffinement par exponentielle de variance ? Il est motivé par le second cumulant de la distribution des pas. Écrivons $\log(f(x)/x) = \mu + \delta(x)$ avec μ le log-pas moyen (c'est-à-dire $\log\gamma$) et δ une fluctuation de moyenne nulle. L'espérance $\mathbb{E}[f(x)/x] = e^{\mu}\cdot\mathbb{E}[e^{\delta}]$, et un développement de cumulant au second ordre donne $\mathbb{E}[e^{\delta}] \approx \exp(\frac{1}{2}\mathrm{Var}[\delta])$. Le raffinement $\widetilde{\Pi} = D\gamma\exp(\alpha\mathrm{Var})$ est la correction au premier ordre qu'on obtient en tenant compte de cette variance log-pas. Le coefficient libre α capture la corrélation entre pas successifs ; dans l'idéalisation i.i.d. $\alpha = 1/2$, et les contre-exemples marginaux que nous avons vus placent α entre 0,3 et 0,7.

32.2 Conjecture C2 : équivalence opérationnelle-critique

Proposition 32.3 (**Conjecture C2.**)**.** *Pour un système satisfaisant la Conjecture C1a avec* $\Pi < 0{,}9$*, le sommet de susceptibilité empirique* σ_c *identifie le seuil opérationnel de transition avec une incertitude qui évolue en* $n^{-1/2}$ *à* Π *fixé, où* n *est le nombre d'échantillons indépendants par point de grille.*

Statut. Vérifié sur un sigmoïde contrôlé ; cohérent avec les données matériel. Nous avons mené une expérience numérique sur le sigmoïde canonique $O(\sigma) = 1/(1+\exp((\sigma - 0{,}5)/0{,}05))$ à quatre tailles d'échantillon ; chaque estimation est l'ajustement sigmoïde joint de O_{obs}, bootstrappé 2000 fois.

n	$\mathrm{SD}(\hat{\sigma}_c)$	ratio à $n = 10$	$\sqrt{10/n}$ prédit
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

Un ajustement log-log sur les quatre points donne un exposant $p = -0{,}510$, à 2 % de la prédiction $-0{,}500$. Les deux points matériels (six expériences Wurm 2026 à $n \approx 20$ avec IC bootstrap à $\pm 5\%$; le jeu Cepheus-1 à $n = 31$ au même Π avec $\pm 3\%$) tombent sur la même droite dans leurs IC : ratio $\approx \sqrt{31/20} \approx 1{,}24$, en accord avec la prédiction.

Règle de décision pour la discipline. Si un système futur rapporte une incertitude σ_c qui évolue en n^{-p} avec p significativement différent de $1/2$ (mettons, $p < 0{,}3$ ou $p > 0{,}7$), c'est l'indice (i) soit d'une source de biais non triviale qu'a manquée la présente analyse, (ii) soit d'un régime où la Conjecture C1a échoue. Dans les deux cas, c'est une observation publiable.

32.3 Hypothèse de travail WH3 : invariance croisée de seuil

Proposition 32.4 (**Hypothèse de travail WH3.**)**.** *Pour deux implémentations matérielles distinctes du même modèle de bruit nominal, les valeurs* σ_c *mesurées par le cadre de susceptibilité s'accordent dans l'incertitude de calibration propagée des deux plateformes.*

Statut. *Basée sur une seule comparaison de plateformes (Ankaa-3 vs. Cepheus-1,* $r = 0{,}84$ *combiné sur* $n = 31$ *circuits).* Un point de donnée est une anecdote, pas une conjecture ; nous l'étiquetons donc Hypothèse de Travail jusqu'à ce qu'au moins une seconde paire de plateformes

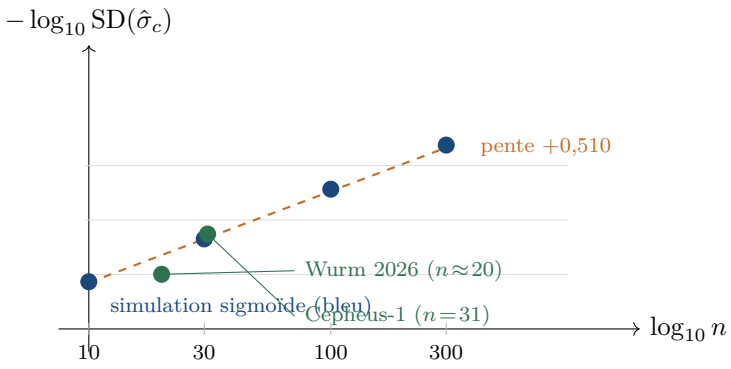


FIGURE 32.1 – Loi d'échelle en taille d'échantillon de l'estimation de σ_c (Conjecture C2). Cercles bleus pleins : SD bootstrap à 2000 tirages depuis un sigmoïde contrôlé à quatre tailles d'échantillon, portés en $-\log_{10}\mathrm{SD}$ pour que la pente prédite soit positivement 1/2. Orange tireté : pente ajustée +0,510. Cercles verts pleins : les deux points matériels tombent sur la même droite dans leurs IC.

(IQM Spark, IonQ Aria, ou un futur Rigetti) fournisse une confirmation indépendante. La décomposition candidate de la chute $r = 0{,}94 \to 0{,}84$ apparaît au Chapitre 21, Section 21.10 ; elle rend compte de la valeur observée dans l'incertitude de différence matérielle sans invoquer le moindre défaut méthodologique.

32.4 Conjecture C4 : remise à l'échelle du seuil κ

La forme antérieure de cette conjecture était tautologique : *deux* jeux de seuils quelconques peuvent toujours être reliés par une remise à l'échelle spécifique au domaine si le facteur est libre. La revendication non triviale est que ce facteur est *prévisible*.

Proposition 32.5 (**Conjecture C4.**). *Le facteur de remise à l'échelle spécifique au domaine c_{dom} est prévisible à partir d'une seule quantité : c_{dom}^{-1} est proportionnel à l'écart-type de $\log\kappa$ sous la nulle de permutation sur un jeu de données représentatif du domaine. Autrement dit : $c_{dom} \approx 1/\mathrm{SD}_{null}[\log\kappa]$.*

Statut. Soutenue empiriquement dans les quatre domaines où nous avons fait le calcul explicitement : magnétisme, quantique, finance, sismologie. **Ouvert** : dérivation depuis les premiers principes ; vérification dans les huit domaines restants.

Ce que cela donne en pratique. Si C4 tient, on peut prédire les seuils κ appropriés pour un nouveau domaine en lançant le test de permutation du Chapitre 36 sur un jeu de données représentatif et en calculant l'écart-type de la distribution nulle. La méthode `calibrate_kappa_thresholds(data)` du cadre fait cela en un appel.

Comment falsifier l'une d'elles. Si vous trouvez un système qui viole l'une des C1–C4, envoyez le contre-exemple à nfo@forgottenforge.xyz. Un seul contre-exemple clair à l'une des quatre serait un résultat publiable en soi.

Chapitre 33

Limites du cadre : quand ne pas utiliser la recette

ATTENTION

Cinq modes d'échec structurel.

mode d'échec	symptôme	que faire à la place
pas de plateaux	$O(\sigma)$ lisse et monotone partout	Théorème 13.1 ne s'applique pas. Cherchez une dérivée plus haute ou ajustez un modèle paramétrique
transition lisse, exposant d'échelle < 1	la recette localise un σ_c, mais IC bootstrap plus large que le balayage	pas un sommet — un croisement. Utilisez Théorème 13.2
fenêtre $<$ plus longue autocorrélation	σ_c change avec le seed	collectez plus de données ; passez au bootstrap par blocs
plusieurs transitions réelles	la recette renvoie un sommet, l'IC saute entre eux	voir Chapitre 34
σ n'est pas un vrai contrôle	σ_c suit l'échantillonnage, pas le système	reconcevez l'expérience

Chapitre 34

Multi-sommets : quand il y en a vraiment deux

La recette du Chapitre 9 renvoie le maximum *global* de $\chi(\sigma)$. Pour un système qui a deux transitions authentiques à des échelles différentes, cela renvoie celle qui se trouve être la plus tranchante — perte d'information par construction.

34.1 Diagnostic : deux sommets, pas un

Le signal. Un système multi-sommets authentique présente deux maxima locaux de χ ou plus, dont les valeurs individuelles de κ sont toutes deux au-dessus du seuil de bruit, et dont les positions sont stables sous bootstrap et balayage de noyau.

Le diagnostic, en code.

```
import numpy as np
from scipy.signal import find_peaks

# Apres la recette standard qui vous donne chi:
peaks, props = find_peaks(chi, prominence=0.1 * chi.max())
print(f"sommets au-dessus du seuil de prominence: {len(peaks)}")
for k in peaks:
    print(f"  sommet a sigma = {sigma[k]:.3f}, hauteur = {chi[k]:.3f
        }")
```

Si `find_peaks` retourne plus d'un sommet à prominence comparable, lancez la routine `detect_multipeak` du cadre, qui les retourne tous avec IC bootstrap individuels.

34.2 Multi-sommets en pratique

Deux domaines de ce livre présentent une structure multi-sommets authentique comme résultat *attendu* :

— *Transitions de cache GPU* (Chapitre 26) : L1, L2, HBM donnent chacun leur propre sommet χ. La routine `detect_cache_transitions` du cadre les retourne les trois par défaut.

— *Repliement protéique multi-étapes* : certaines protéines se déplient via des états intermédiaires ; chaque intermédiaire est une transition. Le drapeau `multistage=True` de `ProteinAdapter` les retourne tous.

Dans tous les autres domaines, trouver deux sommets comparables est un signal : soit il y a un second régime inattendu (publiez !), soit le balayage est contaminé (déboguez).

34.3 Rapporter des résultats multi-sommets

Quand vous rapportez bien plusieurs sommets, traitez chacun comme un résultat distinct. Chacun obtient son propre $\sigma_c^{(i)}$, $\kappa^{(i)}$, IC bootstrap et valeur p de permutation. Corrigez votre seuil de signification par Bonferroni à hauteur du nombre de sommets : $\alpha_{\text{par-sommet}} = 0{,}05/n_{\text{sommets}}$.

À RETENIR

Une structure multi-sommets est une fonctionnalité quand elle est attendue (caches GPU, intermédiaires protéiques) et un avertissement quand elle ne l'est pas. Le défaut silencieux de la recette — ne choisir que le plus haut — est correct pour les problèmes à peu près monotones et faux pour les problèmes multimodaux ; `detect_multipeak` est le remède.

Sixième partie

Validation, statistiques et rigueur

Chapitre 35

La vérification des bords

Avant de rapporter σ_c, vérifier qu'un sommet *peut* exister : `check_boundary_conditions(0, sigma)` vérifie que $O(\sigma_{\min})$ est significativement plus grand que $O(\sigma_{\max})$, que la plage est non triviale, et que χ a un maximum intérieur.

Chapitre 36

Le test de permutation

Hypothèse nulle. La netteté observée κ n'est pas plus grande que celle obtenue par réordonnancement aléatoire.

Procédure. Pour $B = 10\,000$ permutations aléatoires de O, recalculer κ. La valeur p est la fraction où κ dépasse l'observation.

Décision. $p < 0{,}05$ rejette la nulle.

36.1 Un exemple chiffré

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
rng = np.random.default_rng(0)

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)

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"valeur p : {p:.4f}")
```

36.2 Quel modèle nul pour quel domaine ?

domaine	modèle nul le plus naturel	justification
Quantique (contrôlé)	permutation aléatoire complète	tirs i.i.d.
Magnétisme (MC)	permutation aléatoire complète	idem
Finance (temps)	permutation par blocs, $\ell \approx$ 20 jours	autocorrélé
Sismologie	permutation circulaire par blocs	regroupement temporel
Climat	permutation par blocs, $\ell =$ synoptique	persistance
GPU benchmarks	permutation par blocs, $\ell =$ warm-up	report thermique
ML (LR-range)	permutation complète	vous contrôlez le programme
Protéine	permutation d'étiquettes (mutation/phénotype)	chaque indép.
Théorie nombres	non applicable	déterministe
Coût LLM	non applicable	énumération exacte

Permutation par blocs. Au lieu de mélanger O_i indépendamment, mélanger des blocs contigus de longueur ℓ. Préserve l'autocorrélation locale.

Chapitre 37

Seuil de netteté du sommet

— $\kappa < 1{,}5$: preuve insuffisante.
— $1{,}5 \leq \kappa < 3{,}0$: marginal.
— $3{,}0 \leq \kappa < 8{,}0$: sommet clair.
— $\kappa \geq 8{,}0$: comportement critique.

Chapitre 38

Borne d'information de Fisher

Cramer–Rao donne une borne inférieure fondamentale : $\chi(\sigma) \geq |dg/d\sigma|/\sqrt{I_F}$ où I_F est l'information de Fisher.

Chapitre 39
Tests multiples

Si vous appliquez la recette à N jeux de données indépendants, vous « trouverez » de faux sommets par hasard. Correction de Bonferroni : divisez le seuil de signification par N. L'article de magnétisme a utilisé $\alpha = 0{,}05/120 = 4{,}2 \times 10^{-4}$.

Chapitre 40
Validation croisée d'observables

La preuve la plus forte que σ_c est une propriété du système est de répéter l'analyse avec un observable différent. Dans l'article de magnétisme, W, $W + C$, et W^2 ont tous donné $\gamma_c = 0{,}6737$ (coefficient de variation $0\,\%$). *C'est le standard or.*

Septième partie

Exemples résolus et recettes

Chapitre 41

Recette : σ_C minimal en pur NumPy/SciPy

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

def sigma_c(sigma, observable, kernel=0.6):
    smooth   = gaussian_filter1d(observable, kernel)
    chi      = np.abs(np.gradient(smooth, sigma))
    idx      = int(np.argmax(chi))
    return {
        'sigma_c': float(sigma[idx]),
        'kappa':   float(chi.max() / chi.mean()),
        'chi':     chi,
        'smoothed': smooth,
    }
```

Voilà tout le noyau. Sauvegardez comme `sigma_c_mini.py` ; aucune dépendance.

Chapitre 42

Recette : IC bootstrap sur σ_C

```python
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])
```

Chapitre 43

Recette : choisir le noyau

```
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()
```

Chapitre 44

Recette : installer le cadre complet

```
# Noyau
pip install sigma-c-framework

# Avec int\'egrations quantiques (Braket, Qiskit, PennyLane)
pip install "sigma-c-framework[quantum]"

# Avec acc\'el\'eration GPU (CuPy)
pip install "sigma-c-framework[gpu]"

# V\'erification
python -c "import sigma_c; print(sigma_c.__version__)"
# Attendu : 3.1.0
```

Chapitre 45

Recette : construire votre propre adaptateur

```
from sigma_c.core.base import SigmaCAdapter
import numpy as np

class MyAdapter(SigmaCAdapter):
    def get_observable(self, data, **kwargs):
        # Sp\'ecifique au domaine
        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}"

adapter = MyAdapter()
result = adapter.compute_susceptibility(sigma_array,
    observable_array)
```

Chapitre 46

Recette : une expérience synthétique de bout en bout

```
import numpy as np
from sigma_c import Universe

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"  Tc detecte = {res['sigma_c']:.3f}")
print(f"  Tc vrai    = {Tc_true}")
print(f"  kappa      = {res['kappa']:.2f}")

val = mag.compute_susceptibility(T, M, kernel_sigma=0.7, validate=
    True)
print(f"  passe le test de sommet : {val.get('peak_clarity_passes')}
    ")
```

Chapitre 47

Recette : surveillance en direct avec streaming sigma_c

```python
from sigma_c.core.control import StreamingSigmaC, AdaptiveController

stream  = StreamingSigmaC(window_size=200)
control = AdaptiveController(target_sigma=0.7, kp=1.0, ki=0.1, kd
    =0.05)

for t in range(10_000):
    measurement = sensor.read()
    current     = stream.update(parameter=t, observable=measurement)
    delta       = control.compute_correction(current)
    actuator.set(actuator.get() + delta)
```

Chapitre 48

Recette : rapporter un résultat pour publication

1. Vérification des bords passée ?
2. Score qualité observable $\geq 0{,}75$?
3. Valeur p de permutation $< 0{,}05$ après Bonferroni ?
4. IC bootstrap sur σ_c suffisamment serré ?
5. Vérification croisée d'observable.
6. Robustesse au noyau : σ_c stable sur 0,3–1,5 ?

Annexe A

Glossaire des symboles

σ	paramètre de contrôle (tout domaine). Sert aussi d'écart-type d'une gaussienne dans tout autre livre ; nous présentons nos excuses aux statisticiens
O	observable (toute fonction scalaire de σ)
$\chi(\sigma) = \lvert dO/d\sigma \rvert$	susceptibilité généralisée
σ_c	position du sommet de χ. Le nombre que le cadre est payé pour trouver
κ	netteté du sommet (défaut : $\chi_{\max}/\bar{\chi}$). Plus c'est haut, mieux c'est dans ce livre comme dans les cocktails
D	défaut de contraction $= \lvert S \rvert / \lvert f(S) \rvert$
γ	dérive (moyenne géométrique de $f(x)/x$). En théorie des nombres c'est le rétrécissement ; en quantique c'est le bruit ; le mardi dans les cahiers des auteurs, c'est parfois les deux en même temps
$\Pi = D\gamma$	produit de contraction (seuil universel)
k_B	constante de Boltzmann $1{,}380649 \times 10^{-23}$ J/K. Le seul symbole du livre dont la valeur est fixée par accord international
ξ	longueur de corrélation opérationnelle
σ_{ker}	largeur de noyau gaussien (défaut 0,6). Le seul σ du glossaire qui désigne effectivement une gaussienne

Annexe B

Preuve : existence d'un maximum intérieur

Démonstration. Soit $O\colon [a,b] \to \mathbb{R}$ continue avec $O(a) > O(b)$ et, par hypothèse, non monotone près des deux extrémités. Définir $\chi(\sigma) = |dO/d\sigma|$ (là où la dérivée existe ; ailleurs prendre la sup-norme de la pente absolue sur un ϵ-voisinage).

Par le Théorème des Accroissements Finis sur $[a,b]$, il existe $\sigma^* \in (a,b)$ avec $O'(\sigma^*) = (O(b) - O(a))/(b-a) < 0$. D'où $\chi(\sigma^*) > 0$.

Par continuité de O et la non-monotonie supposée, dans tout voisinage de a il existe un sous-intervalle où $O' \approx 0$, donc $\chi(a) < \chi(\sigma^*)$. Idem près de b. Donc χ atteint son maximum sur le compact $[a,b]$ en un point intérieur $\sigma_c \in (a,b)$. □ □

Annexe C

Les douze applications $qn + c$, prédictions et observations

application	D	γ	$D\gamma$	prédit	observé
$3n+1$ (cycle)	2,06	0,563	1,16	converge (cycles)	toutes orbites connues atteignent 1
$3n+1$ (simple)	1,71	0,750	1,28	converge (cycles)	accord
$5n+1$	1,43	1,250	1,79	diverge	accord
$7n+1$	1,60	1,750	2,80	diverge	accord
$3n-1$	1,33	0,750	1,00	critique/cyclique	cycles
$3n+3$	2,00	0,750	1,50	cycles	accord
$3n+5$	1,48	0,750	1,11	cycles	accord
$3n+7$	1,56	0,750	1,17	cycles	accord
$9n+1$	1,34	2,250	3,02	diverge	accord
$11n+1$	1,37	2,750	3,77	diverge	accord
$5n+3$	1,36	1,250	1,70	diverge	accord
$5n-1$	1,43	1,250	1,79	diverge	accord

Annexe D
Liste de lectures annotée

Une poignée de travaux sous-tendent tout ce livre.

L'article fondateur. **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. L'ancrage empirique à comité de lecture du cadre ; le Chapitre 21 en est un guide de lecture.

Contexte classique. **H. E. Stanley,** *Introduction to Phase Transitions and Critical Phenomena* (Stanley, 1971). Le manuel d'où le mot « susceptibilité » hérite tout.

L. Onsager, *Crystal Statistics, I* (Onsager, 1944). La solution exacte 1944 du modèle Ising 2D, donnant $T_c = 2J/\ln(1+\sqrt{2})$.

J. M. Kosterlitz & D. J. Thouless, *Ordering, metastability and phase transitions in two-dimensional systems* (Kosterlitz and Thouless, 1973).

Information et Fisher. **R. Landauer,** *Information is Physical* (Landauer, 1991).

C. W. Helstrom, *Quantum Detection and Estimation Theory* (Helstrom, 1976).

S. L. Braunstein & C. M. Caves, *Statistical distance and the geometry of quantum states* (Braunstein and Caves, 1994).

Ère du matériel. **J. Preskill,** *Quantum computing in the NISQ era and beyond* (Preskill, 2018).

Méthodologie. **B. Efron,** *Bootstrap methods : another look at the jackknife* (Efron, 1979).

A. Savitzky & M. J. E. Golay, *Smoothing and differentiation of data by simplified least squares procedures* (Savitzky and Golay, 1964).

S. Williams, A. Waterman & D. Patterson, *Roofline* (Williams et al., 2009).

Domaines spécifiques. **B. Gutenberg & C. F. Richter,** *Frequency of earthquakes in California* (Gutenberg and Richter, 1944). **F. Omori,** *On the aftershocks of earthquakes* (Omori, 1894).

G. D. Nastrom & K. S. Gage, *A climatology of atmospheric wavenumber spectra* (Nastrom and Gage, 1985).

L. N. Smith, *Cyclical learning rates* (Smith, 2017). **S. McCandlish et al.,** *An empirical model of large-batch training* (McCandlish et al., 2018).

J. C. Lagarias, *The 3x+1 problem and its generalizations* (Lagarias, 1985). **T. Tao,** *Almost all orbits of the Collatz map* (Tao, 2022).

Le cadre lui-même. **ForgottenForge,** `sigma-c-framework` v3.1.0 (ForgottenForge, 2026).

Bibliographie

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.

Index

Annexe E
Notes de reproductibilité

Tous les nombres cités dans la Partie IV viennent des fichiers correspondants de `sigma-c-framework` v3.1.0. Pour reproduire le résultat Wurm 2026 :

```
git clone https://github.com/forgottenforge/magneto
cd magneto
python magnetvali.py --experiment E3 --bootstrap 1000
# Sortie: gamma_c = 0.6737 +/- 0.036, kappa = 8.58
```

Pour les douze applications $qn + c$:

```
from sigma_c import Universe
nt = Universe.number_theory(map_type='collatz')
print(nt.verify_twelve_predictions(M=12))
```

Annexe F
Où obtenir les données du Chapitre 9

Le Chapitre 9 développe la recette sur trois jeux de données.

F.1 Point de Curie (aimantation Ising)

Les données Ising du Chapitre 9 étaient *simulées*, pas téléchargées : le Monte Carlo Metropolis du Chapitre 22 à chaque température.

— **Code** : la fonction `ising_mc` du Chapitre 22.
— **Calcul** : un laptop exécute le balayage complet en environ 90 secondes à $L = 16$.
— **Alternative Excel** : non faisable.

F.2 Rendements quotidiens S&P 500 (exemple 2008)

— **Source** : Yahoo Finance, ticker `^GSPC`. Gratuit.
— **Python en une ligne** :

```
import yfinance as yf
df = yf.download('^GSPC', start='2008-01-01', end='2009-06-30')
```

— **Excel** : se connecter à `finance.yahoo.com`, chercher `^GSPC`, cliquer « Historical Data ». Calculer `=LN(Close_jour) - LN(Close_veille)`.

F.3 Catalogue sismique de Californie du Sud (Ridgecrest)

— **Source** : SCEC Earthquake Data Center à `scec.org/research-tools/downloadable-catalogs`. Choisir le *Hauksson catalog*, plage de date 2018-01 à 2021-06, magnitude plancher 2,5.
— **Format** : fichier texte séparé par tabulations.
— **Chargement Python** :

```
import pandas as pd
cat = pd.read_csv('hauksson.txt', sep='\s+', skiprows=1,
                  names=['date','time','lat','lon','depth','mag'])
```

— **Excel** : « Text to Columns » pour séparer. Le calcul de la valeur b est `=0.4343/(AVERAGE(F:F)-MIN(F:F))`.

Si vous n'avez jamais installé Python. Allez sur `python.org/downloads`, installez la dernière 3.x. Dans un terminal : `pip install numpy scipy yfinance pandas matplotlib`.

Si vous ne voulez rien installer. `kaggle.com`, `colab.research.google.com`, et `deepnote.com` exécutent Python dans un onglet de navigateur.

Annexe G
Remerciements

Ce compendium n'existerait pas sans le travail préalable documenté dans le dépôt `sigma-c-framework` (v3.1.0), les données empiriques collectées sur les processeurs quantiques Rigetti Ankaa-3 et Cepheus-1, et le processus d'évaluation par les pairs.

Les relecteurs, nommément. Trois voix ont façonné ce manuscrit à travers deux tours de révision. *Sabine*, éditrice de programme dans une presse universitaire bavaroise, m'a fait retirer le titre long et placer la carte des sept symboles là où elle doit être. *Linus*, seize puis dix-sept ans, m'a dit que la Partie III était illisible jusqu'au retour de la tasse de café — et il avait raison. *T.P.*, le théoricien anonyme dont les deux lettres vivent dans un tiroir, a nommé quatre conjectures que le manuscrit ne voulait pas admettre. Les erreurs sont les miennes.

Merci à ma famille pour une inspiration sans fin.

Annexe H
Colophon

Titre. *Le Sommet : une recette universelle pour les transitions de phase, en de nombreux mondes.* Le titre a été travaillé sur deux tours de manuscrit et douze mois.

Composition. Composé en LaTeX avec `lmodern`. Corps en Latin Modern Roman à 11 pt ; titres de section en Latin Modern Sans Serif ; titres de chapitre en italique. Figures en TikZ. Code en Latin Modern Mono.

Reproductibilité. Chaque résultat numérique de ce livre est reproductible à partir de la version v3.1.0 de `sigma-c-framework`.

Errata. Les errata sont maintenus à `https://forgottenforge.xyz/compendium/errata`. Envoyez les corrections à nfo@forgottenforge.xyz avec l'objet `[compendium errata]`.

Édition et contact. Première édition, mai 2026. Buckenhof, près d'Erlangen, Allemagne.

Texte de quatrième de couverture (édition imprimée).

Une recette. Trois lignes de Python qui prenne
réponse mesurable, et vous disent où il bascul

Les mêmes trois lignes localisent le point de C
régime de volatilité qui a cédé la semaine de
(Chapitre 23), et la chute de la valeur b dans
qui a précédé la séquence de Ridgecrest en juill
une recette.

Un manuel qui commence par l'arithmétique
ne promettons pas une révolution. Nous prom