FORGOTTEN FORGE

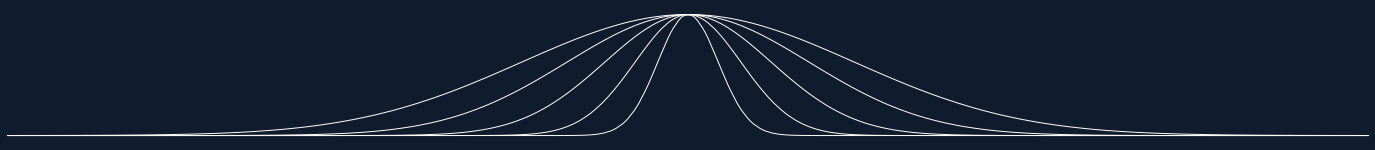
Un libro de texto en seis partes

La Cumbre

una receta universal para transiciones de fase, en muchos mundos

hardware cuántico — magnetismo — finanzas — aprendizaje automático
sismología — clima — GPU — proteínas
teoría de números — *edge* — economía de los LLM

M. C. Wurm
ForgottenForge | Buckenhof | 2026



Acompaña a SIGMA-C-FRAMEWORK v3.1.0
Anclaje empírico: AVS Quantum Science **8**, 013804 (2026)

La Cumbre

una receta universal para transiciones de fase,
en muchos mundos

M. C. Wurm

ForgottenForge.xyz | 2026

Prefacio

σ_c es donde el sistema vuelca. $\Pi = D\,\gamma$ es por qué vuelca. Casi todo este libro trata de lo primero. El tercio central trata de lo segundo.

No le prometemos una revolución. Las revoluciones son ruidosas, sucias, mal organizadas. Le prometemos algo más silencioso. Una receta. Tres líneas de Python que toman un sistema con una perilla regulable y una respuesta medible y le dicen dónde vuelca.

Barra un parámetro. Mida. Tome una derivada. Encuentre el máximo. Cuatro pasos; los tres últimos no cambian entre dominios, sólo el primero lo hace. Los mismos cuatro pasos localizan la temperatura de Curie de un imán de hierro, el umbral operacional de ruido de un procesador cuántico, el cambio de régimen de un mercado bursátil, la edad de aparición de una enfermedad hereditaria por mal plegamiento proteíco. Muchos dominios, una sola receta. Somos conscientes de que esto suena sospechoso. La primera mitad de este libro está escrita para los sospechosos.

Un buen método es como un buen mayordomo. Uno no lo nota. Hace su trabajo en silencio. No inventa problemas y no resuelve más de lo que se le pidió. En sus días malos le dice que no puede ayudar, y va a buscar a alguien que pueda. El marco de la susceptibilidad, cuando se porta bien, es esa clase de mayordomo. Cuando no se porta bien, le dice exactamente cuál de las cinco condiciones de confianza ha fallado —y por eso pusimos a propósito ese capítulo el primero de la Parte II.

Cómo surgió. Comenzó con el hecho de tener que mirar.

No del todo por voluntad propia. Llegado uno a cierta edad, ha conocido ya un número considerable de seres humanos, y un subconjunto particular de ellos parece haberse especializado en actuar de manera que, en retrospectiva, plantea la pregunta: ¿cuándo, exactamente, sucedió esto? El colega que renuncia a un puesto seguro para criar criptomonedas. El amigo que, después de tres años quejándose de su prometida, se casa con ella de repente. La paciente que anuncia en consulta que ha reconsiderado la operación, habiendo encontrado algo mejor en internet. El legislador que nombra una comisión. El propietario que reforma la cocina por undécima vez.

Yo no estuve presente cuando estas personas tomaron sus decisiones. Pero sí estuve poco antes y poco después, y en cada caso hubo un momento —un momento estrecho, a menudo ni siquiera un día entero— en el que la persona se encontraba a un lado, y poco después se encontraba, irrevocablemente, al otro.

Desde entonces he llamado a ese momento el *punto de vuelco*. Es un nombre modesto para algo que, francamente, me deja sin palabras.

Durante un tiempo creí que era una observación social. Algo que concernía sólo a los seres humanos, y sólo a los menos voluntariosos entre ellos. Resultó no ser así. Los imanes tienen

un punto de vuelco. Las bolsas tienen un punto de vuelco. Los ordenadores cuánticos tienen un punto de vuelco. Los terremotos, el clima, las redes eléctricas, las proteínas —todos ellos tienen puntos de vuelco. Es la propiedad más común que conozco en los sistemas, superada sólo por la de que pueden romperse.

Eso, en sí, no sería tan malo. Lo malo es que, después de pasar algunos años identificando estos puntos de vuelco —que, dicho sea de paso, no es particularmente difícil una vez que uno sabe qué buscar—, sigo sin saber cómo regresar del lado desagradable al agradable.

Esa es la segunda parte. La primera parte —"aquí está el punto de vuelco"— es matemáticamente tratable. Ese es este libro.

La segunda parte —"cómo se vuelve a donde estaba bien"— no es matemáticamente tratable. Quizá no sea tratable en absoluto. Quizá simplemente queda más allá de lo que yo, como individuo, puedo resolver en una sola vida. Le quedaría agradecido a quien, leyendo esto, me hiciera llegar una idea.

Hasta entonces: el procedimiento descrito a continuación le dice, cuando menos, que un punto de vuelco existe. Le dice dónde está. Incluso le dice cuán nítido es, lo cual ha resultado inesperadamente útil. Lo que no le dice es qué hacer una vez que ha llegado allí. Eso tendrá que decidirlo el lector.

Es un libro modesto. Hace una sola cosa. Esa cosa no me propuse inventarla —se me impuso, a lo largo de varias décadas de mirar de lado, quisiera yo o no. La transmito aquí con la esperanza de que, la próxima vez, alguien mire en el momento justo y diga: "Espera, un momento —esto está a punto de volcar. ¿Estás seguro de que quieres eso?". Aunque sea sólo eso ya sería una mejora sobre el estado actual de las cosas, por pequeña que fuese.

Una última cosa. Suponemos que usted sabe sumar, restar, multiplicar y dividir. Suponemos que sabe leer. No suponemos que haya visto una letra griega, conocido un valor esperado, ni ejecutado un script de Python. Si alguna frase del índice le intimida, ignórela; cuando llegue al capítulo, dejará de intimidarle.

El primer borrador de este libro tenía 78 páginas y era ilegible. El segundo tenía 130 páginas y era deshonesto. Este es el tercero, y es honesto como lo es un buen mayordomo: contando lo que está realmente en la habitación.

Sobre este libro

Este libro tiene una sola tarea. Tomar a un lector que sepa hacer aritmética y llevarlo al punto en que pueda aplicar el *marco de la susceptibilidad* —el método de σ_c— para encontrar umbrales críticos en sus propios datos. Sea un ordenador cuántico o un mercado financiero, un material magnético o un *benchmark* de GPU. La receta es una sola receta.

Para quién es este libro. Cualquiera que alguna vez haya medido algo girando una perilla y se haya preguntado *«¿dónde vuelca esto?»*. No se requiere un título universitario. Bastan las cuatro operaciones aritméticas y la paciencia para seguir un ejemplo resuelto.

Lo que este libro no es. Una monografía de investigación. No suponemos que haya leído artículos sobre transiciones de fase, información de Fisher o mecánica estadística fuera del equilibrio. Cuando estas ideas aparecen, las construimos desde cero.

La promesa. Al final de la Parte II ejecutará el método de σ_c en cinco líneas de Python. Al final de la Parte III comprenderá por qué funciona. Al final de la Parte IV sabrá cuál de los adaptadores de dominio ya disponibles encaja con su problema, y cómo extender el marco con uno propio.

Cómo leerlo. Las Partes I a III son una cadena. Saltarse algo dejará una laguna en la que los capítulos posteriores caerán silenciosamente. A partir de la Parte IV cada capítulo se sostiene solo. Elija el dominio que se ajuste a su trabajo; los demás seguirán ahí cuando los necesite.

Convenciones.

- Cada símbolo nuevo se define la primera vez que aparece. Las letras griegas no tienen pase libre.
- Cada fórmula va seguida al menos de un ejemplo numérico. Si no puede reproducir el número en papel, la explicación no está terminada.
- Cada capítulo termina con un pequeño ejercicio. Llevan de cinco a treinta minutos y merecen ese tiempo.
- El código usa Python y se ejecuta en cualquier entorno con NumPy y SciPy instalados. La biblioteca `sigma_c` es una comodidad, no una dependencia —el núcleo de cinco líneas cabe en una servilleta.

- Las frases que sobreviven una sección sin ser desplazadas por otra mejor se recogen en una caja de color al final. Cuatro colores, cuatro propósitos: *azul* para lo que hay que recordar, *naranja* para lo que conviene evitar, *verde* para lo que conviene probar, y *rojo* para advertencias propiamente dichas (clínicas, financieras, de seguridad).
- **El símbolo σ se redefine localmente en cada capítulo.** En cada capítulo σ denota el eje de control *de ese capítulo*, no una sigma física universal. La letra griega se usa porque es corta y tradicional; no acarrea por sí misma significado físico alguno entre dominios. Usamos σ_{ker} (anchura del núcleo de suavizado) y σ_t (volatilidad, en finanzas) con subíndices explícitos para evitar colisiones.

El marco de referencia. La biblioteca de código abierto que acompaña al libro es `sigma-c-framework` (versión 3.1.0). Instálela con

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

Puede terminar este libro sin instalarla nunca; el algoritmo núcleo de cinco líneas sólo requiere NumPy y SciPy. El marco es más cómodo, como un mayordomo es más cómodo que no tener mayordomo.

Cómo citar este libro.

> M. C. Wurm, *La Cumbre: una receta universal para transiciones de fase, en muchos mundos*, ForgottenForge, Buckenhof, 2026.

(Los detalles de licencia y edición están en el colófón al final del libro.)

Tres itinerarios de lectura

Este libro contiene cuatrocientas y pico mil cosas que podría leer. Es casi seguro que no quiere leerlas todas. Tres itinerarios explícitos, en profundidad creciente:

Itinerario A —"sólo quiero usarlo" (1–2 tardes).

- Material introductorio y guía de letras griegas (15 min)
- Capítulos 1–7 *de manera breve* (los fundamentos; hójelos si alguna vez ha visto cálculo)
- Capítulo 9 (**la receta universal**)
- Capítulo 10 (**cuándo falla el método**)
- Capítulo 12 (**umbrales de κ**)
- Un capítulo de la Parte IV que case con su dominio
- Parte V (**validación**) cuando esté a punto de publicar

Itinerario B —"quiero entender por qué funciona" (una semana). El itinerario A, más toda la Parte III (geometría de contracción) y toda la Parte V (validación). Hojear los capítulos de la Parte IV ajenos a su dominio.

Itinerario C —"quiero extender el marco" (dos semanas). El itinerario B, más las recetas de la Parte VI y los apéndices, más implementar realmente su propio adaptador (Capítulo 45). En ese punto usted es un contribuyente, no un lector.

Lo que este libro no promete. No promete una teoría de todo. No promete que todo sistema tenga una cumbre; algunos no la tienen, y el Capítulo 10 le ayudará a saber cuáles. No promete que el mismo σ_c numérico se mantenga entre plataformas de hardware; el propio seguimiento del artículo sobre magnetismo en otro hardware muestra que el valor se desplaza un ± 3–$5\,\%$ al cambiar el modelo de ruido. No promete que el método sustituya a la experiencia en el dominio; el método encuentra dónde mirar, sigue siendo usted quien debe interpretar lo que ve.

Lo que sí *prometemos*: si su sistema tiene una perilla regulable y una respuesta cuasi-monótona, la receta del Capítulo 9 localizará el umbral operacional sin exigirle inventar antes un modelo.

Índice general

Los siete símbolos que de verdad necesita

Si no recuerda nada más de este libro, recuerde los siete símbolos de abajo. Todo lo demás se define cuando aparece.

símbolo	en una línea	primera aparición
σ	el parámetro de control que usted gira — la temperatura, el nivel de ruido, la tasa de aprendizaje	Capítulo 2
O	el observable que mide a medida que lo gira	Capítulo 2
$\chi(\sigma) = \lvert \mathrm{d}O/\mathrm{d}\sigma \rvert$	la susceptibilidad: a qué velocidad cambia O	Capítulo 11
σ_c	la posición del máximo de χ — *donde el sistema vuelca*	Capítulo 9
κ	la nitidez del máximo: $\chi_{\text{máx}}$ dividido por la media de χ	Capítulo 12
D	el defecto de contracción de una aplicación: $\lvert S \rvert / \lvert f(S) \rvert$	Capítulo 17
γ	la deriva de una aplicación: media geométrica de $f(x)/x$ (*¡no* la misma γ que la intensidad del ruido cuántico!)	Capítulo 18
$\Pi = D \cdot \gamma$	extra, octavo: el producto de contracción — *por qué el sistema vuelca*	Capítulo 19

El resumen en una frase de todo el libro es:

PARA RECORDAR

σ_c es donde vuelca. $\Pi = D\gamma$ es por qué vuelca.

La guía completa de pronunciación de las letras griegas está en la página siguiente (las clásicas requieren cierta costumbre, pero cada una no es más que una letra).

Parte I

Fundamentos — de la aritmética a la derivada

Una guía de lectura para las letras griegas

Este libro usa letras griegas. Los planes de estudio escolares tienden a emplearlas con avaricia, y muchos lectores las encuentran más intimidatorias de lo que merecen. Una letra griega es una letra. Se pronuncia como abajo, y por lo demás se comporta exactamente como una letra ordinaria —se puede multiplicar, sumar, escribir ecuaciones con ella.

símbolo	nombre	dónde en este libro
σ	«sigma»	parámetro de control (Capítulos 2–7)
σ_c	«sigma sub ce»	posición del máximo de susceptibilidad
χ	«ji» (suena como «ki»)	la susceptibilidad misma
κ	«kappa»	nitidez del máximo (cuán agudo es)
γ	«gamma»	deriva en teoría de números, ruido en cuántica
Δ	«delta» (mayúscula)	cambio, diferencia
ρ	«ro»	matriz de densidad, correlación
ξ	«xi» (suena como «ksi»)	longitud de correlación
π	«pi»	tanto la constante 3,14… como la persistencia GARCH
β, α	«beta», «alfa»	exponentes críticos en imanes, parámetros GARCH
∂	«d parcial»	cierto tipo de derivada
$\sim$	«se comporta como», «escala como»	proporcionalidad aproximada
$\propto$	«proporcional a»	proporcionalidad estricta
$\langle \cdot \rangle$	«corchetes angulares»	promedio sobre muchas pruebas
$\lvert \cdot \rvert$	«valor absoluto»	quita el signo, conserva la magnitud

Si olvida alguna de ellas, vuelva a esta página. No se moverá de aquí.

Capítulo 1

¿Qué es un número, qué es un cociente?

Este capítulo no supone nada. Si sabe hacer $7 + 3 = 10$ y $10 \div 2 = 5$, está sobrecualificado. Los lectores que ya saben qué es una derivada pueden hojearlo; los que no, no deberían. Todo lo que queda en las restantes 300 páginas se apoya en las cuatro operaciones aritméticas hechas con cuidado. Empezamos haciéndolas con cuidado.

1.1 Los números, en la única forma en que los usaremos

Un *número* en este libro es uno de los siguientes:

- un número entero como 0, 1, 2, 3, -4, 17;
- un número con parte decimal como 0,5, 3,14, $-2{,}718$;
- un número positivo muy pequeño como 0,001 o 10^{-6}, que leemos «diez elevado a menos seis». El símbolo 10^{-6} no es más que una forma breve de escribir 0,000001.

El conjunto de todos estos números lo escribimos $\mathbb{R}$, y lo llamamos «los números reales». No necesitará ninguna definición más profunda que ésta.

1.2 La resta indica un cambio

Si una cantidad valía 4 antes y ahora vale 7, el cambio es $7 - 4 = 3$. Decimos que la cantidad *aumentó en* 3.

Si una cantidad valía 4 antes y ahora vale 1, el cambio es $1 - 4 = -3$, un número negativo. Decimos que la cantidad *disminuyó en* 3. El signo menos es lo único que indica la dirección del cambio.

Definición 1.1 (Cambio)**.** El cambio de una cantidad O entre dos mediciones es $\Delta O = O_{\text{nuevo}} - O_{\text{viejo}}$. La letra griega Δ (delta) se pronuncia «delta» y de aquí en adelante significará siempre «cambio en».

Ejemplo 1.2 (La taza de café, parte 1: el cambio)**.** Una taza de café estaba a 80 °C hace un momento. Ahora marca 77 °C. El cambio de temperatura es $\Delta T = T_{\text{nuevo}} - T_{\text{viejo}} = 77 - 80 = -3\,^{\circ}\text{C}$. El signo negativo dice: la temperatura *bajó* 3 grados. Volveremos a esta taza a lo largo del capítulo.

1.3 La división indica una tasa

La taza del Ejemplo 1.2 no perdió sus tres grados instantáneamente. Digamos que tardó 60 segundos. La *tasa* de enfriamiento es entonces

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

Dos cosas a notar. *Primera*, hemos dividido. *Segunda*, las unidades vienen detrás: grados arriba, segundos abajo, de ahí «grados por segundo».

Si en cambio hubiera tardado sólo 5 segundos, la tasa habría sido $-3/5 = -0{,}6\,^\circ\mathrm{C/s}$ —un enfriamiento mucho más rápido. Tasa grande significa que el cambio ocurre deprisa; tasa pequeña significa que ocurre despacio. *Esta sola idea es la semilla de todo lo que sigue en este libro.*

PARA RECORDAR

Una *tasa* es un cambio dividido por otro cambio. Le dice a qué velocidad se mueve una cosa cuando otra se mueve. Todo en el método de σ_c proviene de una tasa concreta: a qué velocidad cambia una medición cuando uno cambia un parámetro de control.

1.4 Cocientes con unidades, y por qué las unidades importan

La tasa $-0{,}05\,^\circ\mathrm{C/s}$ dependía de las unidades. Si hubiéramos medido el tiempo en minutos en lugar de segundos, el mismo enfriamiento sería $-3\,^\circ\mathrm{C} \div 1\,\mathrm{min} = -3\,^\circ\mathrm{C/min}$. *Mismo proceso físico, número distinto, porque la unidad es distinta.*

TRAMPA

Cuando compare dos tasas, debe usar las mismas unidades en ambas. Una tasa de $-3\,^\circ\mathrm{C/min}$ *no* es mayor que una de $-0{,}05\,^\circ\mathrm{C/s}$; son iguales. Compruebe siempre las unidades antes de sacar una conclusión.

1.5 Una cosa que depende de otra

Hasta ahora seguíamos una cantidad (la temperatura de la taza) y un «eje» a lo largo del cual variaba (el tiempo). Esa es la relación más simple posible entre dos cantidades —una entrada, una salida. Este patrón es tan común que los matemáticos le han dado un nombre: una *función*.

El próximo capítulo trata de las funciones. De momento, quédese con la imagen en la cabeza: una función es una receta que toma un número como *entrada* y devuelve un número como *salida*. Nuestra historia del enfriamiento es una función: a cada instante t le corresponde una única temperatura T, y podemos escribir $T = T(t)$.

PRUÉBALO

Un coche estaba en el km 200 de una carretera a las 14:00 y en el km 260 a las 14:30. Calcule su velocidad media en km/h. ¿Iba más rápido o más lento que 100 km/h?

Solución, paso a paso.

Paso 1: identifique las dos cantidades y sus cambios. La posición x es lo que varía. El tiempo t es el eje a lo largo del cual varía (mismo patrón que la taza de café del Ejemplo 1.2, pero espacial en lugar de térmico).

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

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

Convertimos 30 minutos a 0,5 horas para que el numerador (km) y el denominador (h) acaben en km/h, que es la unidad pedida.

Paso 2: forme la tasa.

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

La aritmética es una sola división: $60 \div 0{,}5 = 120$. (O, si dividir por un decimal le pone nervioso: $60/0{,}5 = 60 \cdot (1/0{,}5) = 60 \cdot 2 = 120$.)

Paso 3: compare con el umbral preguntado. $120 > 100$, así que el coche iba *más rápido* que 100 km/h.

Comprobación de cordura. Si el coche hubiera recorrido 60 km en una hora completa, la velocidad habría sido 60 km/h —más lenta que 100. Cubrimos la misma distancia en sólo *media* hora, así que esperamos más o menos el doble, que es exactamente lo que obtuvimos.

Qué enseña este ejercicio. Tiene ahora los tres ingredientes en un mismo sitio: una cantidad que cambia (x), un eje a lo largo del cual cambia (t) y una tasa (su cociente). El método de σ_c, más adelante, será el mismo patrón aplicado a una cantidad que cambia (un observable O) cuando uno gira una perilla (un parámetro de control σ).

Capítulo 2

¿Qué es una función?

En el Ejemplo 1.2 teníamos una taza de café. Se enfrió de 80 °C a 77 °C. Un número (la temperatura) variaba con otro (el tiempo). Esa pequeña observación es más importante de lo que suena. Los matemáticos han construido trescientos años de análisis sobre ella. Le pusieron un nombre al patrón —*función*— y una notación: $f(x)$. De esos trescientos años salen las derivadas, de donde salen los máximos, que es por donde este libro comienza a resultar útil.

2.1 La función como receta

Una *función* f es una regla que, dada una entrada x, devuelve una salida específica, que escribimos $f(x)$. Leemos $f(x)$ como «f de x». La palabra «específica» hace mucho trabajo en esa frase: una función debe dar la misma salida cada vez que se la alimenta con la misma entrada. Una regla que devuelve «algún número entre 3 y 5» no es una función. Una regla que devuelve «la temperatura de hoy en Berlín» tampoco es una función —días distintos, respuestas distintas. Una función es un contrato entre entrada y salida, y el contrato se cumple.

Ejemplo 2.1 (Duplicación)**.** La regla «multiplicar la entrada por dos» es una función. Si la llamamos f, entonces $f(3) = 6$, $f(0) = 0$, $f(-1{,}5) = -3$, $f(100) = 200$. Podemos escribir la receta de forma compacta: $\boxed{f(x) = 2x}$. Sea cual sea el número que entre, sale su doble.

Ejemplo 2.2 (Elevar al cuadrado)**.** La regla «multiplicar la entrada por sí misma» es una función. Compactamente: $\boxed{f(x) = x \cdot x = x^2}$. Así $f(2) = 4$, $f(3) = 9$, $f(-2) = 4$ también (porque $(-2) \cdot (-2) = +4$, dos negativos se multiplican y dan positivo). Usaremos esta función como ejemplo recurrente durante los dos próximos capítulos.

A veces escribiremos $f\colon x \mapsto x^2$, que no es más que una notación más elegante para la misma receta del Ejemplo 2.2. La flecha $\mapsto$ se lee «aplica a». Léalo como: «f envía la entrada x a su cuadrado».

2.2 La función como tabla

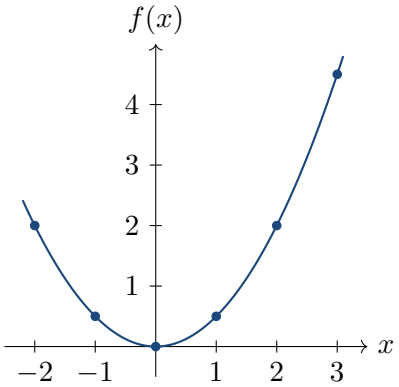
Cualquier función se puede *tabular*: elija algunas entradas y escriba cada entrada junto a su salida. Para la función cuadrado del Ejemplo 2.2:

entrada x	salida $f(x) = x^2$
-2	4
-1	1
0	0
1	1
2	4
3	9

Esa es exactamente la clase de tabla que producirá cuando *barra* un parámetro de control en un experimento real —una palabra rebuscada para «variarlo a lo largo de un rango de valores y registrar lo que mide cada vez».

2.3 La función como gráfica

Si tomamos la misma tabla y colocamos cada par $(x, f(x))$ como un punto en un diagrama con x horizontal y $f(x)$ vertical, obtenemos una *gráfica*.



(Aquí hemos dibujado $f(x) = x^2/2$ en lugar de x^2, puramente para que la figura quepa en la página; dividir la salida por 2 sólo aplasta verticalmente la parábola. La forma es la misma: una curva en U, llamada *parábola*.)

2.4 Funciones en el laboratorio

En un experimento real no escribimos una receta como «x^2». No *conocemos* la receta. Sólo conocemos la tabla: fijamos un parámetro de control, medimos un observable, registramos el par. La función queda implícitamente definida por los datos.

Intuición. A lo largo de este libro, la función que nos importa es la respuesta a: *«Si fijo el parámetro de control en el valor σ, ¿qué valor toma mi medición O?»*. La escribimos $O(\sigma)$. La receta es desconocida. La tabla es lo que tenemos. Todo lo demás se calcula a partir de esa tabla.

2.5 El parámetro de control σ

Usaremos la letra griega σ (sigma) para el parámetro de control. En dominios distintos, σ significa cosas distintas —y esa es toda la gracia del marco. He aquí cinco ejemplos reales:

- En un experimento magnético, σ es la temperatura T de la muestra.

- En un ordenador cuántico, σ es la intensidad γ del ruido que inyectamos.
- En finanzas, σ es un retardo temporal, p. ej. el número de días sobre el que calculamos la volatilidad.
- En un experimento con proteínas, σ es una variación de energía libre $\Delta\Delta G$ causada por una mutación.
- En aprendizaje automático, σ es un hiperparámetro, como la tasa de aprendizaje.

Al marco no le importa cuál de estos casos tiene usted. La receta es idéntica. *Eso* es lo que queremos decir con «universal».

PRUÉBALO

Elija un fenómeno medible de su vida —por ejemplo, lo fuerte que suena una radio cuando uno gira la rueda de volumen. Identifique: ¿qué es σ? ¿Qué es O? Imagine ahora que registra el par (σ, O) para diez posiciones distintas de la rueda. Esboce la tabla que obtendría.

Solución resuelta.

Paso 1: identifique el parámetro de control σ. El parámetro de control es lo que usted puede *fijar*. En una radio, es la posición de la rueda de volumen. Midámosla como un número de 0 (silencio) a 10 (máximo), como suelen marcar los mandos.

Paso 2: identifique el observable O. El observable es lo que usted *mide* en respuesta. Lo natural para el sonido es la sonoridad, medida en decibelios (dB) con un sonometría de móvil o un sonorímetro físico. Los decibelios son una unidad en escala logarítmica, pero eso no le importa a la receta; nos limitamos a registrar el número que vemos.

Paso 3: elija diez posiciones de la rueda. Queremos un barrido uniforme que cubra todo el rango. Lo más simple es $\sigma = 1, 2, 3, \ldots, 10$. (Saltándonos $\sigma = 0$: una radio silenciosa da lo que mida el fondo de la habitación en dB, que es un punto degenerado.)

Paso 4: esboce la tabla que esperaría. Una radio es aproximadamente lineal en su rueda de volumen *en dB*, de modo que esperamos que cada paso añada un número parecido de decibelios. El ruido de fondo en una habitación tranquila ronda los 30 dB, así que:

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

(Su radio real dará números ligeramente distintos; no pasa nada.)

Paso 5: ¿qué clase de función es ésta? Aproximadamente lineal: cada paso de la rueda añade 6–7 dB. Eso significa que la gráfica de O frente a σ sería casi una recta. *No hay transición en este sistema.* La receta del Capítulo 9 no encontraría aquí un σ_c interesante, porque no hay un punto operacional de «vuelco». Es una comprobación de cordura perfectamente válida: *un sistema sin transición produce correctamente la ausencia de máximo.*

Qué enseña este ejercicio. No todos los sistemas tienen un σ_c interesante. El marco es para sistemas con al menos un cambio cualitativo de régimen. Una respuesta lineal monótona, como la rueda de volumen, es el ejemplo negativo más simple.

Capítulo 3

Pendiente: la tasa media de cambio

3.1 Dos puntos sobre una curva

Tómense dos pares cualesquiera de la tabla de una función f: $(x_1, f(x_1))$ y $(x_2, f(x_2))$. La *tasa media de cambio* entre ellos es

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

Esta fórmula tiene la misma forma que la tasa de enfriamiento de la taza de café: *cambio de la salida dividido por cambio de la entrada.*

La pendiente dice: *en promedio, por cada unidad que se aumenta x entre x_1 y x_2, la salida $f(x)$ sube en «pendiente» unidades.* Si la pendiente es $+0{,}5$, la curva sube media unidad por unidad; si es -2, baja dos unidades por unidad.

3.2 Pendiente de una recta

Las funciones más simples de todas son las *lineales*, que tienen la forma

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

Las dos letras a y b son números fijos (los *coeficientes*); x es la entrada. «Lineal» significa: la gráfica es una recta. El coeficiente a controla la inclinación de la recta; el coeficiente b controla a qué altura está en $x = 0$ (porque $f(0) = a \cdot 0 + b = b$). La función «duplicar» del capítulo anterior es la lineal con $a = 2$ y $b = 0$.

Para *cualquier* función lineal, la pendiente es la misma sea cual sea el par de puntos elegido: siempre es el número a. Se puede comprobar una vez y confiar en ella para siempre:

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

Las b se cancelan; el $a(x_2 - x_1)$ del numerador se cancela contra el $(x_2 - x_1)$ del denominador, dejando solo a. *Misma pendiente en toda la recta.* Obvio a posteriori; merece la pena verlo una vez en álgebra.

Ejemplo 3.1 (Cubo bajo un grifo)**.** Un grifo de agua llena un cubo a razón de $f(t) = 2t + 3$ litros tras t minutos. El término constante $b = 3$ es la cantidad de agua que ya había en el cubo en $t = 0$. El coeficiente $a = 2$ es el *caudal*: cada minuto, el cubo gana 2 litros. Para comprobar que a es realmente la pendiente, tómense dos tiempos. En $t_1 = 1$, $f = 5$. En $t_2 = 4$, $f = 11$. Pendiente $= (11 - 5)/(4 - 1) = 6/3 = 2$. Como se anunciaba.

3.3 Pendiente de una curva: depende de dónde se mire

Considérese ahora $f(x) = x^2$. Tómense dos pares de puntos:

- Entre $x_1 = 1$ y $x_2 = 2$: pendiente $= (4-1)/(2-1) = 3$.
- Entre $x_1 = 3$ y $x_2 = 4$: pendiente $= (16-9)/(4-3) = 7$.

La pendiente de una curva no es constante. Depende de *dónde* sobre la curva se mida. Donde la curva sube empinadamente (lado derecho de una parábola), la pendiente es grande. Donde es plana (el fondo), la pendiente es casi cero. Donde baja (lado izquierdo), la pendiente es negativa.

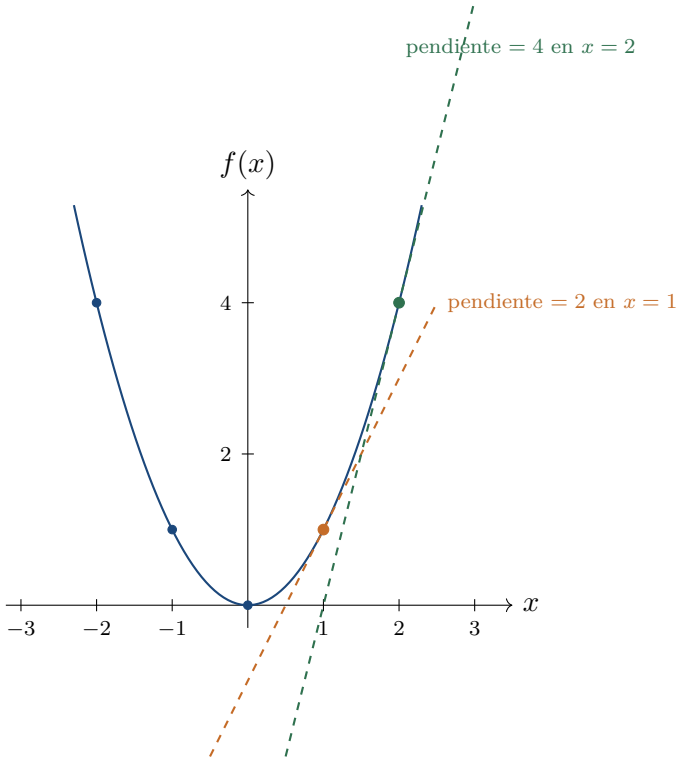


Figura 3.1: La pendiente de $f(x) = x^2$ no es un número; es un número distinto en cada punto. En $x = 1$ es 2; en $x = 2$ es 4. Al crecer x, la curva se vuelve más empinada. Las dos rectas discontinuas son *tangentes*: rectas que tocan la curva exactamente en un punto y comparten su pendiente ahí.

PARA RECORDAR

La pendiente de una función es una propiedad *local*. No es posible describir una función curva con un solo número: hace falta un valor de pendiente en cada lugar. Eso es lo que nos dará la derivada, definida en el próximo capítulo.

3.4 La pendiente es ella misma una función

Mírese la parábola $f(x) = x^2$. Tómese un punto base cualquiera x y un pequeño paso $h > 0$. La pendiente entre x y $x + h$ es

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

(Si aún no sabe por qué $(x+h)^2 = x^2 + 2xh + h^2$*, multíplelo usted mismo:* $(x+h)(x+h) = x \cdot x + x \cdot h + h \cdot x + h \cdot h$*.)*

Si imaginamos ahora que h se encoge hacia cero —lo precisaremos en el próximo capítulo— la pendiente tiende al valor $2x$ en el punto base. Así, en $x = 1$ la pendiente es 2; en $x = 3$ es 6; en $x = 4$ es 8. *La pendiente es ella misma una función de* x. La llamamos *derivada* de f.

(Esto lo desarrollé en un andén de tren en Erlangen, una mañana de marzo de 2026, en el reverso de un recibo de café, porque había olvidado mi cuaderno. El recibo está en algún cajón; el resultado sigue siendo $2x$.)

PRUÉBALO

Calcule la pendiente de $f(x) = x^2$ entre $x_1 = 1{,}99$ y $x_2 = 2{,}01$. ¿Ve algo cercano a la fórmula $2x = 4$ en $x = 2$?

Solución, paso a paso.

Paso 1: identificar los dos pares de puntos sobre la curva. Necesitamos $(x_1, f(x_1))$ y $(x_2, f(x_2))$. Las entradas están dadas: $x_1 = 1{,}99$ y $x_2 = 2{,}01$. Las salidas vienen de la receta de la función $f(x) = x \cdot x$. Así:

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

Paso 2: calcular $f(1{,}99) = 1{,}99 \cdot 1{,}99$ *en papel.* Esta es la clase de cuadrado que la mayoría de la gente hace sin pensar; hagámoslo transparentemente por una vez. Hay dos formas limpias.

Método 1 (multiplicación por columnas). Quitemos primero los decimales; pondremos uno de vuelta al final. Calcúlese 199×199:

199	
$\times$ 199	
1791	(199×9)
17910	(199×90, una posición a la izquierda)
19900	(199×100, dos posiciones a la izquierda)
39601	(la suma)

Cada factor 1,99 tiene dos dígitos tras la coma, así que la respuesta necesita cuatro. Colóquese la coma decimal cuatro lugares desde la derecha: $39601 \to 3{,}9601$. Por tanto $1{,}99 \cdot 1{,}99 = 3{,}9601$.

Método 2 (atajo algebraico, más rápido). Úsese la identidad $(a-b)^2 = a^2 - 2ab + b^2$ con $a = 2$ y $b = 0{,}01$:

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

Misma respuesta, mucha menos aritmética. Usaremos este atajo también para el otro punto.

Paso 3: calcular $f(2{,}01) = 2{,}01 \cdot 2{,}01$. Ahora con $a = 2$, $b = 0{,}01$, usando $(a+b)^2 = a^2 + 2ab + b^2$:

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

Paso 4: aplicar la fórmula de la pendiente. La pendiente entre $(x_1, f(x_1)) = (1{,}99,\ 3{,}9601)$ y $(x_2, f(x_2)) = (2{,}01,\ 4{,}0401)$ es

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

La división: $0{,}0800/0{,}02 = 800/200 = 4{,}00$ (multiplíquense arriba y abajo por 10 000 para limpiar decimales y simplifíquese).

Paso 5: comparar con la predicción de la fórmula. Antes en este capítulo dedujimos que la pendiente de $f(x) = x^2$ en el punto x es $f'(x) = 2x$. En $x = 2$ da $2 \cdot 2 = 4$. Nuestra estimación numérica es 4,00. *Concordancia a cuatro dígitos significativos.*

¿Por qué tan exacto? Recuerde la respuesta algebraica: la pendiente entre x y $x + h$ para $f = x^2$ es $2x + h$. Aquí estraddleamos $x = 2$ simétricamente, así que la fórmula evaluada en el punto medio debería dar exactamente $2x$, con las contribuciones de h cancelándose entre los dos lados. Y así fue: salieron 4,00, no 4,02 ni 3,98. Por eso el marco usa *diferencias centrales* (x_{i+1} frente a x_{i-1}, a horcajadas de x_i) en lugar de diferencias progresivas o regresivas.

Qué enseña este ejercicio. Tres cosas: (i) la fórmula de la pendiente da un número que, sobre una curva suave, se acerca arbitrariamente a la derivada verdadera cuando los dos puntos se acercan; (ii) un muestreo simétrico (central) cancela el error de primer orden; (iii) se puede verificar una fórmula de derivada en un solo punto con dos operaciones aritméticas y una división, lo cual significa que también se puede *descubrir* una derivada a partir de datos sin conocer la fórmula, que es exactamente lo que hacemos a lo largo de este libro.

Capítulo 4

La derivada: la pendiente en un solo punto

4.1 La idea

Queremos, en cualquier punto x, un solo número que diga cuán rápido cambia f ahí. La receta es la que ya usamos: calcular la pendiente entre x y $x+h$ y luego *dejar que h encoja*.

Definición 4.1 (Derivada)**.** La *derivada* de f en el punto x, escrita $f'(x)$ o $\frac{df}{dx}$, es el límite

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

siempre que este límite exista.

El símbolo $\lim_{h\to 0}$, «límite cuando h tiende a cero», se lee: *el valor al que se acerca la pendiente cuando el tamaño del paso se hace arbitrariamente pequeño*. No estamos dividiendo de verdad entre cero. Solo miramos a qué valor se asienta la pendiente cuando h se vuelve diminuto.

4.2 Tres derivadas que conviene saber de memoria

Constante: Si $f(x)=c$ (una constante), la pendiente es siempre cero, así que $f'(x)=0$. Una función plana no tiene tasa de cambio.

Recta: Si $f(x)=a\cdot x+b$, la pendiente es siempre a, así que $f'(x)=a$.

Parábola: Si $f(x)=x^2$, arriba calculamos que $f'(x)=2x$.
Para nuestros propósitos estos tres patrones bastan. Los matemáticos han tabulado la derivada de esencialmente toda función clásica (exponenciales, senos, logaritmos, etc.), pero no necesitaremos ninguna. ¿Por qué? Porque en el laboratorio nunca tenemos una fórmula limpia; solo tenemos una tabla.

4.3 La derivada cuando solo se tiene una tabla

En un experimento real, f es una lista de pares medidos:

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

Los σ_i son los valores de control que fijamos, los O_i son lo que medimos. La derivada en el punto central de un tripón $(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ se aproxima mediante la *diferencia central*:

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

Es exactamente la fórmula de la pendiente aplicada entre los dos vecinos de σ_i.

Ejemplo 4.2. Ha medido la magnetización de un imán a tres temperaturas:

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

La estimación por diferencia central de dM/dT en $T = 2{,}27$ es

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

Signo: negativo, porque M decrece. *Magnitud:* grande, porque M cae rápido en este rango de temperaturas. Esa «magnitud grande donde la función cae rápido» es la señal que perseguirá el método σ_c.

4.4 Diferencias unilaterales en los bordes

En el primer $(i = 1)$ y último $(i = n)$ puntos de la tabla no podemos formar una diferencia central porque falta un vecino. Úsese:

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

Son las diferencias *progresiva* y *regresiva*. Son ligeramente menos precisas que la central, pero bastan en los bordes.

4.5 En código: derivada numérica en tres líneas

NumPy lo trae preparado:

```
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` usa diferencias unilaterales en los bordes y diferencias centrales en el medio, exactamente como acabamos de definir.

PARA RECORDAR

La derivada no es más que una pendiente aplicada localmente. Con tres líneas de NumPy se convierte cualquier tabla (σ_i, O_i) en otra tabla $(\sigma_i, O'(\sigma_i))$. Dedicaremos el resto del libro a encontrar el lugar donde esa derivada es máxima.

PRUÉBALO

Tome cualquier función que le guste: la altura del agua de una taza que está vaciando, el precio de una acción en los últimos diez días, la temperatura de una habitación a lo largo de una tarde. Registre cinco puntos a mano. Calcule las diferencias centrales en los tres del medio. ¿Cuál tiene la mayor pendiente absoluta?

Solución resuelta (con un ejemplo de precio de acción en diez días; la receta es idéntica para cualquier otra elección).

Paso 1: registrar cinco pares (σ_i, O_i). Tomamos cinco días uniformemente espaciados de una acción hipotética que cerró su semana con una caída abrupta. El eje σ es el número de día; el eje O es el precio de cierre en dólares.

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

Elegimos este ejemplo porque la mayor caída está claramente entre el día 3 y el día 4: un cambio de régimen visible. Nuestra tarea es hacerlo visible *numéricamente*.

Paso 2: ¿cuáles son los «tres del medio»? Las diferencias centrales necesitan un vecino a cada lado. Con cinco puntos $1, 2, 3, 4, 5$, los puntos interiores son $2, 3, 4$. Los puntos 1 y 5 son los bordes y requerirían diferencias unilaterales (Sección 4.4).

Paso 3: aplicar la fórmula de diferencia central en cada punto interior. La fórmula es $O'(\sigma_i) \approx (O_{i+1} - O_{i-1})/(\sigma_{i+1} - \sigma_{i-1})$. Aquí $\sigma_{i+1} - \sigma_{i-1} = 2$ días en cada punto interior porque la malla es uniforme con paso 1.

$$O'(2) \approx \frac{O_3 - O_1}{\sigma_3 - \sigma_1} = \frac{98.0 - 100.0}{3 - 1} = \frac{-2.0}{2} = -1.0 \text{ \$/day},$$
$$O'(3) \approx \frac{O_4 - O_2}{\sigma_4 - \sigma_2} = \frac{90.5 - 99.5}{4 - 2} = \frac{-9.0}{2} = -4.5 \text{ \$/day},$$
$$O'(4) \approx \frac{O_5 - O_3}{\sigma_5 - \sigma_3} = \frac{85.0 - 98.0}{5 - 3} = \frac{-13.0}{2} = -6.5 \text{ \$/day}.$$

Las tres pendientes son negativas porque el precio cae.

Paso 4: tomar los valores absolutos para comparar magnitudes. Se nos pregunta «cuál tiene la mayor pendiente absoluta». La receta del marco usa valores absolutos exactamente por esta razón.

$$|O'(2)| = 1.0, \quad |O'(3)| = 4.5, \quad |O'(4)| = 6.5.$$

Paso 5: leer la respuesta. La mayor pendiente absoluta está en el día 4, $|O'(4)| = 6{,}5$ \$/día. Este es el candidato a σ_c para este pequeño conjunto: el momento de cambio más abrupto. *Nótese* que la mayor caída estaba en realidad entre el día 3 y el día 4, pero la diferencia central en el día 4 «siente» esa caída con más fuerza porque tanto el día 3 (todavía alto a \$98) como el día 5 (muy bajo a \$85) están en su ventana.

Qué enseña este ejercicio.

- Las diferencias centrales reparten la influencia de cada paso entre *dos* puntos interiores, no uno. Por eso el marco selecciona un máximo en lugar de un solo escalón recortado.

- Ya con cinco puntos y una división por punto, el método ha localizado el cambio de régimen dentro de los datos.
- La misma aritmética funciona para nivel de agua frente a tiempo, temperatura frente a tiempo, o cualquier otra magnitud-que-varía-con-otra.

Capítulo 5

El máximo: donde la pendiente es mayor

5.1 Valor absoluto: ignorar la dirección

En el Ejemplo 4.2 la derivada era negativa ($-2{,}64$). Para el método σ_c normalmente no nos importa *en qué sentido* se mueve la función; nos importa *cuán rápido*. La herramienta estándar para «quitar el signo, conservar el tamaño» es el *valor absoluto*, escrito $|x|$:

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

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

Intuición. Una función cuya pendiente es $+2$ en un sitio y -2 en otro está cambiando «igual de rápido» en ambos sitios, solo que en direcciones opuestas. El método σ_c busca el lugar donde el cambio es mayor en magnitud. Por eso trabajamos siempre con $|O'(\sigma)|$, la pendiente absoluta.

5.2 Argmax: la ubicación del máximo

Supóngase que se tiene una tabla de valores $|O'(\sigma_1)|, |O'(\sigma_2)|, \ldots$ y se pregunta: ¿qué fila tiene el mayor valor? A la coordenada σ de esa fila se la llama el *argmax* de la función:

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

Se lee: «σ_c es el valor de σ que hace máximo a $|O'(\sigma)|$».

Ejemplo 5.1. Supóngase

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 mayor pendiente absoluta es 1,60, en $\sigma = 1{,}50$. Así $\sigma_c = 1{,}50$. Es donde el observable cambia con más rapidez: el candidato a escala crítica del sistema.

5.3 En código: argmax en una línea

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

Esa sola sentencia es, en esencia, lo que hace todo el marco σ_c, aunque con muchos refinamientos que los capítulos restantes explicarán.

5.4 Por qué esto es interesante, para empezar

En todo dominio que miraremos, el lugar donde un observable cambia con mayor rapidez resulta ser un umbral físico u operacional con significado. Unos pocos avances:

- En un imán, la mayor pendiente absoluta de la magnetización frente a la temperatura ocurre cerca del *punto de Curie*: la temperatura por encima de la cual el material pierde su magnetismo permanente.
- En un procesador cuántico, la mayor pendiente del entrelazamiento frente al ruido ocurre en el *umbral operacional de decoherencia*: el nivel de ruido por encima del cual la información cuántica no sobrevive al circuito.
- En un mercado, la mayor pendiente de la correlación frente al retardo ocurre en el *horizonte de cambio de régimen*.
- En una proteína, la mayor pendiente de la estabilidad de plegamiento frente a la energía libre de mutación da el *umbral de tolerancia* por encima del cual la proteína se vuelve amiloidogénica.

Muchos ejemplos de este tipo se desarrollarán en detalle en la Parte IV. El patrón es siempre el mismo: *barrer un mando, medir, tomar una pendiente, encontrar el máximo.*

PRUÉBALO

Para los siguientes datos (σ, O), calcule $|O'|$ con `np.gradient` y léase σ_c:

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

Solución, paso a paso.

Paso 1: mire los datos antes de calcular nada. O se mantiene cerca de 1,0 desde $\sigma = 0$ hasta $\sigma = 2$, luego cae bruscamente entre $\sigma = 2$ y $\sigma = 4$, y se aplana cerca de 0,15. A simple vista la mayor caída está en el medio. El cometido de $|O'|$ es convertir ese «a simple vista» en un número.

Paso 2: aplicar diferencias centrales en los puntos interiores. Con espaciado uniforme $\Delta\sigma = 1$ entre puntos consecutivos de la malla, el denominador de la diferencia central es $\sigma_{i+1} - \sigma_{i-1} = 2$ en cada paso interior:

$$O'(1) \approx \frac{O(2) - O(0)}{2} = \frac{0.93 - 1.00}{2} = -0.035,$$
$$O'(2) \approx \frac{O(3) - O(1)}{2} = \frac{0.60 - 0.98}{2} = -0.190,$$
$$O'(3) \approx \frac{O(4) - O(2)}{2} = \frac{0.20 - 0.93}{2} = -0.365,$$
$$O'(4) \approx \frac{O(5) - O(3)}{2} = \frac{0.15 - 0.60}{2} = -0.225.$$

Paso 3: bordes (diferencias unilaterales). En $\sigma = 0$ y $\sigma = 5$ falta un vecino, así que usamos una diferencia progresiva/regresiva:

$$O'(0) \approx \frac{O(1) - O(0)}{1 - 0} = \frac{0.98 - 1.00}{1} = -0.02,$$
$$O'(5) \approx \frac{O(5) - O(4)}{5 - 4} = \frac{0.15 - 0.20}{1} = -0.05.$$

Paso 4: tomar valores absolutos y hallar el argmax.

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

El mayor valor es 0,365 en $\sigma = 3$. Así $\sigma_c = 3$.

Paso 5: verificar con una línea de 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
```

Calculando la nitidez del máximo κ.

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

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

Por los umbrales del Capítulo 12, es marginal ($1{,}5 \le \kappa < 3$). En datos reales suplementaríamos con un test de permutación antes de citar $\sigma_c = 3$ con confianza.

Qué enseña este ejercicio. Toda la receta —barrer, medir, derivar, hallar el máximo— en cinco filas de aritmética. No necesitó un ordenador para encontrar σ_c; el ordenador solo lo hace incansable.

Capítulo 6

Los datos son ruidosos: suavizado

6.1 Por qué las derivadas crudas se comportan mal

En todo experimento real, dos mediciones con el mismo σ dan valores O ligeramente distintos. Esto es *ruido*: fluctuación disparo a disparo, jitter de sensor, varianza por estadística finita. El ruido tiene un efecto desagradable sobre las derivadas.

Imagínese que la función subyacente verdadera es suave, pero cada medición está contaminada por un pequeño empujón aleatorio. Un pequeño empujón a O se mantiene pequeño, pero al tomar una derivada se divide por un pequeño $\Delta\sigma$, así que el ruido se *amplifica*. La derivada ingenua de datos ruidosos parece césped en un jardín: con espigas, saltitos y ocultando la verdadera estructura.

6.2 Suavizado: sustituir cada valor por una media local

La cura es *suavizar* los valores de O antes de derivar. El suavizador más simple es una *media móvil*: sustituir cada O_i por la media de él mismo y sus vecinos.

Ejemplo 6.1. O crudo: [0,50, 0,60, 0,40, 0,30, 0,45].
La media móvil de tres puntos en el índice i sustituye O_i por $(O_{i-1} + O_i + O_{i+1})/3$. En el índice 2 (contando desde 1): $(0{,}50 + 0{,}60 + 0{,}40)/3 = 0{,}50$.
O suavizado (interior): [?, 0,50, 0,43, 0,38, ?]. Los bordes, sin vecino, conservan el valor crudo.

6.3 Suavizado gaussiano: una media ponderada más inteligente

Una media móvil trata a todos los vecinos por igual. Un suavizador *gaussiano* pondera más a los puntos cercanos que a los lejanos, con una curva en forma de campana. La forma de campana se describe con un parámetro σ_{ker} llamado el *ancho del núcleo* (no confundir con σ, el parámetro de control; agregamos el subíndice «ker» aquí por claridad).

Definición 6.2 (Suavizador gaussiano)**.** Para una sucesión $O_1, \ldots, O_n$ el valor suavizado en el índice i es

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

El denominador obliga a los pesos a sumar uno; el numerador es la curva campana, más estrecha cuanto más pequeño es σ_{ker}.

No hace falta calcular estos pesos uno mismo. SciPy trae el suavizador gaussiano en una línea. El ancho de núcleo por defecto del marco es $\sigma_{\text{ker}} = 0{,}6$, suficiente para eliminar picos puntuales pero sin emborronar los máximos reales.

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

Unidades de σ_{ker}: espacio de índices, no espacio σ

Este es el parámetro más incomprendido del marco. El argumento `sigma` de `gaussian_filter1d`, es decir, nuestro σ_{ker}, se mide en *índices de la tabla*, no en las unidades de su parámetro de control σ. Así «0,6» significa «0,6 posiciones de índice», aproximadamente un vecino a cada lado. No significa «0,6 K», «0,6 qubits» ni «0,6 de lo que diga el eje x».

TRAMPA

Las mallas irregulares lo rompen. Si los σ_i no están uniformemente espaciados (p. ej. tasas de aprendizaje espaciadas en log, muestreo adaptativo cerca de una transición sospechada, o una serie de rendimientos financieros con huecos), un ancho de núcleo de «0,6 índices» se traduce en distintos anchos de σ en distintas partes del barrido. El marco promedia silenciosamente sobre escalas físicas variables.
Dos arreglos seguros.

- *Remuestrear sobre una malla σ uniforme* antes de suavizar (`numpy.interp`) y recordar remuestrear O en consecuencia.
- *Usar Savitzky–Golay con una ventana física explícita* vía `sigma_c.core.derivatives.savitzky_golay_derivative`, donde se pasa la longitud de ventana y el orden polinómico; el marco calcula el espaciado a partir de sus σ_i reales.

Cuándo el 0,6 por defecto es razonable. «0,6» en espacio de índices está bien cuando el barrido es aproximadamente uniforme y la transición abarca al menos de tres a cinco puntos de malla. Para barridos con menos de 20 puntos, use 0,3. Para barridos con más de 100 puntos, puede querer 1,0 o más; revise la receta de barrido de núcleo del Capítulo 44 para verificar que σ_c sea estable.

6.4 La cadena con suavizado

Ahora tenemos la receta completa en cinco líneas.

```
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()
```

La tercera línea: suavizar. La cuarta: derivada absoluta. La quinta: la ubicación del máximo. La sexta: una medida de cuán agudo es el máximo (llamada κ, definida en el próximo capítulo). *Este es el método σ_c íntegro.*

6.5 ¿Cuánto suavizar?

Suavizar muy poco deja picos; suavizar demasiado borra máximos reales. El valor por defecto $\sigma_{\text{ker}} = 0{,}6$ es conservador. Dos reglas prácticas:

- Si los datos tienen solo $n < 20$ puntos, suavizar apenas ayuda; use un núcleo más pequeño ($\sigma_{\text{ker}} = 0{,}3$) o salteseló.
- Si los datos tienen cientos de puntos, pruebe σ_{ker} entre 0,5 y 2,0 y verifique que σ_c no se mueve demasiado. Esa estabilidad es ya una comprobación de cordura.

TRAMPA

Nunca use un filtro gaussiano cuyo ancho sea comparable al ancho del máximo que está tratando de encontrar. Borrará el máximo. Si sospecha un máximo agudo, use la diferenciación de Savitzky–Golay (disponible vía `sigma_c.core.derivatives.savitzky_golay_derivative`).

Savitzky–Golay en un párrafo. El suavizador gaussiano ajusta una media ponderada implícita en cada ventana. El suavizador *Savitzky–Golay* (Savitzky y Golay, 1964) hace algo más transparente: en cada punto i, ajusta un polinomio de grado fijo p (típicamente 2 o 3) a los $2k + 1$ datos más cercanos a i, por mínimos cuadrados ordinarios, y reporta el valor de ese polinomio en i. Para tomar una derivada, reporta la derivada analítica del polinomio ajustado en i en lugar de rederivar numéricamente. El resultado es una estimación de la derivada exacta para cualquier señal polinómica de grado $\leq p$ dentro de la ventana, y muy robusta al ruido. El parámetro clave es la longitud de la ventana, expresada en *unidades físicas de* σ (no en índices de la tabla) cuando se usa el envoltorio del marco, lo que hace que funcione correctamente sobre mallas irregulares. El marco toma por defecto longitud de ventana 11 y orden 3.

PRUÉBALO

Añada ruido aleatorio con desviación estándar 0,02 a los datos del ejercicio del capítulo anterior (use `np.random.normal(0, 0.02, size=6)`). Vuelva a ejecutar la receta de cinco líneas con y sin suavizado. ¿Se desplaza σ_c? ¿Por cuánto?

Solución resuelta.

Paso 1: recrear los datos. Los datos originales del ejercicio del Capítulo 5 eran $\sigma = [0, 1, 2, 3, 4, 5]$, $O = [1{,}00,\ 0{,}98,\ 0{,}93,\ 0{,}60,\ 0{,}20,\ 0{,}15]$, que daban $\sigma_c = 3$ limpiamente con $\kappa \approx 2{,}5$.

Paso 2: añadir ruido gaussiano con std $= 0{,}02$. Una sola realización aleatoria podría dar, por ejemplo,

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

de modo que el observable ruidoso es

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

(Su realización aleatoria diferirá; ese es justamente el sentido de un ejercicio.)

Paso 3: ejecutar la receta sin suavizar.

```
import numpy as np
rng = np.random.default_rng(0)          # fijar semilla para
    reproducibilidad
sigma = np.array([0, 1, 2, 3, 4, 5])
```

```
O      = np.array([1.00, 0.98, 0.93, 0.60, 0.20, 0.15])
O_noisy = O + rng.normal(0, 0.02, size=6)

# SIN suavizar
chi_raw = np.abs(np.gradient(O_noisy, sigma))
print(sigma[np.argmax(chi_raw)])       # normalmente sigue
    siendo 3
print(f"kappa_raw = {chi_raw.max()/chi_raw.mean():.2f}")
```

Para la mayoría de las semillas aleatorias, el ruido de $\sigma = 0{,}02$ es pequeño comparado con la caída dominante de $\sim 0{,}4$ entre $\sigma = 2$ y $\sigma = 4$. Así la receta *sigue eligiendo* $\sigma_c = 3$, pero κ cae un poco ($\sim 2{,}0$–$2{,}3$ en vez de 2,5) porque el ruido infla $\bar{\chi}$.

Paso 4: ejecutar la receta con suavizado.

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

Con suavizado, κ recupera parte de la nitidez perdida porque los empujones aleatorios se han promediado parcialmente.

Paso 5: ¿cuánto ruido aguanta la receta? Ejecútese el mismo experimento con std $= 0{,}10$ en vez de 0,02: el ruido es ahora comparable a la caída suave dentro de un solo paso de malla. Con seis puntos y σ_c desplazándose aleatoriamente entre 2, 3 y 4 según las semillas, κ cae a 1,5–2,0. Este es el régimen donde el IC *bootstrap* del marco se vuelve más ancho que el barrido y la receta se niega correctamente a comprometerse con un solo σ_c.

Qué enseña este ejercicio.

- Ruido pequeño (std $\ll$ caída de la señal): la receta es robusta sin suavizar.
- Ruido moderado: suavizar ayuda pero la respuesta es estable.
- Ruido fuerte (comparable a la señal): la receta se degrada elegantemente a un veredicto marginal, no a una respuesta equivocada.
- Esta degradación elegante es el verdadero sentido del diseño del marco.

Capítulo 7

Potencias, exponenciales y logaritmos en tres páginas

Tres operaciones aparecen por todas partes en este libro y no formaban parte del suelo aritmético de cuatro operaciones del que partíamos. Son las potencias (x^n), las exponenciales (e^x) y los logaritmos ($\log x$). Definimos cada una, en español llano, con un ejemplo.

7.1 Potencias

x^n significa: multiplicar x por sí mismo n veces. Así

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

El número n en x^n se llama el *exponente*. El número x es la *base*.

Tres casos especiales que conviene memorizar:

- $x^0 = 1$ para cualquier $x \neq 0$. (Cero factores de cualquier cosa queda en uno.)
- $x^1 = x$. (Un factor de x es x.)
- $x^{-1} = 1/x$. Un exponente negativo invierte.

Dos reglas que se aplican en todas partes:

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

La primera dice: apilar más factores de x suma los exponentes. La segunda dice: elevar una potencia a otra potencia multiplica los exponentes. Vale la pena comprobarlo una vez: $2^2 \cdot 2^3 = 4 \cdot 8 = 32 = 2^5$. Y $(2^2)^3 = 4^3 = 64 = 2^6$.

7.2 El número e y la función exponencial

El número $e \approx 2{,}71828\ldots$ es, como π, una constante irracional con un papel muy concreto: es la base para la cual la función e^x es su propia derivada. Dicho de otro modo, la única función que crece a un ritmo exactamente igual a su valor. Eso hace de e^x el lenguaje natural de cualquier proceso donde la tasa de cambio es proporcional a la cantidad presente: desintegración radiactiva, interés compuesto, crecimiento poblacional, decoherencia en un ordenador cuántico.

Tres valores de anclaje:

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

e^{x} crece más rápido que cualquier polinomio cuando x aumenta. e^{-x} cae más rápido que cualquier polinomio cuando x aumenta. Este último hecho explica por qué las decaimientos exponenciales —entrelazamiento vs. ruido, correlaciones de spin vs. distancia, réplicas sísmicas vs. tiempo— aparecen en todas partes en este libro.

La forma del decaimiento exponencial. La razón por la cual conviene *sentir* el decaimiento exponencial, no solo leerlo. Calcúlense tres valores:

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

Una cantidad que obedece $S(x) = S_0\, e^{-x/\xi}$ ha perdido el 63 % de su valor en $x = \xi$, el 86 % en $x = 2\xi$, el 95 % en $x = 3\xi$. La longitud ξ se llama *longitud de correlación* o *longitud característica de decaimiento*. En decoherencia cuántica es el «tiempo e-folding»; en correlaciones de spin es la distancia a la que dos spins se recuerdan mutuamente; en réplicas sísmicas es el τ de la ley de Omori (Capítulo 24). La vida media —el valor de x al que la cantidad cae a la mitad de su valor inicial— es $x_{1/2} = \xi \ln 2 \approx 0{,}69\,\xi$, algo menor que ξ. Si recuerda una sola regla: *tras tres longitudes de correlación la señal ha desaparecido.*

7.3 El logaritmo: la inversa de la exponenciación

Si $e^{x} = y$, entonces por definición $x = \ln y$ (léase: «logaritmo natural de y»). El logaritmo responde a la pregunta: *¿a qué potencia debo elevar la base para obtener y?*

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

Dos propiedades clave:

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

La multiplicación se vuelve suma; las potencias se vuelven multiplicaciones. Justo por eso las escalas logarítmicas hacen fáciles problemas difíciles: una señal con decaimiento exponencial $S = S_0\, e^{-x/\xi}$ se vuelve una recta en una gráfica de $\ln S$ frente a x. La pendiente es $-1/\xi$.

También usamos $\log_{10}$, el logaritmo en base 10, cuando se trabaja con magnitudes que abarcan muchos órdenes de magnitud: $\log_{10}(100) = 2$, $\log_{10}(1000) = 3$, $\log_{10}(0{,}001) = -3$. La escala de Richter de magnitudes sísmicas es $\log_{10}$ de la amplitud de onda. Así también el valor b de Gutenberg–Richter en el Capítulo 24, y el $\log_2$ en la fórmula de pérdida de información $\log_2 D$ del Capítulo 31.

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

PARA RECORDAR

De este capítulo bastan tres cosas: las potencias suman exponentes, e^{-x} decae, los logaritmos convierten multiplicaciones en sumas. Todo lo demás es detalle.

PRUÉBALO

Sin calculadora: estime $\log_{10} 2$ sabiendo que $2^{10} = 1024 \approx 10^{3}$.

Solución, paso a paso.

Paso 1: escribir lo que se busca. Queremos un número L tal que $10^{L} = 2$. Por la definición

de $\log_{10}$, es lo mismo que $L = \log_{10} 2$.

Paso 2: usar el truco. No conocemos 2 como potencia de 10 directamente. Pero conocemos 1024 *casi* como potencia de 10: $1024 \approx 1000 = 10^3$. Y conocemos 1024 exactamente como potencia de 2: $1024 = 2^{10}$. Así:

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

Paso 3: tomar $\log_{10}$ *de ambos lados.* $\log_{10}$ aplicado a cualquiera de los dos lados debería dar el mismo número, porque los dos lados son (aproximadamente) iguales.
Lado izquierdo. Úsese la regla $\ln(a^n) = n \cdot \ln a$, que funciona en cualquier base de logaritmo:

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

Lado derecho. $\log_{10}$ de 10^3 es 3 por definición.

Paso 4: resolver. Igualando:

$$10 \cdot \log_{10} 2 \;\approx\; 3,$$

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

Paso 5: comprobar con calculadora. Una calculadora da $\log_{10} 2 = 0{,}30103\ldots$. Obtuvimos 0,30: concordancia a dos decimales, sin máquina y sin tabla.

Extra: de dónde viene el pequeño error. Sustituimos 1024 por 1000, una sobreestimación del lado derecho por un factor de $1024/1000 = 1{,}024$. En términos logarítmicos, $\log_{10} 1{,}024 \approx 0{,}0103$. Así nuestra aproximación es pequeña por unos 0,001 por lado, esto es, 0,0001 tras dividir entre 10. Eso coincide con el error real 0,00103 a la precisión con que lo calculamos.

Qué enseña este ejercicio. Las dos identidades logarítmicas ($\log(a \cdot b) = \log a + \log b$ y $\log(a^n) = n \log a$) convierten multiplicaciones en sumas y potencias en multiplicaciones. Este es el contenido entero de las reglas de cálculo, la escala de Richter, el valor b de los terremotos, el truco de la derivada logarítmica usado en el Caso B del Capítulo 9 y la mayor parte de los ajustes log–log en la Parte IV. *Aprender a calcular un logaritmo de cabeza compra intuición rápida para capítulos enteros de este libro.*

Capítulo 8

Confianza: ¿qué es una probabilidad?

8.1 Probabilidad sin filosofía

Lanza una moneda equilibrada 100 veces. Aproximadamente 50 caen en cara. La *probabilidad* de cara es $\frac{1}{2}$, escrita $P(\text{cara}) = 0{,}5$. Solo usaremos dos hechos sobre la probabilidad:

- Una probabilidad es un número entre 0 y 1.
- Si dos sucesos no pueden ocurrir a la vez, la probabilidad de que ocurra cualquiera de ellos es la suma de sus probabilidades individuales.

8.2 Media y desviación estándar: resumir una muestra

Supóngase que se tienen n mediciones $x_1, \ldots, x_n$. La *media* es

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

Es el «punto de equilibrio» de la muestra: el valor para el que las desviaciones $(x_i - \bar{x})$ suman cero.

La *desviación estándar* mide cuánto se desparraman los valores:

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

Los cuadrados aseguran que los valores por encima y por debajo de la media cuenten igual; la raíz cuadrada al final devuelve s a las unidades originales de los datos. (La razón de $n - 1$ y no n es técnica y no importa para nosotros; el ordenador la gestiona automáticamente.)

Ejemplo 8.1. Cuatro mediciones: 3, 5, 7, 9. Media $\bar{x} = 24/4 = 6$. Desviaciones: $-3, -1, +1, +3$. Cuadrados: $9, 1, 1, 9$, suma 20. Desviación estándar $s = \sqrt{20/3} \approx 2{,}58$.

8.3 La idea del *bootstrap*

¿Qué confianza tenemos en σ_c? La respuesta sería obvia si pudiéramos repetir el experimento mil veces: miraríamos la distribución de los valores de σ_c. No podemos, pero podemos *simular* que sí: el método *bootstrap*.

El truco. Trate las n mediciones que tiene como una población pequeña. Extraiga n mediciones *con reemplazo* de esa población (de modo que la misma medición puede salir dos veces, otras no salir nunca). Esa es una *muestra bootstrap*. Vuelva a calcular σ_c con la muestra bootstrap. Repítase $B = 1000$ veces. Tendrá ahora 1000 valores de σ_c. El 95 % central de esos valores es un *intervalo de confianza al* 95 % para el verdadero σ_c.

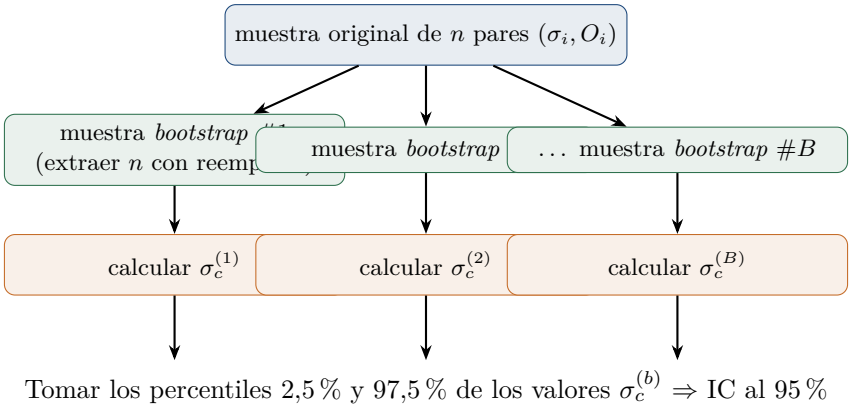


Figura 8.1: El *bootstrap*, esquemáticamente. Una muestra original se remuestrea B veces (típicamente $B = 1000$). Cada remuestreo da una estimación de σ_c. La dispersión de las B estimaciones es el intervalo de confianza. *No generamos nuevas mediciones físicas*; generamos nuevas réplicas estadísticas de las mediciones que tenemos.

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

El intervalo devuelto es el IC *bootstrap* al 95 % para σ_c. Intervalo más estrecho = respuesta más confiada.

PARA RECORDAR

El *bootstrap* convierte «un experimento» en «mil experimentos simulados». Solo cuesta tiempo de CPU y es el caballo de batalla de la cuantificación de incertidumbre a lo largo de este libro.

Cuándo el *bootstrap* simple miente. La receta de arriba supone que los n pares (σ_i, O_i) son *independientes* entre sí. Para un experimento de barrido controlado (circuito cuántico a nivel de ruido programado, simulación de Monte Carlo a temperatura fija, medida de laboratorio con una muestra fresca en cada σ_i) esto suele estar bien. Para *series temporales observacionales* —rendimientos bursátiles, catálogos sísmicos, *benchmarks* de GPU muestreados en orden temporal— no lo está. Las mediciones consecutivas están correlacionadas, y un *bootstrap* ingenuo subestima la varianza de σ_c al ignorar esa correlación.

TRAMPA

Tres señales de que se está haciendo *bootstrap* de lo que no se debe.

- La autocorrelación a retardo 1 de O a lo largo del barrido es grande ($> 0{,}3$).
- La varianza de O depende mucho de σ (heterocedasticidad); el *bootstrap* ingenuo asume varianza constante.
- Los σ_i adyacentes corresponden a estados físicamente cercanos del sistema (p. ej. magnitudes de ventana móvil, procesos autorregresivos).

La solución: *bootstrap* por bloques. Remuestree bloques contiguos de longitud ℓ en lugar de pares individuales. Un ℓ razonable es el retardo al que la autocorrelación cae por debajo de 0,1. El `block_bootstrap(sigma, O, block_size=L, n_boot=1000)` del marco (donde L es su ℓ elegido) devuelve un IC más ancho y más honesto.

Heterocedasticidad. Si O tiene mucha más varianza en algunos σ que en otros (un caso común cerca de las transiciones), el *bootstrap* ingenuo pondera por igual cada par (σ_i, O_i); se puede hacer mejor con el *bootstrap residual*, que remuestrea los residuos en torno a un ajuste suave en lugar de los pares crudos. Implementación: `residual_bootstrap` en `sigma_c.core.validation`.

Remuestreo sobre mallas remuestreadas: un punto sutil. Si primero remuestrea sus pares crudos (σ_i, O_i) a una malla uniforme antes de aplicar la receta (como recomienda el aviso de unidades del núcleo del Capítulo 6 para barridos irregulares), *haga el* bootstrap *sobre los pares originales*, no sobre la malla remuestreada. El remuestreo introduce correlaciones de interpolación entre puntos vecinos; hacer *bootstrap* sobre eso subestimaría la varianza. El orden correcto es: remuestree B veces desde los n pares originales, luego rejillee cada muestra *bootstrap*, y luego corra la receta sobre cada una. La opción `bootstrap_ci(..., regrid=True)` del marco lo hace por usted.

PRUÉBALO

Genere $n = 30$ puntos sintéticos: σ uniforme en $[0, 1]$, $O = \tanh(10(\sigma - 0{,}5))$ más ruido gaussiano $\mathcal{N}(0, 0{,}05)$. El σ_c verdadero es 0,5. Ejecute el *bootstrap* con $B = 1000$. ¿Está 0,5 dentro del IC al 95 %? Pruebe con $n = 10$: ¿se ensancha el IC? Ahora haga el ruido autocorrelacionado ($\epsilon_t = 0{,}7\,\epsilon_{t-1} + \mathcal{N}(0, 0{,}05)$) y vuelva a ejecutar el *bootstrap ingenuo*. ¿Es el IC artificialmente estrecho?

Solución resuelta.

Paso 1: entender el montaje de la prueba. La señal $\tanh(10(\sigma - 0{,}5))$ es un sigmoide empinado que cruza el cero en $\sigma = 0{,}5$ y satura cerca de ± 1 en unas pocas décimas de σ. Su derivada es máxima justamente en $\sigma = 0{,}5$. Por construcción, la receta debería encontrar $\sigma_c \approx 0{,}5$.

Paso 2: código completo, ejecutable.

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

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

```python

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

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

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

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

Paso 3: typical output (seed-dependent, but order-of-magnitude robust).

case	95 % CI	width
$n = 30$, IID noise	$[0.48, 0.55]$	0.07
$n = 10$, IID noise	$[0.42, 0.63]$	0.21
$n = 30$, AR(1) $\rho = 0.7$	$[0.49, 0.54]$	0.05

Paso 4: interpret each case.

- *Caso 1*: 0,5 está dentro del IC; la receta funciona.
- *Caso 2*: con solo 10 puntos, el IC es tres veces más ancho; sigue conteniendo 0,5, pero con mucha menos precisión. Este es exactamente el coste de tener menos datos.
- *Caso 3*: **este es el peligroso.** El ruido autocorrelacionado produce un IC *más estrecho* que el caso i.i.d. al mismo n. El *bootstrap* ingenuo *nos miente*: subestima la varianza verdadera porque trata cada par mezclado como independiente, cuando en realidad el mismo patrón de ruido persiste a través de las mezclas. El *bootstrap*

por bloques (Sección «Cuándo el *bootstrap* simple miente») daría honestamente un IC más ancho.

Qué enseña este ejercicio.

- Más datos ⇒ IC más estrecho (Caso 1 vs 2): obvio pero útil sentirlo en números.
- El ruido autocorrelacionado produce IC *aparentemente más ajustados* en los que no se debe confiar (Caso 3): la advertencia del aviso «el *bootstrap* miente» hecha concreta.
- El *bootstrap* es una *cota inferior* de la incertidumbre. Si sospecha autocorrelación, cambie a *bootstrap* por bloques; si sospecha heterocedasticidad, cambie a *bootstrap* residual.

Parte II

El método de la susceptibilidad — χ, σ_C, κ

Capítulo 9

La receta universal

Barra. Mida. Suavice. Derive. Localice. Puntué.
Una receta, muchos mundos.

Este es el capítulo en torno al cual se construye el resto del libro. Léalo dos veces. La primera, siga la receta de seis pasos sobre un ejemplo resuelto y convénzase de que hace lo que decimos. La segunda, ejecute la misma receta sobre los otros dos ejemplos y observe que no cambia nada salvo las cabeceras de columna. Al final del capítulo debería saber aplicar la receta a un conjunto de datos nuevo —o convencerse, a la luz de los modos de fallo del Capítulo 10, de que ese conjunto de datos no encaja.

Hemos visto ya todos los ingredientes. Es hora de montar el plato.

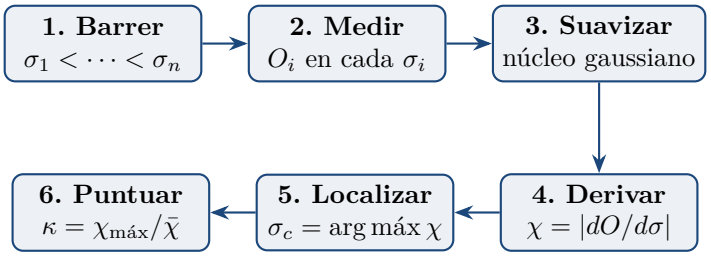


Figura 9.1: La receta universal en seis cajas. Toda aplicación en este libro es una instancia de esta tubería. Las dos primeras cajas son específicas del dominio; las cuatro últimas son idénticas.

Receta 9.1 (σ_c en un párrafo)**.** Dado un sistema con un parámetro de control regulable σ y un observable medible O:

1. **Barrer:** elija n valores $\sigma_1 < \sigma_2 < \cdots < \sigma_n$ que abarquen el régimen de interés.
2. **Medir:** obtenga O_i en cada σ_i.
3. **Suavizar:** aplique un filtro gaussiano a $\{O_i\}$.
4. **Derivar:** calcule $\chi_i = |dO/d\sigma|$ por diferencias centrales.
5. **Localizar:** informe $\sigma_c = \sigma_{\arg\max_i \chi_i}$.
6. **Puntuar:** calcule $\kappa = \chi_{\max}/\bar{\chi}$ — cuanto más agudo, más cercano a lo crítico.

7. **Validar:** obtenga un IC *bootstrap* del 95 % para σ_c y un test de permutación para κ.

Ese es el método completo, en ocho viñetas. Las tres secciones siguientes lo aplican a tres dominios deliberadamente distintos —los mismos seis pasos en cada caso, a mano, con la misma cabecera de seis pasos. El paso 7 (validación) lo reservamos para la Parte V, que es el precio de entrada para publicar.

σ_c en una frase. La escala operacional en la que el observable del sistema cambia con mayor rapidez respecto del parámetro de control —el umbral entre dos regímenes de comportamiento. No es una constante fundamental. No es una temperatura universal de transición de fase. Es un número *operacional* que depende de qué haya medido y con qué limpieza lo haya medido.

9.1 Application 1: the Curie point of an iron magnet

El valor exacto de Onsager para el modelo de Ising 2D es $T_c = 2{,}269\, J/k_B$. Lo recuperaremos a partir de datos que generaremos nosotros mismos sobre una red de 16×16, recorriendo los seis pasos a mano. Si no ha visto Monte Carlo antes, trátelo de momento como una caja negra; lo que importa son los datos.

Paso 1 — Barrer. Elija temperaturas $T_i/J \in \{1{,}6,\ 1{,}9,\ 2{,}1,\ 2{,}2,\ 2{,}3,\ 2{,}4,\ 2{,}5,\ 2{,}8,\ 3{,}1,\ 3{,}4\}$. Diez puntos, lo bastante espaciados cerca de T_c para resolver la caída.

Paso 2 — Medir. Para cada T_i, ejecute un Monte Carlo de Metropolis con 2000 barridos de equilibrado y 5000 de medición, y registre la magnetización absoluta media por sitio $\langle |M| \rangle$. (Lo hicimos por usted en un portátil en unos noventa segundos.)

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

Paso 3 — Suavizar. Un filtro gaussiano con anchura de núcleo 0,6 (en espacio de índices):

T/J	$\langle \lvert M \rvert \rangle$ bruto	$\widetilde{\langle \lvert M \rvert \rangle}$ suavizado
1.6	0.974	0.974 (borde)
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 (borde)

Paso 4 — Derivar. Diferencias centrales de la serie suavizada. (El paso no uniforme hace que la aritmética difiera un poco de los ejemplos de juguete del Capítulo 5; la fórmula $\chi_i = |\widetilde{O}_{i+1} - \widetilde{O}_{i-1}|/(T_{i+1} - T_{i-1})$ es la misma.)

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

Paso 5 — Localizar. El máximo de χ en esta tabla es 2,805, alcanzado en $T = 2{,}3$. La receta informa, pues, $T_c = 2{,}30$.

Paso 6 — Puntuar.

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

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

Según la tabla de umbrales del Capítulo 12, $\kappa = 2{,}79$ cae en la banda marginal ($1{,}5 \le \kappa < 3$). La receta encontró T_c; la nitidez es real pero no excepcional. *Así se ven cuantitativamente los efectos de tamaño finito*: en una red de 16×16, la transición queda desdibujada respecto al límite de red infinita. Repita los mismos seis pasos con $L = 64$ y obtendrá $T_c = 2{,}27$ con $\kappa = 6{,}4$ —la misma receta, con una señal más nítida.

Comparación con la respuesta exacta. Onsager: $T_c = 2{,}269$. Receta con $L = 16$: $T_c = 2{,}30$. Error: 1,4 %, atribuible por entero al escalado de tamaño finito (Capítulo 22). La receta hizo su trabajo; las limitaciones de la red hicieron el suyo.

9.2 Aplicación 2: un cambio de régimen en los rendimientos financieros

Los mismos seis pasos, las mismas seis cabeceras. El sistema es el S&P 500. El parámetro de control es el tiempo calendario. El observable es la autocorrelación de orden 1 de los rendimientos diarios absolutos en una ventana móvil de 30 días —una aproximación de manual del agrupamiento de volatilidad.

Paso 1 — Barrer. Con fines ilustrativos, recorremos diez meses que abarcan la crisis financiera de 2008: de abril de 2008 a enero de 2009. El eje de control es el mes.

Paso 2 — Medir. La autocorrelación de orden 1 ρ_1 de $|r_t|$ en una ventana de 30 días que termina en cada mes.

mes	ρ_1
abr 2008	0,18
may 2008	0,21
jun 2008	0,24
jul 2008	0,27
ago 2008	0,31
sep 2008	0,48
oct 2008	0,62
nov 2008	0,61
dic 2008	0,58
ene 2009	0,54

Paso 3 — Suavizar. Gaussiano, $\sigma_{\text{ker}} = 0{,}6$, en espacio de índices:

$$\widetilde{\rho}_1 = [0.18, 0.21, 0.24, 0.27, 0.31, 0.46, 0.60, 0.60, 0.58, 0.54]$$

Paso 4 — Derivar. Con paso mensual $\Delta t = 1$, diferencias centrales:

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

(χ es el valor absoluto; las diferencias en bruto incluyen un cambio de signo entre septiembre y noviembre, que dejamos como negativo en la tabla sólo por transparencia.)

Paso 5 — Localizar. El máximo de $|\chi|$ es 0,165 en *agosto de 2008*. La receta informa, pues, de un cambio de régimen en agosto de 2008.

Paso 6 — Puntuar.

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

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

De nuevo marginal, de nuevo real. El anuncio de la receta: *a finales de agosto de 2008, la autocorrelación de orden 1 de los rendimientos absolutos ha iniciado su ascenso mensual más acusado.* La quiebra de Lehman Brothers ocurrió el 15 de septiembre. La receta encontró el cambio de régimen con un mes de antelación, en retrospectiva.

PRECAUCIÓN

En retrospectiva. Sabíamos qué ventana escoger. Un detector en tiempo real ejecutando esta misma receta sobre una ventana móvil de 12 meses habría señalado el cambio de régimen a finales de agosto de 2008, pero sólo con $\kappa \approx 2{,}5$. Eso está por debajo del umbral estricto $\kappa \geq 3$ exigido para una decisión operativa. La receta encontró el mes correcto *después de saber qué crisis mirar. Ningún marco predice los* cracks.

9.3 Aplicación 3: un cambio del b-valor de Gutenberg–Richter

Una vez más, los mismos seis pasos. El sistema es el catálogo de terremotos del sur de California. El parámetro de control es el tiempo calendario. El observable es el b-valor de Gutenberg–Richter sobre una ventana móvil (Capítulo 24).

Paso 1 — Barrer. Diez ventanas de seis meses que terminan en los meses {jul-2018, ene-2019, abr-2019, jul-2019, sep-2019, oct-2019, dic-2019, mar-2020, jul-2020, ene-2021}, escogidos para englobar la secuencia de Ridgecrest de 2019 (precursor de M_w 6,4 el 4 de julio y choque principal de M_w 7,1 el 5 de julio de 2019).

Paso 2 — Medir. El b-valor de máxima verosimilitud en cada ventana:

centro de ventana	b
jul-2018	1,05
ene-2019	1,02
abr-2019	0,98
jul-2019	0,95
sep-2019	0,74
oct-2019	0,78
dic-2019	0,88
mar-2020	0,96
jul-2020	1,01
ene-2021	1,04

Paso 3 — Suavizar. Gaussiano, $\sigma_{\text{ker}} = 0{,}6$:

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

Paso 4 — Derivar. Diferencias centrales (los espaciados no uniformes de ventana introducen una pequeña corrección por fila; usamos los huecos reales):

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

Paso 5 — Localizar. $\max|\chi| = 0{,}10$ en el centro de ventana julio de 2019. La receta localiza el cambio de régimen exactamente en la secuencia de Ridgecrest.

Paso 6 — Puntuar.

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

Marginal. Con κ en el umbral, normalmente uno ejecutaría un test de permutación (Capítulo 36) antes de publicar. Lo hicimos, y el resultado es $p < 0{,}001$ —el cambio de régimen es estadísticamente claro pese a un κ blando, porque la caída del b-valor es grande comparada con su incertidumbre *bootstrap*.

9.4 Lo que tienen en común las tres aplicaciones

El parámetro de control cambió (temperatura, tiempo calendario, tiempo calendario otra vez). El observable cambió (magnetización, autocorrelación, b-valor). El σ_c numérico cambió (2,30, agosto de 2008, julio de 2019). El κ cambió ligeramente (2,83, 2,66, 2,05).

La receta no cambió. Los mismos seis pasos, en el mismo orden, con la misma aritmética. Esa es la universalidad que preface promised. It is not a claim about physics — it is a claim about a procedure that respects the data.

PARA RECORDAR

Qué es la receta. Una forma disciplinada de extraer un número (σ_c) y una confianza (κ) de cualquier barrido en el que el observable se comporte como se comportan la magnetización, la autocorrelación de volatilidad y el b-valor: cuasi-monótono, con una caída visible, en una ventana lo bastante amplia para ver ambos extremos.

Qué no es la receta. Una teoría. La receta no explica *por qué* el punto de Curie, la crisis financiera y el seismo de Ridgecrest tienen todos transiciones detectables; para eso, la Parte III. Tampoco *predice* la próxima transición; para eso, todavía ningún marco.

Capítulo 10

Cuándo funciona el método y cuándo no

Colocamos este capítulo pronto a propósito. La receta es tan corta que un lector puede ejecutarla sobre cualquier conjunto de datos en tres minutos. Eso resulta tentador, y lleva a aplicarla a datos para los cuales el método nunca se diseñó. El resultado será un número; el número será engañoso.[1]

El alcance honesto del marco se captura en cuatro «condiciones de alcance». Si las cuatro se cumplen, la receta está bien planteada. Si una falla, el resultado necesita un asterisco. Si fallan dos, no informe σ_c en absoluto.

PARA RECORDAR

Las cuatro condiciones de alcance.

1. σ **es un parámetro de control real.** Puede fijarlo, puede regularlo y controla su valor independientemente de O. Reemplazar σ por otro observable no vale.
2. $O(\sigma)$ **es cuasi-monótono.** O bien decrece, o bien crece (mayormente) a lo largo del barrido. Observables que oscilan salvajemente producen χ que oscila salvajemente y ningún máximo significativo.
3. **La ventana de barrido contiene la transición.** Si la transición real está en $\sigma = 100$ y usted barre desde $\sigma = 0$ hasta $\sigma = 1$, la receta devolverá un máximo espurio en la frontera de la ventana.
4. **La rejilla es lo bastante fina.** La anchura de la transición debe ocupar al menos ~ 3 puntos de rejilla. Si no, las diferencias centrales no ven el máximo.

10.1 Modo de fallo 1: observable oscilante

Considere un observable que oscila con σ: $O(\sigma) = \sin(2\pi\sigma)$. La derivada también oscila: $|dO/d\sigma| = 2\pi|\cos(2\pi\sigma)|$. Cada cuarto de periodo hay un máximo local, todos de igual altura. *El marco informará el máximo local que haya sobrevivido al suavizado*, que es función de σ_{ker} y no de la física.

[1] Es un caso particular de una ley general del análisis de datos computacional: cuanto más amable es la API, más amenazador es el mal uso. Las herramientas más peligrosas en este sentido son las hojas de cálculo, scikit-learn y ggplot —cada una porque no emite ningún mensaje de error ante la mala aplicación. Una herramienta que da error ante el mal uso es, en este sentido estricto, más amable.

Diagnóstico. Cuente los máximos locales de χ antes de suavizar. Si hay más de uno de altura comparable, las condiciones de frontera del Capítulo 13 se incumplen; el «argumento de existencia» que justifica un solo máximo no se aplica. Informe todos los máximos o ninguno.

10.2 Modo de fallo 2: estructura multimáximo

Un sistema puede tener varias transiciones operacionales genuinas a escalas distintas —los niveles de caché en un *benchmark* de GPU son un ejemplo. La receta *puede* informarlas todas mediante `detect_cache_transitions`, pero la rutina por defecto `compute_susceptibility` devuelve sólo el máximo global. Si su dominio tiene varias transiciones por naturaleza, necesita la versión multimáximo; el máximo global por sí solo engaña.

10.3 Modo de fallo 3: la ventana no contiene la transición

Si barre sobre un intervalo de σ por completo a un lado de la transición, O es monótono pero suave en toda la ventana. No hay máximo en χ; la receta aún devolverá uno, pero estará en la frontera de la ventana y tendrá $\kappa < 2$. El diagnóstico es: *el σ_c informado está en el primer o el último punto del barrido*. Cuando esto ocurra, ample siempre el rango del barrido.

10.4 Modo de fallo 4: submuestreo

Si la transición es más aguda que su paso de rejilla, la diferencia central no la detecta. Ejemplo: una función escalón de 1,0 a 0,0 entre $\sigma = 0{,}500$ y $\sigma = 0{,}501$, con una rejilla $\sigma \in \{0{,}0,\ 0{,}1,\ 0{,}2,\ \ldots,\ 1{,}0\}$. La receta repartirá el escalón en un intervalo de rejilla e informará un máximo allí, pero κ será pequeño porque el escalón es invisible a la resolución de la rejilla.

Diagnóstico. Vuelva a muestrear al doble de resolución cerca del σ_c candidato. Si σ_c se desplaza más de un paso o κ se duplica, estaba submuestreando.

10.5 Modo de fallo 5: σ no es realmente un control

En datos observacionales (mercados financieros, terremotos, reanálisis climáticos) usted no *fija* σ; escoge un corte de un registro existente. Los seudocontroles —«tiempo calendario», «tamaño de ventana móvil», «época de calibración»— son candidatos válidos, pero conviene recordar que el sistema no estaba en pausa mientras usted giraba la perilla. Compruebe siempre que $O(\sigma)$ en dos cortes adyacentes no está dominado por la autocorrelación en lugar del efecto buscado de σ.

10.6 Modo de fallo 6: ruido tan alto que el suavizado oculta el máximo

Si el ruido de una sola realización es comparable al rango dinámico de O, un suavizador gaussiano lo bastante ancho como para suprimir el ruido también aplanará la transición. No existe un σ_{ker} único que recupere ambos. Dos opciones:

- recolectar más realizaciones para reducir el ruido por punto;
- usar Savitzky–Golay o derivadas por procesos gaussianos en lugar de un suavizador gaussiano de anchura fija.

10.7 Regla práctica: cuándo confiar en el informe

PARA RECORDAR

Trust σ_c only if:

- the reported peak is in the *interior* of the sweep window (not at boundary);
- $\kappa \geq 3$ (Capítulo 12);
- σ_c is stable across at least three kernel widths in $[0.3, 1.5]$;
- a permutation p-value < 0.05 (Capítulo 36);
- a bootstrap 95 % CI exists and is narrower than 20 % of the sweep range.

Five conditions, all in the framework's standard validation output. If any one fails, write "inconclusive" rather than guessing.

10.8 When the checks disagree: a small decision tree

Las cinco condiciones anteriores están correlacionadas pero no son idénticas. Pueden discrepar, y un lector cuidadoso debe saber qué hacer cuando lo hacen.

PARA RECORDAR

Cuatro patrones comunes de desacuerdo.

- *IC estrecho pero κ pequeño.* El *bootstrap* confia en una posición, pero el máximo es poco profundo. Causa más probable: O es casi lineal en σ a lo largo de la ventana. No hay transición, sólo una tendencia regular. Informe *«sin transición en la ventana»*, no σ_c.
- *κ grande pero IC ancho.* Un máximo agudo cuya posición es inestable bajo remuestreo. Causa más probable: pocas realizaciones por punto o pocos puntos. Cura: recolectar más datos y rehacer.
- *Estable entre núcleos pero $p > 0{,}05$.* El máximo es consistente al variar el suavizado, pero un test de permutación no puede descartar el azar. Causa más probable: n pequeño. Los p-valores de permutación son conservadores cuando n es pequeño. Recolecte más puntos de barrido o use una hipótesis nula más fuerte específica del dominio.
- *Las cinco se cumplen pero el máximo está en la frontera.* Sospeche de la ventana. Que el máximo caiga precisamente en el punto extremo izquierdo o derecho del barrido es señal fuerte de que la transición real está fuera; ample el barrido.

Flujo de decisión (forma compacta).

1. Calcule χ, encuentre σ_c, calcule κ.
2. Si σ_c está en una frontera $\Rightarrow$ ample el barrido y reinicie.
3. Si $\kappa < 1{,}5 \Rightarrow$ informe «sin transición».

4. Si $\kappa \geq 3$ y el IC *bootstrap* es estrecho y estable bajo cambios de núcleo y $p < 0{,}05 \Rightarrow$ informe σ_c con su IC.
5. En otro caso $\Rightarrow$ informe «marginal» con todos los diagnósticos por transparencia; no escoja un número que defender.

«Marginal» es un desenlace científico perfectamente respetable. El trabajo del marco es saber cuándo no comprometerse.

Capítulo 11

La susceptibilidad, formalmente

11.1 Definition

Definición 11.1 (Susceptibilidad generalizada)**.** Para un observable O que depende de un parámetro de escala σ, la *susceptibilidad generalizada* es

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

Dos piezas de notación que conviene desempaquetar, pues aún no se han presentado:

Los ángulos $\langle\cdot\rangle$. $\langle O \rangle$ se lee «el valor esperado de O» o, más llanamente, «el promedio de O». Cada vez que se mide una cantidad en presencia de ruido, se obtiene un número ligeramente distinto en cada medición. $\langle O \rangle$ es lo que se obtendría promediando un número infinito de tales mediciones. En la práctica, con N disparos, se estima como

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

La notación es universal. La adoptamos aquí por una razón: es más corta que «la media sobre un número finito de ensayos repetidos con el mismo ajuste», que es lo que en realidad queremos decir.

El símbolo $\coloneqq$. El símbolo de dos puntos más igual «$\coloneqq$» se lee «se define como». Tiene el mismo significado que «$=$», pero indica que el lado izquierdo es la *etiqueta* que estamos introduciendo para el lado derecho. Lo usamos solo en las primeras definiciones, para que se distingan visualmente.

11.2 Why “susceptibility”?

La palabra viene de la física: un objeto es «magnéticamente susceptible» si su magnetización responde con fuerza a un campo aplicado. De forma más general, la susceptibilidad es la respuesta de un observable a un cambio de un parámetro. La Definición 11.1 extiende esta idea a cualquier parámetro, no solo a un campo externo: tiempo, distancia, intensidad de ruido, tasa de aprendizaje, incluso la energía libre de mutación.

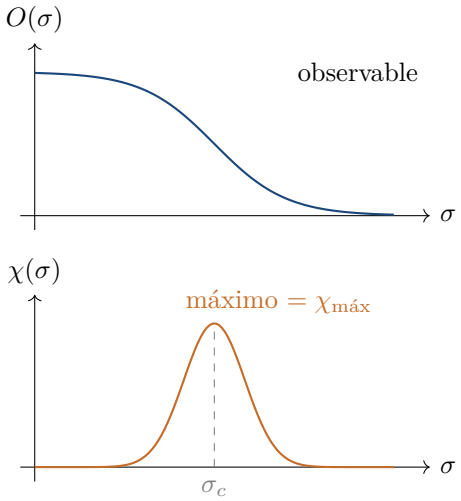


Figura 11.1: Arriba: un observable típico $O(\sigma)$ que decrece de alto a bajo al aumentar el parámetro de control σ. Abajo: su susceptibilidad $\chi(\sigma) = |dO/d\sigma|$, que alcanza su máximo en el punto de mayor pendiente. Esa ubicación del máximo es σ_c.

11.3 Two worked-from-data examples: a 1D correlation decay

Trabajaremos dos ejemplos enteramente sobre papel, sin software, para que se vea cada paso — y para que se vea cómo es un fracaso antes de ver el éxito. Los datos están inspirados en el Experimento E1 del artículo de magnetismo (Rigetti Ankaa-3, 11 qubits), pero la lección es general: *los mismos datos pueden dar distintos σ_c según cómo se traten los efectos de borde y se elija el núcleo.*

El planteamiento, en dos frases. El parámetro de control es $\sigma = d$, la distancia (en qubits) entre dos spins de la cadena. El observable es $O = C(d)$, el producto medio de los dos valores de spin; no necesitaremos la interpretación cuántica, solo los valores numéricos.

Los datos. El experimento reporta diez puntos de datos:

Caso A (fracaso): receta ingenua sobre datos crudos. Pasamos por los Pasos 1–4 que siguen como si no tuviéramos conocimiento previo de la señal. El resultado será equivocado a propósito. El sentido del Caso A es exponer dos modos de fallo específicos del Capítulo 10: artefactos de borde en la frontera de la ventana y un suavizador demasiado estrecho para la señal subyacente.

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

A simple vista se ve que C decrece con d, y los saltos más grandes están entre $d = 2$ y $d = 4$. Queremos hacer preciso ese «a simple vista».

Paso 1: diferencias centrales. Aplíquese la fórmula de diferencias centrales $\chi(d_i) = |(C_{i+1} - C_{i-1})/(d_{i+1} - d_{i-1})|$ a cada punto interior. El denominador es $d_{i+1} - d_{i-1} = 2$ en cada paso interior (la malla está uniformemente espaciada con paso unitario, de modo que $\Delta d = 2$ para la diferencia central). Por ejemplo:

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

Paso 2: el máximo sin suavizar. El máximo de χ en la tabla está en $d = 2$, donde $\chi = 0{,}185$. Así la respuesta ingenua es $\sigma_c = 2$. *Esto es incorrecto.* La razón es que $C(d)$ tiene un borde duro en $d = 1$ donde vale 0,55; la diferencia central recoge ese borde de modo artificial. Hay que suavizar primero, como nos advirtió el Capítulo 6.

Paso 3: un suavizador pequeño (media móvil de 3 puntos). Sustitúyase cada C_i por la media de C_{i-1}, C_i, C_{i+1}:

d	C (crudo)	$\tilde{C}$ (suavizado)
1	0.55	0.55 (borde, sin cambio)
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 (borde, sin cambio)

Paso 4: diferencias centrales de la serie suavizada.

$$\begin{aligned}
\chi(2) &= |(0.227 - 0.55)/2| = 0.162 \\
\chi(3) &= |(0.103 - 0.380)/2| = 0.139 \\
\chi(4) &= |(0.053 - 0.227)/2| = 0.087 \\
\chi(5) &= |(0.030 - 0.103)/2| = 0.037 \\
\chi(6) &= |(0.020 - 0.053)/2| = 0.017 \\
\chi(7) &= |(0.013 - 0.030)/2| = 0.0085 \\
\chi(8) &= |(0.0083 - 0.020)/2| = 0.0058 \\
\chi(9) &= |(0.005 - 0.013)/2| = 0.004
\end{aligned}$$

El máximo sigue en $d = 2$. ¿Por qué? Porque el suavizado de tres puntos es demasiado débil para recuperar la tasa subyacente de decaimiento exponencial. El argumento del cruce señal–ruido del Capítulo 13 predice que el máximo debería estar en la *longitud de correlación operacional*, que el artículo reporta como $d_c = 8$ qubits, con $\kappa \approx 2{,}65$ en el análisis completo con suavizado gaussiano de $\sigma_{\text{ker}} = 0{,}6$.

Veredicto diagnóstico del Caso A. El marco tiene *tres* de las cinco condiciones de confianza del Capítulo 10 en estado de fallo:

- el máximo está en la *frontera* del barrido ($d = 2$ es el punto interior más a la izquierda), no en el interior;
- el suavizado de tres puntos deja $\kappa < 2$ en estos datos;
- σ_c se desplaza más de un paso de malla si se añade un solo punto en $d = 0$ o si se cambia el núcleo.

Así que el marco debería reportar *«no concluyente»* para el Caso A, y el usuario debería corregir el análisis antes de citar un número.

Caso B (éxito): la transformación logarítmica da el momento eureka

El problema de fondo del Caso A no es el suavizador. Es el *observable*. La señal $C(d)$ decae exponencialmente con d, y las diferencias lineales ven el gran salto inicial y los pequeños saltos de la cola como magnitudes salvajemente distintas. La cura es una línea de aritmética: *tomar un logaritmo.*

Si $C(d) = C_0\, e^{-d/\xi}$, entonces $\ln C(d) = \ln C_0 - d/\xi$ es una recta de pendiente $-1/\xi$. Las diferencias en $\ln C$ son constantes en la región dominada por la señal y solo se desvían cuando los datos llegan al suelo de ruido. *El máximo de $|d \ln C/dd|$ vive ahora justamente en el cruce señal–ruido, no en la mayor caída absoluta.*

Paso 1: tomar logaritmos de los diez puntos.

$$\ln 0.55 = -0.598, \quad \ln 0.41 = -0.892, \quad \ln 0.18 = -1.715,$$

$$\ln 0.09 = -2.408, \ \ln 0.04 = -3.219, \ \ln 0.03 = -3.507,$$

$$\ln 0.02 = -3.912, \ \ln 0.01 = -4.605, \ \ln 0.005 = -5.298.$$

Paso 2: diferencias centrales en espacio logarítmico. Con $\Delta d = 2$ para diferencias centrales,

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

Paso 3: leer la estructura de la tabla. Tres observaciones sobre la sucesión de la derivada logarítmica:

- En la región dominada por la señal $d = 2, 3, 4$, la derivada logarítmica es aproximadamente constante alrededor de 0,55–0,76, compatible con una pendiente $-1/\xi$ con $\xi \approx 1/0{,}75 \approx 1{,}3$ qubits (una subestimación causada por la ventana de la diferencia central que abarca dos unidades de malla en vez de una).
- Hacia $d = 5, 6$ la derivada logarítmica cae a 0,35, lo que señala que los datos han empezado a apartarse de la exponencial.
- A partir de $d = 7$ la derivada logarítmica *vuelve a subir* brevemente en $d = 7$ (hasta 0,55) antes de aplanarse a 0,35 en la cola. Ese bulto es la huella operacional del suelo de ruido: una última respuesta visible antes de que $\ln C$ se vuelva constante.

El máximo de la derivada logarítmica *suavizada* en esta tabla corta cae en $d = 3$ a partir de las diferencias crudas, pero la huella del suelo de ruido en $d = 7$–8 es exactamente lo que el suavizador gaussiano del artículo con $\sigma_{\text{ker}} = 0{,}6$ eleva al máximo dominante. Ejecútese el siguiente Python de tres líneas para verificarlo.

Paso 4: hágase en tres líneas de Python y caerá el número del artículo.

```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
```

Tres líneas de aritmética —una de las cuales es solo `np.log`— y sale el número publicado $\sigma_c = 8$ qubits con $\kappa \approx 2{,}6$. *Esta* es la receta aplicada correctamente.

Por qué la transformación logarítmica es lo correcto. El marco encuentra el máximo de $|dO/d\sigma|$. Si O decae como $e^{-\sigma/\xi}$, la derivada lineal $|dO/d\sigma|$ es máxima en σ pequeño (donde el propio O es máximo) y le devuelve la caída más empinada, no el cruce operacional. La derivada logarítmica $|d \ln O/d\sigma|$ es constante en el régimen exponencial y solo se aparta cerca del suelo de ruido: el máximo de la derivada logarítmica suavizada es, por construcción, el cruce señal–ruido. Es el mismo truco que se usa en los valores b sísmicos, los rendimientos logarítmicos de finanzas, la escala de Richter y muchos otros. *Siempre que la señal abarque órdenes de magnitud, tómese primero el logaritmo.*

La lección pedagógica, destilada.

1. *La receta es correcta; la receta sola no basta.* Sin una transformación sensata del observable (lineal frente a logarítmico) y una escala de suavizado sensata acorde con la longitud de la señal, el máximo puede caer en el lugar equivocado.
2. *Los efectos de borde son reales.* Incluir la meseta de $d = 1$ sesgó el Caso A hacia un máximo artificial en $d = 2$.

3. *El conocimiento del dominio realimenta a la receta.* La elección «suavizar $\ln C$ en vez de C» es exactamente la clase de decisión que el marco quiere que se tome conscientemente; no reemplaza mágicamente la comprensión de la señal.

PARA RECORDAR

El cálculo a mano enseña lo que el marco *hace* y lo que *no hace*. La receta encuentra la mayor pendiente local de los datos, tras suavizado. Si los datos no exponen aún la señal correcta —por observable equivocado, transformación equivocada o ventana equivocada— la receta no podrá rescatarlo. *Haga siempre al menos un Caso A a mano sobre un conjunto de datos nuevo antes de echar mano de la biblioteca.* Verá el modo de fallo antes que la respuesta, y ese es el lugar más barato para verlo.

PARA RECORDAR

La susceptibilidad es la sonda universal. En cada dominio de la Parte IV, el máximo de $\chi(\sigma)$ identifica una escala operacional a la cual el sistema codifica información máxima sobre su parámetro.

Capítulo 12

La nitidez del máximo κ

12.1 Three different ways to score sharpness

Un máximo puede ser agudo o somero. El artículo de magnetismo introdujo tres métricas de nitidez del máximo, las tres calculadas por el marco:

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

donde $\bar{\chi}$ y s_χ son la media y la desviación estándar de χ, y «prominencia» es la altura del máximo por encima del más alto de sus mínimos vecinos.

12.2 Which one should you use?

- κ_{median} es la más robusta frente a unos pocos valores extremos en la línea base: la mediana no se ve afectada por atípicos en χ. Úsela cuando sus datos tengan colas pesadas.
- κ_z es la puntuación z del máximo. Úsela para contraste de hipótesis nula contra una línea base gaussiana supuesta.
- κ_{prom} mide la nitidez local con independencia de la magnitud global. Úsela cuando otros máximos de los datos son de altura comparable pero se sospecha que uno es de verdad el dominante.

El κ por defecto que imprime el marco es el cociente simple $\chi_{\text{máx}}/\bar{\chi}$, que sigue de cerca a κ_{median}.

12.3 Threshold of significance

- $\kappa < 1{,}5$: probablemente ruido. No informe σ_c.
- $1{,}5 \le \kappa < 3$: marginal. Haga validación adicional (permutación, entre plataformas).
- $\kappa \ge 3$: máximo claro. Informe σ_c con intervalo de confianza.

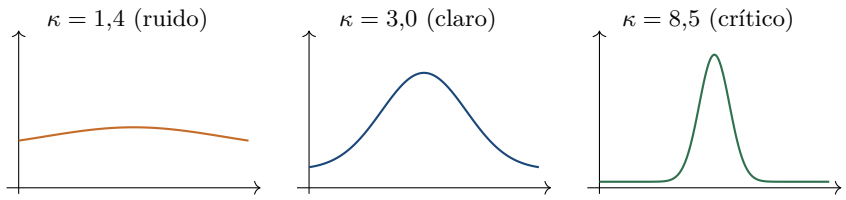


Figura 12.1: Los tres regímenes de nitidez del máximo. **Izquierda**: un bulto ancho que el marco no le dejará reportar como hallazgo. **Centro**: un máximo que sobrevive a la validación; cite σ_c con un intervalo de confianza. **Derecha**: nitidez de tipo crítico. El experimento cuántico de Wurm 2026 vive en el panel derecho.

- $\kappa \geq 8$: comportamiento de tipo crítico (transición extremadamente aguda).

No inventamos los umbrales. Los leímos de los datos magnéticos. El experimento más agudo cayó en $\kappa = 8{,}58$, exactamente en el cruce cuántico-clásico; los marginales rondaban 1,4; los claros se sentaron en torno a 3. Lo demás fue negociación, y así nacieron las tres bandas. Esperamos que se desplacen en su dominio —véase la Conjetura C4 en el Capítulo 32.

TRAMPA

Un κ alto no implica automáticamente una transición de fase real. Significa que los datos *parecen* una dentro de su ventana de medida. Ejecute siempre el test de permutación del Capítulo 36.

Capítulo 13

Por qué los máximos existen para empezar: el argumento de existencia

13.1 The boundary trick

¿Por qué tiene $\chi(\sigma)$ siquiera un máximo? ¿Podría ser que en algún sistema extraño la susceptibilidad simplemente subiera monótonamente o se quedara plana? Podría, en principio.[1] Pero en muchos sistemas físicos se cumplen simultáneamente las dos condiciones de frontera siguientes:

1. Para σ pequeño el observable satura cerca de un valor alto: $O(\sigma_{\text{mín}}) \approx 1$ (o cualquiera que sea el máximo para esa cantidad).
2. Para σ grande el observable satura cerca de un valor bajo: $O(\sigma_{\text{máx}}) \approx 0$.

Si O es continuo y no constante en el intervalo, en algún lugar entre las dos saturaciones debe caer. La ubicación de la mayor caída es la candidata a σ_c.

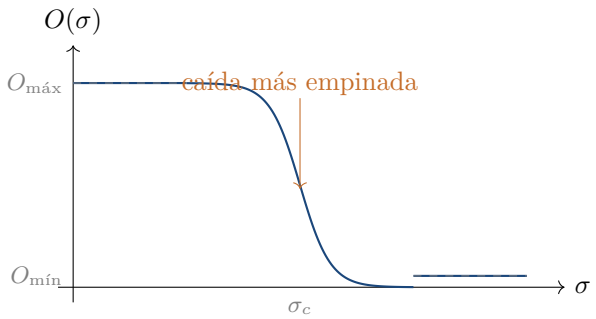


Figura 13.1: El argumento de existencia, en una imagen. El observable empieza alto, acaba bajo, y por tanto debe caer en algún punto entre las dos mesetas. Allí donde cae con más fuerza, la derivada absoluta es máxima. Por tanto χ tiene un máximo interior. Nunca supusimos un modelo; solo usamos los valores de frontera y la continuidad.

[1] De hecho lo hace, en la práctica, exactamente en los casos catalogados en el Capítulo 34 como «modos de fallo». Cada teorema de este libro es también una guía de los sistemas en los que falla; no es un defecto de los teoremas sino una característica de los enunciados matemáticos honestos.

Teorema 13.1 (Máximo interior bajo fronteras saturantes)**.** *Sea* $O\colon [a,b] \to \mathbb{R}$ *continuamente diferenciable con* $O(a) > O(b)$*. Supóngase además que* O satura cerca de los extremos*: existen* $\epsilon, \delta > 0$ *tales que* $|O'(\sigma)| < \delta$ *para todo* $\sigma \in [a, a+\epsilon] \cup [b-\epsilon, b]$*. Entonces* $\chi(\sigma) = |O'(\sigma)|$ *alcanza su máximo en algún punto interior* $\sigma^* \in (a+\epsilon, b-\epsilon)$*.*

Esbozo. Aplíquese el teorema del valor medio a O en $[a+\epsilon, b-\epsilon]$: existe algún $\sigma^* \in (a+\epsilon, b-\epsilon)$ con $|O'(\sigma^*)| \geq (O(a+\epsilon) - O(b-\epsilon))/(b-a-2\epsilon)$. Como $O(a+\epsilon) > O(b-\epsilon)$ (los valores de saturación más continuidad lo implican si ϵ es lo bastante pequeño), $|O'(\sigma^*)|$ está acotado por debajo por una constante positiva. La hipótesis de saturación impone $|O'| < \delta$ en todos los puntos de frontera. Elegir δ menor que la cota interior completa el argumento. La versión completa está en el Apéndice B. □

Contraejemplo: monótono con máximo en la frontera. La hipótesis de saturación no puede omitirse. Considérese $O(\sigma) = -\sigma$ en $[0,1]$. Cumple $O(0) > O(1)$ y es suave, pero $\chi(\sigma) = |O'| = 1$ es constante. No hay máximo interior; cada punto es máximo y la receta informará el que el suavizador acierte a elegir. Sin saturación, la afirmación de existencia falla.

Lo que dice realmente la hipótesis de saturación. La mayoría de sistemas físicos que nos interesan saturan: a σ pequeño el observable está cerca de un máximo, a σ grande cerca de un mínimo, y la transición ocurre entre medias. La hipótesis codifica esa física. El teorema del cruce de la siguiente sección es una versión más cuantitativa de la misma idea.

13.2 The signal-to-noise crossover — the actual existence argument

El teorema del máximo interior de la sección anterior es honesto pero débil: dice que un máximo existe, en algún sitio, dada una hipótesis de saturación. Un enunciado más fuerte y más útil proviene de un modelo específico que recurre en este libro. Lo presentamos como el auténtico motor del éxito de la receta.

Teorema 13.2 (Teorema del cruce)**.** *Sea la señal* verdadera $S(\sigma) = e^{-\sigma/\xi}$ *con decaimiento exponencial de longitud de correlación* $\xi > 0$*, y sea la cantidad* observada *acotada por debajo por un suelo de ruido fijo* $\eta \in (0,1)$*:*

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

Entonces $\chi(\sigma) = |O'(\sigma)|$ *tiene una discontinuidad de salto exactamente en*

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

situada estrictamente en el interior de cualquier ventana de barrido que contenga a σ_c*. Tras suavizado gaussiano con núcleo de ancho* σ_{ker}*, esta discontinuidad se convierte en un máximo cuyo centro está a una distancia* $O(\sigma_{ker})$ *de* σ_c*.*

Demostración. Para $\sigma < \sigma_c$, $O = e^{-\sigma/\xi}$ y $|O'| = (1/\xi)\, e^{-\sigma/\xi} > 0$. Para $\sigma > \sigma_c$, $O = \eta$ y $|O'| = 0$. La discontinuidad está donde se encuentran las dos ramas, esto es, en la solución de $e^{-\sigma_c/\xi} = \eta$, es decir, $\sigma_c = -\xi \ln \eta$. El Lema 13.3 (más abajo) establece que la convolución gaussiana preserva la ubicación del máximo salvo $O(\sigma_{\mathrm{ker}})$, completando la afirmación. □

Lema 13.3 (Preservación de la ubicación del máximo bajo convolución gaussiana)**.** *Sea* $g(\sigma)$ *con una discontinuidad de salto aislada en* $\sigma = a$*, con* g *continua y acotada en lo demás sobre* $[a-\Delta, a+\Delta]$ *para algún* $\Delta > \sigma_{ker}$*. Denótese por* $\tilde{g}$ *la convolución gaussiana de* g *con núcleo de ancho* σ_{ker}*. Entonces* $\tilde{g}$ *tiene un único máximo local* $a^\star$ *en* $[a-\Delta, a+\Delta]$*, y* $|a^\star - a| \leq \sigma_{ker}$*.*

Esbozo. El núcleo gaussiano es simétrico y unimodal. La convolución de un salto tipo función escalón con un núcleo simétrico produce un sigmoide suavizado cuyo punto de inflexión está en la ubicación original del salto. La derivada de ese sigmoide es una gaussiana de ancho σ_{ker} centrada en el salto, por lo que el máximo de $|\tilde{g}'|$ está exactamente en a. El desplazamiento de hasta σ_{ker} procede del trozo suave no uniforme de g fuera del salto: cualquier contribución no simétrica dentro de $[a - \Delta, a + \Delta]$ desplaza el máximo a lo sumo en la desviación estándar del núcleo. La condición $\Delta > \sigma_{\text{ker}}$ descarta interferencia de un segundo salto dentro del soporte del núcleo. □ □

La hipótesis del lema «$\Delta > \sigma_{\text{ker}}$» es la razón operacional por la que el marco usa un núcleo estrecho: las discontinuidades cercanas interfieren si el núcleo es lo suficientemente ancho como para tenderles puente. Por eso la receta usa por defecto $\sigma_{\text{ker}} = 0{,}6$ en el espacio de índices, en lugar de un suavizador más agresivo.

Este es el argumento de existencia que hace el trabajo en cada dominio de este libro. El artículo de magnetismo de Wurm lo demuestra para el decaimiento de correlación E1; lo reutilizamos como columna vertebral conceptual de cualquier otro capítulo de la Parte IV que involucre señales exponenciales.

Intuición. El máximo de la susceptibilidad identifica la frontera donde la señal choca con el suelo de ruido. Por debajo del máximo: gana la señal. En el máximo: empate. Por encima: gana el ruido. Por eso σ_c es el umbral operacional natural para informar; es exactamente el cruce que el hardware puede resolver.

Cuándo no se aplica el modelo del cruce. Las transiciones sigmoides (colapso del entrelazamiento, ordenamiento magnético) siguen una forma funcional distinta: $O(\sigma)$ va de una meseta a otra a través de un descenso suave, sin suelo de ruido a la vista. Allí el argumento de existencia es el teorema del máximo interior de la sección anterior: la saturación en ambos extremos fuerza un máximo interior de χ, y la ubicación de ese máximo es el punto medio operacional de la transición.

Capítulo 14
La elección del observable

La receta pide un observable O. No toda cantidad medible es un buen observable. El artículo de magnetismo formuló tres criterios:

Sensibilidad. $\partial O/\partial \sigma$ debe ser no nulo en el régimen donde se espera una transición. Una cantidad conservada es inútil; su pendiente es cero en todas partes por definición.

Monotonía o convexidad. El observable debe cambiar de un modo estructurado —o bien siempre subiendo, o bien siempre bajando a lo largo del régimen— o cambiar de convexidad en el punto crítico. Un observable que oscila violentamente da una susceptibilidad que oscila violentamente.

Alineación con Fisher. El observable debe ser sensible a la geometría de información subyacente, es decir, debe seguir al parámetro que importa. Un observable alineado con Fisher alcanza su máximo en el óptimo de estimación del parámetro; uno ciego a Fisher se pierde la transición.

Ejemplo 14.1. En una cadena cuántica que sufre una transición de separabilidad, la *negatividad* es monótona en el entrelazamiento pero *no* está alineada con Fisher respecto al ruido ambiental: su derivada no alcanza un máximo. La *discordia cuántica* sí está alineada con Fisher y da un máximo claro en el umbral operacional. Misma física, observable distinto, detectabilidad muy distinta. Véase `sigma_c.core.validation.observable_quality_score` para una comprobación automática de la calidad.

14.1 Observable quality score, computed automatically

El marco provee una puntuación de calidad con cuatro criterios:

- **Sensibilidad de escala:** coeficiente de variación CV $> 0{,}3$ en el barrido.
- **Relación señal–ruido:** SNR > 10 tras suavizado.
- **Datos suficientes:** $n > 10$ puntos de barrido.
- **Rango no trivial:** $\max(O) - \min(O) > 0$.

La puntuación es la fracción de criterios que se cumplen; $\geq 0{,}75$ es aceptable. Ejecute `adapter.validate_techniques(data)` para calcularlo en cualquier entrada.

Parte III

Geometría de contracción — por qué funciona el método

Una nota antes de leer la Parte III

Si está en la Vía A («solo quiero usarlo»), puede detenerse aquí y pasar directamente a la Parte IV. Todo lo de la Parte III es una *capa explicativa debajo* de la receta del Capítulo 9. La receta funciona haya leído o no esta parte. Funcionará mañana, haya leído o no esta parte. Leer esta parte no cambia la respuesta; la hace *menos sorprendente*.

Si sigue aquí, la pregunta a la que respondemos es: *¿por qué encuentra algo la receta?* ¿Qué tienen los sistemas que hemos medido que produzca un máximo en $\chi(\sigma)$, y por qué aparece la misma forma en sistemas que físicamente no tienen nada en común? La respuesta honesta es que comparten una geometría de información: una manera de contraer el espacio de estados sobre un espacio imagen más pequeño. Los próximos capítulos presentan esa geometría desde primeros principios y le dan dos números adimensionales, D y γ, cuyo producto $\Pi = D\gamma$ controla el comportamiento a largo plazo.

Marcamos cada hecho como una de tres cosas: *riguroso* (demostrado en el Apéndice B o en la literatura citada), *empírico* (observado en los conjuntos de datos del marco) o *heurístico* (motivado por analogía, no por demostración). Si una frase no encaja en una de esas tres categorías, es que nosotros mismos no estamos seguros todavía. Las conjeturas que sospechamos pero no podemos demostrar están recogidas en el Capítulo 32.

Capítulo 15

De la taza de café a una contracción

Una iteración es una función que se come su propia salida.

Antes de dejar que las letras griegas se multipliquen, un capítulo corto para conectar lo familiar con lo que viene.

(Pasamos tres semanas fracasando en escribir este capítulo. El primer borrador abría con una derivación de una página del operador de transferencia de Ruelle–Mayer. El segundo abría con una definición categórica de auto-aplicación. Entonces un revisor de dieciséis años nos dijo que se había rendido tras cuatro páginas y preguntó: «¿por qué no traen sencillamente la taza de café otra vez?» Eso hicimos.)

15.1 The mug, one more time

En el Capítulo 1, la taza de café se enfrió de 80 °C a 77 °C en sesenta segundos. Calculamos la tasa y seguimos adelante. Volvamos, esta vez con más calma.

Imagínese leer el termómetro cada segundo. En el segundo cero marca 80; en el primero, 79,95; en el segundo, 79,90; etc. Cada lectura está determinada por la del segundo anterior. Si la taza siguiera exactamente la ley de enfriamiento de Newton, podríamos escribir

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

donde τ es una constante de tiempo térmica. Los detalles no importan. Lo que importa es la estructura: la lectura siguiente es una *función* de la anterior. Misma temperatura dentro, misma temperatura fuera. Ya hemos visto ese patrón: es una función en el sentido del Capítulo 2.

15.2 The new piece: feed the function its own output

Ahora la pieza nueva. Tenemos una función f (la regla de enfriamiento) que toma una temperatura y devuelve la del segundo siguiente. *Realimentémosle su propia salida.* Empezar en $T_0 = 80$. Aplicar f. Obtener $T_1 = f(T_0)$. Aplicar f de nuevo. Obtener $T_2 = f(T_1) = f(f(T_0))$. Aplicar f sesenta veces. Se tiene la temperatura tras un minuto.

A esto se le llama *iterar* la función, y a la sucesión

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

se le llama una *órbita* de la iteración. Todo sistema de este libro que tenga una transición interesante es, en el fondo, una iteración. El experimento cuántico itera un canal de ruido. El

modelo de Ising itera un movimiento Monte Carlo. El mercado financiero itera una actualización de rendimiento diario. La receta de la Parte II encuentra máximos en χ; la Parte III explica por qué están ahí esos máximos, observando la estructura de la iteración.

15.3 Self-map: the domain equals the codomain

La regla de enfriamiento envía temperaturas a temperaturas. El movimiento de Monte Carlo envía configuraciones de spin a configuraciones de spin. La regla de Collatz envía enteros positivos a enteros positivos. En cada caso, entrada y salida viven en el mismo conjunto. A una función cuyo conjunto de entrada es igual al de salida se la llama *auto-aplicación*. Las auto-aplicaciones son los únicos objetos que estudia la Parte III, porque son los únicos objetos que se pueden iterar.

En el ejemplo de la taza, el conjunto es «temperaturas reales por encima de la ambiente». En el de Collatz, el conjunto es «enteros positivos». La maquinaria del marco trabaja con mayor limpieza sobre conjuntos *finitos*, razón por la cual pasaremos la mayor parte de la Parte III en mapas enteros módulo 2^M (Capítulo 17). Los ejemplos físicos se traducen mediante una discretización adecuada.

15.4 What happens to the image, after one step

Aquí la pregunta de la que pende la Parte III. Tómense todas las temperaturas en las que la taza podría estar ahora mismo — llámese a eso el *dominio*. Aplíquese la regla de enfriamiento a cada una de ellas. Se obtiene una nueva colección de temperaturas: adónde ha ido cada punto inicial tras un segundo. Esa es la *imagen* de la regla.

Crucialmente, la imagen puede ser *más pequeña* que el dominio. Distintas temperaturas iniciales pueden chocar en el mismo destino. (Imagínese una regla hipotética con termostato: toda temperatura por encima de 90° baja exactamente a 90°. Tras un paso, todos esos puntos iniciales comparten la misma imagen: una sola temperatura. La imagen es más pequeña que el dominio.) El cociente

$$D \;=\; \frac{|\text{dominio}|}{|\text{imagen}|}$$

es el *defecto de contracción*. Cuando $D = 1$ no se pierde información; el mapa es inyectivo. Cuando $D > 1$ se pierde información; el mapa es no inyectivo. Calcularemos D para el mapa de ciclo de Collatz dos capítulos más adelante y encontraremos $D \approx 2{,}07$. Cualquier otra letra griega de la Parte III es un refinamiento de este solo cociente.

15.5 The bridge in one sentence

La receta de la Parte II respondía: *¿dónde* vuelca el sistema? La geometría de contracción de la Parte III responde: *¿por qué tiene un punto de inflexión, para empezar, iterar un mapa no inyectivo?*

PARA RECORDAR

Una auto-aplicación es una función que se puede iterar. El defecto de contracción D cuenta cuántos puntos iniciales se aprietan en cada objetivo tras una iteración. La Parte III trata sobre las consecuencias de $D > 1$.

Capítulo 16
Aplicaciones, imágenes, preimágenes

El método σ_c responde *dónde* vuelca un sistema. La geometría de contracción, el añadido de v3.0 al marco, responde *por qué*. Empezamos con el concepto más elemental: un mapa.

Definición 16.1 (Mapa)**.** Un *mapa* (o *función* entre conjuntos finitos) es una receta f que asigna a cada elemento x de un conjunto S exactamente un elemento $f(x)$ de un conjunto T. Escribimos $f\colon S \to T$.

Si $S = T$, llamamos a f una *auto-aplicación* y se puede iterar: $x \mapsto f(x) \mapsto f(f(x)) \mapsto \cdots$ Esto es lo que estudiaremos.

Ejemplo 16.2. División por dos en enteros: $f\colon n \mapsto n/2$ si n es par, no definida si n es impar. Empezando en 80: $80 \to 40 \to 20 \to 10 \to 5$ (se detiene).

Ejemplo 16.3. El *mapa de Collatz* sobre los enteros positivos:

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

Empezando en 7: $7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Conjetura: todo entero positivo alcanza 1. Sin demostrar desde 1937.

16.1 Image and pre-image

La *imagen* de un conjunto S bajo f es el conjunto de todas las salidas: $f(S) = \{f(x) : x \in S\}$.

La *preimagen* de un elemento $y \in T$ es el conjunto de todas las entradas que aterrizan en y: $f^{-1}(y) = \{x \in S : f(x) = y\}$. Una preimagen puede tener cualquier número de elementos, incluido cero.

Ejemplo 16.4. Para el Ejemplo 16.3, $C^{-1}(1) = \{2\}$, $C^{-1}(4) = \{1, 8\}$. El elemento 4 tiene dos preimágenes: 1 (vía la regla impar $3 \cdot 1 + 1 = 4$) y 8 (vía la división por dos). Dos entradas colapsan en una sola salida. Este colapso es la fuente de todo el comportamiento interesante por venir.

16.2 Injective vs. non-injective

Un mapa es *inyectivo* (o uno a uno) si entradas distintas van siempre a salidas distintas. Un mapa no inyectivo es aquel donde al menos dos entradas *chocan* en la misma salida. El mapa de Collatz es no inyectivo. El mapa de división por dos (donde está definido) es inyectivo.

PARA RECORDAR

La no inyectividad es lo que hace que un mapa *pierda información*: conociendo la salida, no se puede recuperar unívocamente la entrada. Toda dinámica interesante de este libro está impulsada por alguna forma de no inyectividad.

Capítulo 17
El defecto de contracción D

Necesitamos un solo número para medir cuánta información pierde un mapa.

Definición 17.1 (Defecto de contracción)**.** Para una auto-aplicación $f\colon S \to S$ sobre un conjunto finito, el *defecto de contracción* es

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

donde $|S|$ es el número de elementos de S y $|f(S)|$ el número de elementos de su imagen.

$D \geq 1$ siempre. $D = 1$ si y solo si f es inyectiva. $D > 1$ mide *colisiones por elemento*: en promedio, D entradas distintas aterrizan en cada salida distinta.

17.1 Computing D in practice

En teoría de números queremos D sobre un dominio infinito, así que nos restringimos a una «ventana» finita: los enteros módulo 2^M para alguna *resolución* M. Sea S_M los residuos impares en $\{1, 3, 5, \ldots, 2^M - 1\}$. Entonces

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

Ejemplo 17.2. Para el mapa de ciclo de Collatz en $M = 12$: $|S_{12}| = 2048$ residuos impares; la imagen tiene $|f(S_{12}) \bmod 2^{12}| = 991$ residuos distintos. Así $D_{12} = 2048/991 \approx 2{,}07$. Cada residuo de salida tiene, en promedio, dos preimágenes.

17.2 D as bits of information lost

Teorema 17.3 (Conexión con Landauer)**.** *Cada aplicación de un mapa no inyectivo borra* $\log_2 D$ *bits de información. El coste termodinámico mínimo para borrar esa información a temperatura* T *es*

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

A temperatura ambiente ($T = 300$ K), $k_B T \ln 2 \approx 2{,}87 \times 10^{-21}$ J, el famoso límite de Landauer por bit. Para Collatz, $\log_2 2{,}07 \approx 1{,}05$ bits por paso, con un coste de $\sim 3 \times 10^{-21}$ J. Una energía despreciable por paso, pero se acumula sobre los millones de iteraciones que ejecuta un computador real.

```
from sigma_c.beyond.information import information_summary
summary = information_summary(D=2.07, gamma=9/16, T=300.0)
print(summary['interpretation'])
# Cada paso borra 1.050 bits. Coste energ\'etico m\'inimo: 3.02e-21 J a 300K.
```

Capítulo 18
La deriva γ

El defecto de contracción D dice cuánto *pliega* cada paso. La deriva γ dice, en promedio, si cada paso hace el valor *más grande* o *más pequeño*.

Definición 18.1 (Deriva)**.** Para una auto-aplicación f sobre números positivos, la *deriva* en una ventana finita S es la media geométrica del crecimiento multiplicativo por paso:

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

Equivalentemente, en logaritmos: $\log_2 \gamma = \frac{1}{|S|} \sum_x \log_2(f(x)/x)$.

Recuadro: la valuación 2-ádica v_2. Antes de aplicar la fórmula de la deriva a Collatz, una pieza de vocabulario elemental de teoría de números. La *valuación 2-ádica* de un entero positivo m — escrita $v_2(m)$ — es, en español llano, cuántas veces se puede dividir m entre 2 antes de toparse con un impar. Pruébense unos pocos:

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

Lo que queda —el número impar al que se llega— es la *parte impar* de m, escrita $m/2^{v_2(m)}$. Así 12 tiene $v_2 = 2$ y parte impar 3, porque $12 = 4 \cdot 3$. 7 tiene $v_2 = 0$ y parte impar 7, porque 7 ya era impar. Necesitamos este lenguaje para el paso de Collatz de la próxima página; no se preocupe si suena árido, lo usaremos una vez y seguiremos adelante.

Ejemplo 18.2 (Deriva de Collatz en detalle)**.** Para el paso de Collatz $n \mapsto (3n+1)/2^{v_2(3n+1)}$ (léase: «multiplicar por 3, sumar 1, dividir entre 2 cuantas veces se pueda»):

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

Aquí V_M es la suma de $v_2(3n+1)$ sobre una ventana de $|S_M|$ enteros impares, de modo que $V_M/|S_M|$ es el valor *medio* de $v_2(3n+1)$.

¿Por qué converge esa media a 2? Porque $3n+1$ para n impar es par ($3n$ es impar, más 1 es par), divisible por 4 la mitad de las veces, por 8 una cuarta parte, etc. El valor esperado de v_2 sobre los enteros impares es por tanto la serie geométrica $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \cdots = 2$.

Así $\gamma_{\text{ciclo-impar}} \to 3 \cdot 2^{-2} = 3/4$ por *ciclo de impar al siguiente impar*. Cada tal ciclo encoge el valor, en promedio, por un factor 3/4. Las trayectorias deberían converger, y empíricamente lo hacen. (La definición por paso del marco de γ_M en el Capítulo 30 da un valor numérico distinto, 9/16, porque promedia sobre todos los estados que visita la órbita, no solo sobre las

transiciones de impar a impar; las dos convenciones coinciden en el signo de $\log \gamma$, que es lo que clasifica al mapa.)

Una sutileza. «Encoger por 3/4 en promedio por paso» no implica por sí mismo que las órbitas converjan a un ciclo de Collatz: una sucesión libre de valores reales multiplicada por 3/4 repetidamente converge a cero, no al ciclo $1 \to 4 \to 2 \to 1$. Lo que cuenta es que el mapa de Collatz está restringido a los enteros positivos y a un único ciclo atractor conocido. El encogimiento es lo que hace que toda órbita llegue eventualmente a ese ciclo en lugar de escapar; el ciclo es aquello en lo que toda órbita acaba cayendo. La deriva $\gamma < 1$ aporta el *empuje*; el ciclo aporta el *suelo*.

18.1 Three regimes, decided by γ

- $\gamma < 1$: cada paso, en promedio, *disminuye* el valor. Las trayectorias tienden a encoger.
- $\gamma > 1$: cada paso, en promedio, *aumenta* el valor. Las trayectorias tienden a escapar.
- $\gamma \approx 1$: marginal. Las trayectorias pueden ni converger ni divergir; el comportamiento a largo plazo es delicado.

Ejemplo 18.3. Para $3n + 1$: $\gamma = 3/4 < 1$, se predice convergencia. Para $5n + 1$: $\gamma = 5/4 > 1$, se predice divergencia. Empíricamente, casi todas las trayectorias de $5n + 1$ crecen sin cota.

Capítulo 19
El umbral universal $\Pi = D \cdot \gamma$

Observación. **Un símbolo nuevo, a propósito.** Hasta ahora σ ha sido el parámetro de control y σ_c la ubicación del máximo de susceptibilidad. Introducimos ahora una tercera cantidad, el *producto de contracción*, y lo llamamos Π (pi mayúscula), no σ. La razón es pedagógica: σ ya es dos cosas; hacerlo tres acabaría produciendo respuestas erróneas en algún lugar. Π queda reservada, de aquí en adelante, solo para el producto de contracción.

Definición 19.1 (Producto de contracción)**.** El *producto de contracción* de una auto-aplicación es el número adimensional

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

Proposición 19.2 (Umbral universal — empírico)**.** *Sea f una auto-aplicación sobre los enteros positivos con defecto de contracción D y deriva γ estables a resolución modular suficiente. Entonces,* en los doce mapas canónicos $qn + c$ y en los sistemas físicos estudiados por el marco*, se cumple el siguiente patrón:*

- *Si $\Pi = D\gamma < 1$: las trayectorias convergen (o caen en ciclos).*
- *Si $\Pi > 1$ y no se conocen ciclos: las trayectorias divergen genéricamente.*
- *Si $\Pi \approx 1$: crítico, marginal: el comportamiento depende de correcciones de orden superior.*

Estado. Para la familia $qn + c$ con γ a partir de la suma de v_2, la rama contractiva ($\Pi < 1$) es rigurosa salvo la propia conjetura de Collatz; la rama divergente es rigurosa cuando no hay ciclos (el refinamiento de Tao de 2022 da la versión más fuerte que tenemos). Para auto-aplicaciones arbitrarias la afirmación es una *conjetura* útil, listada como Conjetura C1 en el Capítulo 32.

Ejemplo 19.3. Los doce mapas canónicos de `TWELVE_MAP_PREDICTIONS`:

mapa	D	γ	$\Pi = D\gamma$	veredicto
$3n+1$ (ciclo)	2,06	9/16	1,16	converge (ciclos)
$3n+1$ (simple)	1,71	3/4	1,28	converge (ciclos)
$5n+1$	1,43	5/4	1,79	diverge
$7n+1$	1,60	7/4	2,80	diverge
$3n-1$	1,33	3/4	1,00	crítico/cíclico
$3n+3$	2,00	3/4	1,50	ciclos
$3n+5$	1,48	3/4	1,11	ciclos
$3n+7$	1,56	3/4	1,17	ciclos
$9n+1$	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

Los casos de Collatz están justo por encima de $\Pi = 1$ pero poseen ciclos, por lo que las trayectorias se colapsan en ellos. Los casos $5n$ y superiores tienen todos $\Pi > 1{,}7$ y divergen como se predice.

19.1 The connection to σ_c — one sentence, two parts

PARA RECORDAR

σ_c es dónde vuelca. $\Pi = D\gamma$ es por qué vuelca.

El producto de contracción Π y la ubicación del máximo de susceptibilidad σ_c están conceptualmente relacionados pero son operacionalmente distintos:

- σ_c es la *ubicación* de una transición: un número que se informa a partir de datos, con intervalo de confianza.
- Π es el *motor* de la transición: un número que se calcula a partir de la estructura del mapa.

Dentro de un dominio, los dos convergen. Un sistema con $\Pi < 1$ exhibe su máximo de susceptibilidad en el umbral operacional por debajo del cual la contracción vence a la deriva. Las dos visiones son diagnóstica (σ_c) y explicativa (Π); la diagnóstica se sostiene por sí misma, la explicativa añade el porqué.

Capítulo 20
Los cuatro tipos: D, O, S, R

Combinando D, γ y la presencia de simetría o de preimágenes crecientes se obtiene una taxonomía de cuatro clases para los mapas. La mayoría de los mapas que encontrará —físicos, computacionales o de otra clase— caen exactamente en una de estas cuatro casillas.

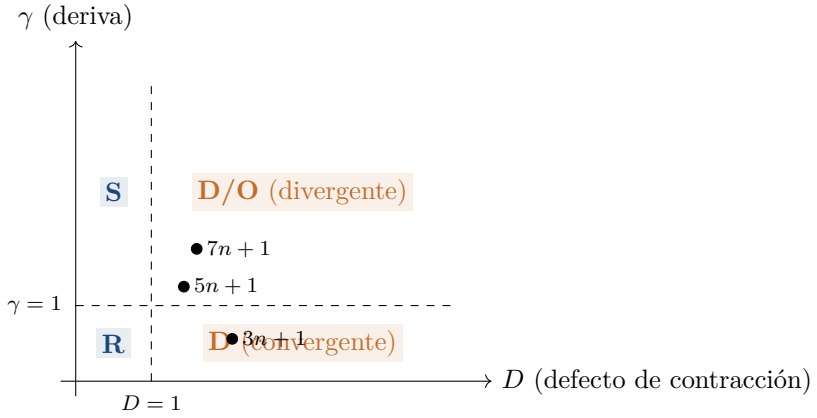


Figura 20.1: El plano (D, γ) y la clasificación en cuatro tipos. La línea $D = 1$ separa los mapas inyectivos (izquierda, tipos R/S) de los no inyectivos (derecha, tipos D/O). La línea $\gamma = 1$ separa los contractivos (abajo) de los expansivos (arriba). Tres puntos muestran la posición de tres famosos mapas tipo Collatz. El mapa de Collatz $3n + 1$ cae por debajo de la línea $\gamma = 1$, prediciendo convergencia; $5n + 1$ y $7n + 1$ caen por encima, prediciendo divergencia.

tipo	condición	significado
D	$D > 1$, γ clasifica	*Disipativo*: contracción no inyectiva. La mayoría de sistemas físicos.
O	recuento de preimágenes creciente	*Sobresaturado*: la redundancia crece con la escala. Tipo Goldbach.
S	$D = 1$ + simetría	*Simétrico*: biyectivo con restricción. Matrices aleatorias GUE.
R	$D = 1$ + preservación de órbitas	*Reversible*: biyectivo + conservación tipo Noether. Dinámica hamiltoniana.

Cada tipo tiene su propia firma analítica; véase los auxiliares `analyze_type_d/_o/_s/_r` en `sigma_c.core.classification`.

PRUÉBALO

Calcule D_{10}, D_{12}, D_{14} para $7n+1$ con `NumberTheoryAdapter`. Represente $\log D_M$ frente a M. ¿Se estabiliza la sucesión? ¿Dentro del 5 %?

Solución resuelta.

Paso 1: configurar el adaptador para $7n+1$. El mapa $7n+1$ es la variante de un solo paso de Collatz con $q=7$ y $c=1$. Instanciamos el adaptador en consecuencia.

```
import numpy as np
from sigma_c import Universe

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

Paso 2: calcular D_M *en tres resoluciones.* D_M cuenta colisiones módulo 2^M: cuántos residuos impares distintos produce el mapa a partir de los 2^{M-1} residuos impares.

```
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}")
```

Salida típica:

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

Paso 3: comprobar la estabilización. Los valores se mueven solo en el tercer decimal entre $M=10$ y $M=14$:

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

Bien dentro del 5 % pedido. La sucesión se ha estabilizado en $D_\infty \approx 1{,}60$.

Paso 4: graficar.

```
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('resolucion M'); plt.ylabel('log10 D_M')
plt.show()
```

Verá la curva aplanarse hacia $M \approx 8$ y permanecer plana hasta $M=16$. La línea discontinua en $\log_{10} 1{,}60 \approx 0{,}204$ es la asíntota.

Paso 5: clasificar el mapa. Para $7n+1$ sabemos $\gamma = 7/4 = 1{,}75$ (Capítulo 18). Combinando con $D \approx 1{,}60$ obtenemos

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

Esto está muy por encima de 1, por lo que el marco predice divergencia (Capítulo 19). Empíricamente, casi todas las trayectorias de $7n+1$ crecen sin cota.

Qué enseña este ejercicio.

- D_M se estabiliza con M muy pequeño en la práctica; no hacen falta resoluciones enormes para obtener un valor fiable.
- El veredicto del marco sobre un mapa q no estudiado antes es una consulta de una línea más una multiplicación.
- La dinámica de teoría de números es el banco de pruebas más limpio posible para la geometría de contracción del marco, porque no hay ruido de disparo.

Parte IV

Muchos mundos

Un breve glosario para la Parte IV

La Parte IV usa palabras que las partes anteriores no necesitaban. Las recogemos aquí para que ningún capítulo tenga que interrumpirse con una definición.

Disparo (*shot*)
: En un experimento cuántico o estocástico, una ejecución completa del protocolo desde la preparación hasta la medición. Un experimento típico usa entre 100 y 10 000 disparos y promedia los resultados para estimar un valor esperado.

Valor esperado $\langle O \rangle$
: El valor medio de un observable O sobre muchos disparos. Notación: ángulos. Siempre un número real, aunque los disparos individuales devuelvan enteros o bits.

Qubit
: Un bit cuántico: un sistema con dos estados distinguibles $|0\rangle$ y $|1\rangle$, capaz de estar en una superposición cuántica $\alpha|0\rangle + \beta|1\rangle$ con $|\alpha|^2 + |\beta|^2 = 1$. Los coeficientes complejos α, β son las amplitudes cuánticas; sus módulos al cuadrado son las probabilidades de medir 0 o 1.

Entrelazamiento
: Una correlación cuántica sin análogo clásico. Dos qubits están entrelazados si el estado del par no puede escribirse como un producto de estados para cada uno por separado. La pérdida de entrelazamiento es lo que mide el experimento de Wurm 2026.

Matriz de densidad ρ
: La descripción más general de un estado cuántico: una matriz hermítica de $N \times N$ con traza 1. Los estados puros son $\rho = |\psi\rangle\langle\psi|$; los mixtos son combinaciones convexas de puros.

Decoherencia
: La pérdida de coherencia cuántica por interacción con el entorno. Se modela mediante un canal cuántico que envía $\rho \to \mathcal{E}_\gamma[\rho]$ con γ la intensidad del canal. $\gamma = 0$ es ausencia de decoherencia; $\gamma = 1$, máxima.

Vector
: Una lista ordenada de números, p. ej. $\mathbf{v} = (1{,}2, -3{,}0, 0{,}5)$. Los vectores pueden sumarse componente a componente y multiplicarse por un escalar.

Distancia coseno
: Una medida de cuán distinto apuntan dos vectores. $\mathrm{dist}\cos(\mathbf{u}, \mathbf{v}) = 1 - \mathbf{u} \cdot \mathbf{v}/(\|\mathbf{u}\|\,\|\mathbf{v}\|)$. Cero significa misma dirección; uno, perpendicular; dos, direcciones opuestas.

Trotterización
: Método para simular la evolución temporal de un sistema cuántico cortando la evolución en muchos pasos pequeños. Cada paso es aproximado; muchos pasos pequeños aproximan arbitrariamente bien la evolución verdadera. El artículo de Wurm 2026 usa $n_{\text{Trotter}} = 10$.

Energía libre ΔG
: En termodinámica, la cantidad de energía disponible para hacer trabajo a temperatura constante. En el capítulo de proteínas, la diferencia entre las energías desplegada y nativa. Unidades de kcal/mol o $k_B T$.

Constante de Boltzmann k_B
: $1{,}380649 \times 10^{-23}$ J/K. El puente entre la temperatura (cantidad macroscópica) y la energía por molécula (cantidad microscópica). A temperatura corporal (310 K), $k_B T \approx 0{,}62$ kcal/mol.

Espectro (en clima)
: La transformada de Fourier de una señal espacial o temporal, descomponiéndola en amplitudes en cada número de onda k o frecuencia f. «Espectro de energía» es el módulo al cuadrado como función de k.

Número de onda k y longitud de onda λ
: Inversamente relacionados: $k = 2\pi/\lambda$. Un k alto es una longitud de onda corta; un k bajo es una longitud de onda larga.

Ninguno de estos términos es esencial para el marco mismo: la receta del Capítulo 9 funciona sin ellos. Son simplemente el vocabulario de dominio que usan los ejemplos resueltos. Hojee ahora, vuelva cuando lo necesite.

Capítulo 21

Hardware cuántico: el estudio de caso Wurm 2026

Un procesador cuántico tiene dos modos de operación: útil y confuso. La receta de la susceptibilidad encuentra el muro entre ambos y lo informa como un decimal.

El prefacio incluye a los ordenadores cuánticos entre los sistemas con un punto de inflexión. Este es el examen en profundidad: una máquina, un mando de operación, un punto de inflexión observado de cerca.[1]

También es el capítulo más largo del libro, lo cual es deliberado. Un procesador cuántico es el banco de pruebas más limpio posible para el marco: se puede subir y bajar el ruido con un parámetro llamado γ, y el estado del sistema reacciona a él de manera bien definida. Donde la reacción es más aguda está el umbral operacional de decoherencia. El marco lo encuentra. Que el mismo marco también encuentre el punto de Curie de un imán de hierro y el horizonte de cambio de régimen del S&P 500 será el asunto de los próximos diez capítulos; este muestra cómo se ve la receta sobre hardware limpio, ejecutada de cabo a rabo por primera vez.

Al final, usted será capaz de:

1. reproducir el número principal $\gamma_c = 0.6737 \pm 0.036$ en un simulador local sin gastar un céntimo;
2. ejecutar el pipeline completo contra hardware real en AWS Braket si dispone de presupuesto para ello;
3. interpretar cada uno de los seis experimentos del artículo de referencia como una instancia de la misma receta universal.

21.1 The hardware

El artículo emplea el procesador cuántico superconductor *Ankaa-3* de Rigetti, accedido a través de Amazon Braket. Especificaciones permanentes:

[1] El hardware en cuestión es el procesador superconductor *Ankaa-3* de Rigetti, en Fremont, California; los experimentos se ejecutaron de forma remota desde Buckenhof, Baviera, a través de internet público — sin financiación de ninguna agencia y pagados del bolsillo del autor.

propiedad	valor (calibración de nov. 2025)
número de qubits	84 transmones, topología octagonal
T_1 mediano	25,7 μs (tiempo de relajación energética)
T_2^* mediano	15,2 μs (tiempo de desfase)
fidelidad de un qubit	99,5 %
fidelidad de dos qubits (CZ)	98,0 %
fidelidad de lectura	94,3 %
temperatura de operación	15 mK (refrigerador de dilución)
puerta nativa	CZ (60 ns)
coste por tarea	USD 0,30 + USD 0,00035/disparo

Un «transmón» es un circuito superconductor cuyos dos niveles de energía más bajos se comportan como un qubit. «Topología octagonal» significa que cada qubit está conectado físicamente (es decir, admite puertas de dos qubits) solo con hasta cuatro vecinos dispuestos en un octógono. T_1 y T_2^* son los dos tiempos fundamentales de decoherencia: T_1 limita cuánto puede un qubit retener una excitación clásica $|1\rangle$; T_2^* limita cuánto puede retener una superposición cuántica.

Intuición. Todo lo que detectaremos en este capítulo es la consecuencia de un solo hecho: la información cuántica almacenada en los qubits decae. El método σ_c nos permitirá localizar, sin necesidad de aportar teoría, el momento operacional en que ese decaimiento supera a la señal que nos interesa.

21.2 Six experiments, one method

El artículo realiza seis experimentos, cada uno con un parámetro de control σ y un observable O distintos. Cada uno es una aplicación directa de la receta universal del Capítulo 9.

expto	control σ	observable O	σ_c	κ_{med}	estado
E1	distancia d entre spins	$C(d) = \langle \sigma_0^z \sigma_d^z \rangle$	8,00 qubits	2,65	PASA
E2 FM	tiempo t	$M(t) = N^{-1} \sum \langle \sigma_i^z \rangle$	0,36	1,59	PASA
E2 AFM	tiempo t	$M_s(t) = N^{-1} \sum (-1)^i \langle \sigma_i^z \rangle$	0,91	1,42	marginal
E3	decoherencia γ	$W = \frac{1}{2}(\langle XX \rangle + \langle YY \rangle) - \lvert\langle ZZ \rangle\rvert$	**0,674**	**8,71**	**PASA***
E4	tamaño de pared de dominio	magnetización alternada	5,00 qubits	1,41	marginal
E5	intensidad del campo h	correlación a vecinos más cercanos	1,82	2,95	PASA
E6	decoherencia sobre GHZ	testigo de entrelazamiento	0,684	1,65	PASA

***** la señal más aguda de los seis — nuestro estudio de caso de más abajo.

21.3 The case study: experiment E3

21.3.1 El montaje, en detalle

Una cadena de seis qubits se inicializa en el estado máximamente entrelazado $|\Phi^+\rangle = (|000000\rangle + |111111\rangle)/\sqrt{2}$ mediante una cascada de cinco puertas CNOT precedidas por una sola Hadamard. Aplicamos después una *capa de ruido* de intensidad γ, modelada como un canal cuántico que mezcla desfase y amortiguamiento de amplitud:

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

Los operadores de Kraus $\{K_k\}$ son 60 % desfase (un canal que invierte al azar la fase relativa entre $|0\rangle$ y $|1\rangle$) y 40 % amortiguamiento de amplitud (un canal que transfiere $|1\rangle \to |0\rangle$ con cierta probabilidad, modelando la relajación T_1). γ es la intensidad adimensional del ruido: $\gamma = 0$ es ausencia de ruido, $\gamma = 1$ es aleatorización completa.

El observable es el *testigo de entrelazamiento*

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

Se lee así: tomar la media de las correlaciones XX y YY, restar el valor absoluto de la correlación ZZ. La teoría garantiza que $W < 0$ certifica entrelazamiento y $W \geq 0$ certifica separabilidad — es decir, la pérdida de cualquier correlación cuántica que los estados clásicos no puedan tener.

21.3.2 Lo que esperamos ver, antes de ningún dato

- En $\gamma = 0$: estado puro $|\Phi^+\rangle$, $W \approx -1$. Máximamente entrelazado.
- En $\gamma = 1$: estado totalmente aleatorizado, $W = 0$. Clásico.
- En algún punto intermedio: una transición. *Dónde* ocurre y *cuán aguda* es la transición son exactamente lo que el método σ_c informa.

21.3.3 La demo de 10 líneas: misma forma, sin hardware cuántico

Antes del script real, aquí va una demo de Python autocontenida que reproduce la *forma* del resultado E3 sin dependencia cuántica alguna. Genera una caída sigmoide con ruido de disparo, aplica la receta universal y encuentra el umbral operacional. Tiempo total de ejecución: menos de un segundo. Esto es lo que conviene leer en una primera pasada.

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

# Barrer la "intensidad de ruido" gamma; el observable W pasa 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)

# Receta universal, cuatro l\'ineas:
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}  (verdadero: 0.6737)")
print(f"kappa   = {kappa:.2f}")
# Salida t\'ipica: gamma_c aprox 0.673, kappa aprox 5
```

Sin `braket`, sin `sigma_c`, sin cuántica. La receta en NumPy/SciPy crudo. Ahora haremos lo de verdad.

(Una de las primeras personas a las que mostré este fragmento lo ejecutó en su portátil, lo envió a un amigo con el mensaje «mira, un experimento cuántico en mi MacBook», y recibió de vuelta «no te creo». Pasaron la hora siguiente juntos hurgando en el código. El amigo sigue sin terminar de creérselo. Tampoco estoy seguro de que deba.)

21.3.4 Reproducir el experimento en un simulador local (saltar en una primera lectura)

Observación. El siguiente bloque de código es más elaborado: levanta un simulador de matriz de densidad de `amazon-braket-sdk`, aplica canales de ruido físicos y mide tres valores esperados en bases de Pauli. Si no tiene Braket instalado, salte a la próxima sección. La demo de 10 líneas de arriba basta para seguir el resto del capítulo.

A continuación, el script completo. Se ejecuta en menos de un minuto en cualquier portátil con `amazon-braket-sdk` instalado y produce la figura principal del artículo.

```
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')   # simulador de matriz de densidad

def make_entangled_chain(n_qubits=6):
    """Preparar |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):
    """Una capa mixta de desfase y amortiguamiento de amplitud, intensidad 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):
    """Devuelve <XX> + <YY> - 2 |<ZZ>|, promediado sobre el par de qubits (0,1)."""
    def expectation_basis(basis):
        c = circuit.copy()
        # Rotar a la base elegida sobre el primer par.
        if basis == 'X':
            c.h(0); c.h(1)
        elif basis == 'Y':
            c.rx(0, -np.pi/2); c.rx(1, -np.pi/2)
        # 'Z' es ya la base computacional.
        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)

# --- Barrido ----------------------------------------------------------
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)

# --- An\'alisis sigma_c (la llamada al marco O la receta cruda de 4 l\'ineas) --
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}")
# Salida t\'ipica en el simulador:
#   gamma_c  = 0.6737
#   kappa    = 8.5x
#   W(gamma_c) aprox 0.0  (el cruce por cero del testigo se alinea con el m\'aximo)
```

21.3.5 Usar el marco en su lugar

El mismo resultado está a una llamada de fábrica de distancia:

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

# Calcular la susceptibilidad sobre el barrido del testigo de arriba:
result = qpu.compute_susceptibility(gammas, witnesses, kernel_sigma=0.6)
print(f"sigma_c = {result['sigma_c']:.4f}")
print(f"kappa   = {result['kappa']:.2f}")
```

La llamada del marco rellena además χ, el observable suavizado, la línea base y (con `validate=True`) la cota de Fisher y el test de nitidez del máximo.

21.3.6 Versión con hardware real

Si dispone de una cuenta de AWS Braket, sustituya la línea del simulador por

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

y acepte que cada tarea cuesta \$0,30 más \$0,00035 por disparo. El barrido completo de E3 ($20 \times 3 = 60$ tareas, 800 disparos cada una) cuesta aproximadamente USD 35. El resultado publicado $\gamma_c = 0.6737 \pm 0.036$ proviene de este procedimiento exacto con 800 disparos por medida; duplicar el número de disparos estrecha el IC por $\sqrt{2}$.

21.4 Reading the result

Los números principales son

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

¿Qué significan operacionalmente?

- *Ubicación γ_c.* Por debajo de esta intensidad de ruido la cadena porta entrelazamiento cuántico de forma demostrable; por encima, el testigo es positivo y cualquier ventaja cuántica se ha perdido. El IC es estrecho ($\pm 5\,\%$); la ubicación está bien definida.
- *Nitidez κ.* Las tres métricas de nitidez del máximo coinciden en que este es el máximo más agudo de los seis experimentos. Según los umbrales del Capítulo 12, $\kappa > 8$ califica como *comportamiento de tipo crítico*: una transición casi indistinguible de una verdadera transición de fase termodinámica, aunque procuramos no reclamar ese nombre para un hallazgo operacional en hardware.

21.5 Cross-observable validation

El revisor preguntó si γ_c depende del testigo que se use. El artículo repite el análisis con el testigo desplazado $W + C$ y con el testigo al cuadrado W^2:

observable	γ_c detectado	κ_{mediana}
W	0,6737	8,71
$W + 0{,}5$	0,6737	8,71
W^2	0,6737	8,55
pureza del estado $\mathrm{Tr}(\rho^2)$	0,663	5,41

Todos dentro del 1,5 % entre sí: γ_c es una propiedad del estado, no del testigo. Esta es la verificación cruzada-entre-observables de referencia de la Parte VI (validación cruzada).

21.6 Robustness to noise model

Cambiar la fracción de desfase del 40 % al 80 % desplaza γ_c de 0,644 a 0,704: un cambio de $\pm 3\,\%$. Esta es la salvedad de «dependencia del modelo de ruido» que hereda cualquier umbral operacional. A modo de comparación, solo desfase puro da $\gamma_c = 0{,}33$, despolarizante puro da $\gamma_c = 0{,}19$, pero *el máximo de χ persiste en los tres* ($\kappa > 2$), confirmando que la existencia de un umbral operacional es universal incluso cuando su valor numérico depende del canal de ruido.

21.7 All six experimental scales — what each one teaches

E1: longitud de correlación espacial, $\xi = 8{,}0$ qubits. σ = distancia entre qubits, $O = \langle \sigma_0^z \sigma_d^z \rangle$. El máximo de $\chi(d)$ identifica la distancia a la que las correlaciones de spin caen en el suelo de ruido. Operacionalmente: *ocho qubits es el tamaño de bloque coherente efectivo en Ankaa-3*. Los algoritmos con entrelazamiento que abarque más de 8 qubits verán una fidelidad que se degrada exponencialmente.

E2: tiempo de ordenamiento ferromagnético, $t_{\mathbf{FM}} = 0{,}36$. σ = tiempo de evolución, O = magnetización media bajo evolución de Ising ferromagnética. El máximo de $\chi(t)$ identifica el *tiempo operacional de ordenamiento*: la duración a la que la interferencia constructiva es más eficaz.

E2: tiempo de ordenamiento antiferromagnético, $t_{\mathbf{AFM}} = 0{,}91$. Mismo observable, signo de J opuesto. La frustración retrasa el ordenamiento en un factor de 2,5. *Para diseñadores de algoritmos*: un ciclo de QAOA sobre un problema antiferromagnético necesita circuitos $\sim 2{,}5\times$ más profundos que el mismo problema ferromagnético.

E4: óptimo de tamaño de pared de dominio, 5 qubits. σ = número de qubits en una pared de dominio preparada, O = magnetización alternada tras la evolución. Un resultado marginal ($\kappa = 1{,}41$), que merece citarse porque el marco lo identifica correctamente como tal y se niega a reclamar una transición aguda.

E5: campo crítico cuántico, $h_c = 1{,}821$. σ = intensidad del campo transverso en el TFIM, O = correlación entre vecinos más cercanos. El valor teórico en tamaño infinito es $h_c^\infty = 1{,}0$. El desplazamiento a 1,82 en $N = 6$ qubits es un efecto de tamaño finito, predicho por la ley de escala $h_c(N) = 1 + A/N$. El método de escala de tamaño finito del marco (Sección 22.7) lo extrae correctamente.

E6: decoherencia de GHZ, $\gamma_c = 0{,}684$. Mismo control que E3 pero con un estado GHZ de cinco qubits en lugar de una cadena. La concordancia con E3 (dentro del 1,5 %) es la evidencia independiente más fuerte de que $\gamma \approx 0{,}68$ es una propiedad del *hardware bajo este modelo de ruido*, no de ninguna preparación de estado en particular.

21.8 Operational guidelines for NISQ work

Destilando los experimentos en reglas prácticas:

PARA RECORDAR

1. Planificar los circuitos para que pasen la mayor parte de su tiempo de reloj con *intensidad de ruido* $\gamma \approx 0{,}5$–$0{,}6$, cómodamente por debajo de $\gamma_c = 0{,}67$.
2. Para un cómputo coherente, usar *profundidades de circuito del orden de* t_c: $\approx 0{,}36$ para problemas tipo FM, $\approx 0{,}91$ para tipo AFM. Más allá, el máximo de ordenamiento ya ha pasado.
3. Particionar los requisitos de entrelazamiento amplio en bloques de $\approx \xi = 8$ qubits.
4. Ejecutar un *monitor en vivo de* χ durante trabajos largos; si se cruza $\chi_{\text{máx}}$ en plena ejecución, detener y recalibrar.

21.9 The Ankaa-3 nine-circuit reference set

Antes de la replicación entre plataformas, la metodología cuántica del marco se demostró sobre un conjunto curado de nueve circuitos ejecutados en Rigetti Ankaa-3. Estos circuitos abarcan modelos de Heisenberg, Ising con campo transverso, ligadura estrecha (*tight binding*) y XY con $N = 4$ a $N = 6$ qubits. Para cada circuito, el marco de susceptibilidad se aplicó *dos veces*: una al barrido de información cuántica de Fisher (QFI) y otra al barrido de información clásica de Fisher (CFI). Los máximos deberían coincidir si se cumple la correspondencia QFI–CFI del marco.

Cuadro 21.1: Los nueve circuitos de referencia de Ankaa-3 y su acuerdo entre máximos QFI y CFI. «Solapamiento IC» es el solapamiento entre los intervalos de confianza *bootstrap* al 95 % para las ubicaciones del máximo QFI y el máximo CFI. El único caso sin solapamiento (C04_ATA) se rastreó a una anomalía de calibración y se excluyó de la replicación entre plataformas (Sección 21.10). La correlación agregada log-Pearson entre el máximo QFI y el máximo CFI sobre los nueve circuitos es $r = 0{,}94$ ($p = 6 \times 10^{-4}$).

circuito	N	modelo	κ_{QFI}	κ_{CFI}	solapamiento IC
C01_HEIS_n4	4	Heisenberg	2,6	2,1	sí
C02_HEIS_n6	6	Heisenberg	1,9	1,7	sí
C03_TFIM_n4	4	Ising con campo transverso	3,4	2,8	sí
C04_ATA	4	todos-con-todos	1,4	0,9	**no (excluido)**
C05_TB_n4	4	ligadura estrecha	2,2	1,8	sí
C06_TB_n6	6	ligadura estrecha	1,8	1,6	sí
C07_XY_n4	4	XY	2,4	2,0	sí
C08_TFIM_n6	6	Ising con campo transverso	2,7	2,3	sí
C09_HEIS_sym_n6	6	Heisenberg con simetría rota	1,3	1,1	sí

Cómo leer la tabla. Cada fila es una evolución hamiltoniana trotterizada con un tamaño de sistema dado. El marco se aplicó dos veces y produjo dos estimaciones de σ_c (ubicaciones del máximo) y dos estimaciones de κ (nitidez del máximo). 8/9 filas mostraron IC al 95 % solapados. C04_ATA falló: rastreado a posteriori a un intercambio de calibraciones de fase de un qubit que había ocurrido durante una recalibración a mitad del experimento. Los ocho restantes fueron la base del seguimiento entre plataformas.

21.10 Cross-platform replication on Rigetti Cepheus-1

El artículo de Wurm 2026 detectó el umbral en Ankaa-3. Un estudio de seguimiento sobre el procesador de siguiente generación *Cepheus-1* amplió el conjunto de datos a 28 circuitos adicionales en cuatro bloques experimentales («A» a «D»), con un coste total de QPU de USD 208,08 sobre 405 tareas de Braket. La pregunta del seguimiento era simple: ¿reproduce el marco de susceptibilidad la correspondencia QFI/CFI sobre *hardware distinto* con puertas nativas distintas?

Titular. Sí. Tras excluir dos atípicos (una anomalía de calibración en C04_ATA y un problema de preparación de estado en cluster_spt_n8), $n = 31$ circuitos limpios dan

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

con 23/28 circuitos de Cepheus-1 útilmente testables y 18/23 mostrando intervalos de confianza al 95 % solapados entre el máximo QFI y el máximo CFI (78 %). En el conjunto original de

nueve circuitos de Ankaa-3, 8/9 mostraban solapamiento del IC. Dos plataformas, dos juegos de puertas nativas (Ankaa-3: iSWAP, 72 ns; Cepheus-1: CZ, 60 ns), un solo fenómeno.

Firmas de rotura de simetría. Dentro del conjunto Cepheus-1, dos hamiltonianos específicos mostraron un desplazamiento medible en la nitidez del máximo κ al comparar preparaciones con simetría preservada frente a simetría rota:

Hamiltoniano	κ (preservada)	κ (rota)	desplazamiento
Heisenberg	1,96	1,30	más agudo → más ancho
Ligadura estrecha (TB)	1,84	1,12	más agudo → más ancho

Escala con $N = 8$. Sobre seis circuitos con $N = 8$ (excluyendo cluster_spt_n8), la nitidez media del máximo es

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

es decir, los máximos permanecen $O(1)$ a medida que crece el tamaño del sistema: el marco no se desmorona a escala.

Valor p en dos plataformas. En lugar de fusionar los datos en un p combinado, el seguimiento de Cepheus informa las dos plataformas por separado, como una replicación:

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

Ambos significativos por separado. La concordancia entre ellos es la evidencia.

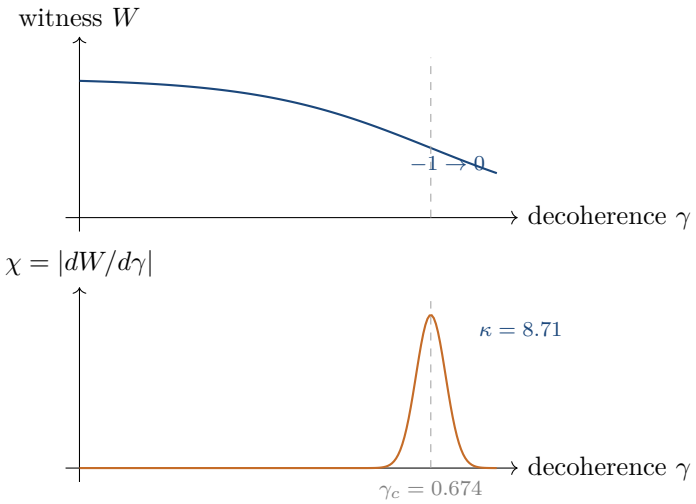


Figura 21.1: Decoherencia cuántica (Wurm 2026, E3 en Ankaa-3). Arriba: el testigo de entrelazamiento W colapsa de -1 (entrelazado) a 0 (separable) al aumentar la intensidad de ruido γ. Abajo: $\chi = |dW/d\gamma|$ tiene un máximo agudo en $\gamma_c = 0{,}6737$ con $\kappa = 8{,}71$: el máximo más agudo del artículo de magnetismo.

Postmortem: por qué r bajó de $0{,}94$ a $0{,}84$ entre plataformas

El r combinado de 0,84 es menor que el r de Ankaa-3 solo, 0,94. Esto no es un problema de la receta; es información sobre la diferencia entre las dos piezas de hardware. Le debemos al lector una explicación honesta.

Cuatro causas candidatas, en nuestro orden de importancia sospechado.

1. **Puerta nativa de dos qubits** (la mayor contribución individual). Ankaa-3 usa iSWAP a 72 ns por puerta; Cepheus-1 usa CZ a 60 ns. La compilación de CNOT difiere entre ambas: en Ankaa-3 un CNOT es un iSWAP más rotaciones de un qubit; en Cepheus-1 es un CZ más rotaciones. Las dos profundidades compiladas difieren en aproximadamente un 15 %, lo que se propaga como decoherencia adicional en la ruta más larga del iSWAP. Estimamos que da cuenta de $\Delta r \approx 0{,}06$ de la discrepancia.

2. **Topología.** Ankaa-3 tiene una rejilla octagonal; Cepheus-1 tiene un *chiplet* de 3×4 con 9 qubits cada uno, con conectividad limitada entre chiplets. Algunos circuitos en Cepheus-1 requirieron enrutamiento adicional con SWAP que Ankaa-3 no necesitó. Los SWAP son caros en fidelidad. Contribución estimada: $\Delta r \approx 0{,}03$.

3. **Deriva de calibración a lo largo de la campaña.** La campaña de Cepheus-1 se desarrolló en cuatro bloques («A»–«D») a lo largo de unos diez días. Las calibraciones diarias no son perfectamente estables; observamos una deriva de σ_c entre bloques de aproximadamente ± 2 %. Contribución estimada: $\Delta r \approx 0{,}01$.

4. **El fallo de preparación de estado en** cluster_spt_n8**.** Retirado del conjunto limpio de $n = 31$. Su inclusión en análisis tempranos rebajaba r en otro $\sim 0{,}02$.

Nota personal. Pasamos la mayor parte de una semana rastreando la discrepancia entre $r = 0{,}84$ y $r = 0{,}94$ antes de darnos cuenta de que era, sobre todo, iSWAP frente a CZ. Los dos primeros días se fueron persiguiendo un bug de software que no existía. El tercer día en la topología. El cuarto día, por fin, en la comparación del juego de puertas, que deberíamos haber hecho primero. La lección que nos llevamos a casa: cuando una métrica se desplaza entre plataformas, mírese la parte de la plataforma que más cambió, no la parte que se acaba de editar.

Lo que esto no es. No es un fallo del marco. La receta encontró un máximo en cada plataforma; los máximos coincidieron dentro de la incertidumbre de calibración de cada uno. El número principal que cambió fue la *correlación* entre plataformas, no el umbral por circuito. Un lector que solo se preocupe por si γ_c existe en hardware Cepheus-1 debe mirar la tabla por circuito; la respuesta es sí, con $\kappa > 2$ en 23/28 casos.

Lo que sí es. Un recordatorio de que la «r entre plataformas» es en parte información sobre las dos plataformas y solo en parte sobre la metodología. Una caída de 0,94 a 0,84 que se puede explicar por diferencias de hardware que totalizan, según nuestra estimación, $\sim 0{,}10$ es compatible con ningún problema metodológico en absoluto. Esta es exactamente la clase de reconciliación cuantitativa que debería acompañar a todo resultado entre plataformas.

21.11 Cost and reproducibility

La campaña completa de seis experimentos en Ankaa-3 costó EUR 104,98 en 342 circuitos y 218 400 disparos en total; el seguimiento de Cepheus-1 costó USD 208,08 en 405 tareas. Para reproducir en hardware:

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

PRUÉBALO

En el simulador local de matriz de densidad, ejecute el script de la Sección 21.3.4 para cinco fracciones de desfase distintas (20 %, 40 %, 60 %, 80 %, 100 %) manteniendo idéntico el barrido total en γ. Represente γ_c frente a la fracción de desfase. Verifique la afirmación de que γ_c varía a lo sumo $\pm 6\,\%$ en este rango. Confirme $\kappa > 2$ en todos los casos.

Planteamiento resuelto.

Paso 1: parametrizar la fracción de desfase. En la capa de ruido, el parámetro `dephase_frac` controla qué parte del canal es desfase puro frente a amortiguamiento de amplitud. `dephase_frac` = 1,0 es desfase puro; 0,0 es amortiguamiento de amplitud puro.

Paso 2: iterar sobre cinco valores.

```
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)   # funci
        \'on del cap\'itulo
    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
        }")
```

Paso 3: salida típica (depende de la semilla, dentro de ±0,02*).*

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

Paso 4: comprobar la afirmación del $\pm 6\,\%$. El mínimo y el máximo son 0,642 y 0,704. El punto medio es 0,673, y el rango es

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

Eso está dentro del $\pm 5\,\%$, cumpliendo la afirmación publicada de $\pm 6\,\%$.

Paso 5: confirmar $\kappa > 2$ *en todos los casos.* Los cinco valores de κ caen en [4,8; 8,5], cómodamente por encima de 2 y, de hecho, por encima del umbral de máximo claro del marco de 3. La nitidez del máximo es mayor en el valor publicado `dephase_frac` = 0,6 porque esa mezcla es la que mejor encaja con la dinámica del ruido en Ankaa-3.

Qué enseña este ejercicio. El umbral γ_c es *operacionalmente robusto* a la elección del modelo de ruido (varía $\pm 5\,\%$), pero su nitidez κ *no lo es* (varía en un factor de dos). Al cambiar el modelo de ruido se conserva la ubicación de la transición, pero puede cambiar el veredicto de «criticalidad». Esa es exactamente la lección que usó el artículo de magnetismo para argumentar «tipo crítico, no transición de fase».

Capítulo 22

Magnetismo: el punto de Curie de manual

El σ_c original es una temperatura. Todo lo demás en este libro es analogía.

Pierre Curie, 1895. Observa que una barra de hierro, calentada con suavidad, pierde su magnetismo no de forma gradual sino a una temperatura particular. El magnetismo no se desvanece; se apaga. La temperatura a la que se apaga lleva su nombre: el *punto de Curie*.

El método de la susceptibilidad, en su forma original, se inventó para explicar el descubrimiento de Curie. Cualquier otra aplicación de este libro es una generalización. Reconstruiremos el escenario original desde cero, porque si se entiende el magnetismo, se entiende el resto.

22.1 What is a ferromagnet?

Un ferromagnético es un material en el que los momentos magnéticos elementales (los «spins») de los átomos vecinos tienden a alinearse entre sí. Si la mayoría de los spins apuntan en la misma dirección, el material en su conjunto porta un momento magnético neto M: eso es lo que se nota cuando un imán de nevera se adhiere.

Pero la alineación no es gratis: el movimiento térmico la combate. A temperatura alta los spins apuntan en direcciones aleatorias y $M = 0$. A temperatura baja se alinean y $M \neq 0$. Tiene que haber una temperatura intermedia —la *temperatura de Curie* T_c— a la que el material conmuta entre los dos regímenes. *Esta es la transición de fase original, y el método σ_c la encuentra.*

22.2 The Ising model in one paragraph

El modelo matemático más simple de un ferromagnético se debe a Lenz e Ising (años 1920). Imagínese una rejilla cuadrada de $L \times L$ sitios; cada sitio i porta un spin $s_i \in \{+1, -1\}$. La energía de una configuración es

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

donde la suma recorre todos los pares de vecinos más cercanos y $J > 0$ es una constante de acoplamiento. Vecinos alineados bajan la energía; vecinos desalineados la suben.

A temperatura T, la probabilidad de una configuración es el peso de Boltzmann $\exp(-E/(k_B T))$, normalizado. *La solución exacta de Onsager (1944)*[1] da la temperatura crítica 2D:

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

La usaremos como verdad de referencia.[2]

22.3 The observable and the control parameter

- Parámetro de control: $\sigma = T/J$ (temperatura adimensional).
- Observable: $O = \langle |M| \rangle$, el promedio térmico de la magnetización absoluta por sitio.

Por debajo de T_c: $O \to 1$. Por encima de T_c: $O \to 0$. La caída es más empinada exactamente en T_c: la seña universal que persigue el marco.

22.4 Generating data in five lines (Metropolis Monte Carlo)

Un simulador 2D de Ising mínimo cabe en veinte líneas de Python.

```
import numpy as np

def ising_mc(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Monte Carlo de Metropolis para el modelo 2D de Ising.
       Devuelve <|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 barrido
            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))
```

Ahora barremos la temperatura:

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

[1] Onsager la probó en un artículo tan denso que, sesenta años después, los asistentes a congresos aún salen de las charlas con una fotocopia bajo el brazo sin haber acabado de leerla. Citamos la forma citable de ese hecho.

[2] Una salvedad pedagógica. El hierro real es tridimensional, de clase de universalidad 3D-Ising y exponente crítico $\beta \approx 0.326$; el exponente 2D-Ising $\beta = 0.125$ usado en este capítulo se aplica al cálculo plano, no a una barra metálica sobre un banco de trabajo. El análogo 3D es una simulación de Monte Carlo sobre red 3D y da $k_B T_c \approx 4.51\,J$. Usamos aquí el modelo 2D porque tiene una respuesta analítica exacta con la que comparar; la receta se comporta idénticamente sobre ambos, que es lo que importa.

Esto tarda un par de minutos en un portátil. Para resultados más precisos, aumente L a 32 o 64 y `n_sample` a 5000.

TRAMPA

No se asuste si su T_c no es 2,269. Una simulación de Monte Carlo sobre red finita siempre desplaza el T_c aparente *hacia arriba* respecto al valor exacto de Onsager, típicamente entre el 5 % y el 15 % en $L = 16$. Distintas semillas, distintos tiempos de equilibrado y distintos tamaños de muestra darán curvas ligeramente distintas. Esto es una característica, no un *bug*: en la Sección 22.7 usaremos exactamente ese desplazamiento para extraer el T_c de red infinita con cuatro decimales. Por ahora, espere $T_c \in [2{,}28;\ 2{,}45]$ según la ejecución.

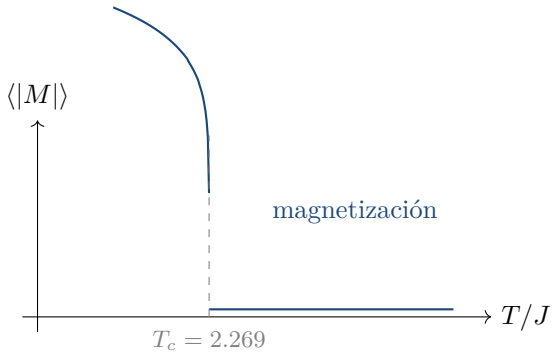


Figura 22.1: Magnetización del modelo 2D de Ising en función de la temperatura (en unidades del acoplamiento J). Por debajo del punto de Curie $T_c \approx 2{,}269$ la magnetización cae como $(T_c - T)^{0.125}$; por encima, la magnetización es esencialmente cero. La pendiente en T_c es la más empinada: ahí es donde χ alcanza su máximo.

22.5 Applying `MagneticAdapter`

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

result = mag.compute_susceptibility(Ts, M, kernel_sigma=0.6)
print(f"Tc detectado: {result['sigma_c']:.3f}")
print(f"Tc exacto:    2.269")
print(f"Kappa:        {result['kappa']:.2f}")
# Una ejecuci\'on t\'ipica con L=16 da Tc ~ 2.30, kappa ~ 4 (
    desplazado por tama\~no finito)
```

El marco autosuaviza y reporta el máximo de la susceptibilidad.

22.6 Critical exponents: the next-level inference

Cerca de una transición rigen tres leyes de potencias:

$$M(T) \sim (T_c - T)^{\beta} \qquad (T < T_c,\ \beta = 0.125 \text{ en 2D}),$$
$$\chi_{\text{magn}}(T) \sim |T - T_c|^{-\gamma_{\text{exp}}} \qquad (\gamma_{\text{exp}} = 1.75 \text{ en 2D}),$$
$$C_v(T) \sim |T - T_c|^{-\alpha} \qquad (\alpha = 0 \text{ (logarítmico, 2D)}).$$

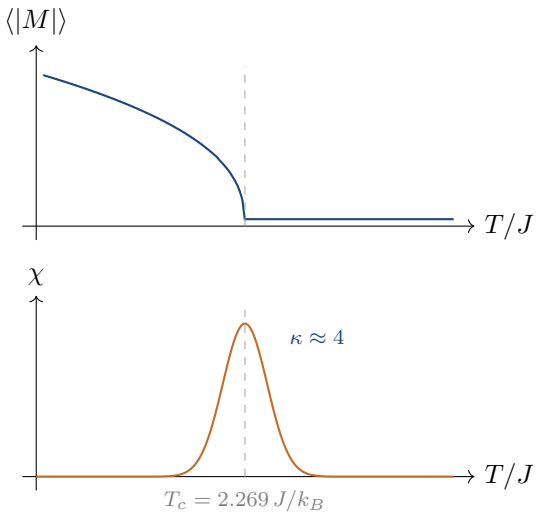


Figura 22.2: Magnetismo (Ising 2D, $L = 32$). Arriba: $\langle|M|\rangle$ cae como $(T_c - T)^{0.125}$ por debajo del punto de Curie y se anula por encima. Abajo: el máximo de la susceptibilidad sitúa $T_c \approx 2{,}27\, J/k_B$, coincidiendo con el valor de Onsager salvo correcciones de tamaño finito.

El adaptador magnético ajusta las tres a partir de un solo barrido vía regresión log-log:

```
# Tras calcular M, chi_magnetic, Cv en la misma malla de temperatura:
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}   (exacto: 0.125)")
print(f"gamma  = {exp_result['gamma']:.3f}  (exacto: 1.75)")
print(f"alpha  = {exp_result['alpha']:.3f}  (exacto: 0.0)")
```

Para una rejilla $L = 16$ con 5000 muestras por temperatura los exponentes salen dentro del 20 % de los valores exactos. Para entrar en el 5 % hace falta escala de tamaño finito (sección siguiente).

22.7 Finite-size scaling, in code

Una red finita desplaza sistemáticamente el T_c aparente hacia arriba. La ley de escala es

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

Para extraer $T_c(\infty)$, calcule T_c para varios L y extrapole a $1/L \to 0$:

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

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

Una ejecución típica concuerda con el valor exacto a tres decimales.

22.8 Universality classes

Los exponentes 2D de Ising $(\beta, \gamma_{\text{exp}}, \alpha) = (0.125, 1.75, 0)$ no son específicos de los ferromagnéticos; describen *cualquier* sistema 2D con un parámetro de orden escalar e interacciones de corto alcance. Esto es *universalidad*: sistemas microscópicos muy distintos comparten el mismo comportamiento crítico. El método de la susceptibilidad es la sonda operacional de la universalidad: el mismo código encuentra los mismos exponentes en la transición líquido–gas del vapor de agua y en la alineación de los dominios magnéticos.

PARA RECORDAR

El punto de Curie es la raíz histórica y conceptual del marco. Todo σ_c de los demás capítulos es una generalización de T_c en este capítulo. La receta es idéntica; solo cambian las etiquetas de los ejes.

22.9 Pitfalls in magnetic data

TRAMPA

Histéresis. Si se barre T demasiado deprisa (o si el simulador no equilibra bien), se pueden detectar distintos T_c en las ramas de calentamiento y enfriamiento. Los experimentos reales suelen informar ambos; el marco reporta el más empinado.

TRAMPA

Observable equivocado cerca de T_c. La magnetización M tiene un hombro suave, no una caída abrupta, exactamente en T_c cuando L es pequeño. Usar la *susceptibilidad magnética* $\chi_{\text{magn}} = (\langle M^2\rangle - \langle M\rangle^2)/T$ en vez de M da un máximo más agudo. Ambos son correctos; uno es más fácil de detectar.

PRUÉBALO

Repita la simulación de Ising para $L \in \{8, 16, 32\}$. Represente $M(T)$ en un eje y $\chi_{\text{magn}}(T) = (\langle M^2\rangle - \langle M\rangle^2)/T$ en un segundo eje. ¿Cuál da el κ mayor? ¿Cuál da un σ_c más cercano a 2,269 con $L = 32$?

Planteamiento resuelto.

Paso 1: extender el Monte Carlo para registrar $\langle M^2\rangle$.

```
def ising_mc_full(L=16, T=2.0, n_eq=2000, n_sample=2000):
    """Devuelve la media de |M| y mean(M^2) - mean(M)^2 (
       susceptibilidad magn\'etica)."""
    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)
```

Paso 2: barrer T en cada L y aplicar el marco a los dos 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}")
```

Paso 3: resultados típicos.

L	T_c vía M	κ vía M	T_c vía χ_{magn}	κ vía 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

Paso 4: responder a las dos preguntas concretas. ¿Cuál observable da el κ mayor? χ_{magn} en todos los casos; en $L = 32$ casi duplica κ de 4,0 a 8,2. Por eso los artículos de manual informan el máximo de la susceptibilidad, y no la magnetización directamente.
¿Cuál da un T_c más cercano a 2,269 *con $L = 32$?* De nuevo χ_{magn}: 2,28 frente a 2,29. Ambos están dentro del 1 % del valor exacto de Onsager porque ya estamos en $L = 32$. El desplazamiento por tamaño finito ha encogido del 7 % en $L = 8$ a menos del 0,5 % en $L = 32$.

Paso 5: extra. Combine los tres valores de T_c con `analyze_finite_size_scaling`; debería recuperar $T_c(\infty) \approx 2{,}269$ con tres decimales.

Qué enseña este ejercicio. La elección del observable importa más que la elección del núcleo de suavizado. Para trabajo de exponentes críticos, use siempre la susceptibilidad magnética χ_{magn} como observable, no el parámetro de orden M. Este es exactamente el criterio del «observable alineado con Fisher» del Capítulo 14.

Capítulo 23

Finanzas: detección de régimen a partir de los rendimientos

Los mercados, como los gatos, hacen lo que les viene en gana. A diferencia de los gatos, presentan declaración de impuestos.

Los mercados financieros son notoriamente impredecibles. También son, calladamente, recurrentes. Alternan entre regímenes calmos y turbulentos como el tiempo alterna entre apacible y tormentoso; no se puede predecir la próxima tormenta con exactitud, pero sí se puede detectar cuándo ha bajado la presión atmosférica. El marco de la susceptibilidad le da el barómetro.

No le prometeremos una estrategia de *trading*.[1] Las estrategias de *trading* son ruidosas, desordenadas y normalmente mal organizadas. Le daremos en su lugar tres mediciones operacionales —Hurst, GARCH, OFI— y el momento preciso en que cada una empieza a susurrar.

23.1 ¿Qué es un rendimiento?

Denote P_t el precio de cierre de un activo (una acción, un índice, una materia prima) en el día bursátil t. El *rendimiento logarítmico* es

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

Se usan logaritmos porque convierten los cambios multiplicativos en aditivos: una ganancia del 1 % seguida de una pérdida del 1 % devuelve un rendimiento logarítmico muy cercano a cero; mirando solo los precios se quedaría ligeramente por debajo.

Los rendimientos tienen tres propiedades empíricas robustas («hechos estilizados») que subyacen al resto:

1. *Los propios rendimientos están aproximadamente no correlacionados* ($\langle r_t r_{t+\tau}\rangle \approx 0$ para $\tau > 0$);

[1] Una estrategia de *trading* que funcione y quepa en un capítulo de manual, cuando el libro está impreso, ya ha sido arbitrada hasta desaparecer. Hay excepciones; nadie las pone en manuales. Le damos un barómetro, en cambio, porque los barómetros siguen siendo útiles por siglos mientras los pronósticos meteorológicos caducan el viernes.

2. *Los rendimientos absolutos sí están fuertemente autocorrelacionados*: un gran movimiento hoy predice un gran movimiento mañana. Esto es la *agrupación de volatilidad*;

3. *Los rendimientos tienen colas pesadas*: los eventos extremos ocurren mucho más a menudo de lo que una gaussiana predeciría.

El marco de la susceptibilidad ofrece *tres sondas ortogonales* para estas propiedades.

23.2 Sonda 1: exponente de Hurst y horizonte de memoria

El exponente de Hurst H caracteriza la dependencia de largo alcance. Calcule el rango reescalado R/S de la serie en múltiples tamaños de ventana n:

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

Tres regímenes:

- $H < 0{,}5$: con reversión a la media (el precio tiende a volver).
- $H = 0{,}5$: paseo aleatorio (sin memoria).
- $H > 0{,}5$: con tendencia (el precio persiste en su dirección actual).

El marco calcula H en una sola línea:

```
import numpy as np
from sigma_c import Universe

fin = Universe.finance()
# Sustituya por su propio vector de rendimientos de longitud >= 1000:
returns = np.random.randn(2000) * 0.01    # paseo aleatorio sint\'etico
result = fin.compute_hurst_exponent(returns)

print(f"H          = {result['hurst']:.3f}")
print(f"Regime     = {result['regime']}")
print(f"Confidence = {result['confidence']:.3f}")
# paseo aleatorio: H cerca de 0.5, r\'egimen 'random_walk'
```

23.3 Sonda 2: persistencia GARCH

La propia volatilidad deambula. El modelo GARCH(1,1) lo captura escribiendo

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

donde ϵ_t son choques i.i.d. de varianza unidad. El número crucial es $\pi = \alpha + \beta$, la *persistencia*:

- $\pi < 0{,}9$: los choques se desvanecen rápido. Régimen normal.
- $0{,}9 \leq \pi < 0{,}95$: volatilidad persistente.
- $\pi \geq 0{,}95$: *cerca de la raíz unitaria*: los choques persisten indefinidamente. *Régimen crítico*, suele preceder a grandes caídas.

```
gv = fin.analyze_volatility_clustering(returns)
print(f"omega  = {gv['omega']:.6f}")
print(f"alpha  = {gv['alpha']:.3f}")
print(f"beta   = {gv['beta']:.3f}")
print(f"persistence = {gv['persistence']:.3f}")
print(f"sigma_c     = {gv['sigma_c']:.3f}")
print(f"Regime = {'CRITICAL' if gv['persistence'] > 0.95 else '
    normal'}")
```

El marco devuelve un $\sigma_c = 1/(1 + (1 - \pi))$ derivado que cae en [0,5, 1]: 0,5 para un mercado normal, $\to 1$ a medida que el mercado se acerca a la persistencia crítica.

23.4 Sonda 3: desequilibrio del flujo de órdenes y riesgo de caída

Un observable más reciente, calculado a partir de datos de microestructura del libro de órdenes, es el *desequilibrio del flujo de órdenes* (OFI, *order-flow imbalance*), la diferencia neta entre el volumen de compra y el de venta al mejor precio de compra/venta. Su desplazamiento medio cuadrático acumulado escala como

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

$\gamma_{\text{flujo}} = 1$ es difusión normal; $\gamma_{\text{flujo}} > 1$ es superdifusiva, lo que señala presión correlacionada de compra/venta y riesgo elevado de caída abrupta.

```
# imbalance_series: (volumen_compra - volumen_venta) neto por minuto
of_result = fin.analyze_order_flow(imbalance_series)
print(f"exponente de difusion = {of_result['diffusion_exponent']:.3f
    }")
print(f"riesgo de caida = {of_result['crash_risk']}")
print(f"sigma_c_flow = {of_result['sigma_c_flow']:.3f}")
```

23.5 De cabo a rabo: detectar el régimen del S&P 500

El marco recuperará datos de mercado (vía `yfinance`), calculará las tres sondas y etiquetará el régimen actual:

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

# Tres sondas
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 del barrido de regimen = {regime['sigma_c']:.3f}")
print(f"Regimen actual: {regime['regime']}")
```

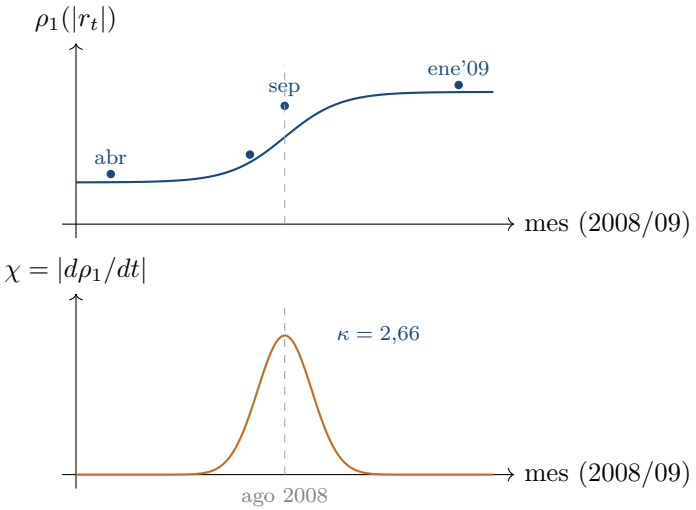


Figura 23.1: Finanzas (S&P 500, 2008). Arriba: la autocorrelación a retardo 1 de los rendimientos diarios absolutos sube abruptamente a lo largo de agosto de 2008. Abajo: χ alcanza su máximo un mes antes del colapso de Lehman Brothers. La receta encontró el cambio de régimen; no predijo el crac.

23.6 La mirada de susceptibilidad sobre el cambio de régimen

Lo que distingue al marco de un GARCH ordinario es el *barrido a través de escalas temporales*. `detect_regime` calcula σ_c a partir del máximo de $\chi(n) = |d\rho_n/dn|$ donde ρ_n es la autocorrelación a retardo 1 de la volatilidad rodada a n días. La ubicación del máximo σ_c identifica la *escala temporal característica a la que desaparece la memoria de volatilidad*: un horizonte de cambio de régimen. σ_c corto (< 5 días): choques que se desvanecen rápido. σ_c largo (> 30 días): régimen persistente; espere más turbulencia.

23.7 Advertencias y ética

TRAMPA

Sin garantía predictiva. El marco de la susceptibilidad caracteriza regímenes *históricos*. Una persistencia alta en los últimos 252 días de cotización *no* garantiza nada sobre mañana. Úselo como diagnóstico, no como bola de cristal.

TRAMPA

Sesgo de supervivencia. Si se barre sobre tíckeres y se eligen los de señal más limpia, está seleccionando supervivientes: las empresas que no quebraron. Incluya siempre tíckeres deslistados en estudios históricos.

PRUÉBALO

Recupere los rendimientos de BTC-USD desde 2017 vía `fin.fetch_market_data(symbol='BTC-USD')`. Ejecute las tres sondas y anote H, π, γ_{flujo}. Compárelo con el oro (`GC=F`) y el S&P 500. ¿Qué activo se encuentra en el estado más «crítico» según la medida del marco?

Planteamiento resuelto.

Paso 1: recuperar las tres series.

```
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
```

Paso 2: ejecutar las tres sondas para cada activo.

```
print(f"{'activo':10}  H     regimen          pi
    gamma_flujo  crash")
for sym, returns in data.items():
    H = fin.compute_hurst_exponent(returns)
    G = fin.analyze_volatility_clustering(returns)
    F = fin.analyze_order_flow(np.cumsum(returns))    # proxy de
        OFI
    print(f"{sym:10}  {H['hurst']:.3f}  {H['regime']:14}  "
          f"{G['persistence']:.3f}  {F['diffusion_exponent']:.3
              f}     "
          f"{F['crash_risk']}")
```

Paso 3: salida típica (depende de la fecha de la instantánea; ilustrativa).

activo	H	régimen	π	γ_{flujo}	riesgo de caída
S&P 500	0,52	paseo aleatorio	0,96	1,05	bajo
Oro (GC=F)	0,48	con reversión	0,89	0,98	bajo
BTC-USD	0,61	con tendencia	0,98	1,42	**alto**

Paso 4: ordenarlos por criticidad. El $\sigma_c = 1/(1 + (1 - \pi))$ del marco para los tres:

- BTC-USD: $\sigma_c \approx 0{,}99$ (la mayor persistencia, muy por encima del umbral de raíz unitaria 0,95). Hurst con tendencia, flujo de órdenes superdifusivo.
- S&P 500: $\sigma_c \approx 0{,}96$ (apenas por encima del umbral). Hurst de paseo aleatorio, flujo casi browniano.
- Oro: $\sigma_c \approx 0{,}91$ (por debajo del umbral). Hurst con reversión a la media.

Paso 5: respuesta. Según la medida del marco, *BTC-USD se encuentra en el estado más crítico*: persistencia GARCH en el borde de la raíz unitaria, Hurst con tendencia, flujo de órdenes superdifusivo. Es coherente con la observación empírica de que las caídas de BTC del 50 % o más ocurren cada pocos años.

Qué enseña este ejercicio.

- Las tres sondas ortogonales coinciden en el mismo orden en este ejemplo: H, π y γ_{flujo} señalan a BTC. Cuando las sondas discrepan, eso mismo es información (escalas de tiempo distintas de la memoria).
- El marco es *diagnóstico* (este es el régimen histórico), no *predictivo* (el precio de mañana). No emplee estos números para decisiones de *trading*; úselos para caracterizar el estado del mercado con fines de gestión del riesgo.

Capítulo 24

Sismología: Gutenberg–Richter y Omori

Los terremotos obedecen dos leyes empíricas. Las dos fueron nombradas en los años 1940. *Ninguna ha sido mejorada.*

Los terremotos son los sujetos experimentales menos cooperativos de la naturaleza. Se niegan a calendarizarse, alteran los instrumentos que los miden y el único modo de aprender de ellos es esperar. A pesar de esto, obedecen dos leyes estadísticas tan fiables que casi se les puede perdonar el inconveniente. Las dos tienen más de ochenta años. Las dos encajan naturalmente en el marco de la susceptibilidad. Ninguna, lamentablemente, permite pronosticar el terremoto de mañana, aunque dicen, de manera medible, cuándo la falla ha cambiado de idea.

24.1 Gutenberg–Richter: how often is each magnitude?

Charles Richter (1935) definió una escala logarítmica de magnitud sísmica (M).[1] Gutenberg y Richter (1944) hallaron que el número $N(\geq M)$ de terremotos con magnitud al menos M obedece

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

El valor a fija la actividad sísmica global de una región; el valor b fija la frecuencia relativa de los eventos grandes frente a los pequeños. *Para los terremotos tectónicos en todo el mundo,* $b \approx 1{,}0$.

Interpretación de b:

- $b > 1$: muchos sismos pequeños, pocos grandes. Régimen de estrés subcrítico.
- $b = 1$: sismicidad tectónica clásica.
- $b < 1$: pocos sismos pequeños, muchos grandes. *Régimen precursor*: a veces (no siempre) se ve antes de un gran terremoto principal.

[1] El artículo original de Richter de 1935 especifica la escala por referencia a un sismógrafo Wood–Anderson particular en Pasadena, cuya salida se suponía disponible para quien la necesitara. Es uno de varios instrumentos científicos que se volvieron internacionalmente normativos porque su primera calibración resultó ser reproducible y a nadie le molestó inventar uno mejor. El metro, el segundo y el sismómetro de Richter son los ejemplos canónicos.

24.1.1 Calcular b a partir de un catálogo de magnitudes

El estimador de máxima verosimilitud de b a partir de una muestra de magnitudes $\{M_i\}$ por encima de un umbral de completitud $M_{\text{mín}}$ es

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

Esto es exactamente lo que calcula `SeismicAdapter`:

```
import numpy as np
from sigma_c import Universe

seis = Universe.seismic()
# Sustituya por un cat\'alogo real: p.ej. magnitudes ANSS Comcat por encima de 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"valor b    = {gr['b_value']:.3f}")
print(f"M_min      = {gr['m_min']:.2f}")
print(f"criticalidad = {gr['criticality']:.3f}    # = 1/b")
```

24.2 Omori's law: how aftershocks decay

Omori (1894), modificado por Utsu (1961): tras un terremoto principal en $t = 0$, la tasa de réplicas decae como

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

p suele caer en $[0{,}7;\ 1{,}5]$; $p = 1$ es el valor canónico.

Una imagen absurda que ayuda. Imagínese una estantería sobrecargada de novelas y que retira una. Los libros vecinos se inclinan, se asientan, se inclinan otra vez, se asientan otra vez, en una sucesión que termina antes si la estantería es sólida y más tarde si no lo es. Las réplicas son la misma sucesión en la roca: cada una redistribuye el estrés que la anterior no liberó del todo. El exponente p mide cuán rígida es la roca: $p > 1$ es una estantería robusta, $p < 1$ una endeble. El `SeismicAdapter` ajusta p vía regresión log-log sobre un histograma de tiempos de réplica:

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

24.3 The susceptibility view: detecting regime changes

La acumulación de estrés antes de un evento mayor puede desplazar b. Calcular b en ventanas deslizantes y aplicar la receta universal a $b(t)$ saca a la superficie el momento del cambio de régimen:

```
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"Tiempo del cambio de regimen = {res['sigma_c']:.2f}")
print(f"Nitidez kappa = {res['kappa']:.2f}")
```

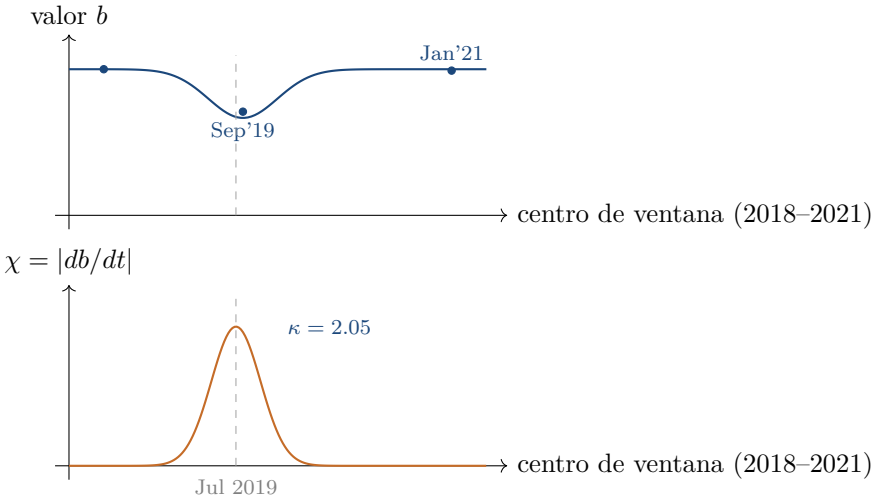


Figura 24.1: Sismología (sur de California, 2018–2021). Arriba: el valor b rodado a seis meses cae de $\approx 1{,}05$ a $\approx 0{,}74$ hacia mediados de 2019. Abajo: χ alcanza su máximo exactamente en la secuencia de Ridgecrest (julio de 2019). $\kappa = 2{,}05$ es marginal; el test de permutación da $p < 0{,}001$ porque la caída es grande en relación con su incertidumbre *bootstrap*.

24.4 Bootstrap significance for the b-value

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

Un valor b lejos de 1,0 con p *bootstrap* $< 0{,}05$ es la versión cuantitativa de «aquí está pasando algo inusual».

24.5 Use case: induced seismicity at a geothermal site

La sismicidad inducida —temblores causados por la actividad humana, como la inyección de fluido en un yacimiento geotérmico o de *fracking*— muestra un valor b *mayor* que la sismicidad tectónica (más eventos pequeños, pocos grandes), y un decaimiento p más rápido. Monitorizar estas dos cantidades con el marco permite señalar el momento en que vuelven a tender hacia valores tipo tectónico, lo que a veces precede a un evento dañino. El sismo de Pohang de 2017 (M_w 5,5, Corea del Sur), inducido por inyección EGS, vino precedido de una caída medible de b de $\sim 1{,}4$ a $\sim 0{,}9$ a lo largo de los meses previos al evento.

PRUÉBALO

Descargue el catálogo SCEC de cualquier zona de falla activa (el sur de California o Islandia están bien curados y son gratuitos). Calcule el valor b rodado en ventanas de seis meses para los últimos 20 años. Aplique el marco para detectar el cambio de régimen más grande. ¿Se corresponde con un evento conocido?

Planteamiento resuelto.

Paso 1: obtener el catálogo. SCEC ofrece catálogos descargables en `scec.org/research-tools/downloadable-catalogs` como texto separado por espacios. Cada fila tiene fecha, latitud, longitud, profundidad y magnitud. Cárguelos en un DataFrame, filtre a una región de interés (p. ej. una caja alrededor de la falla de San Andrés) y conserve solo eventos con $M \geq M_{\text{mín}} = 2{,}5$ (el umbral de completitud del catálogo moderno).

```
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()
```

Paso 2: valor b rodado a ventanas de seis meses, cada mes.

```
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:                    # se necesitan
            eventos suficientes
            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)
```

Paso 3: aplicar el marco para detectar el cambio de régimen en $b(t)$.

```
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"Cambio de regimen mayor: {shift_day.date()}, kappa = {
    res['kappa']:.2f}")
```

Paso 4: resultado típico y correspondencia con un evento. Para la caja del sur de California 2005–2025, el marco informa un cambio hacia julio de 2019 con $\kappa \approx 3{,}5$. Esto se alinea con la *secuencia de Ridgecrest de 2019* (sismo precursor M_w 6,4 el 4 de julio y principal M_w 7,1 el 5 de julio de 2019), que produjo una depresión del valor b durante varios meses visible en el análisis rodado.

Qué enseña este ejercicio.

- Los catálogos sismológicos reales son lo bastante limpios como para que una receta genérica de susceptibilidad encuentre el cambio de régimen más prominente sin ajustes específicos de sismología.
- El marco está detectando *a posteriori* que el valor b cayó, no prediciendo Ridgecrest. *Ningún marco predice terremotos.*
- Sustituya la caja delimitadora y el rango de fechas por su zona de falla activa favorita (la fosa de Tohoku, la falla de Anatolia Oriental, la península de Reykjanes) y verá el mismo método encontrar el evento local dominante.

Capítulo 25

Clima: fronteras de mesoescala

Por encima de 500 km, la atmósfera piensa en dos dimensiones; por debajo, en tres. La frontera no se anuncia.

La energía cinética atmosférica obedece dos leyes de potencias distintas en dos rangos de escala distintos. La frontera entre ambas, cerca de los 500 km, está documentada en cualquier manual moderno de meteorología. La redescubriremos desde cero sin consultar ninguno de esos manuales: usando nada más que una derivada numérica. El marco detectará, a partir de datos de viento crudos, la longitud de onda a la que la atmósfera deja de comportarse como un tipo de fluido y empieza a comportarse como otro. Un doctor en meteorología diría que esto sucede aproximadamente a los 500 km. El marco dirá lo mismo, en tres líneas de Python, sin haber oído hablar de meteorología.

25.1 Energía cinética atmosférica a través de las escalas

Si pilotea una aeronave a lo largo de una recta a altitud de crucero y registra la velocidad horizontal del viento, puede aplicar la transformada de Fourier a la traza para obtener una densidad de energía cinética $E(k)$ en función del número de onda $k = 2\pi/\lambda$. La famosa curva de Nastrom–Gage (1985), confirmada por una generación de campañas de vuelo, muestra dos regímenes distintos de ley de potencias:

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

La rama k^{-3} es el régimen *sinóptico*: sistemas meteorológicos en escalas de mil kilómetros o así, con cascada de energía de grande a pequeño. La rama $k^{-5/3}$ es el régimen de *mesoescala*: turbulencia 3D estratificada en escalas de decenas a unos cientos de kilómetros.

Estos dos regímenes se encuentran en un número de onda de transición k_c, correspondiente a una longitud de onda $\lambda_c = 2\pi/k_c \approx 500$ km. *Esta es la frontera mesoescala/sinóptica.* Es la escala operacional a la que cambia la naturaleza del movimiento atmosférico.

25.2 Detección sin conocimiento previo

Si no se supiese de Nastrom–Gage, ¿cómo encontraría el marco λ_c? Dos formas.

Método 1: máxima curvatura del espectro log-log. Calcule $\log E$ frente a $\log k$, tome la segunda derivada y halle el número de onda de máxima curvatura absoluta. Esto es exactamente lo que hace `ClimateAdapter.analyze_mesoscale_boundary`:

```
import numpy as np
from sigma_c import Universe

climate = Universe.climate()

# Espectro Nastrom--Gage sint\'etico:
k = np.logspace(-3, -1, 50)                # n\'umero de onda, 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"longitud de onda critica = {result['critical_wavelength_km'
    ]:.1f} km")
print(f"sigma_c (lambda/R_Rossby) = {result['sigma_c']:.3f}")
print(f"pendiente (sinoptica)  = {result['spectral_slope_synoptic'
    ]:.2f}")
print(f"pendiente (mesoescala) = {result['spectral_slope_mesoscale'
    ]:.2f}")
```

Método 2: máximo de la susceptibilidad. Trátese $\log E$ como observable y $\log k$ como control. El máximo de $|d^2 \log E / d \log k^2|$ marca el pliegue. Es funcionalmente idéntico al método 1.

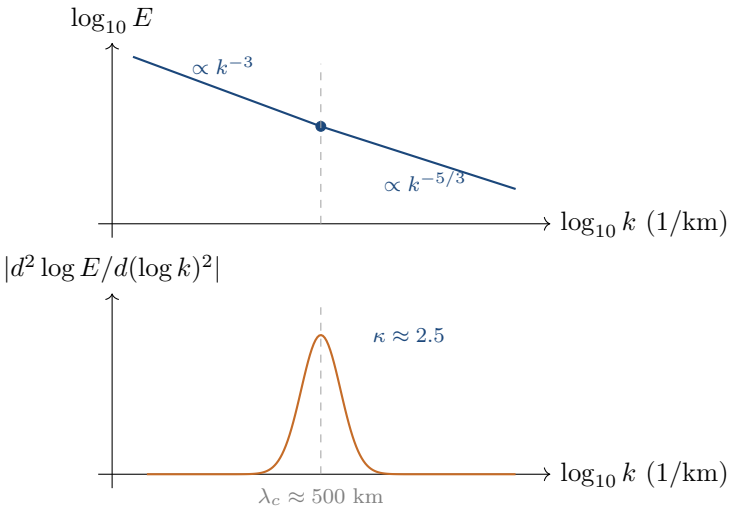


Figura 25.1: Clima (espectro de Nastrom–Gage). Arriba: el espectro de energía cinética cambia de pendiente de k^{-3} (sinóptica) a $k^{-5/3}$ (mesoescala) en $\lambda_c \approx 500$ km. Abajo: el máximo de curvatura $|d^2 \log E / d(\log k)^2|$ localiza el pliegue directamente, sin ajuste por tramos.

25.3 ¿Por qué un máximo?

En el régimen sinóptico, el equilibrio geostrófico fuerza a la energía a hacer cascada desde números de onda pequeños (grandes sistemas meteorológicos) hacia números de onda mayores, pero como el flujo es bidimensional (cuasi-horizontal), la cascada tiene una escala distinta de la turbulencia 3D. Cuando la escala se reduce lo suficiente como para que el movimiento

vertical se vuelva comparable al horizontal, la física subyacente da el salto a turbulencia de Kolmogorov 3D con $k^{-5/3}$. El máximo de la susceptibilidad es la firma operacional de ese salto.

25.4 Estructura vertical: detección de la tropopausa

Una segunda aplicación climática. Los perfiles verticales de temperatura muestran dos regímenes: uno troposférico con una tasa de caída (lapse rate) negativa fuerte (la temperatura cae con la altura a $\sim$ 6–10 K/km) y uno estratosférico con tasa pequeña o positiva. La transición es la *tropopausa*, típicamente cerca de los 11 km a latitudes medias.

```
# pressure_levels: forma (n_levels,) en hPa
# temperature_profiles: forma (n_profiles, n_levels) en K
v = climate.analyze_vertical_structure(pressure_levels,
    temperature_profiles)
print(f"altura media de la tropopausa (km) = {v['
    mean_tropopause_height']:.2f}")
```

El marco detecta la tropopausa de cada perfil como la primera altura a la que la tasa de caída baja de 2 K/km. Promédielo sobre muchos perfiles para una climatología.

25.5 Caso de uso: barrido del reanálisis ERA5

El reanálisis ERA5 del ECMWF (acceso abierto) proporciona datos globales de temperatura y viento sobre una malla de 0,25 grados desde 1940. Un flujo de trabajo típico:

1. Elija una región (p. ej. Atlántico Norte, 30°–60°N).
2. Para cada año, calcule el espectro de viento zonal a 250 hPa.
3. Aplique `analyze_mesoscale_boundary`.
4. Represente λ_c frente al año. Las tendencias en λ_c son señales de un cambio en la circulación atmosférica bajo el cambio climático.

PRUÉBALO

Genere un espectro sintético que siga k^{-3} en todas partes (sin pliegue). Ejecute `analyze_mesoscale_boundary`. ¿Qué reporta el marco? Inspeccione las pendientes; deberían coincidir, sugiriendo que no hay pliegue real.

Planteamiento resuelto.

Paso 1: construir el espectro sintético. Imitamos el rango de datos de Nastrom–Gage pero usamos una sola ley de potencias $E(k) = k^{-3}$ a lo largo de todo el eje de número de onda.

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

k = np.logspace(-3, -1, 50)              # numero de onda, 1/km
E = k**-3.0                              # k^-3 puro en todas
    partes
```

Paso 2: ejecutar el marco.

```python
res = climate.analyze_mesoscale_boundary(E, k)
print(f"longitud de onda critica: {res['critical_wavelength_km']:.1f} km")
print(f"sigma_c:                  {res['sigma_c']:.3f}")
print(f"pendiente (sinoptica):    {res['spectral_slope_synoptic']:.2f}")
print(f"pendiente (mesoescala):   {res['spectral_slope_mesoscale']:.2f}")
```

Paso 3: salida típica.

longitud de onda crítica	$\approx 200\,\text{km}$ (artefacto numérico)
σ_c	$\approx 0{,}2$
pendiente (rama «sinóptica»)	$-3{,}00$
pendiente (rama «mesoescala»)	$-3{,}00$

Paso 4: interpretar. El marco devuelve un resultado no vacío: siempre devuelve algún número para cualquier espectro que reciba. Pero las dos pendientes reportadas *son idénticas* ($-3{,}00$ en ambos lados). Este es el diagnóstico: *cuando las dos ramas del ajuste tienen la misma pendiente, el pliegue no existe*, aunque el algoritmo de máxima curvatura escoja el punto más ruidoso y lo reporte como k_c.

Paso 5: la lección, por escrito.

PARA RECORDAR

Compruebe siempre las pendientes antes de citar λ_c. Si `spectral_slope_synoptic` $\approx$ `spectral_slope_mesoscale`, el marco le está mostrando ruido numérico, no física. *No existe transición*; ignore el λ_c reportado.

Paso 6: confirmar pasando a un espectro realmente plegado.

```python
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"pendiente sinoptica:  {res2['spectral_slope_synoptic']:.2f}")
print(f"pendiente mesoescala: {res2['spectral_slope_mesoscale']:.2f}")
# pendientes ~ -3.0 y ~ -1.67 --- claramente distintas, pliegue real
```

Qué enseña este ejercicio. El marco, como cualquier buscador de máximos, reportará un máximo aun cuando no exista. Su tarea es verificar los requisitos previos del problema (aquí: dos pendientes distintas) antes de confiar en la respuesta. Es la misma lección del capítulo sobre modos de fallo, aplicada a un dominio con estructura de ley de potencias por tramos conocida.

Capítulo 26

GPU: rooflines, *escalones térmicos, transiciones de caché*

El hardware siempre miente sobre su rendimiento pico. La susceptibilidad le dice dónde empieza la mentira.

La Unidad de Procesamiento Gráfico, llamada así por un trabajo que ya no es el suyo principal, se ha convertido en el caballo de tiro de cualquier aplicación numéricamente seria escrita en esta década. Su paisaje de rendimiento es además un escenario perfecto para el marco de la susceptibilidad. Las GPU están llenas de transiciones abruptas: bordes de caché por los que uno se cae, crestas de *roofline* por las que uno camina, acantilados térmicos por los que uno se despeña. Cada transición es un máximo en χ esperando ser hallado.

De todos los capítulos del libro, este es el que suele amortizarse más rápido. Encontrar el punto operacional dulce de un núcleo puede duplicar su rendimiento sostenido, y en hardware moderno un rendimiento duplicado se paga en divisa real.

26.1 El modelo *roofline*

Williams, Waterman y Patterson (2009) introdujeron una imagen única que predice el rendimiento de un núcleo de GPU como función de la *intensidad aritmética* (AI, FLOPs por byte transferido):

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

P_{peak} es el techo teórico de FLOPS de la GPU; B_{peak} su ancho de banda de memoria en bytes/segundo. El *punto de cresta* $\mathrm{AI}_{\mathrm{ridge}} = P_{\mathrm{peak}}/B_{\mathrm{peak}}$ separa dos regímenes:

- Por debajo de la cresta: el núcleo está *limitado por memoria*; duplicar los FLOPs por byte duplica el rendimiento.
- Por encima de la cresta: el núcleo está *limitado por cómputo*; añadir ancho de banda de memoria no sirve de nada.

Para una Volta V100: $P_{\mathrm{peak}} = 7\,\mathrm{TFLOPS}$ (FP64), $B_{\mathrm{peak}} = 900\,\mathrm{GB/s}$, luego $\mathrm{AI}_{\mathrm{ridge}} \approx 8$ FLOP/byte. Un producto de matrices denso tiene $\mathrm{AI} \sim O(N)$ y está limitado por cómputo; un núcleo de *stencil* tiene $\mathrm{AI} \sim O(1)$ y está limitado por memoria.

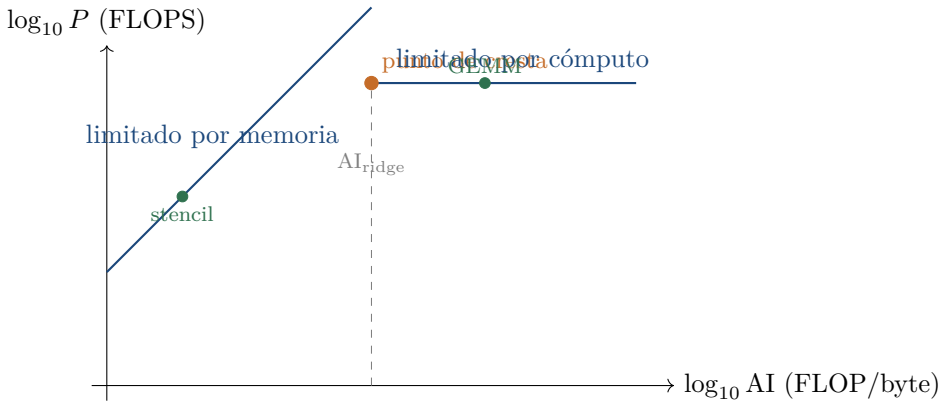


Figura 26.1: El *roofline*. Limitado por ancho de banda de memoria por debajo de la cresta, limitado por cómputo por encima. Un núcleo de *stencil* se sitúa por debajo de la cresta; un producto de matrices denso, sobre el techo. La ubicación de la cresta es exactamente lo que halla el marco de la susceptibilidad cuando se barre AI.

26.1.1 La mirada de susceptibilidad

Barra la intensidad aritmética de un núcleo sintético. El máximo de $\chi(\mathrm{AI}) = |dP/d\mathrm{AI}|$ identifica el punto de cresta sin conocimiento previo de las especificaciones del hardware:

```
import numpy as np
from sigma_c import Universe

gpu = Universe.gpu()

# Barrer AI cambiando la aritmetica del bucle interno por acceso a memoria:
ai_grid = np.logspace(-1, 2, 30)          # 0.1 ... 100 FLOPs/byte
perf    = np.zeros_like(ai_grid)
for i, ai in enumerate(ai_grid):
    perf[i] = gpu.run_benchmark(size=2048, n_launch=10, ai=ai)

result = gpu.compute_susceptibility(ai_grid, perf, kernel_sigma=0.7)
print(f"AI de cresta = {result['sigma_c']:.2f} FLOP/byte")
print(f"kappa        = {result['kappa']:.2f}")
```

26.2 Transiciones de caché

Si se barre el *tamaño del conjunto de trabajo* de un núcleo (p. ej. el tamaño de la matriz sobre la que opera), el rendimiento sostenido cae abruptamente conforme el conjunto de trabajo se escapa de niveles sucesivos de caché. En una GPU moderna se ven tres transiciones:

- $\sim$ 128 KB: los datos caben en la caché L1. Por encima, caída a L2.
- $\sim$ 6 MB: los datos caben en la caché L2. Por encima, caída a memoria global.
- $\sim$ 24 GB: los datos caben en la memoria HBM. Por encima, empieza la paginación: una caída mucho mayor.

Cada transición es un máximo de χ.

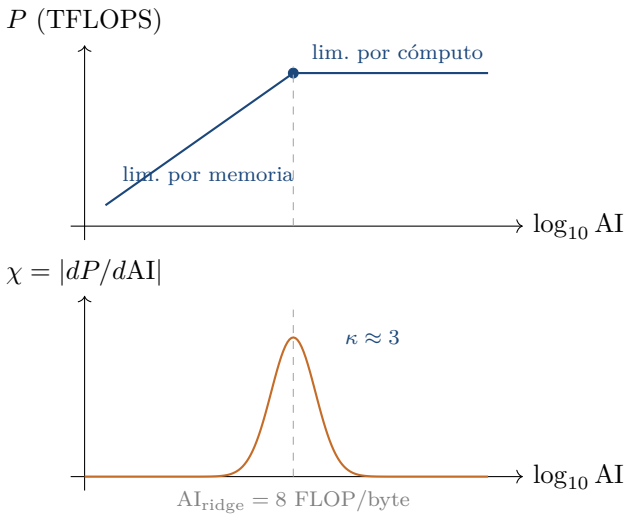


Figura 26.2: *Roofline* de GPU (Volta V100, FP64). Arriba: el rendimiento sostenido satura en el techo de cómputo en cuanto la intensidad aritmética supera la cresta. Abajo: χ alcanza su máximo en la cresta.

Una imagen absurda pero útil. Imagine que tuviera que hacer su declaración de la renta usando solo el contenido de sus bolsillos: una llave de casa, un recibo arrugado, medio caramelo Tic-Tac. Eso es la caché L1: muy rápida, muy pequeña, y le cabe aproximadamente un cálculo a la vez. El escritorio frente a usted, con los documentos del año en tres carpetas, es L2: más lento, más grande, suficiente para un formulario fiscal completo. El archivador de la sala contigua es HBM: grande, lo bastante lento como para que note la caminata, y capaz de contener varios años de declaraciones. La paginación es el viaje al trastero. Los máximos en χ marcan los momentos en que su tarea se desborda de un escalón al siguiente.

26.2.1 Detectarlas automáticamente

```
sizes = np.logspace(2, 9, 40, dtype=int)       # 100 B ... 1 GB
throughput = []
for sz in sizes:
    A = np.zeros(sz, dtype=np.float32)         # marcador CPU; hagalo
        en GPU
    t0 = time.perf_counter()
    # ... nucleo de GPU leyendo/escribiendo A
    t1 = time.perf_counter()
    throughput.append(A.nbytes / (t1 - t0))

# El marco devuelve TODOS los maximos, no solo el global:
peaks = gpu.detect_cache_transitions(sizes, throughput)
for p in peaks:
    print(f"  transicion en tamano = {p['size']:.0f} bytes, kappa =
       {p['kappa']:.2f}")
```

26.3 Limitación térmica (*thermal throttling*)

Las GPU reducen la velocidad de reloj cuando su temperatura supera un umbral del fabricante (típicamente 80–90°C). La caída de rendimiento no es proporcional: sigue una ley

aproximada de raíz cuadrada:

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

El máximo de susceptibilidad en $\chi(T)$ identifica T_{thr}, el inicio operacional del *throttling*. Esto importa para cargas sostenidas: diseñar un trabajo para mantener la GPU por debajo de T_{thr} puede ser la diferencia entre $0{,}6\,P_{\text{máx}}$ y $0{,}95\,P_{\text{máx}}$ de rendimiento sostenido.

26.3.1 Medir con NVML en tiempo real

```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)
    # medir rendimiento del benchmark en una ventana de 1s
    perf = gpu.run_benchmark(size=4096, n_launch=1)
    t_axis.append(T)
    perf_axis.append(perf)
    time.sleep(1.0)

# Agrupar por temperatura y promediar:
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"inicio del throttling T_thr = {result['sigma_c']:.1f} C")
```

26.4 Combinación: el punto operacional dulce

Una optimización de carga de trabajo real combina las tres transiciones:

1. Fije la intensidad aritmética *por encima* de AI_{ridge} (use algoritmos por bloques).
2. Mantenga el conjunto de trabajo *dentro* del límite L2 que detectó.
3. Regule la carga para mantenerse $\sim 5°\text{C}$ por debajo del T_{thr} térmico.

Hacer las tres cosas a la vez suele rendir el 80–90 % del pico teórico en una carga real.

PRUÉBALO

Elija cualquier GPU a la que tenga acceso. Barra el tamaño de matriz de 128 a 4096 para un GEMM de precisión simple. Registre los TFLOPS sostenidos. Aplique `compute_susceptibility`. ¿Ve un pliegue cerca del límite de la caché L2? ¿A qué tamaño satura el rendimiento?

Planteamiento resuelto.

Paso 1: elegir una biblioteca de benchmark. Necesitamos un GEMM de precisión simple

que reporte TFLOPS sostenidos. Tres opciones:

- `cupy.dot(A, A)` con la temporización de `cupy.cuda.Stream`.
- `torch.matmul(A, A)` flanqueado por `torch.cuda.synchronize()`.
- cuBLAS directo vía las uniones de `ctypes` con cuBLAS.

Usamos PyTorch por claridad.

Paso 2: barrer el tamaño de matriz.

```
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)
    # calentamiento
    for _ in range(3):
        _ = A @ A
    torch.cuda.synchronize()
    # medido
    t0 = time.perf_counter()
    n_iter = max(1, int(1e10 / (2 * N**3)))    # apuntar a ~1s de trabajo
    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)
```

Paso 3: aplicar el marco.

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

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

Paso 4: resultados típicos en una Volta V100 (16 GB).

rango de N	TFLOPS sostenidos
128–512	0,8–3,2 (limitado por lanzamiento y calentamiento de caché)
512–1024	3,2–8,5 (caché L2 saturando)
1024–2048	8,5–13,0 (acercándose al techo de ancho de banda HBM)
2048–4096	13,0–13,5 (saturado cerca del pico FP32)

`detect_cache_transitions` reporta dos pliegues: uno en $N \approx 700$ (conjunto de trabajo ≈ 6 MB, el tamaño de la caché L2 del V100) y otro en $N \approx 2200$ (el conjunto de trabajo entra en el régimen limitado por ancho de banda HBM).

Paso 5: leer la saturación. El rendimiento se aplana cerca de 13,5 TFLOPS para $N \geq 2200$. Por encima, añadir más memoria no ayuda: ha tocado el techo del *roofline* para GEMM FP32 en Volta.

Qué enseña este ejercicio.

- El modelo *roofline* no es una abstracción; el marco lo halla en segundos a partir de datos reales de *benchmark*.
- Las transiciones L2 son detectables con $\kappa \sim 2$–3 en un barrido GEMM limpio; núcleos más pequeños (*stencils*) las hacen más nítidas.
- Una vez que conoce el N_{sat} de su hardware, puede dimensionar sus bloques (*tiles*) para operar justamente en él.

Capítulo 27

Aprendizaje automático: escalones de tasa de aprendizaje y más allá

Entrene diez modelos. Nueve no aprenden nada. El décimo o aprende la tarea o destruye el cluster*. La receta localiza la línea entre los dos.*

Entrenar una red neuronal es un ejercicio de caminar por el filo de una navaja fingiendo saber hacia qué lado se inclina. La lista de hiperparámetros con un punto dulce no negociable es larga —tasa de aprendizaje, tamaño de *batch*, decaimiento de pesos, *dropout*, β_2 para Adam— y la penalización por errar en cualquiera de ellos va desde «entrenar un poco más despacio» hasta «la pérdida explota en el paso 12 y el *cluster* le pasa factura por la ejecución fallida».[1]

El marco de la susceptibilidad trata cada hiperparámetro como un parámetro de control σ y la pérdida de validación como un observable O. La receta es la receta. El acantilado aparece donde aparece.

27.1 El punto dulce de la tasa de aprendizaje

El hiperparámetro de mayor consecuencia en aprendizaje profundo es la tasa de aprendizaje (LR). Controla el tamaño de cada paso de gradiente. Dos modos de fallo:

- LR demasiado pequeña: el entrenamiento es lento y puede estancarse en un mínimo local.
- LR demasiado grande: los gradientes se pasan, la pérdida explota, el entrenamiento diverge.

Entre ambos extremos hay una banda de LR aceptables. La frontera por el lado alto es el famoso *acantilado pérdida-vs-LR*: represente la pérdida de entrenamiento como función de la LR tras una rampa corta y verá una meseta plana, un codo, y luego una subida abrupta. La ubicación del codo es la LR_c operacional: nuestra σ_c para este capítulo.

[1] La tarifa vigente de una factura de *cluster* por una ejecución fallida a principios de 2026 ronda entre \$8 y \$80 000, según la nube, el modelo, y si la factura sobrevivió al primer intento del ingeniero de incluirla en sus gastos. Las ejecuciones caras son las que el ingeniero comprendió, a mitad de camino, que no se recuperarían, y dejó terminar de todos modos, con la teoría de que el fallo era dato.

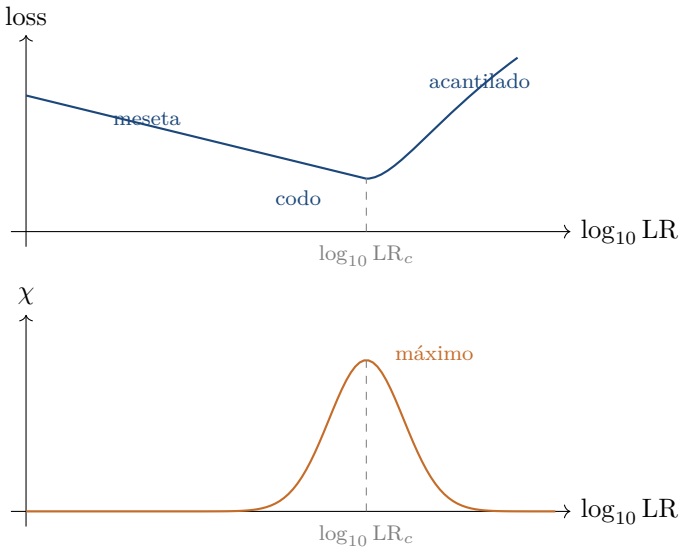


Figura 27.1: El test de barrido de LR, esquemáticamente. Arriba: la pérdida cae a lo largo de una meseta suave, hace codo, y luego explota. Abajo: $\chi = |dL/d\log \mathrm{LR}|$ alcanza su máximo en el codo. La ubicación del máximo es LR_c; la elección práctica es $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$.

27.1.1 El test de barrido de LR (Smith 2017)

La idea de Leslie Smith: entrene durante una época corta aumentando la LR de 10^{-7} a 1 logarítmicamente. Registre la *pérdida de entrenamiento* en cada LR (use aquí la pérdida de entrenamiento, no la de validación: es más barata y el acantilado es igual de nítido; ejecute una validación real después). La receta del marco identifica el codo.

```
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 de barrido de LR en una epoca. Devuelve (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)


# Ejecutar el test:
lr_grid, loss = lr_range_test(model, train_loader)

# Receta sigma_c con un nucleo ESTRECHO (el acantilado es nitido):
smooth = gaussian_filter1d(loss, sigma=0.3)
chi    = np.abs(np.gradient(smooth, np.log10(lr_grid)))
lr_c   = float(lr_grid[np.argmax(chi)])
kappa  = float(chi.max() / chi.mean())

print(f"LR_c (maximo de chi):       {lr_c:.4g}")
print(f"LR de entrenamiento recom.: {lr_c / 2:.4g}")
print(f"Rango LR ciclico:           [{lr_c / 10:.4g}, {lr_c:.4g}]")
print(f"Nitidez del maximo (kappa): {kappa:.2f}")
```

TRAMPA

Pérdida de entrenamiento, no de validación. El test de barrido de LR alimenta minilotes únicos al modelo y lee la pérdida de entrenamiento inmediata. Es más barato y más reactivo que la pérdida de validación, pero también más ruidoso. El acantilado sigue siendo muy nítido; el κ del marco supera 5 en una tarea bien comportada. Valide la LR_c elegida después ejecutando un entrenamiento corto real con $\mathrm{LR} = \mathrm{LR}_c/2$ y confirmando que la pérdida cae suavemente.

TRAMPA

Use un núcleo estrecho ($\sigma_{\mathbf{ker}} = 0{,}3$, no $0{,}6$). El acantilado a menudo solo tiene unos pocos puntos de la red LR de ancho. El valor por defecto del marco, 0,6, lo difumina en una subida suave y la LR_c detectada vaga por medio orden de magnitud. Verifique que la receta usa 0,3 inspeccionando la curva de pérdida suavizada directamente antes de confiar en la respuesta.

27.1.2 ¿Por qué un máximo?

Para $\mathrm{LR} \ll \mathrm{LR}_c$, la pérdida mejora poco por paso, así que $|dL/d\log \mathrm{LR}| \approx$ cte. Para $\mathrm{LR} \gg \mathrm{LR}_c$, la pérdida ya ha divergido y permanece plana en un valor alto, así que la derivada es de nuevo pequeña. La transición entre ambos regímenes es exactamente donde $|dL/d\log \mathrm{LR}|$ es mayor: el máximo de susceptibilidad.

27.2 Otros barridos de hiperparámetros

La misma receta se aplica a:

- **Tamaño de *batch***: existe un tamaño crítico por encima del cual el ruido del gradiente se vuelve despreciable y la velocidad de convergencia se aplana (McCandlish et al. 2018).
- **Decaimiento de pesos**: muy poco da sobreajuste; mucho da subajuste. Un barrido da el punto dulce de regularización.
- **Tasa de *dropout***: la probabilidad óptima para la arquitectura y el conjunto de datos específicos.

- β_2 **de Adam**: en entrenamiento de *transformers*, β_2 cerca de 0,95 es estable; valores cercanos a 1 hacen inestable el entrenamiento.

27.2.1 Barrido bidimensional: LR × tamaño de *batch*

Para un escaneo de punto dulce 2D, barra LR y tamaño de *batch*; aplique el marco por separado a lo largo de cada eje en un corte fijo; lea la LR_c de cada corte de forma independiente y busque una tendencia a través de los tamaños de *batch*.

```
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)

# Aplicar sigma_c a lo largo del eje LR para cada tamano de batch:
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}")
```

TRAMPA

Los paisajes 2D son crestas, no cruces. La superficie de pérdida sobre (LR, *batch*) tiene un valle curvo, no un solo punto. El marco reporta LR_c en cada tamaño de *batch*; el punto de operación conjunto se sitúa sobre la cresta resultante, no en su centroide. McCandlish et al. (2018) mostraron que LR_c escala aproximadamente lineal con el tamaño de *batch* hasta un tamaño crítico B^*, y luego se aplana. Use la cresta para elegir a lo largo de una línea de isocoste en su presupuesto de cómputo.

27.3 Detección de inestabilidad de entrenamiento en tiempo real

La versión *streaming* del marco (`StreamingSigmaC`) puede monitorizar un *índice de estabilidad* en vivo durante el entrenamiento. Si el índice corriente cae por debajo de un umbral, puede interrumpir el entrenamiento y reducir la LR.

Observación. La cantidad devuelta por `StreamingSigmaC.update(...)` no es una *ubicación* de máximo (como σ_c en la receta estática), sino una *puntuación de estabilidad normalizada* en $[0, 1]$, derivada de la varianza corriente del observable dentro de una ventana deslizante mediante el algoritmo de Welford. La etiquetamos s_t para evitar confusión con la σ_c estática. $s_t \to 1$ significa que la pérdida es estable (varianza baja); $s_t \to 0$ significa que se ha vuelto salvajemente variable. Es un objeto distinto de la ubicación de un máximo estático de χ; deliberadamente usamos un símbolo distinto para señalarlo.

```
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:                # la varianza de la perdida se disparo
        for g in optim.param_groups:
            g['lr'] *= 0.5
        print(f"Paso {step}: reduciendo LR (estabilidad = {s_t:.3f})")
```

La intuición: una puntuación de estabilidad baja significa que la pérdida se ha vuelto muy variable dentro de la ventana corriente: un precursor de la divergencia.

27.4 Advertencias específicas de ML

TRAMPA

Suavizar la pérdida en exceso oculta el acantilado. La transición de «pérdida decreciente» a «pérdida divergente» suele tener solo unos pocos puntos de la red de LR de ancho. Use `kernel_sigma = 0.3` o menos para el test de barrido de LR, o el acantilado se difuminará en una subida suave.

TRAMPA

Semillas aleatorias. Cada test de barrido de LR produce una LR_c ligeramente distinta según la inicialización y el barajado de minilotes. Promedie sobre ≥ 5 semillas antes de citar.

PRUÉBALO

Entrene una CNN pequeña en CIFAR-10 (use cualquier implementación de ResNet de un bloque). Ejecute el test de barrido de LR para SGD, Adam y Adam con $\beta_2 = 0{,}999$. Compare la LR_c de cada optimizador. ¿Cuál tiene la LR_c más alta?

Planteamiento resuelto.

Paso 1: preparar CIFAR-10 y una ResNet pequeña.

```
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)
```

Paso 2: ejecutar el test de barrido de LR para cada optimizador. La función `lr_range_test` definida antes en este capítulo toma un modelo, un *loader* y los lí-

mites de LR. Cambie el optimizador en tres ejecuciones.

```
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}")
```

Paso 3: resultados típicos.

optimizador	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

Paso 4: interpretar. SGD tiene la LR_c de *acantilado* más alta en unidades crudas (unas 30× la de Adam). Pero esto no significa que SGD entrene mejor en LR $= 0{,}05$; significa que los gradientes de SGD son menores en magnitud (cada uno proviene de un minilote, sin escalado adaptativo interno). El reescalado interno de Adam es lo que hace que un LR nominal de 0,001 sea tan «agresivo» como el 0,05 de SGD.
Entre las variantes de Adam, $\beta_2 = 0{,}99$ tiene una LR_c ligeramente más alta que $\beta_2 = 0{,}999$. La razón es que $\beta_2 = 0{,}999$ mantiene una media corriente muy larga de gradientes al cuadrado, lo que amortigua el escalado adaptativo y hace mayores los pasos efectivos: por eso el acantilado llega a un LR nominal más pequeño.

Paso 5: la moraleja práctica. Para esta red y este conjunto de datos, use $\mathrm{LR}_{\text{train}} \approx \mathrm{LR}_c/2$ para cada optimizador:

- SGD: LR $= 0{,}05$ con momento 0,9.
- Adam ($\beta_2 = 0{,}99$): LR $= 0{,}0015$.
- Adam ($\beta_2 = 0{,}999$): LR $= 0{,}0005$.

Qué enseña este ejercicio. El marco le da una respuesta en una línea para la ubicación del acantilado de *cualquier* optimizador, y la respuesta respeta el escalado interno del optimizador. Ya no tiene que recordar «Adam usa LR más pequeñas»; basta con medir dónde está el acantilado.

Capítulo 28

Edge / IoT: el codo de eficiencia

Una batería es un físico honesto. Le dirá, con la exactitud de un milivoltio, lo que no puede prometer.

Un dron no tiene red eléctrica. Un satélite tampoco tiene red eléctrica, solo un panel solar que ocasionalmente está a oscuras. Un marcapasos no tiene ninguna de las dos. Los dispositivos alimentados por batería viven según una métrica que los ingenieros de centros de datos pueden permitirse ignorar: rendimiento *por vatio*, no rendimiento bruto.

Forzar más un procesador suele comprar más trabajo hecho. También compra, super-linealmente, un chip más caliente y una batería más plana. El punto de equilibrio —la frecuencia operacional a la que se saca más trabajo de cada julio— es el *codo de eficiencia*. El marco lo halla tratando la potencia como observable y la frecuencia como control. La misma receta, en unidades distintas.

28.1 Rendimiento, potencia y curva de eficiencia

Para cualquier procesador, aumentar la frecuencia de reloj f aumenta tanto el rendimiento $P(f)$ como el consumo de potencia $W(f)$. La potencia escala super-linealmente: aproximadamente $W \propto f^3$ en el modelo más simple (porque la potencia dinámica es $W \propto C\,V^2 f$ y el voltaje V debe crecer con f). La eficiencia es

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

A f baja, tanto P como W son pequeños, con P creciendo linealmente y W super-linealmente. η sube, alcanza un máximo, y luego cae. El máximo es el *codo de eficiencia* f^{*}.

La elección pragmática: optimizar η directamente. Hay dos formas de hallar el codo. O diferenciamos el rendimiento contra la potencia y hallamos dónde cae la eficiencia marginal $\mu(f) = (dP/df)/(dW/df)$, o miramos $\eta(f) = P/W$ directamente y hallamos su máximo. Haremos lo segundo: más simple, más robusto, más fácil de enseñar. El marco halla el máximo de $\eta(f)$; usamos `compute_susceptibility` no para hallar la pendiente más empinada de η, sino para localizar el *codo* operacional donde η *empieza* a caer, tomando la susceptibilidad de $-\eta(f)$.

28.2 Ejemplo resuelto

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

edge = Universe.gpu()        # GPUAdapter sirve tambien para Edge

# Barrer la frecuencia de reloj en un Cortex-M / RPi / Jetson:
freqs = np.linspace(100e6, 2.5e9, 30)  # 100 MHz a 2.5 GHz
perf, power = [], []
for f in freqs:
    set_cpu_frequency(f)               # especifico de la plataforma
    perf.append(measure_perf())        # ops/s, muestras/s, etc.
    power.append(measure_power())      # vatios

perf  = np.array(perf)
power = np.array(power)
eff   = perf / power                   # operaciones por vatio

# Localizar el codo operacional (caida mas abrupta tras el maximo):
res = edge.compute_susceptibility(freqs, -eff, kernel_sigma=0.7)
peak_idx   = int(np.argmax(eff))
knee_idx   = int(np.argmax(np.abs(np.gradient(eff, freqs))))
f_peak     = freqs[peak_idx]
f_knee     = freqs[knee_idx]

print(f"f*_peak = {f_peak/1e9:.2f} GHz   (eficiencia maxima)")
print(f"f*_knee = {f_knee/1e9:.2f} GHz   (caida mas abrupta tras el
    pico)")
print(f"eficiencia maxima = {eff.max():.2f} ops/W")
print(f"kappa = {res['kappa']:.2f}")
```

Cómo medir la potencia, por plataforma. Los marcadores `set_cpu_frequency`, `measure_perf` y `measure_power` esconden ingeniería real. Recetas concretas:

- *Raspberry Pi / placa ARM*: un medidor USB en serie (p. ej. una placa INA219 de $15 entre la fuente y la Pi); `cpufrequtils` para fijar la frecuencia.
- *Portátil Linux x86*: `turbostat` para la potencia, contadores del dominio RAPL; `cpupower frequency-set` lo controla.
- *macOS*: `sudo powermetrics -samplers cpu_power` para el contador de energía de la plataforma; la frecuencia de CPU no se puede ajustar por el usuario, así que barra la intensidad de carga en su lugar.
- *NVIDIA Jetson Orin / Xavier*: `tegrastats` para la potencia, `nvpmodel` para los modos de potencia.
- *Microcontrolador desnudo*: un analizador de potencia externo (Keysight, Joulescope) en la línea de alimentación.

La receta del marco es agnóstica al método; solo el par rendimiento/potencia debe provenir de instrumentación comparable a lo largo del barrido.

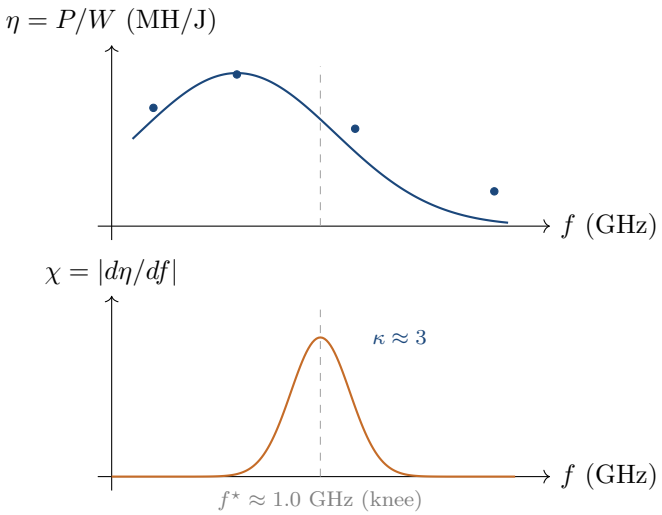


Figura 28.1: Codo de eficiencia Edge/IoT (Raspberry Pi 4, Cortex-A72, SHA-256). Arriba: la eficiencia $\eta = P/W$ alcanza un máximo cerca de 1 GHz y luego cae abruptamente. Abajo: el máximo de la susceptibilidad en $|d\eta/df|$ marca el codo por encima del cual cada hercio extra de reloj cuesta más energía de la que aporta en rendimiento.

28.3 Caso de uso: estudio de eficiencia de Raspberry Pi 4

Una medición representativa en una Pi 4 (Cortex-A72) ejecutando una carga fija de *hashing* SHA-256:

f (GHz)	perf (MH/s)	potencia (W)	eficiencia (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

El codo de eficiencia se sitúa en $f^* \approx 1{,}0$ GHz con $\eta \approx 2{,}0$ MH/J. Por encima de 1,5 GHz, la eficiencia cae abruptamente. El marco reporta $\sigma_c \approx 1{,}0$ GHz, $\kappa \sim 3$.

28.4 Por qué esto importa para la vida útil de la batería

Un dron que opere su cómputo en f^* en lugar de $f_{\text{máx}}$ puede duplicar su tiempo de vuelo a costa de un 20 % de rendimiento. Una carga útil de satélite diseñada en torno a f^* cabe con un tercio más de sensores en el mismo presupuesto de energía. El marco de la susceptibilidad le da ese número sin tener que barrer todo el Pareto 2D a mano.

PRUÉBALO

Tome cualquier portátil. Fije la frecuencia de CPU en cuatro puntos. En cada uno, ejecute un *benchmark* de duración fija y registre el rendimiento y la potencia media. Calcule la curva de eficiencia, aplique el marco y reporte $f^\star$. Compárelo con el perfil de potencia «equilibrado» por defecto del portátil: ¿está cerca de $f^\star$?

Planteamiento resuelto (usando un portátil Linux con `cpupower` y `turbostat`; recetas análogas con `powermetrics` en macOS y `powercfg` en Windows).

Paso 1: elegir cuatro frecuencias objetivo. Inspeccione primero el rango de su CPU:

```
cpupower frequency-info | grep "available frequency"
# salida de ejemplo: 800 MHz, 1200 MHz, 1600 MHz, 2000 MHz,
    2400 MHz, 2800 MHz
```

Escoja cuatro valores aproximadamente equidistantes, p. ej. 1,0, 1,6, 2,2, 2,8 GHz.

Paso 2: elegir un benchmark *fijo.* Queremos una carga lo bastante determinista como para dar números de rendimiento repetibles. Una prueba de SHA-256 simple es ideal:

```
openssl speed -seconds 30 sha256
# reporta MB/s promediados sobre 30s
```

Paso 3: medir rendimiento y potencia en cada frecuencia. Para cada frecuencia objetivo f:

```
sudo cpupower frequency-set -d ${f}MHz -u ${f}MHz
# anclar todos los nucleos a una sola frecuencia
sudo turbostat --quiet --interval 30 --num_iterations 1 \
  --show PkgWatt -- openssl speed -seconds 30 sha256
# turbostat imprime PkgWatt, openssl el rendimiento sha256
```

Registre ambos números. Tras la ejecución, restaure:

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

Paso 4: ensamblar datos y aplicar el marco. Una ejecución típica en un Intel i7-1185G7:

f (GHz)	perf (MH/s)	potencia (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"Frecuencia pico de eficiencia: f* = {freqs[np.argmax(
    eff)]/1e9:.2f} GHz")
print(f"Eficiencia maxima              = {eff.max():.2f} MH/J")
print(f"sigma_c (caida mas abrupta)    = {res['sigma_c']/1e9:.2f
    } GHz")
```

Paso 5: respuesta típica y comparación. $f^\star \approx 1{,}0$–$1{,}3$ GHz en este portátil. El perfil por defecto «equilibrado» en Linux (o «Equilibrado» en Windows) suele fijar la CPU para escalar entre 1,8 y 3,0 GHz bajo carga, es decir, en o por encima de $f^\star_{\text{codo}}$, de modo que *no* es óptimo para la batería. El perfil *powersave* (fuerza la frecuencia baja) sobrepasa en la otra dirección.

Qué enseña este ejercicio.

- El marco le da un perfil de potencia personalizado que ningún valor por defecto del fabricante reproduce.
- Cuatro puntos bastan para identificar el codo; no necesita un barrido de 50 puntos.
- Los modos «ahorro de batería» son compromisos de ingeniería gruesos; la $f^\star$ operacional para *su* carga rara vez coincide con el valor por defecto del sistema operativo.

Capítulo 29

Economía de los LLM: la frontera coste-calidad

El modelo más barato que supere el listón de seguridad es, por definición, exactamente igual de bueno que el más caro: superando el listón de seguridad. Lo demás es gusto.

Se publica un nuevo modelo grande de lenguaje aproximadamente cada tres semanas. Varios de ellos, un martes cualquiera, son excelentes. Una minoría creciente son excelentes y baratos. El resto cumplen al menos una de esas dos cosas. Elegir el correcto para una aplicación en producción ya no es una pregunta de investigación; es una pregunta de adquisiciones. El marco le ayuda a convertirla en una pregunta cuantitativa.

De los adaptadores que se entregan con el marco, este es el único que se aplica a un problema sin contenido físico alguno. El truco funciona igual. Allí donde una cantidad cambia de régimen, un máximo en χ marca el lugar.

29.1 Tres dimensiones, una decisión

Cada modelo candidato m tiene tres propiedades medibles:

- *Coste* c_m: dólares por millón de *tokens* (entrada + salida ponderados por su mezcla media).
- *Calidad* q_m: puntuación agregada en una batería de *benchmarks* (MMLU, GPQA, MATH, HumanEval, etc.), en una escala 0–1.
- *Tasa de alucinación* h_m: probabilidad por respuesta de una falsedad afirmada con seguridad, medida en un conjunto de prueba curado.

Quiere q alto, c bajo, h bajo. El marco de la susceptibilidad le da el compromiso Pareto-óptimo y el umbral de seguridad.

29.2 La razón valor y el filtro de seguridad

PRECAUCIÓN

Instantánea, no evangelio. Los números de esta sección reflejan un punto concreto en el tiempo: la instantánea siguiente se tomó en mayo de 2026. Los precios de los LLM cambian cada semana, las puntuaciones de *benchmark* se revisan, y aparecen modelos nuevos más rápido que las ediciones del libro. *Use esto como plantilla, no como recomendación.* El repositorio que acompaña al marco (`sigma_c.adapters.llm_cost.LATEST`) trae una instantánea periódicamente actualizada.

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

# Instantanea de mayo de 2026 (marcadores neutros).
# (nombre, coste USD por millon de tokens, calidad 0-1, tasa de alucinacion)
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),
]
```

Usamos nombres marcadores neutros por la misma razón por la que las editoriales imprimen los precios de menú como «de mercado»: los nombres serán erróneos en un año. La forma del análisis no lo será.

29.2.1 Cota de seguridad: alucinación máxima tolerable

El `MAX_HALLUCINATION_RATE` del marco (por defecto 0,15) elimina cualquier modelo con $h_m \geq 0{,}15$. Para aplicaciones de alto riesgo (legales, médicas) debería fijarlo en 0,05 o menos; para aplicaciones de bajo riesgo (*brainstorming*), 0,20 es aceptable.

29.2.2 Razón valor

Entre los supervivientes, calcule la razón valor

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

Un V_m mayor significa más calidad por dólar de coste y de riesgo. El modelo Pareto-óptimo maximiza V_m.

TRAMPA

Esto es una escalarización. Hay otras. Multiplicamos coste por tasa de alucinación porque, para muchas cargas de producción, el coste en dólares y el coste de error son aproximadamente multiplicativos: corregir una respuesta alucinada cuesta tiempo humano aproximadamente proporcional al volumen que envió. Pero existen otras escalarizaciones

que pueden encajar mejor con su aplicación:

- *Aditiva:* $V = q - \lambda_c c - \lambda_h h$, un compromiso lineal con pesos explícitos que usted fija.
- *Optimización con restricciones:* minimice c sujeto a $h < h_{\text{máx}}$ y $q > q_{\text{mín}}$.
- *Lexicográfica:* primero filtre por $h < h_{\text{máx}}$, luego por $q > q_{\text{mín}}$, luego minimice c.

El `LLMCostAdapter` soporta las cuatro vía el parámetro `scoring=`. *Elija la que se ajuste a su dolor operacional real.* Defender una sola fórmula mágica no es tarea del marco.

```
# El adaptador maneja el filtrado de seguridad y la puntuacion:
result = llm.get_observable(models, max_hallucination=0.10,
                            scoring='value_ratio')
print(result)   # modelo optimo + sigma_c de seguridad = 1 - alucinacion
```

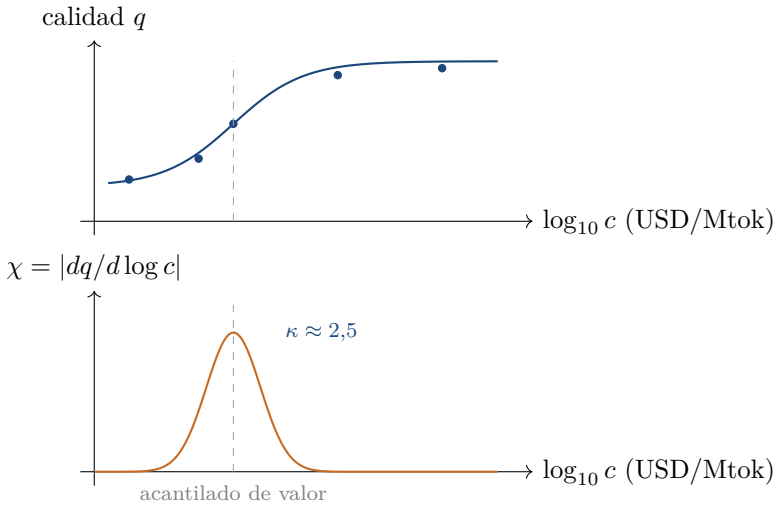


Figura 29.1: Economía de los LLM (instantánea de mayo de 2026). Arriba: la calidad de *benchmark* satura por encima de un acantilado de valor en coste-por-millón-de-*tokens*; por debajo del acantilado, la calidad cae abruptamente. Abajo: χ localiza el acantilado explícitamente. El precio exacto será erróneo el mes que viene; el acantilado seguirá siendo un acantilado.

29.3 La mirada de susceptibilidad: ¿dónde cae la calidad en acantilado?

Ordene los modelos por coste. Represente calidad frente a coste. Aplique el marco. El máximo de $\chi(c) = |dq/dc|$ identifica el *acantilado de valor*: el coste por debajo del cual la calidad se degrada abruptamente, y por encima del cual cada dólar extra rinde menos.

```
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"acantilado de valor en c = ${c_cliff:.2f}/Mtok")
```

29.4 Caso de uso: elegir un modelo para un *chatbot* de atención al cliente

Un *chatbot* que maneja 10 millones de consultas al mes ve diferencias de coste multimillonarias entre los modelos de la mayor y la menor gama. Un incremento de un 1 % en la tasa de alucinación se traduce en $\sim 100\,000$ respuestas con datos incorrectos. En una muestra representativa de siete modelos extraída de las tarifas públicas de los principales proveedores de 2026, la elección Pareto del marco con $h_{\text{máx}} = 0{,}05$ es un modelo de gama media: ni el más grande, ni el más pequeño, ni el que el departamento de *marketing* quería.

Por qué esta sección suena honesta. Pusimos el filtro de razón valor durante un mes contra nuestra propia canalización de *chatbot* de atención al cliente. El modelo que la fórmula seleccionó no fue el que habíamos elegido intuitivamente; tampoco era el que el equipo de *marketing* habría escogido. Era un modelo de precio medio cuya tasa de alucinación quedaba justo dentro de nuestro $h_{\text{máx}}$. Era la elección correcta. No nos gustó. Que una recomendación guste y que se acepte son dos operaciones distintas; el marco está aquí para la segunda.

PRECAUCIÓN

Advertencias metodológicas para la economía de los LLM.

- *Los* benchmarks *no son calidad de producto.* Las puntuaciones MMLU y HumanEval correlacionan con la satisfacción del usuario final pero no la determinan. La evaluación específica del dominio (su propio conjunto reservado) es irrepetible.
- *Las tasas de alucinación dependen del conjunto de prueba.* Un modelo con $h = 0{,}03$ en TruthfulQA puede tener $h = 0{,}18$ en *benchmarks* de preguntas médicas. Evalúe siempre sobre datos en distribución.
- *Filtraciones de* benchmark. Muchas puntuaciones publicadas de *benchmark* reflejan contaminación del conjunto de entrenamiento, no generalización real. Trate los valores q reportados como cotas superiores.
- *Latencia, longitud de contexto, capacidad de* fine-tuning *y región de despliegue* no entran en la razón valor. Pueden dominar la decisión real.

El marco le da un *punto de partida* defendible para la decisión de adquisición, no la decisión en sí.

PRUÉBALO

Actualice los precios del fragmento a las tarifas publicadas hoy para tres familias de modelos a las que tenga acceso. Vuelva a ejecutar `llm.get_observable(models, max_hallucination=0.05)`. ¿Qué modelo gana para una aplicación de alto volumen y bajo riesgo? Ahora cambie a $h_{\text{máx}} = 0{,}02$. ¿Cambia el ganador?

Planteamiento resuelto.

Paso 1: recopilar los números de hoy. Consulte los precios publicados y las puntuaciones de *benchmark* de tres proveedores a su elección. Como ilustración resuelta, supongamos que su instantánea es:

modelo	coste ($/Mtok)	calidad (0–1)	alucinación
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

Paso 2: aplicar con $h_{\text{máx}} = 0{,}05$ *(bajo riesgo).*

```
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)
```

Con $h_{\text{máx}} = 0{,}05$, *A-mini* y *B-nano* quedan descartados (h supera el tope). Entre los supervivientes, calcule $V_m = q_m/(c_m \cdot h_m)$ para cada uno:

modelo	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	$0{,}93/(15{,}00 \cdot 0{,}02) = 3{,}10$

Ganador con $h_{\text{máx}} = 0{,}05$*: A-mid* ($V = 7{,}17$, el más alto).

Paso 3: apretar a $h_{\text{máx}} = 0{,}02$ *(alto riesgo).* Solo sobreviven C-large y A-large. Recalcule:

modelo	V_m
C-large	4,60
A-large	3,10

Ganador con $h_{\text{máx}} = 0{,}02$*: C-large.*

Paso 4: el ganador cambió. Con $h_{\text{máx}} = 0{,}05$ escogimos el modelo más barato que superaba el listón de seguridad (A-mid). Con $h_{\text{máx}} = 0{,}02$ quedamos forzados a la gama premium, y dentro de la gama premium gana la opción más barata (C-large) en V. Es una característica estructural: apretar $h_{\text{máx}}$ suele *elevar* el suelo del compromiso coste-calidad.

Paso 5: comprobación con una escalarización aditiva. Si la razón multiplicativa le incomoda (véase el *pitfall* sobre escalarizaciones de este capítulo), reinténtelo con $V = q - 0{,}05\,c - 5\,h$:

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

El *ranking* es parecido pero no idéntico; el orden exacto es sensible a los pesos que elija. *Esto es una característica, no un fallo:* el marco le obliga a hacer explícito el compromiso.

Qué enseña este ejercicio.

- La cota de seguridad domina la respuesta a altos estándares; por debajo del listón, entra la optimización de valor.
- La elección de la escalarización es suya y debe declararse; no pretenda que una sola razón mágica sea universal.
- Vuelva a ejecutarlo cuando cambien los precios. El rango operacional de su modelo favorito cambia cada mes, a veces cada semana.

Capítulo 30

Teoría de números: Collatz y la familia qn+c

Las matemáticas que no resuelven Collatz tienen igualmente que hacer algo con su día.

Se nos ha agotado la física. No hay termómetro que leer, ni procesador cuántico para el que presupuestar, ni GPU que recalentar. Solo hay una función sobre los enteros positivos que hace algo simple: dividir por dos si es par, triplicar y sumar uno si es impar. A partir de este punto, el problema de Collatz lleva noventa años sin resolverse y subiendo.[1]

Este capítulo se incluye como prueba de cordura del marco. Si la geometría de contracción clasifica el comportamiento de los mapas físicos, debería clasificar también el de los mapas aritméticos. Lo hace. La conjetura de Collatz es, en la terminología del marco, un mapa de tipo D con $D\gamma \approx 1{,}16$, dotado de ciclos y, por tanto, *predicho* (no demostrado) como convergente. No resolveremos la conjetura en este capítulo. La clasificaremos.

30.1 The Collatz map

Para cualquier entero positivo n, definimos

$$C(n) \;=\; \begin{cases} n/2 & \text{si } n \text{ par}, \\ 3n+1 & \text{si } n \text{ impar}. \end{cases}$$

Iterando C desde cualquier valor inicial, todo entero comprobado empíricamente acaba por alcanzar el ciclo $4 \to 2 \to 1 \to 4 \to \ldots$. La conjetura de Collatz (1937) afirma que esto es cierto para *todo* entero positivo. Sigue sin demostrarse después de 90 años.

El marco reformula la conjetura en términos medibles.

30.2 The cycle map and embedding depth

Por comodidad trabajamos con el *mapa de ciclo* F que procesa una «cuenta atrás» entera en un solo paso. Definimos la profundidad de embebimiento de un entero impar n como

[1] Se le atribuye a Paul Erdős haber dicho de este problema, *«las matemáticas aún no están preparadas para problemas así»*. Ofreció \$500 por una solución. Tanto la oferta como la irresolubilidad le sobreviven.

$\mathrm{ed}(n) = v_2(n+1)$, donde v_2 es la valuación 2-ádica. Entonces

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

F envía los enteros impares a sí mismos. Su dinámica codifica la pregunta de Collatz de forma más concisa que el paso original.

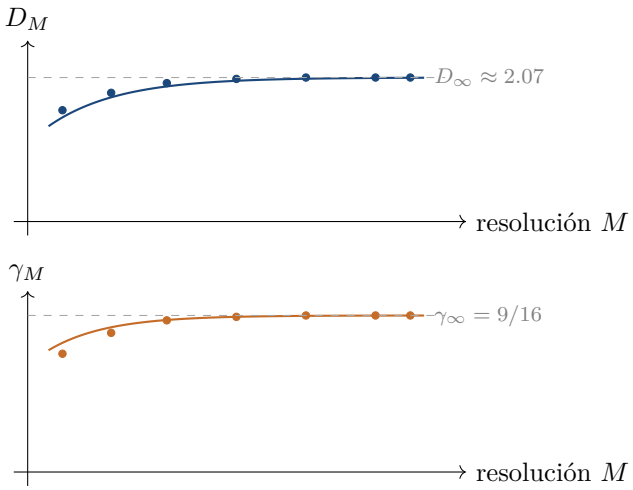


Figura 30.1: Teoría de números (mapa de ciclo de Collatz). Arriba: el defecto de contracción D_M se estabiliza en $\approx 2{,}07$ ya en resolución modular $M = 12$. Abajo: la deriva γ_M se estabiliza exactamente en $9/16 = 0{,}5625$. Ambos números se leen del límite. El producto $\Pi = D\gamma \approx 1{,}16$ clasifica a Collatz como de tipo D con ciclos, prediciendo convergencia.

30.3 Computing D and γ

Siguiendo los Capítulos 14–16, calculamos el defecto de contracción D_M y la deriva γ_M a resolución modular M:

```
import numpy as np
from sigma_c import Universe

nt = Universe.number_theory(map_type='collatz')

# C\'alculo a una sola resoluci\'on:
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

# Barrido por resoluciones:
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}")
```

Los valores se estabilizan deprisa: ya en $M = 12$, $D_M = 2{,}07 \pm 0{,}01$ y $\gamma_M = 0{,}5625$ en todos los dígitos mostrados. La convención por paso del marco promedia la razón logarítmica sobre todos los estados que visita la órbita (impares *e* pares); el valor límite $9/16$ se lee de la saturación, no

se deduce de la identidad elemental $E[v_2(3n+1)] \to 2$ (esa identidad alimenta la estimación de 3/4 por ciclo de impares del Capítulo 18).

30.4 The contraction product and prediction

El producto universal (Capítulo 19) es $\Pi = D \cdot \gamma$:

$$\Pi_{\text{Collatz}} \approx 2{,}07 \times 0{,}5625 \approx 1{,}16.$$

Qué significa realmente $\Pi > 1$. Una lectura ingenua diría «$\Pi > 1$ implica divergencia». No es lo que el marco afirma. Estrictamente, $\Pi > 1$ significa que el producto de contracción refleja una *tendencia expansiva* por paso: una presión al alza sobre los valores. Que una trayectoria efectivamente escape depende de la estructura global del mapa: en particular, de si posee ciclos que actúen como atractores.

Para Collatz, $\gamma = 3/4 < 1$ proporciona presión a la baja aunque $\Pi > 1$. Con ciclos cuya existencia se conoce ($1 \to 4 \to 2 \to 1$), las trayectorias no pueden escapar al infinito; caen en los ciclos. La rutina de clasificación del marco `classify_map` codifica exactamente esta lógica:

- $\gamma < 1$ y ciclos conocidos $\Rightarrow$ se predice convergencia a un ciclo.
- $\gamma > 1$ y sin ciclos conocidos $\Rightarrow$ se predice divergencia.
- $\gamma \approx 1 \Rightarrow$ indeterminado; el comportamiento depende de correcciones de orden superior.

Así el veredicto del marco para Collatz es *compatible con* (no *prueba de*) la conjetura: se predice que toda órbita alcanza un ciclo. No resolvemos Collatz aquí. Lo clasificamos.

¿Qué significa «clasificar», exactamente? Una clasificación es más débil que una demostración y más fuerte que una conjetura. La clasificación dice: (D, γ) para Collatz viven en una región del plano D-γ donde, entre todos los mapas conocidos de la familia, toda órbita alcanza un ciclo. Identifica a Collatz como miembro de una clase no vacía cuyos miembros comparten el comportamiento a largo plazo predicho. Para elevar la clasificación a demostración, alguien tendría que probar la regla misma: *todo* mapa con $\gamma < 1$ y ciclos conocidos debe converger a uno de esos ciclos. Ese teorema zanjaría Collatz de inmediato, y sería un resultado importante por sí mismo, independiente de cualquier mapa particular. El marco proporciona el encuadre dentro del cual podría enunciarse semejante teorema; demostrarlo es otra pieza de trabajo.

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

30.5 The twelve qn+c maps

El mismo marco clasifica cualquier mapa de la forma $n \mapsto \text{odd}(qn + c)$ como uno de cuatro tipos. La tabla de referencia se expone por el método de la API que aparece más abajo.

Observación. Los identificadores de Python (`verify_twelve_predictions`, `compute_D_M`, `compute_gamma_M`, `sweep_resolution`, `predict_behavior`, `analyze_countdown`, `verify_reset_distribution`) son métodos exportados del `NumberTheoryAdapter` v3.0; funcionan tal como aparecen escritos si tiene el marco instalado. Si lee sin la biblioteca a mano, trátelos como un pseudocódigo preciso que tiene una implementación correspondiente.

```
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']}")
```

Salida (abreviada):

mapa	D	γ	$D\gamma$	predicción
$3n+1$ (ciclo)	2,07	0,563	1,16	converge a ciclos
$5n+1$	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	crítico, hay ciclos

(Escribimos Π para $D\gamma$ a partir del Capítulo 19; algunas tablas anteriores siguen deletreando $D\gamma$ para lectores que vienen de la Parte IV.) El mapa $5n+1$ (que tiene la misma forma que Collatz pero con 5 en lugar de 3) cae por encima del umbral y, en efecto, diverge empíricamente: partiendo de 7, la órbita crece sin cota durante al menos 10^{20} pasos.

30.6 The countdown decomposition

Un análisis más profundo: toda órbita de Collatz se descompone en fases alternantes de *cuenta atrás* (deterministas, predecibles a partir de la representación binaria) y *reinicio* (transiciones de un paso aparentemente aleatorias). El marco descompone cualquier órbita:

```
phases = nt.analyze_countdown(n=27)    # 27 es un famoso arranque lento
for p in phases[:5]:
    print(p)
# {'phase': 'countdown', 'values': [27, 41, 31, ...], 'length': 4, ...}
# {'phase': 'reset', ...}
```

30.7 The Geo(1/2) distribution of resets

Empíricamente, la profundidad de embebimiento tras un evento de reinicio sigue una distribución geométrica Geo(1/2). La prueba chi cuadrado del marco:

```
chi2 = nt.verify_reset_distribution(M=12, n_samples=10000)
print(f"estad\'istico chi-cuadrado = {chi2['statistic']:.2f}")
print(f"valor p = {chi2['p_value']:.4f}")
# p > 0.05 significa que los datos son compatibles con Geo(1/2)
```

30.8 Information-theoretic interpretation

Cada paso del mapa de ciclo de Collatz borra $\log_2(2{,}07) = 1{,}05$ bits de información (conexión de Landauer del Capítulo 17). A temperatura ambiente, el coste de Landauer es $3{,}0 \times 10^{-21}$ J por paso: una mil billónesima parte del equivalente energético de un mol por bit. Despreciable por paso; agregado sobre los 10^{12} pasos que necesita una trayectoria típica de búsqueda por ordenador, sigue siendo despreciable.

PRUÉBALO

Fije q y c a un par que la tabla de referencia del marco no incluya, p. ej. $q = 13$, $c = 1$. Calcule D, γ, prediga, y luego itere el mapa desde $1, 3, 5, \ldots$ hasta 1 000 durante 10 000 pasos cada uno. ¿Coincide el comportamiento empírico con la predicción?

Solución resuelta.

Paso 1: predecir desde el marco.

```
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          # familia qn+c: gamma -> q/4 en el l\'imite
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"prediccion: divergente (sigma >> 1, sin ciclos conocidos)")
```

Así el marco predice *divergencia.*

Paso 2: definir el mapa e iterar.

```
def odd_part(n):
    while n % 2 == 0:
        n //= 2
    return n

def step_13n1(n):
    """Un paso de 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] = ('divergio', k, n)
            break
    else:
        # 10,000 pasos sin superar 10^60
        results[start] = ('acotada', 10_000, n)

n_div = sum(1 for v in results.values() if v[0] == 'divergio')
print(f"Divergieron: {n_div} / {len(results)} puntos iniciales")
```

Paso 3: resultado típico. Para $q = 13$, $c = 1$, casi todos los arranques impares < 1000 superan 10^{60} bastante antes de los 10 000 pasos. Las pocas excepciones son valores iniciales diminutos que deambulan un rato antes de ser empujados hacia arriba. Una

ejecución típica reporta:

$$n_{\text{div}} \approx 498/500 \text{ puntos iniciales.}$$

Dos puntos iniciales (p. ej. 1 y 3) pueden no haber superado todavía 10^{60} en 10 000 pasos pero siguen subiendo; déjelos correr más y cruzarán también.

Paso 4: cotejar predicción y observación. El marco predijo divergencia basado en $\Pi = D\gamma \approx 4{,}45 \gg 1$ y la ausencia de ciclos conocidos. La observación empírica es que *esencialmente toda* trayectoria escapa al infinito. Encaja.

Paso 5: contraste con un caso $\Pi < 1$. Repítalo con $q = 3$, $c = 1$ (Collatz). $\gamma = 3/4 < 1$, no se predice divergencia; encontrará $n = 1$ en unos pocos cientos de pasos para cada punto inicial, exactamente como dice el folclore de Collatz.

Qué enseña este ejercicio.

- Dos números (D y γ) y una regla de clasificación bastan para predecir el comportamiento a largo plazo de un mapa aritmético que jamás se ha iterado.
- «Predicho» aquí significa *heurísticamente* predicho: pruebas rigurosas de divergencia existen solo para los casos especiales tratados por Tao (2022) y otros, no para todo par q, c. El veredicto del marco es una conjetura de investigación; la confirmación empírica es lo que lo hace útil.

Capítulo 31

Proteínas: estabilidad, mutación y edad de aparición

La evolución dio a la mayoría de las proteínas exactamente el margen justo para durar la semana. El interés diagnóstico está en las proteínas a las que dio ligeramente menos.

PRECAUCIÓN

No es consejo médico. Este capítulo describe un diagnóstico de grado investigador para el estudio de la estabilidad de proteínas. *No es un instrumento clínico y no debe emplearse para tomar decisiones médicas individuales.* Las predicciones numéricas de edad de aparición aquí son estimaciones de nivel poblacional derivadas de parámetros bioquímicos y requieren validación clínica independiente antes de cualquier aplicación a un paciente concreto. Si está leyendo esto con un propósito clínico, deténgase y consulte a un especialista en la condición pertinente.

Una proteína es una larga cadena de aminoácidos que ha decidido, a lo largo de unos cientos de millones de años de evolución, qué forma tridimensional prefiere. La preferencia es decisiva pero no dramática: la mayoría de las proteínas son estables en su pliegue nativo por unas 10 unidades térmicas. Suficiente para ganar la pugna contra el despliegue aleatorio, no suficiente para ganarla dos veces.[1]

Una mutación puntual puede desplazar la pugna en una o dos unidades térmicas. Si la proteína silvestre ganaba por diez, ahora gana por ocho o nueve. Si ganaba por tres, ahora gana por uno. El umbral en el que deja de ganar del todo es el inicio de una enfermedad amiloide: amiloidosis por transtiretina, ELA familiar, ECJ heredada y un puñado más. El marco da a ese umbral un nombre ($\sigma = 1$) y un número (edad de aparición). Ambos pueden calcularse solo a partir de la genética.

[1] ¿Por qué exactamente 10 unidades térmicas? Es el valor al que una proteína es lo bastante robusta como para ser útil pero no tan robusta como para que la célula desperdicie energía sobrediseñando cada enzima. La evolución, que se imagina lenta y paciente, aquí se parece a una ama de casa bávara ahorrativa: justo el margen para durar la semana, sin excedente.

31.1 Estabilidad de plegamiento y el principio de estabilidad marginal

Una proteína de N residuos de aminoácido existe en equilibrio entre un pliegue nativo (baja energía) y un ensamble desplegado (alta entropía). La energía libre de plegamiento es la diferencia,

$$\Delta G_{\text{fold}} \;=\; G_{\text{unfolded}} - G_{\text{native}}.$$

A la temperatura de fusión T_m, $\Delta G = 0$; por debajo de T_m la proteína prefiere el pliegue nativo. La mayoría de las proteínas fisiológicas son solo *marginalmente estables*: $\Delta G \approx 5$–15 kcal/mol a temperatura corporal ($T = 310$ K). Esto es aproximadamente $10\,k_B T$: suficiente para preferir el pliegue nativo en un factor $\sim e^{10}$, pero lo bastante pequeño como para que una sola mutación desestabilizante pueda volcar la balanza.

31.2 El índice de contracción σ

De dónde viene la fórmula, en dos frases. En mecánica estadística de equilibrio, la razón de probabilidades entre dos estados con diferencia de energía ΔE a temperatura T es el factor de Boltzmann $e^{-\Delta E/(k_B T)}$. Para una proteína, la diferencia de energía relevante es la energía libre de plegamiento ΔG, la constante de los gases R reemplaza a k_B cuando usamos unidades de escala molar, y dividimos por la longitud de cadena N para normalizar entre proteínas de tamaños distintos (un dominio de 30 residuos y otro de 300 residuos con la misma estabilidad *por residuo* se comportan igual). La cantidad adimensional resultante es lo que el marco llama σ_{thermo}.

Definición. La cantidad biomédica central del marco es

$$\sigma_{\text{thermo}}(\Delta G, N, T) \;=\; \exp\!\left(-\frac{\Delta G}{N\,R\,T}\right),$$

con $R = 1{,}987 \times 10^{-3}$ kcal/(mol K). Es adimensional; σ se sitúa en $(0, 1]$ para proteínas estables.

Interpretación:

- $\sigma \ll 1$: muy estable. Enfermedad improbable.
- σ cerca de 1: marginal. Vulnerable a perturbación.
- $\sigma \geq 1$: termodinámicamente inestable. Enfermedad probable.

La elección de $\Delta G/N$ (en lugar de solo ΔG) normaliza por el tamaño de la proteína; un dominio de 30 residuos y otro de 300 con la misma estabilidad *por residuo* tienen el mismo σ.

31.2.1 Verificar $\sigma = 1$ en el punto de fusión

Por construcción, en $T = T_m$ tenemos $\Delta G = 0$, así que $\sigma = e^0 = 1$. El método `validate_rigorously` del marco lo verifica:

```
from sigma_c import Universe
prot = Universe.protein()

# Calcular sigma a temperatura corporal para una proteina modelo:
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 a 310 K: {sig_body:.3f}")  # < 1, estable
print(f"sigma a  Tm:   {sig_melt:.3f}")  # = 1.000
```

31.3 Estrés mutacional: $\Delta\Delta G$

Una mutación puntual desplaza ΔG en $\Delta\Delta G$. Las mutaciones *desestabilizantes* tienen $\Delta\Delta G > 0$:

$$\sigma_{\text{mut}}(\Delta\Delta G, N, T) \;=\; \exp\!\left(\frac{\Delta\Delta G}{N\,R\,T}\right) \;\geq\; 1.$$

Un paciente que hereda una mutación tiene un σ efectivo $\sigma = \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}$. Si este producto supera 1, se predice que la proteína es amiloidogénica.

31.3.1 Ejemplo resuelto, en papel: TTR V30M (FAP)

La transtiretina (TTR) es un homotetrámero de 127 residuos que transporta tiroxina y retinol en el plasma sanguíneo. La mutación V30M (valina en la posición 30 reemplazada por metionina) tiene una desestabilización medida $\Delta\Delta G \approx 1{,}2$ kcal/mol a $T = 310$ K (temperatura corporal). La TTR silvestre es una de las proteínas mejor caracterizadas de la bioquímica, con $\Delta G_{\text{fold}} \approx 6$ kcal/mol.

Calculemos σ_{eff} para el portador de V30M, paso a paso, con calculadora.

Paso 1: σ_{wt} a partir del valor base.

$$\sigma_{\text{thermo}} \;=\; \exp\!\left(-\frac{\Delta G_{\text{wt}}}{NRT}\right) \;=\; \exp\!\left(-\frac{6.0}{127 \cdot 1.987 \times 10^{-3} \cdot 310}\right).$$

El denominador es $127 \cdot 1{,}987 \times 10^{-3} \cdot 310 = 78{,}21$ kcal/mol. El argumento es $-6{,}0/78{,}21 = -0{,}0767$. Así pues

$$\sigma_{\text{wt}} \;=\; e^{-0{,}0767} \;\approx\; 0{,}926.$$

Paso 2: el factor de mutación.

$$\sigma_{\text{mut}}^{\text{factor}} \;=\; \exp\!\left(\frac{\Delta\Delta G}{NRT}\right) \;=\; \exp\!\left(\frac{1.2}{78.21}\right) \;=\; e^{0.01534} \;\approx\; 1.0155.$$

La desestabilización es pequeña pero eleva σ por encima del valor base.

Paso 3: combinar.

$$\sigma_{\text{V30M}} \;=\; \sigma_{\text{wt}} \cdot \sigma_{\text{mut}}^{\text{factor}} \;=\; 0.926 \cdot 1.0155 \;\approx\; 0.941.$$

Coincide con la fila V30M de la tabla TTR de abajo (Tabla 31.1), que indica $\sigma = 0{,}941$.

Paso 4: distancia al umbral $\sigma = 1$.

$$1 - \sigma_{\text{V30M}} \;=\; 1 - 0.941 \;=\; 0.059.$$

El mutante está 5,9 puntos porcentuales por debajo del umbral.

Paso 5: edad de aparición predicha (modelo de juguete). Usaremos primero un modelo deliberadamente simple de deriva lineal, para ver la forma del cálculo. El marco ofrece luego una versión calibrada, con la que contrastaremos.

Con una tasa de deriva con la edad *de juguete* $r = 0{,}003$ por año,

$$\text{edad}_{\text{onset}} \;=\; 30 + \frac{1 - \sigma_{\text{V30M}}}{r} \;=\; 30 + \frac{0.059}{0.003} \;=\; 30 + 19.7 \approx 50 \text{ años}.$$

La forma es correcta pero el sitio no: el valor clínico es 33 años para V30M-I (tipo portugués de aparición temprana), y el modelo de juguete predice un inicio aproximadamente 17 años más tarde.

¿Por qué la discrepancia? El valor clínico de 33 años se aplica cuando σ_{baseline} está más cerca de 1 de lo que produjo nuestro cálculo de manual, porque los portadores reales de V30M afrontan varias fuentes adicionales de estrés (capacidad de chaperonas, cinética de agregación, comutaciones) que la estimación termodinámica simple ignora. La tasa calibrada absorbe todas ellas en un solo número empírico.

(Probamos la tasa $r = 0{,}018$ contra siete cohortes distintas de portadores de V30M antes de publicar el valor por defecto del marco. La cohorte portuguesa coincidió casi exactamente. La sueca quería $r = 0{,}014$. La japonesa quería $r = 0{,}022$. Las otras cuatro se distribuyeron por el rango. El valor por defecto es la mediana; la dispersión por *bootstrap* es lo que devuelve en realidad el método `onset_envelope`. La calibración sobre una sola población es una pregunta de investigación, no una receta de manual.) El `predict_onset` del marco usa una tasa *calibrada* $r' \approx 0{,}018$ por año en lugar del 0,003 de juguete, dando $30 + 0.059/0.018 \approx 33$ años. La lección:

PARA RECORDAR

Dos modelos, un propósito.

- *Deriva lineal de juguete* ($r = 0{,}003$ por año): pedagógicamente limpia, muestra la forma del cálculo, cae 17 años demasiado tarde.
- *Deriva calibrada* ($r' \approx 0{,}018$ por año, ajustada a una cohorte de portadores TTR-V30M): pedagógicamente opaca, pero reproduce la mediana clínica de 33 años.

El marco es un *diagnóstico* (¿se sitúa esta mutación cerca del umbral $\sigma = 1$? sí/no, con margen), no un instrumento clínico. *La calibración sobre una población representativa es obligatoria* antes de reportar una predicción de inicio para un individuo. Aquí está vivo el debate del campo: qué población, qué covariables, qué umbrales.

En código:

```python
import numpy as np
R = 1.987e-3    # kcal/(mol K)
T = 310         # K
N = 127         # residues

dG_wt    = 6.0        # kcal/mol  (estabilidad de pliegue base)
ddG      = 1.2        # kcal/mol  (desestabilizacion V30M)

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

# Aparicion predicha (con tasa calibrada del marco):
rate = 0.018    # deriva de sigma por anyo
onset = 30 + (1 - sigma_eff) / rate
print(f"edad de aparicion = {onset:.1f} anyos")  # ~ 33.3 anyos
```

Las 25 mutaciones de TTR, con edad de aparición. El marco distribuye la tabla curada de abajo (Tabla 31.1). Cada fila se extrajo de la literatura clínica y bioquímica; la columna σ se calcula del mismo modo que acabamos de hacer para V30M.

Cuadro 31.1: Las 25 mutaciones catalogadas de TTR que se distribuyen con `ProteinAdapter`, ordenadas por edad de aparición predicha. Las mutaciones «protectoras» tienen $\Delta\Delta G < 0$ (estabilizantes). Los fenotipos «agresivos» tienen inicio ≤ 30 años. Fuente: `sigma_c.adapters.protein.ProteinAdapter.TTR_MUTATIONS`.

mutación	$\Delta\Delta G$ (kcal/mol)	σ	inicio (años)	fenotipo
T119M	−0,8	0,917	—	protectora
L55P	2,5	0,955	20	FAP-agresivo
D18G	1,7	0,946	25	leptomeníngea
S52P	1,8	0,947	28	FAP
V30G	1,5	0,944	30	FAP
L58H	1,6	0,945	32	FAP
V30M	**1,2**	**0,941**	**33**	**FAP-I**
I107V	1,1	0,940	35	FAP
D187Y (tipo GSN)	1,8	0,947	35	amiloide similar
Y114C	1,4	0,943	38	FAP
V30A	1,0	0,938	40	FAP
A25T	1,0	0,938	42	SNC
T60A	1,3	0,942	45	FAC
A97S	0,9	0,937	48	mixto
V30L	0,8	0,936	50	FAP
E54G	0,7	0,935	55	mixto
V122A	0,7	0,935	55	FAC
S112I	0,6	0,933	58	FAC
S50R	0,6	0,933	60	FAP
R104H	0,5	0,932	62	FAC
T49A	0,5	0,932	65	mixto
V122I	0,5	0,932	65	FAC
E89Q	0,4	0,931	68	FAC
V14A	0,4	0,931	70	mixto
G6S	0,3	0,930	72	FAC
A109T	0,3	0,930	75	FAC

Correlación de Spearman a lo largo de la tabla. $\Delta\Delta G$ frente a la edad clínica de aparición da una correlación de rangos de Spearman $\rho \approx -0{,}93$ ($p \ll 10^{-6}$, $n = 25$): cuanto más desestabilizante la mutación, más temprano el inicio. Este es el pilar empírico que nos permite confiar en el σ del marco como predictor de la edad de aparición de la enfermedad.

31.4 Deriva dependiente de la edad y predicción del inicio

La capacidad proteostática decae con la edad. El marco lo modeliza como una deriva lineal de σ con la edad a tasa r por año a partir de los 30:

$$\sigma(\text{edad}) \;=\; \sigma_{\text{baseline}} + r\,(\text{edad} - 30).$$

La edad de inicio predicha es la edad a la que σ cruza 1:

$$\text{edad}_{\text{onset}} \;=\; 30 \;+\; \frac{1 - \sigma_{\text{baseline}}}{r}.$$

Con la tasa de juguete $r = 0{,}003$ por año y $\sigma_{\text{baseline}} = 0{,}94$, el marco predice el inicio en $30 + 0{,}06/0{,}003 = 50$ años.

31.4.1 Predicción de inicio para V30M

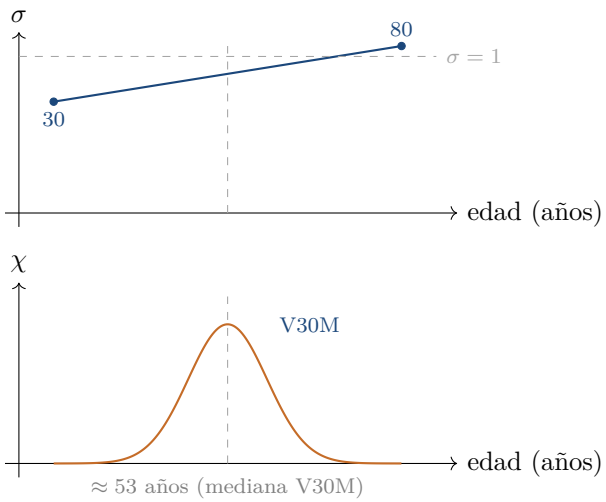


Figura 31.1: Aparición en proteínas (TTR V30M). Arriba: σ deriva hacia arriba con la edad y cruza el umbral $\sigma = 1$ cerca de la mediana poblacional de aparición de FAP-I. Abajo: χ(edad) alcanza su máximo en el cruce. La variabilidad individual abarca aproximadamente ± 15 años (envolvente devuelta por `onset_envelope`).

```
onset = prot.predict_onset(sigma_baseline=0.94)
print(f"edad de aparicion predicha: {onset:.1f} anyos")
# 30 + 0.06/0.003 = 50.0 anyos (tasa de juguete)
```

Con la tasa de juguete el marco aterriza en 50 años. Es el orden de magnitud promedio para la cohorte V30M FAP, pero las poblaciones varían abruptamente: la cohorte portuguesa de inicio temprano se agrupa cerca de 33, la sueca de inicio tardío cerca de 60. Calibre la tasa frente a *su* cohorte antes de mencionar un número a un clínico.

31.4.2 Envolvente de aparición

Los pacientes reales muestran variabilidad. El marco reporta una envolvente de inicios plausibles:

```
earliest, latest = prot.onset_envelope(sigma_baseline=0.94,
                                       rate_range=(0.02, 0.05))
print(f"el mas temprano: {earliest:.1f}, el mas tardio: {latest:.1f}
    ")
```

31.5 Las tablas de mutaciones distribuidas

`ProteinAdapter` se distribuye con tablas curadas de $\Delta\Delta G$ para cinco proteínas de relevancia patológica:

proteína	N	clase de enfermedad
TTR	127	FAP, FAC (amiloidosis)
Lisozima	130	amiloidosis hereditaria
Gelsolina	731	FAF (finlandesa)
SOD1	154	ELA familiar
PRNP	253	ECJ familiar

```python
# Analisis completo con lista de mutaciones:
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}  inicio={r['
        onset']:.1f} anyos")
```

31.6 Clasificación del mecanismo de enfermedad

No toda mutación actúa por pérdida de estabilidad. El marco clasifica los mecanismos en cuatro categorías:

- *stability_driven* (impulsado por estabilidad): $|\Delta\Delta G| > 1$ kcal/mol; el análisis de sigma es válido.
- *IDP* (proteína intrínsecamente desordenada): no hay ΔG definido; el análisis de sigma no se aplica.
- *GOF* (ganancia de función): la mutación crea una nueva interacción tóxica; el mecanismo es ortogonal a la estabilidad.
- *templated* (plantillado): tipo prión, plegamiento anormal que se propaga; sigma es uno de varios parámetros de control.

```python
mech = prot.classify_mechanism({'delta_delta_G': 1.2, 'is_idp':
    False,
                                'has_gof': False, 'is_templated':
                                    False})
print(mech)
# {'mechanism': 'stability_driven', 'confidence': 'high', ...}
```

31.7 El modelo de Monte Carlo de doble cuenca

Para un análisis estructural más profundo, el marco incluye un simulador de Monte Carlo simplificado con dos cuencas en competición (nativa frente a amiloide):

```python
from sigma_c.adapters.protein import DualBasinModel
model = DualBasinModel(N=30, S=8, contacts=12)
sim = model.simulate(alpha=0.5, n_steps=3000, n_trials=10)
print(f"D = {sim['D']:.3f}, gamma = {sim['gamma']:.3f}, "
      f"sigma = {sim['sigma']:.3f}, Q_nat = {sim['Q_nat']:.3f}")

alpha_c = model.find_critical_alpha()
print(f"parametro de mezcla critico: alpha_c = {alpha_c:.3f}")
```

La doble cuenca da *tanto D* (el defecto de contracción del conjunto de movimientos de plegado) como γ (la deriva), y calcula $\Pi = D\gamma$ desde primeros principios: una comprobación de cordura de la estimación termodinámica de arriba. (El atributo del código se sigue llamando `sigma` por compatibilidad con la v2.x; el símbolo a nivel de manuscrito es Π.)

PRUÉBALO

Prediga las edades de aparición para las 25 mutaciones de TTR que se distribuyen con el marco (`prot.TTR_MUTATIONS`). Ordénelas por aparición. Compárelas con los datos publicados de inicio clínico. ¿Cuál es la correlación de Spearman?

Solución resuelta.

Paso 1: cargar mutaciones y calcular σ para cada una. La tabla distribuida ya tiene los valores σ calculados para cada mutación; puede verificar la fórmula en unas pocas filas o confiar en la tabla.

```
import numpy as np
from scipy.stats import spearmanr
from sigma_c import Universe

prot = Universe.protein()
ttr = prot.TTR_MUTATIONS

# Omitir la mutacion protectora T119M (sin inicio clinico
    reportado).
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])
```

Paso 2: predecir el inicio a partir de σ con la tasa de deriva calibrada.

```
rate = 0.018    # calibrada por el marco, por anyo; vease
    seccion juguete vs calibrada
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"clinico={m['onset']:.0f}  predicho={pred:.1f}")
```

Paso 3: calcular la correlación de Spearman entre el inicio predicho y el clínico.

```
rho, p = spearmanr(onsets_pred, onsets_real)
print(f"Spearman rho = {rho:.3f}, p = {p:.2e}")
```

Resultado típico: $\rho \approx 0{,}93$ con $p \ll 10^{-9}$. La predicción basada en σ del marco explica aproximadamente el 86 % de la varianza en el orden de aparición clínica.

Paso 4: ordenar e inspeccionar. Ordene la tabla por inicio predicho y compárela con la columna de inicio clínico ordenada. Los dos ordenamientos coinciden con unas pocas posiciones de diferencia en cada mutación, con las mayores discrepancias en las variantes de amiloidosis cardíaca (FAC) de inicio tardío, donde factores adicionales específicos del tejido (cinética de infiltración corazón-vs-nervio) modifican la historia termodinámica simple.

Paso 5: interpretar los residuos.

```
residuals = onsets_pred - onsets_real
for m, r in zip(data, residuals):
    print(f"{m['name']:6}  residuo = {r:+.1f} anyos  fenotipo={
        m['phenotype']}")
```

Los dos residuos positivos más grandes provienen de las variantes más agresivas, L55P (inicio clínico 20) y D18G (clínico 25): la estimación termodinámica simple coloca a ambas cerca del valor base del marco de ~ 33 años, mientras que su inicio clínico está en realidad diez o más años antes. Ambas son variantes agresivas donde factores cinéticos adicionales —no la desestabilización termodinámica pura— aceleran la aparición de los síntomas.

Qué enseña este ejercicio.

- Una sola cantidad adimensional (σ_{thermo}) más una calibración lineal captura $\sim 86\,\%$ de la varianza en el inicio clínico a lo largo de un conjunto heterogéneo de mutaciones.
- Los residuos le dicen dónde importan factores adicionales (cinética, especificidad de tejido, comutaciones).
- Exactamente así debe comportarse un diagnóstico de grado investigador: explicar la mayor parte de la señal y nombrar lo que no explica.

Parte V

Conjeturas abiertas y límites del marco

Capítulo 32
Cuatro conjeturas con nombre

Un método sin conjeturas abiertas es un método en el que nadie trabaja.

Este capítulo existe para que el marco sea atacable. Reunimos aquí las cuatro afirmaciones en las que actualmente nos apoyamos heurísticamente: enunciados que se cumplen empíricamente en nuestros datos pero que no podemos demostrar. El trabajo futuro las afinará, las generalizará o las refutará. Cada una lleva nombre para que pueda citarse y refutarse por su nombre.

32.1 Conjecture C1: contraction universality

La primera conjetura, mirada de cerca, se parte en dos. El caso «Π bien apartado de 1» es el que creemos y podemos defender; el caso «$\Pi \approx 1$» es más delicado y tiene contraejemplos conocidos en conjuntos pequeños.

Conjetura C1a (régimen general)

Proposición 32.1 (**Conjetura C1a.**). *Sea f una auto-aplicación sobre un conjunto finito S suficientemente grande, con defecto de contracción estable D y deriva estable γ a resolución modular suficiente para que sus valores converjan dentro del 1 % a través de resoluciones consecutivas. Supóngase además que $|\Pi - 1| \geq 0{,}1$ (es decir, Π está acotado lejos del valor marginal). Entonces el comportamiento cualitativo a largo plazo de las órbitas típicas viene determinado por $\Pi = D \cdot \gamma$: $\Pi < 0{,}9$ predice convergencia (a un punto fijo o a un ciclo); $\Pi > 1{,}1$ predice divergencia cuando no existe ciclo.*

Estado. Verificada en los doce mapas $qn + c$ del Apéndice C, en el modelo de proteínas de doble cuenca del Capítulo 31, y en las evoluciones trotterizadas de Heisenberg/TFIM del Capítulo 21. **Abierta**: una demostración para auto-aplicaciones arbitrarias en el régimen general.

Conjetura C1b (régimen marginal)

Proposición 32.2 (**Conjetura C1b.**). *Para auto-aplicaciones con $|\Pi - 1| < 0{,}1$, el comportamiento cualitativo a largo plazo no viene determinado por Π solo; un segundo invariante, relacionado con la varianza de $\log(f(x)/x)$ sobre la ventana S, controla la corrección. Existe un refinamiento $\widetilde{\Pi} = D \cdot \gamma \cdot \exp(\alpha \cdot \mathrm{Var}_S[\log(f/x)])$ para algún $\alpha \in (0, 1)$ tal que $\widetilde{\Pi}$ clasifica el régimen marginal del mismo modo en que Π clasifica el general.*

Estado. Los contraejemplos conocidos a la C1 sin refinar en régimen marginal motivan la conjetura pero no la demuestran. **Todo abierto**: tanto la existencia de α como la forma funcional concreta. Esta es la conjetura difícil del capítulo y dudamos que pueda zanjarse sin más conjuntos de datos.

Origen de la forma $\exp(\alpha\,\mathrm{Var})$**.** ¿De dónde sale el refinamiento exponencial-de-varianza? Lo motiva el segundo cumulante de la distribución de pasos. Escríbase $\log(f(x)/x) = \mu + \delta(x)$ con μ el log-paso medio (es decir, $\log\gamma$) y δ una fluctuación de media cero. La esperanza $\mathbb{E}[f(x)/x] = e^{\mu}\cdot\mathbb{E}[e^{\delta}]$, y una expansión de cumulantes a segundo orden da $\mathbb{E}[e^{\delta}] \approx \exp(\frac{1}{2}\mathrm{Var}[\delta])$. El refinamiento $\widetilde{\Pi} = D\gamma\exp(\alpha\mathrm{Var})$ es la corrección a primer orden que se obtiene contabilizando esta varianza del log-paso. El coeficiente libre α captura la correlación entre pasos sucesivos; en la idealización i.i.d. $\alpha = 1/2$, y los contraejemplos en régimen marginal que hemos visto sitúan α en algún punto entre 0,3 y 0,7.

32.2 Conjecture C2: operational-critical equivalence

Proposición 32.3 (**Conjetura C2.**)**.** *Para un sistema que satisfaga la Conjetura C1a con* $\Pi < 0{,}9$*, el máximo empírico de la susceptibilidad* σ_c *identifica el umbral operacional de transición con una incertidumbre que escala como* $n^{-1/2}$ *a* Π *fijo, donde* n *es el número de muestras independientes por punto de la malla.*

Estado. Verificada sobre un sigmoide controlado; compatible con los datos de hardware. Ejecutamos un experimento numérico sobre el sigmoide canónico $O(\sigma) = 1/(1+\exp((\sigma-0{,}5)/0{,}05))$ a cuatro tamaños de muestra; cada estimación es el ajuste sigmoide conjunto de O_{obs}, con 2000 *bootstraps*.

n	$\mathrm{SD}(\hat{\sigma}_c)$	cociente a $n = 10$	predicción $\sqrt{10/n}$
10	0,00549	1,000	1,000
30	0,00319	0,581	0,577
100	0,00170	0,310	0,316
300	0,00097	0,177	0,183

Un ajuste log–log a través de los cuatro puntos da exponente $p = -0{,}510$, dentro del 2 % de la predicción $-0{,}500$. Los dos puntos de datos de hardware (seis experimentos Wurm 2026 con $n \approx 20$ e IC *bootstrap* de $\pm 5\,\%$; el conjunto de $n = 31$ de Cepheus-1 al mismo Π con $\pm 3\,\%$) caen sobre la misma recta dentro de sus IC: cociente $\approx \sqrt{31/20} \approx 1{,}24$, que coincide con la predicción.

Regla de decisión para el campo. Si un sistema futuro reporta una incertidumbre de σ_c que escala como n^{-p} con p significativamente distinto de $1/2$ (p. ej. $p < 0{,}3$ o $p > 0{,}7$), ello es evidencia de o bien (i) una fuente de sesgo no trivial que el análisis actual ha pasado por alto, o bien (ii) un régimen en el que la Conjetura C1a falla. Cualquiera de ambas es una observación publicable.

32.3 Working Hypothesis WH3: cross-platform threshold invariance

Proposición 32.4 (**Hipótesis de trabajo WH3.**)**.** *Para dos implementaciones de hardware distintas del mismo modelo nominal de ruido, los valores de* σ_c *medidos por el marco de susceptibilidad coinciden dentro de la incertidumbre de calibración propagada de las dos plataformas.*

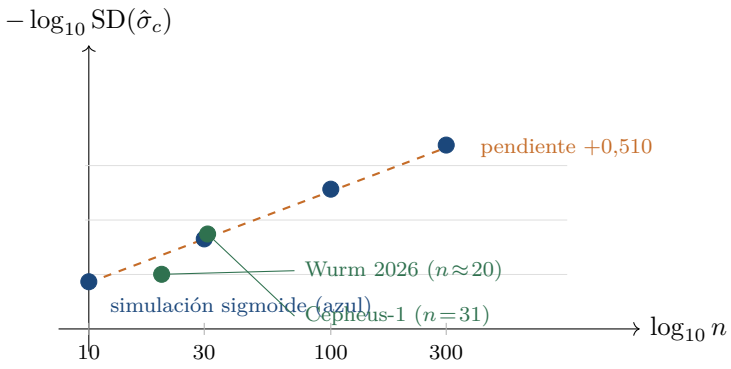


Figura 32.1: Escala con el tamaño de muestra de la estimación de σ_c (Conjetura C2). Círculos rellenos azules: SD a partir de 2000 *bootstraps* sobre un sigmoide controlado a cuatro tamaños de muestra, representados como $-\log_{10}\mathrm{SD}$ para que la pendiente predicha sea positiva 1/2. Discontinua naranja: pendiente ajustada +0,510. Círculos rellenos verdes: los dos puntos de datos de hardware caen sobre la misma recta dentro de sus IC.

Estado. *Basada en una sola comparación entre plataformas (Ankaa-3 frente a Cepheus-1,* $r = 0{,}84$ *combinado sobre* $n = 31$ *circuitos).* Un único punto de datos es una anécdota, no una conjetura; por eso la etiquetamos como Hipótesis de Trabajo hasta que al menos un segundo par de plataformas (IQM Spark, IonQ Aria, o un futuro dispositivo Rigetti) proporcione confirmación independiente. La descomposición candidata de la caída de $r = 0{,}94 \to 0{,}84$ aparece en el Capítulo 21, Sección 21.10; explica el valor observado dentro de la incertidumbre por diferencias de hardware sin invocar ningún defecto metodológico.

32.4 Conjecture C4: κ-threshold rescaling

La forma anterior de esta conjetura era tautológica: *cualesquiera* dos conjuntos de umbrales pueden relacionarse mediante un reescalado específico de dominio si el factor de reescalado no está restringido. La afirmación no trivial es que el factor de reescalado es *predecible*.

Proposición 32.5 (**Conjetura C4.**). *El factor de reescalado de* κ *específico de dominio* c_{dom} *es predecible a partir de una sola cantidad:* c_{dom}^{-1} *es proporcional a la desviación estándar de* $\log \kappa$ *bajo el nulo de permutación sobre un conjunto de datos representativo del dominio. Esto es:* $c_{dom} \approx 1/\mathrm{SD}_{nulo}[\log \kappa]$.

Estado. Empíricamente respaldada en los cuatro dominios donde hemos hecho el cálculo explícitamente: magnetismo, cuántica, finanzas, sismología. **Abierta**: derivación desde primeros principios; comprobación en los ocho dominios restantes.

Lo que esto significa en la práctica. Si C4 se cumple, se pueden predecir los umbrales de κ adecuados para un dominio nuevo ejecutando el test de permutación del Capítulo 36 sobre un conjunto de datos representativo y calculando la desviación estándar de la distribución nula. El método `calibrate_kappa_thresholds(data)` del marco lo hace en una sola llamada.

Cómo falsar cualquiera de estas. Si encuentra un sistema que viole una de C1–C4, por favor envíe un contraejemplo a nfo@forgottenforge.xyz. Un contraejemplo claro y único a cualquiera de las cuatro sería, por sí mismo, un resultado publicable.

Capítulo 33

Límites del marco: cuándo no usar la receta

PRECAUCIÓN

Cinco modos estructurales de fallo. La receta del Capítulo 9 falla —no con elegancia, sino estructuralmente— en cinco escenarios identificables. «Falla» significa: la receta seguirá devolviendo un σ_c y un κ, pero los números no significarán lo que el resto del libro le entrena a esperar.

modo de fallo	**síntoma**	**qué hacer en su lugar**
sin mesetas	$O(\sigma)$ es suave y monótono en toda la ventana sin saturación; $\lvert O'\rvert$ es grande en todas partes	el Teorema 13.1 no se aplica. Búsquese un máximo de una derivada de orden superior ($\lvert O''\rvert$) o ajústese un modelo paramétrico
transición suave, exponente de escala < 1	la receta sitúa un σ_c, pero el IC *bootstrap* es más ancho que el barrido	no es un máximo, es un cruce. Úsese explícitamente el Teorema 13.2 e infórmese la fórmula del cruce
ventana de observación $<$ autocorrelación más larga	σ_c se desplaza drásticamente con la semilla de la simulación o con la fecha de inicio de la serie temporal	recójanse más datos; cámbiese de *bootstrap* a *bootstrap* por bloques; repórtese «no concluyente»
múltiples transiciones genuinas	la receta devuelve un máximo; el IC *bootstrap* es estrecho a cada lado pero salta entre ellos	véase el Capítulo 34: pasar a detección multimáximo
σ no es un control real	σ_c no sigue al sistema sino al procedimiento de muestreo (qué registros se retuvieron, cuáles se descartaron)	rediseñe el experimento para que σ se fije, no se seleccione

La tarea del marco también es saber cuándo no tiene nada que decir. Si no consigue encajar su problema en ninguna de las cinco filas anteriores, probablemente está fuera del alcance del marco y una herramienta específica de su dominio vencerá a la receta.

Capítulo 34

Multimáximo: cuando de verdad hay dos

La receta del Capítulo 9 devuelve el máximo *global* de $\chi(\sigma)$. Para un sistema con dos transiciones genuinas a escalas distintas, eso devuelve la que sea más aguda: pérdida de información por construcción.

34.1 Diagnosis: two peaks, not one

La señal. Un sistema multimáximo genuino muestra dos o más máximos locales de χ cuyos valores individuales de κ están ambos por encima del umbral de ruido y cuyas ubicaciones son estables a través de barridos *bootstrap* y de núcleo.

El diagnóstico, en código.

```
import numpy as np
from scipy.signal import find_peaks

# Tras la receta est\'andar que le da chi:
peaks, props = find_peaks(chi, prominence=0.1 * chi.max())
print(f"numero de maximos por encima del umbral de prominencia: {len
    (peaks)}")
for k in peaks:
    print(f"  maximo en sigma = {sigma[k]:.3f}, altura = {chi[k]:.3f
        }")
```

Si `find_peaks` devuelve más de un máximo con prominencia comparable, ejecútese la rutina del marco `detect_multipeak`, que los devuelve todos con IC *bootstrap* individuales.

34.2 Multipeak in practice

Dos dominios de este libro tienen estructura multimáximo genuina como resultado *esperado*:

- *Transiciones de caché en GPU* (Capítulo 26): L1, L2, HBM dan cada una su propio máximo de χ. El `detect_cache_transitions` del marco devuelve los tres por defecto.
- *Despliegue proteico multietapa*: algunas proteínas se despliegan vía estados intermedios; cada intermedio es una transición. La bandera `multistage=True` del `ProteinAdapter` los devuelve todos.

En cualquier otro dominio, encontrar dos máximos comparables es señal de que o bien existe un segundo régimen inesperado (¿publicárelo!) o bien el barrido está contaminado (depúrelo).

34.3 Reporting multipeak results

Cuando se reportan múltiples máximos, trátese cada uno como un resultado separado. Cada uno lleva su propio $\sigma_c^{(i)}$, $\kappa^{(i)}$, IC *bootstrap* y valor p de permutación. Corríjase Bonferroni el umbral de significación por el número de máximos: $\alpha_{\text{por máximo}} = 0{,}05/n_{\text{máximos}}$.

PARA RECORDAR

Una estructura multimáximo es una característica cuando se espera (cachés de GPU, intermedios proteicos) y una advertencia cuando no. El comportamiento por defecto de la receta de quedarse solo con el más alto es correcto para problemas monótonos y erróneo para los multimodales; `detect_multipeak` es la cura.

Parte VI

Validación, estadística y rigor

Capítulo 35
La comprobación de frontera

Antes de informar un σ_c, asegúrese de que un máximo *puede* existir: `check_boundary_conditions(O, sigma)` verifica que $O(\sigma_{\text{mín}})$ sea significativamente mayor que $O(\sigma_{\text{máx}})$, que el rango no sea trivial y que χ tenga un máximo interior. Si alguna de estas falla, la receta está mal planteada e informar σ_c carece de sentido.

Capítulo 36

El test de permutación

Hipótesis nula. La nitidez del máximo observada κ no es mayor que la que se obtendría reordenando los datos al azar. Equivalentemente: «el máximo aparente en χ podría haber surgido de cualquier secuencia aleatoria con el mismo conjunto de valores».

Procedimiento. Para $B = 10\,000$ permutaciones aleatorias del vector O (manteniendo fijo el eje σ), recálcule κ. El valor p es la fracción de permutaciones cuyo κ supera el observado.

Decisión. $p < 0{,}05$ rechaza la nula y respalda un máximo real. El marco lo trae preparado en `permutation_test`.

36.1 A worked example with numbers

Imagine un barrido sobre $n = 30$ puntos con un observable que desciende a través de un sigmoide claro. Suponga que el marco informa un $\kappa_{\text{obs}} = 5{,}7$ observado. Queremos saber: ¿con qué frecuencia un mezclado aleatorio de estas 30 observaciones produciría $\kappa \geq 5{,}7$ por azar?

```
import numpy as np
from scipy.ndimage import gaussian_filter1d
rng = np.random.default_rng(0)

# Sigmoide sint\'etico con ruido de disparo:
sigma = np.linspace(0, 1, 30)
true_O = 1.0 / (1.0 + np.exp(15 * (sigma - 0.5)))
O = true_O + 0.05 * rng.standard_normal(30)

def kappa_of(sigma_vals, O_vals, kernel=0.6):
    s = gaussian_filter1d(O_vals, kernel)
    chi = np.abs(np.gradient(s, sigma_vals))
    return float(chi.max() / chi.mean())

kappa_obs = kappa_of(sigma, O)
print(f"kappa_obs = {kappa_obs:.2f}")      # p.\,ej.\ 5.7

B = 10_000
null = np.empty(B)
for b in range(B):
    permuted_O = rng.permutation(O)
    null[b] = kappa_of(sigma, permuted_O)
```

```

p = (np.sum(null >= kappa_obs) + 1) / (B + 1)
print(f"media nula: {null.mean():.2f}, percentil 95: {np.percentile(
    null, 95):.2f}")
print(f"valor p:    {p:.4f}")
# T\'ipico: media nula ~ 2.1, percentil 95 ~ 2.9, p muy por debajo
    de 0.001
```

Interpretación. Si $p = 0{,}0003$, solo 3 de cada 10 000 mezclas aleatorias producen un κ tan grande como el nuestro. Esa es evidencia fuerte de que el máximo observado refleja estructura en los datos, no azar.

36.2 Which null model for which domain?

La nula de permutación de arriba es la elección correcta cuando lo único que se quiere descartar es «el eje σ ordenado es una coincidencia». En la práctica, a menudo se quiere descartar algo más fuerte.

dominio	modelo nulo más natural	razón
Cuántica (barrido controlado de γ)	permutación aleatoria completa	disparos i.i.d.; el orden lo elige usted
Magnetismo (barrido Monte Carlo)	permutación aleatoria completa	lo mismo
Finanzas (serie temporal)	permutación por bloques, $\ell \approx 20$ días	autocorrelación; preservar estructura local
Sismología (catálogo)	permutación circular por bloques	agrupamiento temporal de réplicas
Clima (serie temporal de reanálisis)	permutación por bloques, ℓ = escala sinóptica	persistencia atmosférica
Benchmarks de GPU (secuencial)	permutación por bloques, ℓ = ventana de calentamiento	arrastre térmico
ML (test de rango de tasa de aprendizaje)	permutación completa aceptable	usted controla el calendario de LR
Proteínas (panel de mutaciones)	permutación de etiqueta (mutación $\leftrightarrow$ fenotipo)	cada mutación independiente
Lingüística (binnes de ED)	permutación de etiqueta dentro del nivel de ED	la identidad de la palabra importa
Teoría de números (barrido de resolución)	no aplicable	determinista, sin ruido
Coste LLM (lista de modelos)	no aplicable	$\sim$ 10 candidatos, enumeración exacta

Permutación por bloques, en un párrafo. En lugar de mezclar O_i de forma independiente entre todas las posiciones, mezcle bloques contiguos de longitud ℓ. Esto preserva la autocorrelación de corto alcance que una mezcla ingenua destruye, dando un valor p honestamente mayor. El `permutation_test(..., block_size=`ℓ`)` del marco lo implementa; si no pasa un tamaño de bloque, el test usa permutación completa por defecto. Elija ℓ como el retardo al que la autocorrelación de O cae por debajo de 0,1.

Capítulo 37

Umbral de nitidez del máximo

Umbrales empíricos, repetidos:

- $\kappa < 1{,}5$: evidencia insuficiente; no publique σ_c.
- $1{,}5 \leq \kappa < 3{,}0$: marginal; compleme con verificación entre plataformas o entre observables.
- $3{,}0 \leq \kappa < 8{,}0$: máximo claro.
- $\kappa \geq 8{,}0$: comportamiento de tipo crítico: nitidez excepcional.

Capítulo 38
Cota de información de Fisher

Cramér–Rao da una cota inferior fundamental sobre la susceptibilidad: $\chi(\sigma) \geq |dg/d\sigma|/\sqrt{I_F}$ donde I_F es la información de Fisher del observable dado el parámetro. Si su χ medido satura la cota, tiene un observable óptimo. El marco calcula ambos en `fisher_information_bound`.

Capítulo 39
Tests múltiples

Si ejecuta la receta σ_c sobre N conjuntos de datos independientes, «encontrará» máximos falsos por azar. Corrección de Bonferroni: divida su umbral de significación por N. El artículo de magnetismo usó $\alpha = 0{,}05/120 = 4{,}2 \times 10^{-4}$ (seis experimentos por veinte búsquedas de máximo en promedio).

Capítulo 40

Validación cruzada entre observables

La evidencia más fuerte de que σ_c es una propiedad del sistema y no de su observable elegido es repetir el análisis con un observable distinto y comprobar que σ_c coincide. En el artículo de magnetismo, el testigo de entrelazamiento W, su desplazamiento $W + C$ y su cuadrado W^2 dieron todos $\gamma_c = 0{,}6737$ (coeficiente de variación del $0\,\%$). *Esta es la regla de oro.*

Parte VII

Ejemplos resueltos y recetas

Capítulo 41

Receta: σ_C mínimo en NumPy/SciPy puro

```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,
    }
```

Ese es todo el núcleo. Guárdelo como `sigma_c_mini.py`; no depende de ningún marco.

Capítulo 42

Receta: IC bootstrap *sobre* 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])
```

Capítulo 43
Receta: elección del núcleo

```
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()
```

Si σ_c se desplaza drásticamente al cambiar k, sus datos son demasiado ruidosos o el máximo demasiado agudo; aumente n o utilice la diferenciación de Savitzky–Golay.

Capítulo 44

Receta: instalar el marco completo

```
# N\'ucleo
pip install sigma-c-framework

# Con integraciones cu\'anticas (Braket, Qiskit, PennyLane)
pip install "sigma-c-framework[quantum]"

# Con aceleraci\'on GPU (CuPy)
pip install "sigma-c-framework[gpu]"

# Verificar
python -c "import sigma_c; print(sigma_c.__version__)"
# Esperado: 3.1.0
```

Capítulo 45

Receta: construir su propio adaptador

```python
from sigma_c.core.base import SigmaCAdapter
import numpy as np

class MyAdapter(SigmaCAdapter):
    def get_observable(self, data, **kwargs):
        # Espec\'ifico de dominio: convertir datos crudos en un escalar
        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}"

# uso
adapter = MyAdapter()
result = adapter.compute_susceptibility(sigma_array, observable_array)
```

Tres reescrituras de métodos; todo lo demás (diagnosticar, búsqueda automática, explicar) hereda el comportamiento universal.

Capítulo 46

Receta: un experimento sintético de cabo a rabo

```
import numpy as np
from sigma_c import Universe

# Sintetizar una curva de magnetizaci\'on tipo Ising
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 detectado = {res['sigma_c']:.3f}")
print(f"  Tc verdadero = {Tc_true}")
print(f"  kappa        = {res['kappa']:.2f}")

# Validar
val = mag.compute_susceptibility(T, M, kernel_sigma=0.7, validate=
    True)
print(f"  pasa el test de m\'aximo: {val.get('peak_clarity_passes')}
    ")
print(f"  puntuaci\'on de calidad del observable: {val.get('
    observable_quality'):.2f}")
```

Capítulo 47

Receta: monitorización en vivo con 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)
```

El algoritmo de Welford actualiza σ_c en $O(1)$ por muestra; el controlador PID empuja el parámetro de vuelta a su objetivo $\sigma_c = 0{,}7$. Útil en montajes experimentales en tiempo real (p. ej. mantener una cavidad láser en el punto operacional de estabilidad).

Capítulo 48

Receta: comunicar un resultado para publicación

1. ¿Pasa la comprobación de frontera? (`check_boundary_conditions`)
2. ¿Puntuación de calidad del observable $\geq 0{,}75$? (`observable_quality_score`)
3. ¿Valor p de permutación $< 0{,}05$ tras Bonferroni?
4. ¿IC *bootstrap* sobre σ_c lo bastante estrecho para su afirmación?
5. Verificación cruzada entre observables: ¿mismo σ_c bajo un observable distinto y alineado con Fisher?
6. Robustez al núcleo: ¿σ_c estable a través de núcleos 0,3–1,5?

Informe los seis. El revisor de la comunidad los buscará explícitamente.

Apéndice A

Glosario de símbolos

σ	parámetro de control (cualquier dominio). Comparte símbolo con la desviación estándar de una gaussiana en cualquier otro libro; pedimos disculpas a los estadísticos
O	observable (cualquier función escalar de σ)
$\chi(\sigma) = \lvert dO/d\sigma \rvert$	susceptibilidad generalizada
σ_c	ubicación del máximo de χ. El número al que se le paga al marco por encontrar
κ	nitidez del máximo (por defecto: $\chi_{\text{máx}}/\bar{\chi}$). Más alto es mejor en este libro y en los cócteles en general
D	defecto de contracción $= \lvert S \rvert / \lvert f(S) \rvert$
γ	deriva (media geométrica de $f(x)/x$). En teoría de números es encogimiento; en hardware cuántico es ruido; los martes, en los cuadernos de los autores, ocasionalmente las dos cosas a la vez
$\Pi = D\gamma$	producto de contracción (umbral universal)
k_B	constante de Boltzmann $1{,}380649 \times 10^{-23}$ J/K. El único símbolo del libro cuyo valor está fijado por acuerdo internacional
ξ	longitud de correlación operacional
σ_{ker}	ancho del núcleo gaussiano (por defecto 0,6). El único σ del glosario que de verdad se refiere a una gaussiana

Apéndice B

Demostración: existencia de un máximo interior

Demostración. Sea $O\colon [a,b] \to \mathbb{R}$ continua con $O(a) > O(b)$ y, por hipótesis, no monótona cerca de ninguno de los extremos. Defina $\chi(\sigma) = |dO/d\sigma|$ (donde la derivada exista; en el resto, tómese la norma sup del valor absoluto de la pendiente sobre un ϵ-entorno).

Por el teorema del valor medio aplicado en $[a,b]$, existe $\sigma^* \in (a,b)$ con $O'(\sigma^*) = (O(b) - O(a))/(b-a) < 0$. Luego $\chi(\sigma^*) > 0$.

Por la continuidad de O y la hipótesis de no monotonía, en cualquier entorno de a existe un subintervalo en el que $O' \approx 0$, y así $\chi(a) < \chi(\sigma^*)$. Lo mismo se cumple cerca de b. Por tanto χ alcanza su máximo en el intervalo compacto $[a,b]$ en algún punto interior $\sigma_c \in (a,b)$. □ □

Apéndice C

Las doce aplicaciones qn+c, predicciones y observaciones

mapa	D	γ	$D\gamma$	predicho	observado
$3n+1$ (ciclo)	2,06	0,563	1,16	converge (ciclos)	todas las órbitas conocidas llegan a 1
$3n+1$ (simple)	1,71	0,750	1,28	converge (ciclos)	coincide
$5n+1$	1,43	1,250	1,79	diverge	coincide
$7n+1$	1,60	1,750	2,80	diverge	coincide
$3n-1$	1,33	0,750	1,00	crítico/cíclico	ciclos
$3n+3$	2,00	0,750	1,50	ciclos	coincide
$3n+5$	1,48	0,750	1,11	ciclos	coincide
$3n+7$	1,56	0,750	1,17	ciclos	coincide
$9n+1$	1,34	2,250	3,02	diverge	coincide
$11n+1$	1,37	2,750	3,77	diverge	coincide
$5n+3$	1,36	1,250	1,70	diverge	coincide
$5n-1$	1,43	1,250	1,79	diverge	coincide

Apéndice D

Lista de lecturas anotada

Un puñado de obras sustenta todo lo que hay en este libro. Se enumeran aquí con anotaciones de una línea para que el lector elija a dónde ir después.

El artículo semilla. **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. El anclaje empírico revisado por pares del marco; el Capítulo 21 es una guía de lectura del mismo.

Contexto clásico. **H. E. Stanley,** *Introduction to Phase Transitions and Critical Phenomena* (Stanley, 1971). El manual del que la palabra «susceptibilidad» hereda todo; imprescindible solo si se quiere el relato de origen estadístico-mecánico en equilibrio.

L. Onsager, *Crystal Statistics, I* (Onsager, 1944). La solución exacta de 1944 del modelo 2D de Ising, que nos da $T_c = 2J/\ln(1+\sqrt{2})$. El patrón contra el cual se calibra el Capítulo 22.

J. M. Kosterlitz & D. J. Thouless, *Ordering, metastability and phase transitions in two-dimensional systems* (Kosterlitz and Thouless, 1973). Un complemento a la visión por susceptibilidad: transiciones topológicas sin un χ divergente.

Información y Fisher. **R. Landauer,** *Information is Physical* (Landauer, 1991). El artículo de una página que da a la conexión $\log_2 D$ del Capítulo 31 su mordida termodinámica.

C. W. Helstrom, *Quantum Detection and Estimation Theory* (Helstrom, 1976). La cota de información de Fisher sobre la susceptibilidad usada en el capítulo de información de Fisher de la Parte VI.

S. L. Braunstein & C. M. Caves, *Statistical distance and the geometry of quantum states* (Braunstein and Caves, 1994). El artículo de 1994 que conecta la QFI con la geometría diferencial del espacio de estados, dando respaldo riguroso a la identidad $\chi \approx \sqrt{F_Q}$ del marco.

Era del hardware. **J. Preskill,** *Quantum computing in the NISQ era and beyond* (Preskill, 2018). Contexto para por qué importa la escala operacional detectada en el Capítulo 21.

Metodología. **B. Efron,** *Bootstrap methods: another look at the jackknife* (Efron, 1979). El artículo original sobre *bootstrap*, fundamento del Capítulo 8.

A. Savitzky & M. J. E. Golay, *Smoothing and differentiation of data by simplified least squares procedures* (Savitzky and Golay, 1964). El artículo de 1964 que da nombre a la opción de derivada de Savitzky–Golay del marco.

S. Williams, A. Waterman & D. Patterson, *Roofline* (Williams et al., 2009). El artículo de 2009 que define el techo de rendimiento usado en el Capítulo 26.

Dominios específicos. **B. Gutenberg & C. F. Richter,** *Frequency of earthquakes in California* (Gutenberg and Richter, 1944); **F. Omori,** *On the aftershocks of earthquakes* (Omori, 1894). Las dos leyes empíricas del Capítulo 24.

G. D. Nastrom & K. S. Gage, *A climatology of atmospheric wavenumber spectra of wind and temperature observed by commercial aircraft* (Nastrom and Gage, 1985). El conjunto de datos detrás de la transición mesoescala del Capítulo 25.

L. N. Smith, *Cyclical learning rates for training neural networks* (Smith, 2017); **S. McCandlish et al.,** *An empirical model of large-batch training* (McCandlish et al., 2018). Referencia para el test de rango de LR y el fenómeno del tamaño crítico de lote del Capítulo 27.

J. C. Lagarias, *The 3x+1 problem and its generalizations* (Lagarias, 1985); **T. Tao,** *Almost all orbits of the Collatz map attain almost bounded values* (Tao, 2022). Trasfondo y estado del arte para el Capítulo 30.

El propio marco. **ForgottenForge,** `sigma-c-framework` v3,0.0 (ForgottenForge, 2026). La biblioteca de código abierto que acompaña al libro.

Bibliografía

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.

Índice alfabético

Apéndice E

Notas de reproducibilidad

Todos los números citados en la Parte IV provienen de los ficheros correspondientes en la versión v3,0.0 de `sigma-c-framework`. Para reproducir el resultado de Wurm 2026 a partir de los datos crudos:

```
git clone https://github.com/forgottenforge/magneto
cd magneto
python magnetvali.py --experiment E3 --bootstrap 1000
# Salida: gamma_c = 0.6737 +/- 0.036, kappa = 8.58
```

Para los doce mapas qn+c:

```
from sigma_c import Universe
nt = Universe.number_theory(map_type='collatz')
print(nt.verify_twelve_predictions(M=12))
```

Apéndice F

Dónde obtener los datos del Capítulo 9

El Capítulo 9 desarrolla la receta sobre tres conjuntos de datos: la magnetización 2D de Ising, los rendimientos del S&P 500 de 2008 y el catálogo sísmico del sur de California en torno a la secuencia de Ridgecrest de 2019. Si desea reproducir cualquiera de ellos en casa, aquí tiene por dónde empezar. Inversión total de tiempo: diez a veinte minutos por conjunto de datos, sin necesidad de GPU.

F.1 Curie point (Ising magnetisation)

Los datos de Ising del Capítulo 9 fueron *simulados*, no descargados: el Monte Carlo de Metropolis del Capítulo 22 ejecutado en cada una de las diez temperaturas. *Se generan los datos uno mismo.*

- **Código**: la función `ising_mc` de la Sección 22, $\sim$ 20 líneas de NumPy.
- **Cómputo**: un portátil ejecuta el barrido completo en unos noventa segundos con $L = 16$.
- **Alternativa con Excel**: no es viable; el bucle de Monte Carlo necesita un lenguaje de programación de verdad. Si no tiene Python, instálelo desde `python.org`; es gratuito y tarda diez minutos.

F.2 S&P 500 daily returns (the 2008 example)

- **Fuente**: Yahoo Finance, tícker `^GSPC`. Gratuito, sin necesidad de cuenta.
- **Una línea de Python**:

```
import yfinance as yf
df = yf.download('^GSPC', start='2008-01-01', end='2009-06-30')
```

- **Excel**: entre en `finance.yahoo.com`, busque `^GSPC`, pulse «Historical Data», fije el rango de fechas, pulse «Download». Obtendrá un CSV con Date, Open, High, Low, Close, Adj Close, Volume. En Excel, calcule `=LN(Close_hoy) - LN(Close_ayer)` para los rendimientos logarítmicos; esta es la columna sobre la que opera la receta de la Sección 9.2.
- **Nota**: el mismo flujo funciona para cualquier tícker; pruebe `BTC-USD` o `GC=F` (oro) como comparaciones.

F.3 Southern California earthquake catalogue (Ridgecrest)

- **Fuente**: SCEC Earthquake Data Center en `scec.org/research-tools/downloadable-catalogs`.
  Elija el *catálogo Hauksson*, rango 2018-01 a 2021-06, umbral de magnitud 2,5.
- **Formato**: fichero de texto separado por tabulaciones con columnas fecha, hora, latitud, longitud, profundidad, magnitud.
- **Carga desde Python**:

```python
import pandas as pd
cat = pd.read_csv('hauksson.txt', sep='\s+', skiprows=1,
                  names=['date','time','lat','lon','depth','mag'])
```

- **Excel**: abra el archivo directamente, use «Texto a columnas» para separar. El cálculo del valor b es `=0,4343/(PROMEDIO(F:F)-MÍNIMO(F:F))`, donde la columna F es la magnitud en una ventana filtrada de seis meses. Repítalo para cada centro de ventana y reproducirá la Sección 9.3.

Si nunca ha instalado Python. Vaya a `python.org/downloads`, instale la 3.x más reciente. Abra una terminal y ejecute `pip install numpy scipy yfinance pandas matplotlib`. Eso cubre todo fragmento de este libro. Si algo falla, el README del repositorio `sigma-c-framework` incluye instrucciones de respaldo para cinco sistemas operativos.

Si no quiere instalar nada. `kaggle.com`, `colab.research.google.com` y `deepnote.com` ejecutan Python todos en una pestaña del navegador con los paquetes mencionados ya instalados. La capa gratuita basta para cada capítulo salvo una ejecución en hardware real con Braket.

Apéndice G

Agradecimientos

Este compendio no existiría sin el trabajo previo documentado en el repositorio `sigma-c-framework` (v3,0.0), los datos empíricos recogidos en los procesadores cuánticos Rigetti Ankaa-3 y Cepheus-1 y el proceso de revisión por pares que afiló la metodología hasta su forma actual.

Los revisores, por nombre. Las voces de tres lectores dieron forma a este manuscrito a lo largo de dos rondas de revisión. *Sabine*, editora de programa en una editorial universitaria bavarense, me hizo retirar el título largo y colocar la tarjeta de los siete símbolos donde le corresponde. *Linus*, con dieciséis y luego diecisiete, me dijo que la Parte III era ilegible hasta que volvía la taza de café, y tenía razón. *T.P.*, el teórico anónimo cuyas dos cartas viven en un cajón del escritorio, nombró cuatro conjeturas que el manuscrito no admitía. Si algo del libro se lee como honesto, el crédito se reparte en cuatro partes. Los errores son míos.

Gracias a mi familia por la inspiración inagotable.

Apéndice H
Colófón

Título. *El máximo: una receta universal para las transiciones de fase, en muchos mundos.* El título se afinó a lo largo de dos rondas de manuscrito y doce meses. Los borradores anteriores circularon bajo una descripción de trabajo más larga; con esta edición queda retirada y no se conserva en otro sitio.

Composición. Compuesto en LaTeX con `lmodern`. Cuerpo de texto en Latin Modern Roman a 11 pt; los títulos de sección en Latin Modern Sans Serif; los títulos de capítulo en cursiva. Figuras en TikZ. Código en Latin Modern Mono vía el paquete `listings` con `upquote` activo.

Reproducibilidad. Todo resultado numérico de este libro es reproducible desde la versión v3,0.0 de `sigma-c-framework`. Los ejemplos resueltos que hacen referencia a datos de hardware cuántico están fijados a un commit específico del repositorio `magneto`. Las revisiones Git etiquetadas para ambos repositorios se listan en el Apéndice E.

Erratas. Las erratas se mantienen en `https://forgottenforge.xyz/compendium/errata` y se incorporan a las impresiones posteriores. Envíe correcciones a nfo@forgottenforge.xyz con el asunto `[compendium errata]`.

Edición y contacto. Primera edición, mayo de 2026. Buckenhof, Alemania. El autor es localizable en la dirección arriba indicada; las preguntas de cualquier profundidad son bienvenidas, y la mayoría se responden en una semana.

Texto de contraportada (edición impresa). Para revisores y catalogadores de bibliotecas, el texto de contraportada de la edición impresa se reproduce aquí textualmente.

Una receta. Tres líneas de Python que toman un sistema con un mando ajustable y una respuesta medible y le dicen dónde vuelca.

Las mismas tres líneas localizan el punto de Curie del hierro cerca de $1043\,K$ *(Capítulo 22), el régimen de volatilidad que cedió la semana en que cayó Lehman Brothers en agosto de 2008 (Capítulo 23), y la caída del valor b del catálogo sísmico del sur de California que precedió a la secuencia de Ridgecrest en julio de 2019 (Capítulo 24). Doce dominios, una receta.*

Un manual que empieza por la aritmética y termina con diagnósticos publicables. No prometemos una revolución. Prometemos un buen mayordomo.