Mecánica Estadística Avanzada
FIM8451

Montecarlo

Benjamín Loewe
Martes 20 de Agosto, 2024

Herramientas computacionales y recomendaciones

- Sistema Operativo, se recomienda Unix
  - Linux: Debian,Ubuntu, Mint, Fedora, ArchLinux
  - MacOS

- En caso de usar Windows:
  - WSL2: https://learn.microsoft.com/en-us/windows/wsl/install
  - Virtual Machine: https://www.virtualbox.org/

- Editor: Visual Studio Code (https://code.visualstudio.com/)

- Lenguaje: El que Uds. prefieran
  - Recomendado para simulaciones: C++, C (por velocidad)
  - Si ocupan Python, ocupar numpy lo más posible y evitar loops
  - Para data análisis: Python, R OK
  - Otras opciones: Julia, Rust, Fortran, etc …

- Control de versión, backup y sharing:
  - Git (https://git-scm.com)
  - GitHub (https://github.com/)

- Ayuda externa: ChatGPT, copilot, y compañeros

Métodos numéricos son indispensables en el estudio contemporáneo de la mecánica estadística

- Calcular funciones partición es extremadamente difícil en sistemas complejos

- Aunque existen aproximaciones, estas
  i. Muchas veces fallan en capturar la física del problema,
  ii. Involucran cálculos largos y tediosos
  iii. Necesitan ser verificadas

- Métodos computacionales y numéricos nos permiten estudiar sistemas complicados en gran detalle, sirviendo como una nueva forma de “laboratorio”, pero con gran flexibilidad.

- El desafio para estudiar sistemas en mecánica estadística en equilibrio, es precisamente alcanzar la condición de equilibrio.

- Existen múltiples métodos. Por ejemplo
  - MD (Molecular Dynamics)
  - Métodos continuous
  - Métodos de Montecarlo

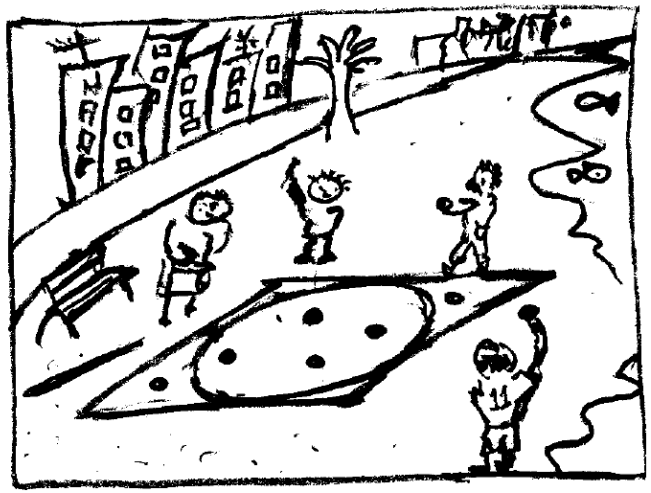
Método de Montecarlo

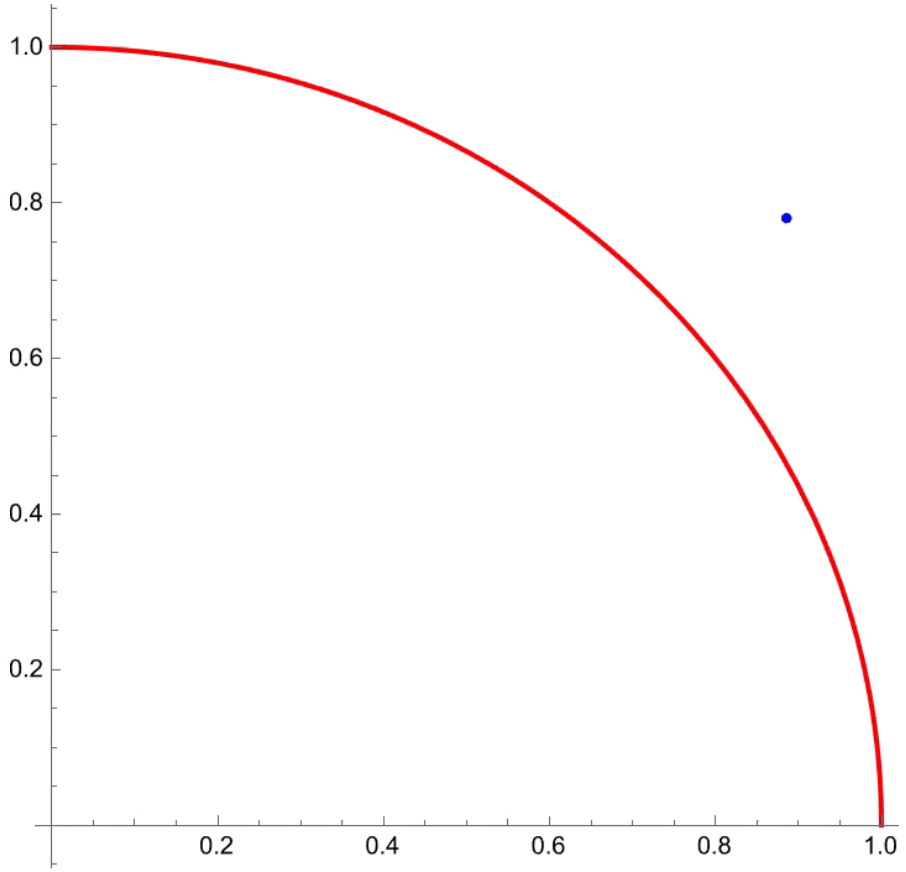
- Es una forma estadística (casi experimental) de calcular integrales.
- La idea es ocupar posiciones aleatorias (samples) cuya distribución se escoge cuidadosamente.
- Hay dos formas fundamentalmente distintas de realizar el sampling
  - Direct sampling (sampling directo)
  - Markov-chain sampling (samplin via cadena de Markov)

Direct Sampling: “Un juego de niños”

Imaginemos a niños jugando en las playas de Monaco

- Los niños dibujan un círculo y un cuadrado inscrito.
- Arrojan piedras (trials), y cuentan cuantas caen dentro del círculo (hits)
- Tomando nota del número de trials y hits, los niños realizan un calculo de direct sampling Montecarlo





Resultados del juego

La razón entre hits y trials debe acercarse a la razón entre las áreas del circulo y el cuadrado: $\pi/4$

Por ejemplo, asumiendo que los niños tiran 4000 piedras cada vez, ven que

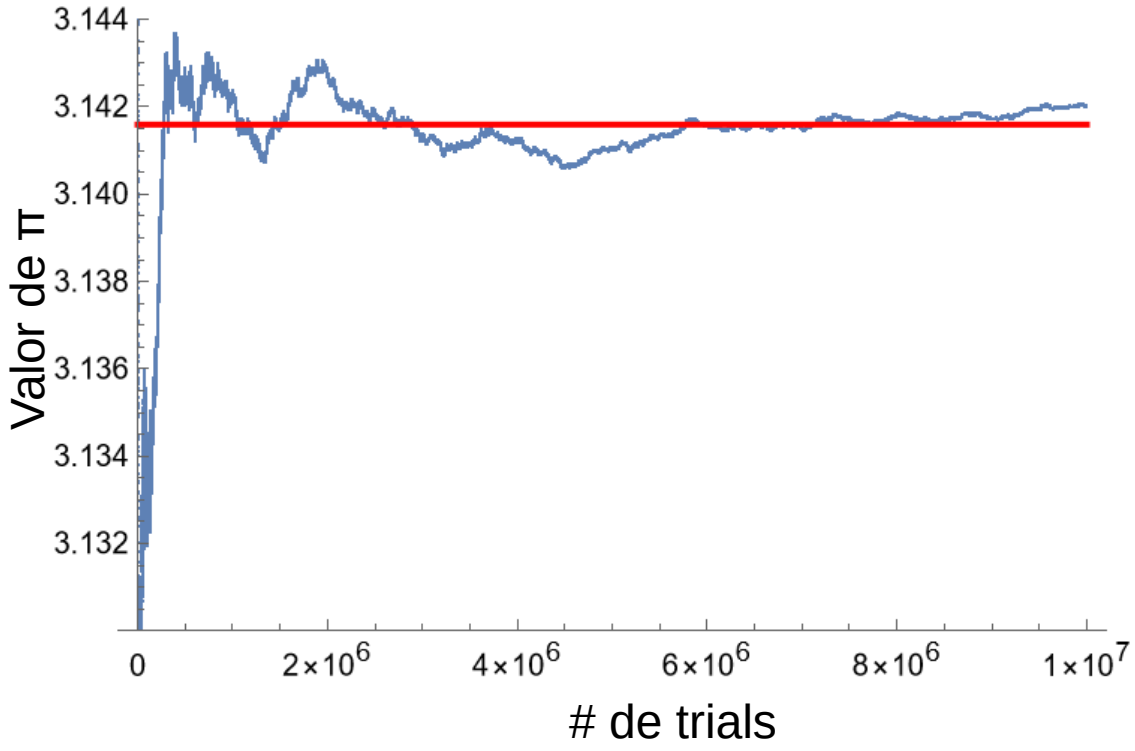
Run	N_{hits}	Estimate of π
1	3156	3.156
2	3150	3.150
3	3127	3.127
4	3171	3.171
5	3148	3.148

Una regla importante: el número de trials se decide antes de empezar.
Un cálculo estocástico no se detiene o prorroga si es que el resultado es correcto anticipadamente o no llegó a la solución que esperamos, respectivamente.

La ley de grandes números al rescate

Aunque los resultados del experimento (código adjunto) no son muy exactos, debido a la ley de grandes números se espera que se vuelva exacto en el límite de un número infinito de trials.

procedure `direct-pi`
$N_{\text{hits}} \leftarrow 0$ (initialize)
for $i = 1, \ldots, N$ **do**
$\begin{cases} x \leftarrow \texttt{ran}\,(-1, 1) \\ y \leftarrow \texttt{ran}\,(-1, 1) \\ \textbf{if } (x^2 + y^2 < 1)\ N_{\text{hits}} \leftarrow N_{\text{hits}} + 1 \end{cases}$
output N_{hits}

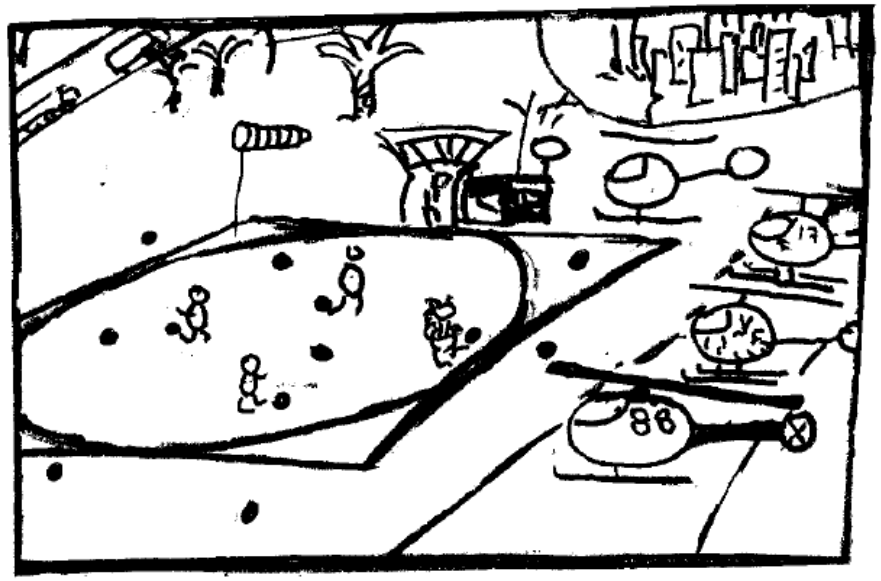


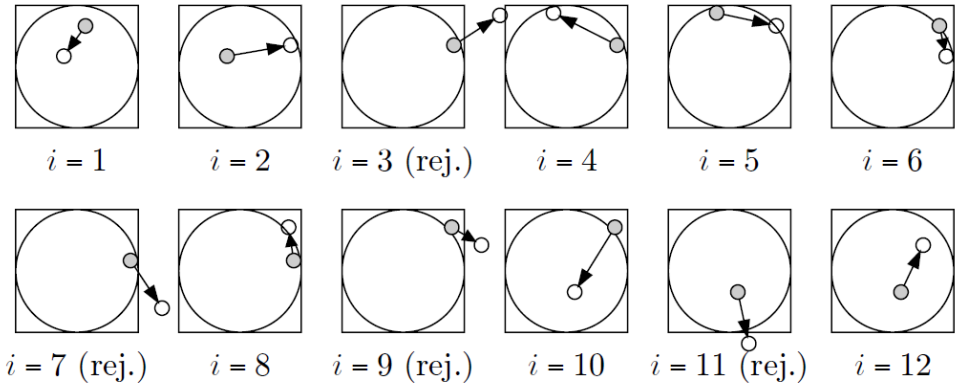
Markov-chain sampling: No solamente los niños juegan en monaco

Ahora, imaginemos un grupo de adultos jugando un juego parecido, pero a mayor escala, por lo que las reglas deben modificarse.

- Cada jugador empieza desde la casa club con un saco de piedras
- Con los ojos cerrados, arrojan la primera piedra en una dirección aleatoria, y caminan hacia donde la piedra aterrizó
- Desde esa posición, se repite el proceso con otra piedra
- Si la nueva piedra cae fuera del cuadrado, el movimiento se rechaza. Un ayudante le devuelve la piedra, la cual se coloca encima de la piedra ya existente.

El objetivo del juego es, al igual que los niños, cubrir el cuadrado del helipuerto en forma uniforme, sólo que las distancias que cada jugador puede cubrir son pequeñas comparadas con el tamaño del sistema.

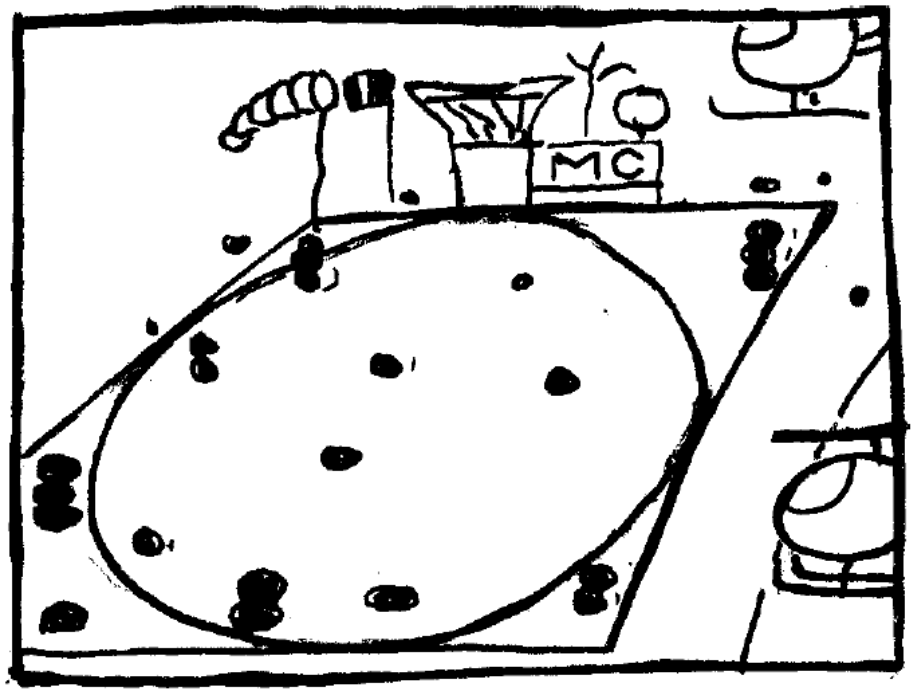




Aunque extrañas, las reglas parecen funcionar

Cuando el juego termina, hay pequeñas torres de piedras, cerca del borde del cuadrado. Aun que no es intuitivo a primera vista, la razón del número de hits con el número de trials sigue aproximando a $\pi/4$. Para 4000 tiros se registra que

Run	N_{hits}	Estimate of π
1	3123	3.123
2	3118	3.118
3	3040	3.040
4	3066	3.066
5	3263	3.263

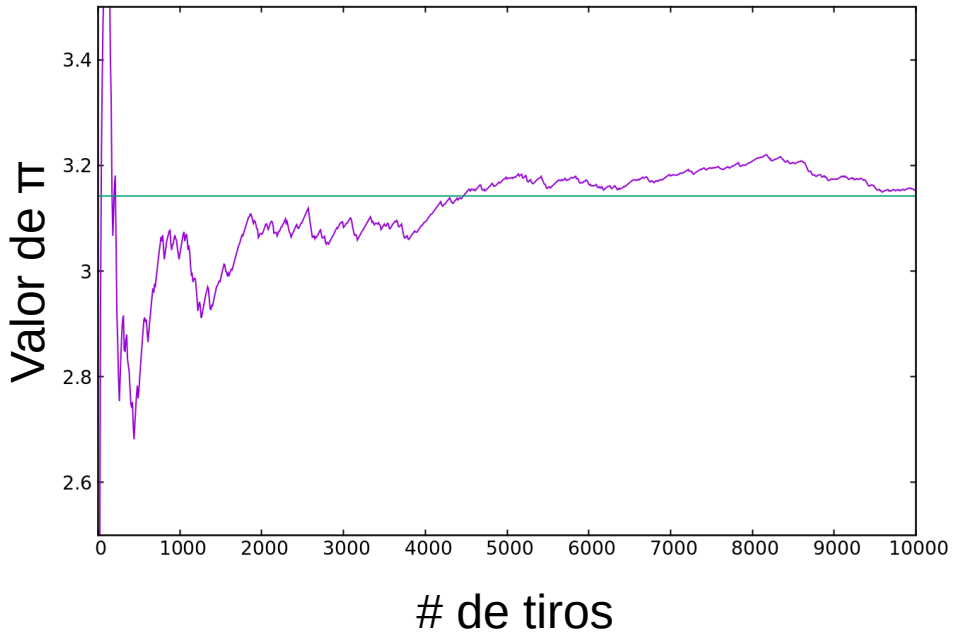


Nótese que hay detrás de esta tabla hay una variable que no hemos discutido, la fuerza con la cual se tira la piera

Un parámetro clave es la fuerza con la que se tira cada piedra

Cuantificamos la fuerza máxima de un tiro con el parámetro: δ (fijo durante el juego)

procedure markov-pi
$N_{\text{hits}} \leftarrow 0;\ \{x, y\} \leftarrow \{1, 1\}$
for $i = 1, \ldots, N$ **do**
  $\Delta_x \leftarrow \texttt{ran}\,(-\delta, \delta)$
  $\Delta_y \leftarrow \texttt{ran}\,(-\delta, \delta)$
  if $(|x + \Delta_x| < 1$ and $|y + \Delta_y| < 1)$ **then**
    $x \leftarrow x + \Delta_x$
    $y \leftarrow y + \Delta_y$
  if $(x^2 + y^2 < 1)\ N_{\text{hits}} \leftarrow N_{\text{hits}} + 1$
output N_{hits}

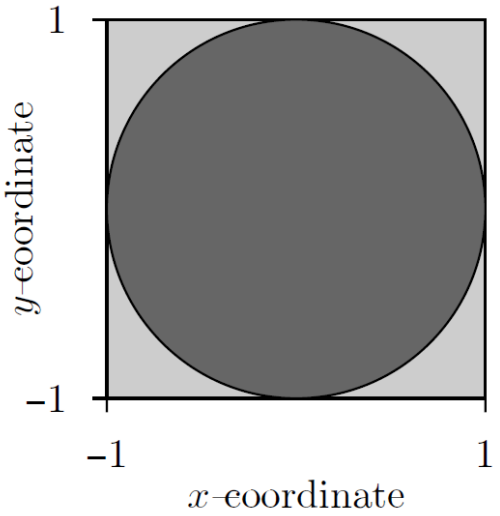


- Si δ es muy pequeño: la taza de aceptación es alta, pero el camino recorrido en cada lanzamiento es pequeño (va a tomar mucho tiempo cubrir el helipuerto)
- Si δ es muy grande, el número de rechazos también es alto, el jugador se queda fijo, no cubre el helipuerto.

En general, queremos una taza de rechazo del 50%

Con estos juegos en realidad calculan un valor de expectación.

$$\underbrace{\frac{N_{\text{hits}}}{\text{trials}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{O}_i}_{\text{sampling}} \simeq \underbrace{\langle\mathcal{O}\rangle = \frac{\int_{-1}^{1}\mathrm{d}x\ \int_{-1}^{1}\mathrm{d}y\ \pi(x,y)\mathcal{O}(x,y)}{\int_{-1}^{1}\mathrm{d}x\ \int_{-1}^{1}\mathrm{d}y\ \pi(x,y)}}_{\text{integration}}$$



$\pi = 1$ inside square, zero outside

$\mathcal{O} = 1$ inside circle, zero outside

Diferencias entre los dos métodos

Direct sampling: “Oro puro”.

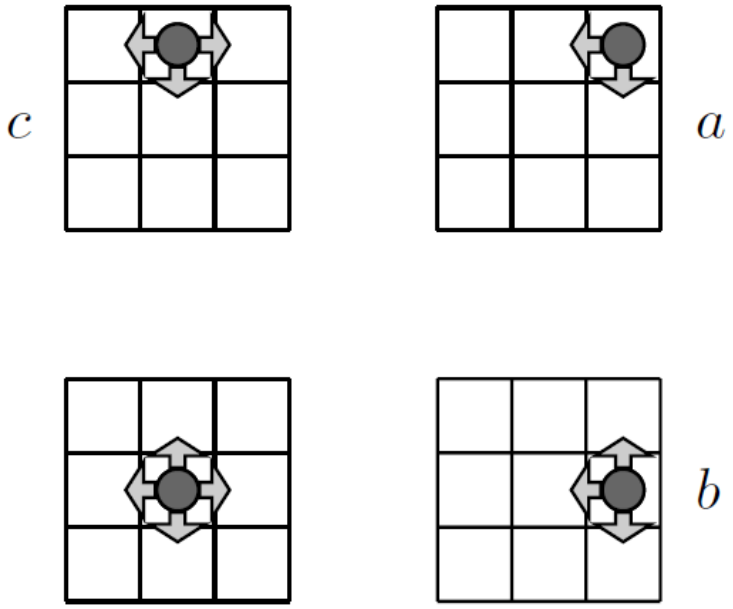
- Una subrutina provee un hit independiente a la distribución $\pi(x)$.
- Es el equivalente computacional a la solución exacta: no hay que preocuparse del rango de cada tiro, ni de condiciones iniciales y un análisis de errores trivial.

Markov-chain sampling:

- En este método hay que tener mucho más cuidado.
- El problema escencial es el tiempo de correlación, durante el cual el jugador mantiene una memoria de la posición inicial. Este tiempo puede ser muy largo. En términos prácticos, “uno se contenta con unas cuantos samplings independientes después de simulaciones que duran semanas”.

Balance detallado: el porqué de las torres de piedras

Pensemos en una versión discreta del juego del helipuerto: una grid de 3x3 con los movimientos descritos en la imagen.
Queremos un algoritmo que después de varias iteraciones nos de una probabilidad uniforme



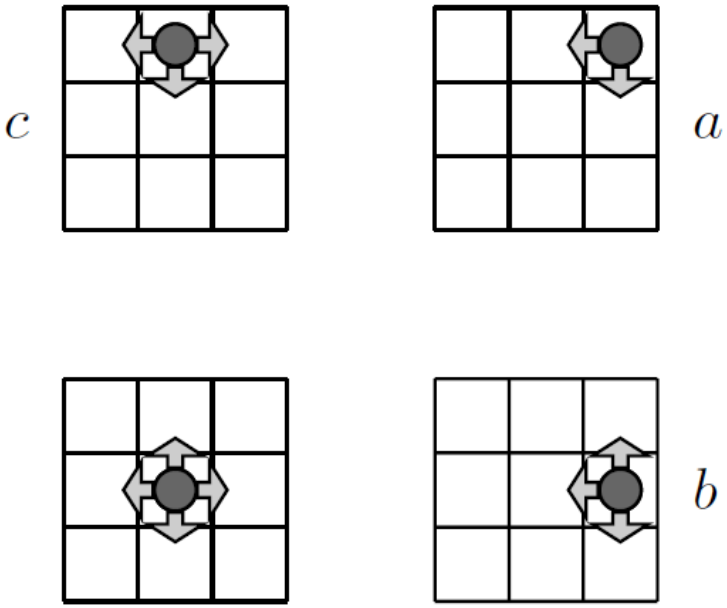
$$\{\pi(a), \pi(b), \ldots\} : \left\{ \begin{array}{c} \text{stationary probability} \\ \text{for the system to be at } a, b, \text{ etc.} \end{array} \right\}$$

$$\{p(a \to b), p(a \to c), \ldots\} : \left\{ \begin{array}{c} \text{probability of the algorithm} \\ \text{to move from } a \text{ to } b, \text{ etc.} \end{array} \right\}$$

$$p(a \to a) + p(a \to b) + p(a \to c) = 1$$

$$\pi(a) = \pi(b)p(b \to a) + \pi(c)p(c \to a) + \pi(a)p(a \to a)$$

La condición de balance detallado



$$\pi(a)[1 - p(a \to a)] = \pi(b)p(b \to a) + \pi(c)p(c \to a)$$

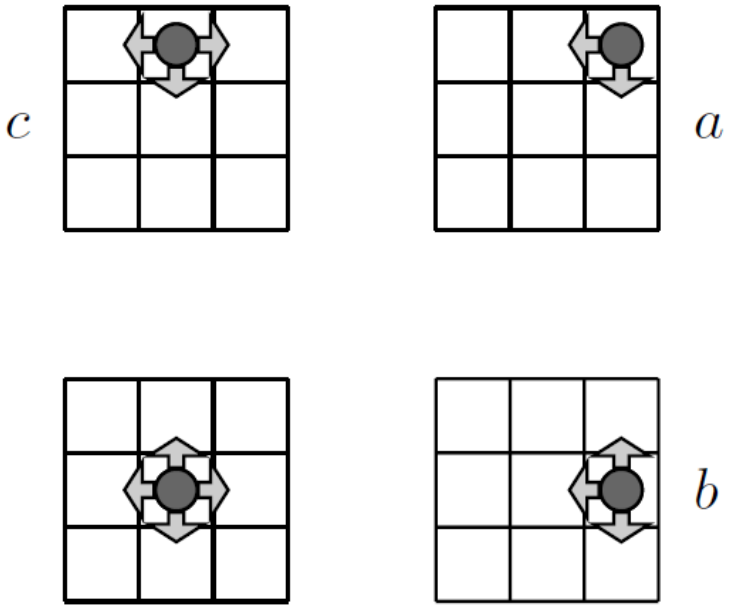
$$1 - p(a \to a) = p(a \to b) + p(a \to c)$$

$$\pi(a)\,p(a \to b) + \pi(a)\,\overbrace{p(a \to c) = \pi(c)}\,p(c \to a) + \pi(b)\,p(b \to a)$$

$$\left\{ \begin{array}{c} \text{detailed} \\ \text{balance} \end{array} \right\} : \begin{array}{l} \pi(a)p(a \to b) = \pi(b)p(b \to a) \\ \pi(a)p(a \to c) = \pi(c)p(c \to a) \end{array}$$

Esta condición de tazas permite que el juego sea consistente.

La condición de balance detallado



$$\left\{\begin{array}{l}\text{detailed}\\ \text{balance}\end{array}\right\}: \quad \begin{array}{l}\pi(a)p(a\to b)=\pi(b)p(b\to a)\\ \pi(a)p(a\to c)=\pi(c)p(c\to a)\end{array}$$

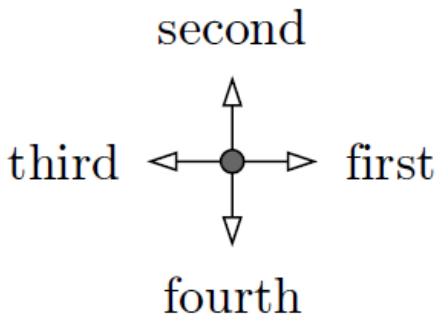
En el juego de las piedras, el balance detallado se mantiene debido a que:
i) Queremos $\pi(a)=\pi(b)=\pi(c)$

ii) La probabilidad de moverse a sitios vecinos es ¼, y la probabilidad p(a→b) es trivialmente igual a p(b→a).

iii) Una esquina (borde) solamente tiene 2 (3), vecinos, por lo que solamente 2/4 (3/4) de las veces nos podemos mover, y el resto de las veces, 2/4 (1/4), nos debemos quedar estacionarios, (p(a→a)) construyendo una pila.

Usando una tabla de vecinos podemos calcular la matriz de transferencia

7	8	9
4	5	6
1	2	3



$$P = \{p(a \to b)\} = \begin{bmatrix} p(1 \to 1) & p(2 \to 1) & p(3 \to 1) & \dots \\ p(1 \to 2) & p(2 \to 2) & p(3 \to 2) & \dots \\ p(1 \to 3) & p(2 \to 3) & p(3 \to 3) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Site k	Nbr$(\dots, k)$ 1	2	3	4
1	2	4	0	0
2	3	5	1	0
3	0	6	2	0
4	5	7	0	1
5	6	8	4	2
6	0	9	5	3
7	8	0	0	4
8	9	0	7	5
9	0	0	8	6

$$\{p(a \to b)\} = \begin{bmatrix} \boxed{\frac{1}{2}} & \frac{1}{4} & . & \frac{1}{4} & . & . & . & . & . \\ \frac{1}{4} & \boxed{\frac{1}{4}} & \frac{1}{4} & . & \frac{1}{4} & . & . & . & . \\ . & \frac{1}{4} & \boxed{\frac{1}{2}} & . & . & \frac{1}{4} & . & . & . \\ \frac{1}{4} & . & . & \boxed{\frac{1}{4}} & \frac{1}{4} & . & \frac{1}{4} & . & . \\ . & \frac{1}{4} & . & \frac{1}{4} & \boxed{0} & \frac{1}{4} & . & \frac{1}{4} & . \\ . & . & \frac{1}{4} & . & \frac{1}{4} & \boxed{\frac{1}{4}} & . & . & \frac{1}{4} \\ . & . & . & \frac{1}{4} & . & . & \boxed{\frac{1}{2}} & \frac{1}{4} & \\ . & . & . & . & \frac{1}{4} & . & \frac{1}{4} & \boxed{\frac{1}{4}} & \frac{1}{4} \\ . & . & . & . & . & \frac{1}{4} & . & \frac{1}{4} & \boxed{\frac{1}{2}} \end{bmatrix}$$

Hay que diferenciar entre la probabilidad estacionaria $\pi(a)$, la probabilidad estacionaria, y el sampling en una iteración i del juego de Montecarlo $\pi^i(a)$.

$$\pi^{i+1}(a) = \sum_{b=1}^{9} p(b \to a)\pi^i(b)$$

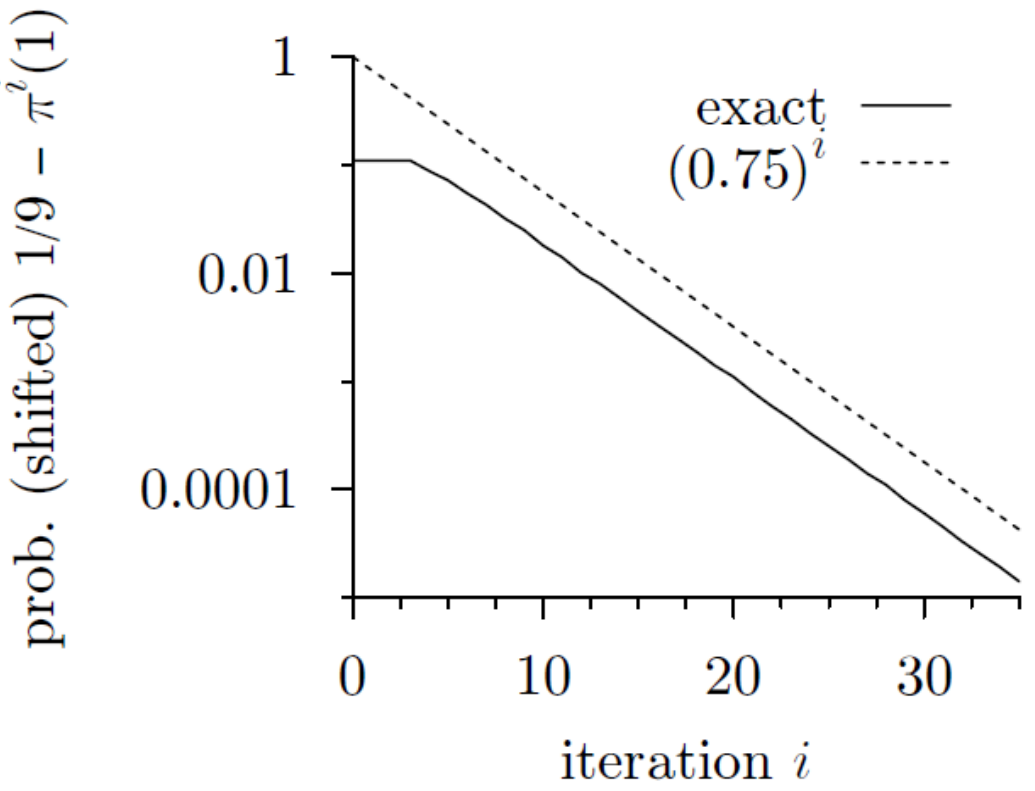
Debemos imaginarnos una simulación de Montecarlo como si fuese realizada sobre un gran número de helipuertos, cada uno con su propio jugador. Esto permite hacer sentido del concepto de “probabilidad de estar en la configuración a en la iteración i”.

Esto nos permite ver como evoluciona con cada iteración la distribución.

$$\{\pi^0(1), \ldots, \pi^0(9)\} = \{0, \ldots, 0, 1\}$$ ← Condición inicial

$$\pi^{i+1}(a) = \sum_{b=1}^{9} p(b \to a)\pi^i(b)$$ ← Evolución con cada iteración

Prob.	Iteration i				
	0	1	2	...	∞
$\pi^i(1)$	0	0	0	...	1/9
$\pi^i(2)$	0	0	0	...	1/9
$\pi^i(3)$	0	0	0.062	...	1/9
$\pi^i(4)$	0	0	0	...	1/9
$\pi^i(5)$	0	0	1/8	...	1/9
$\pi^i(6)$	0	1/4	0.188	...	1/9
$\pi^i(7)$	0	0	0.062	...	1/9
$\pi^i(8)$	0	1/4	0.188	...	1/9
$\pi^i(9)$	1	1/2	0.375	...	1/9



Para estudiar esto en más detalle, hacemos un análisis de los autovetores/autovalores

$$P\boldsymbol{\pi}_e^k = \lambda_k \boldsymbol{\pi}_e^k$$

$$\boldsymbol{\pi} = \alpha_1 \boldsymbol{\pi}_e^1 + \alpha_2 \boldsymbol{\pi}_e^2 + \cdots + \alpha_9 \boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k \boldsymbol{\pi}_e^k$$

$$P^i \boldsymbol{\pi} = \alpha_1 \lambda_1^i \boldsymbol{\pi}_e^1 + \alpha_2 \lambda_2^i \boldsymbol{\pi}_e^2 + \cdots + \alpha_9 \lambda_9^i \boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k (\lambda_k)^i \boldsymbol{\pi}_e^k$$

$$\{\pi^i(1), \ldots, \pi^i(9)\} = \underbrace{\{\tfrac{1}{9}, \ldots, \tfrac{1}{9}\}}_{\substack{\text{first eigenvector} \\ \text{eigenvalue } \lambda_1 = 1}} + \alpha_2 \cdot (0.75)^i \underbrace{\{-0.21, \ldots, 0.21\}}_{\substack{\text{second eigenvector} \\ \text{eigenvalue } \lambda_2 = 0.75}} + \cdots$$

La matriz P de transferencia sólo tiene un autovector con todas las componentes positivas, y por lo tanto, un solo autovector que puede representar un vector de probabilidades. Su autovalor es 1.

Todos los demás autovalores son menores a 1.

Para estudiar esto en más detalle, hacemos un análisis de los autovetores/autovalores

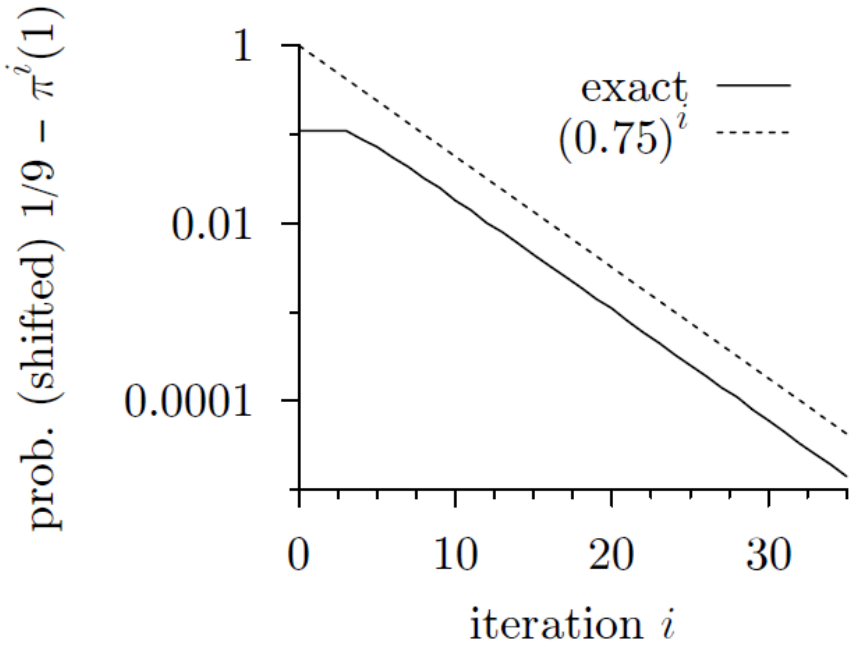
$$\{\pi^i(1), \ldots, \pi^i(9)\} = \underbrace{\{\tfrac{1}{9}, \ldots, \tfrac{1}{9}\}}_{\substack{\text{first eigenvector} \\ \text{eigenvalue } \lambda_1 = 1}} + \alpha_2 \cdot (0.75)^i \underbrace{\{-0.21, \ldots, 0.21\}}_{\substack{\text{second eigenvector} \\ \text{eigenvalue } \lambda_2 = 0.75}} + \cdots$$

Luego, cuando i → ∞, solamente el primer autovector sobrevive. La diferencia entre este limite asintótico y la iteración i es de orden

$$(0.75)^i = \mathrm{e}^{i \cdot \log 0.75} = \exp\left(-\frac{i}{3.476}\right) \propto \mathrm{e}^{-i/\Delta_i}$$

$$\Delta_i = 3.476$$

Es una escala de tiempo que determina si un número de iteraciones es grande o pequeño.



Algoritmo Metrópolis: Cuando no todas las configuraciones son equiprobables

$$p(a \to b) = \min\left[1, \frac{\pi(b)}{\pi(a)}\right]$$

Esto satisface el balance detallado

Case	$\pi(a) > \pi(b)$	$\pi(b) > \pi(a)$
$p(a \to b)$	$\pi(b)/\pi(a)$	1
$\pi(a)p(a \to b)$	$\pi(b)$	$\pi(a)$
$p(b \to a)$	1	$\pi(a)/\pi(b)$
$\pi(b)p(b \to a)$	$\pi(b)$	$\pi(a)$

Para el helipuerto

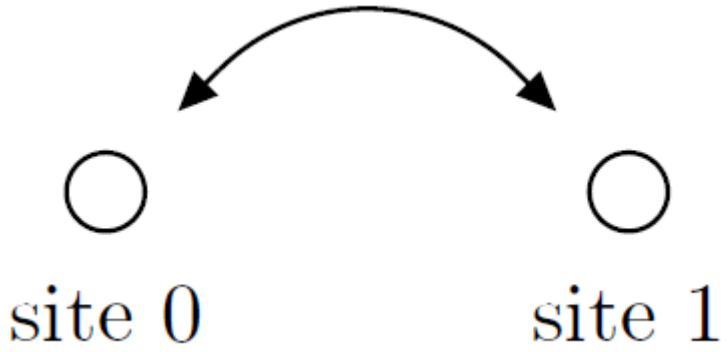
a y b adentro: el movimiento siempre se acepta

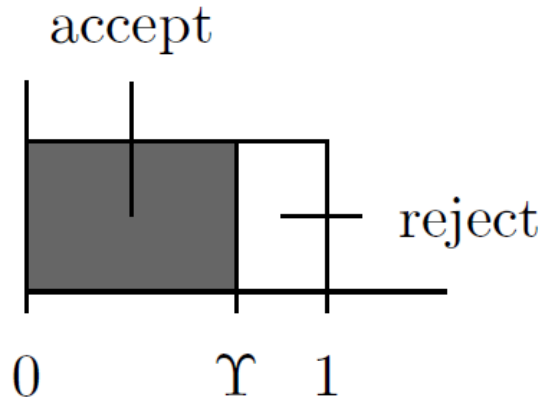
$$\pi(b)/\pi(a) = 1,\ p(a \to b) = 1$$

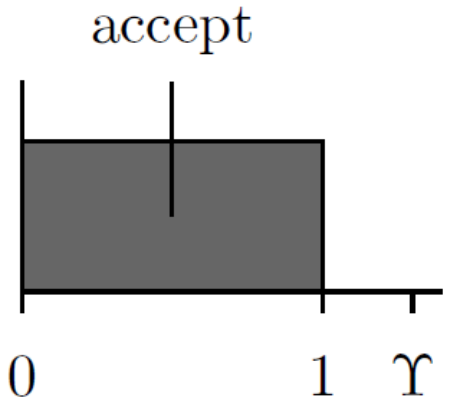
a adentro, b afuera: el movimiento se rechaza

$$\pi(b)/\pi(a) = 0,\ p(a \to b) = 0$$

Ejemplo: El problema de 2 sitios







Las probabilidades de estar en 0 o en 1 son proporcionales a dos números positivos arbitrarios: $\pi(0)$ y $\pi(1)$

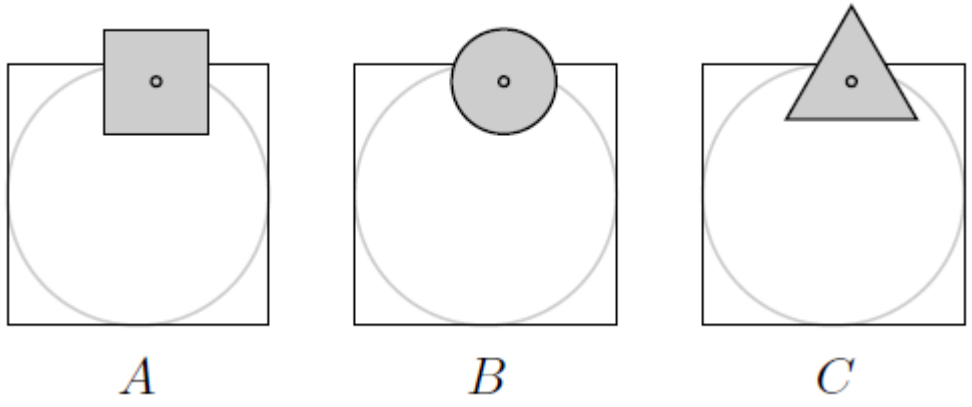
Idea: comparar la razón de pesos estadísticos Υ

$$\pi(1)/\pi(0) \text{ or } \pi(0)/\pi(1)$$

Con un numero aleatorio entre 0 y 1

procedure `markov-two-site`
input k (either 0 or 1)
if $(k = 0)$ $l \leftarrow 1$
if $(k = 1)$ $l \leftarrow 0$
$\Upsilon \leftarrow \pi(l)/\pi(k)$
if $(\texttt{ran}(0,1) < \Upsilon)$ $k \leftarrow l$
output k (next site)

Prioridades a priori (no simétricas): Algoritmo de Metropolis-Hasting



$$\mathcal{P}(a \to b) = \underbrace{\mathcal{A}(a \to b)}_{\text{consider } a \to b} \cdot \underbrace{p(a \to b)}_{\text{accept } a \to b}$$

$$\pi(a)\mathcal{P}(a \to b) = \pi(b)\mathcal{P}(b \to a)$$

$$\frac{p(a \to b)}{p(b \to a)} = \frac{\pi(b)}{\mathcal{A}(a \to b)} \frac{\mathcal{A}(b \to a)}{\pi(a)}$$

En el helipuerto, limitábamos los lanzamientos a un cuadrado de lado 2δ.
Esto es un ejemplo de una distribución de probabilidad a priori

$$\mathcal{A}(a \to b)$$

La cual utilizamos para “samplear” el movimiento a → b.
El balance detallado se puede establecer independientemente de la forma de A

$$p(a \to b) = \min\left[1, \frac{\pi(b)}{\mathcal{A}(a \to b)} \frac{\mathcal{A}(b \to a)}{\pi(a)}\right]$$

Estimar errores en el método de cadenas de Markov es más complicado que en el caso de Direct Sampling

- Después de correr 5 juegos de 4000 piedras, tenemos 20000 piedras distribuidas en el helipuerto.
- Sin embargo, estas no son independientes: tienden a estar agrupadas (incluso apiladas)
- Por lo tanto, lo que podemos aprender de $\pi(x, y)$ con este sampleo es menos detallado de lo que pudiésemos aprender si los lanzamientos fueran independientes (direct sampling)

Run	N_{hits}	Estimate of π
1	3156	3.156
2	3150	3.150
3	3127	3.127
4	3171	3.171
5	3148	3.148

Direct Samspling

Run	N_{hits}	Estimate of π
1	3123	3.123
2	3118	3.118
3	3040	3.040
4	3066	3.066
5	3263	3.263

Markov

- La anchura de la distribución de runniing averages es más grande

Una solución ingenua

- En vez de lidiar con las 5x4000 piedras, podríamos considerar los 5 running averages de $4\ {}^{N_{hits}}/_{N}$ como variables independientes

Run	N_{hits}	Estimate of π
1	3123	
2	3118	3.122
3	3040	$\pm$
4	3066	0.04
5	3263	

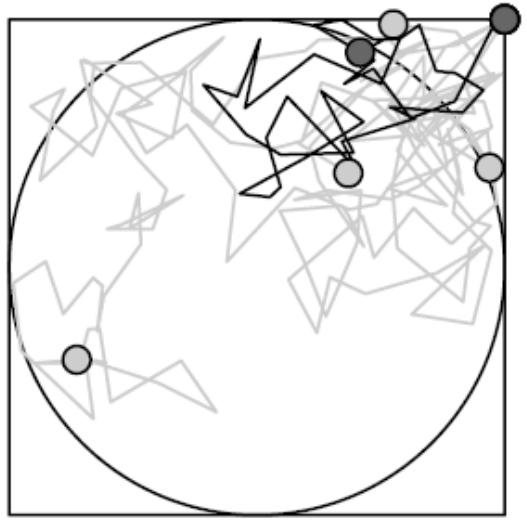
El problema con esta estrategia:

- La elección del punto de partida produce bias, por lo que queremos que cada run sea lo más largo posible
- Por otra parte, queremos un gran número de runs para reducir la incerteza en el error

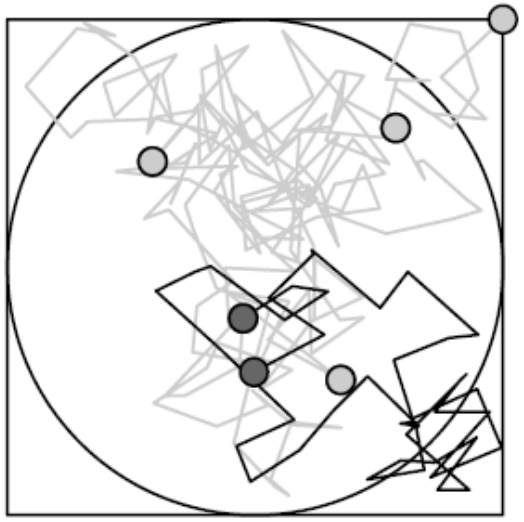
Con esta estrategia, no es claro como optimizar estos dos aspectos contradictorios.

Una solución más astuta: Data Bunching

Otra solución sería considerar que, en vez de tener a todos los jugadores partiendo desde la misma posición (el clubhouse), deberíamos permitir a una cadena empezar donde la anterior terminó. Esto nos deja una sola cadena mucho más larga



Misma posición inicial



Empezar donde la anterior terminó

Dado esto, ¿por qué limitarnos a dividir la cadena en 5 partes?

Podríamos en vez producir “bunches” de tamaño {2, 4, 8, …. } y considerar estos bunches como independientes.

Agrupando en pares, es fácil codificar el procedimiento

procedure `data-bunch`
input $\{x_1, \ldots, x_{2N}\}$ (Markov-chain data)
$\Sigma \leftarrow 0$
$\Sigma' \leftarrow 0$
for $i = 1, \ldots, N$ **do**

$$\begin{cases} \Sigma \leftarrow \Sigma + x_{2i-1} + x_{2i} \\ \Sigma' \leftarrow \Sigma' + x_{2i-1}^2 + x_{2i}^2 \\ x'_i \leftarrow (x_{2i-1} + x_{2i})/2 \end{cases}$$

$\text{error} \leftarrow \sqrt{\Sigma'/(2N) - (\Sigma/(2N))^2}/\sqrt{2N}$
output $\Sigma/(2N), \text{error}, \{x'_1, \ldots, x'_N\}$

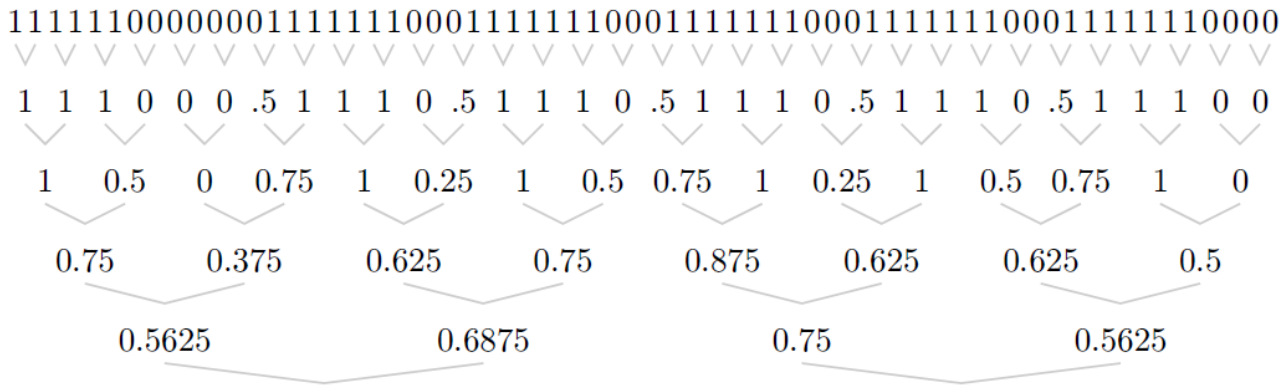


Ilustración del algoritmo

La data proveniente de los bunches está menos correlacionada

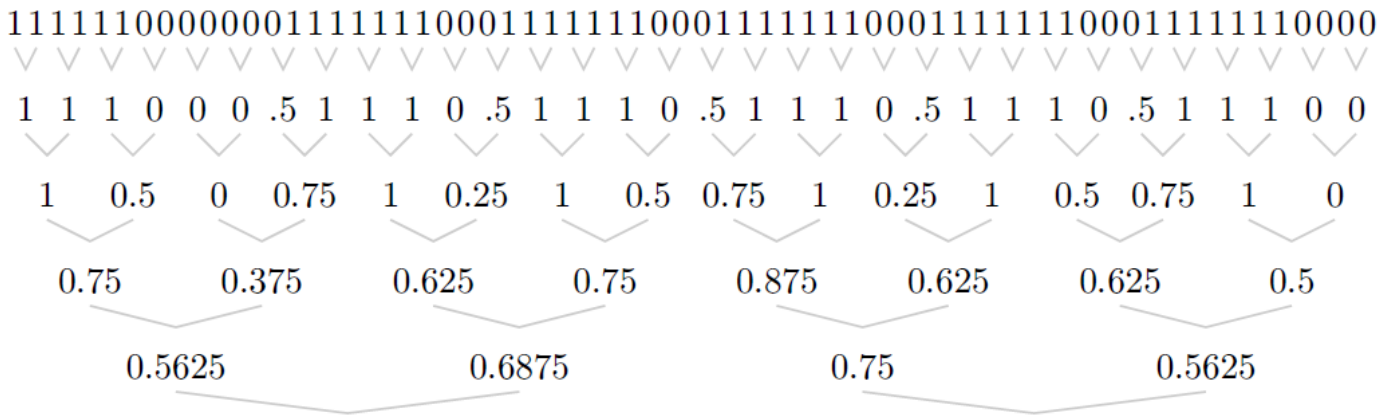
- Supongamos que estamos en la k-esima iteración del algoritmo.

  Luego tenemos bunches de tamaño 2^k.

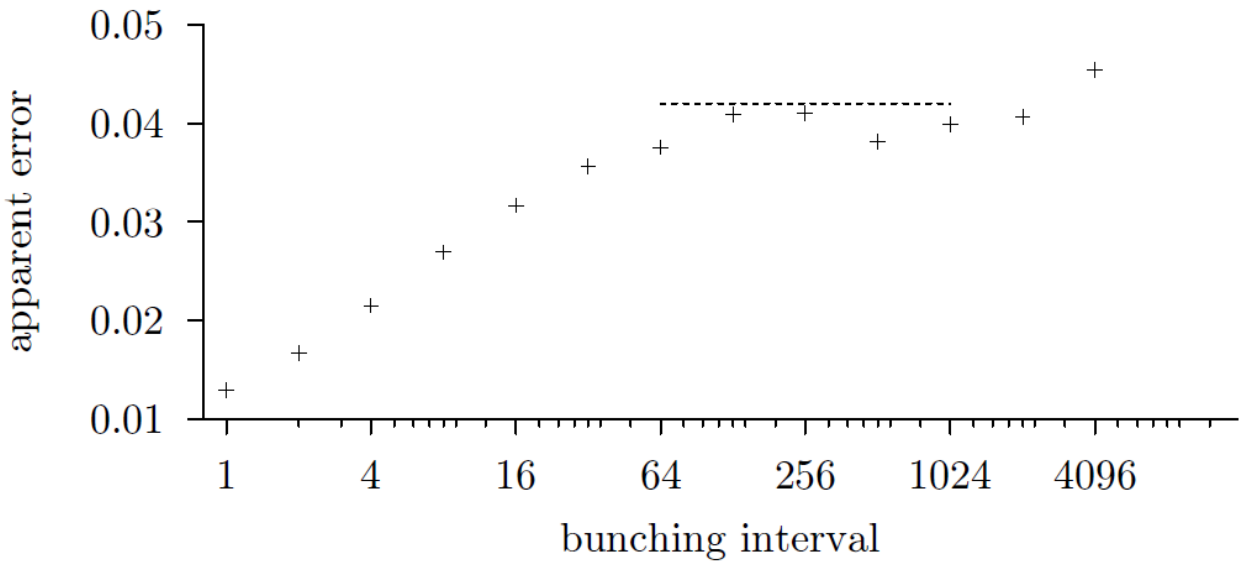
- Supongamos que la data está correlacionada en una lenghtscale $\xi \lesssim 2^k$, y que la data original separada por más de 2^k es completamente independiente

- Luego, al nivel k, las correlaciones afectan a los bunches vecinos, pero no a los “next-nearest”.

- En practica, el largo de correlación se redujo de 2^k a $\lesssim 2$.



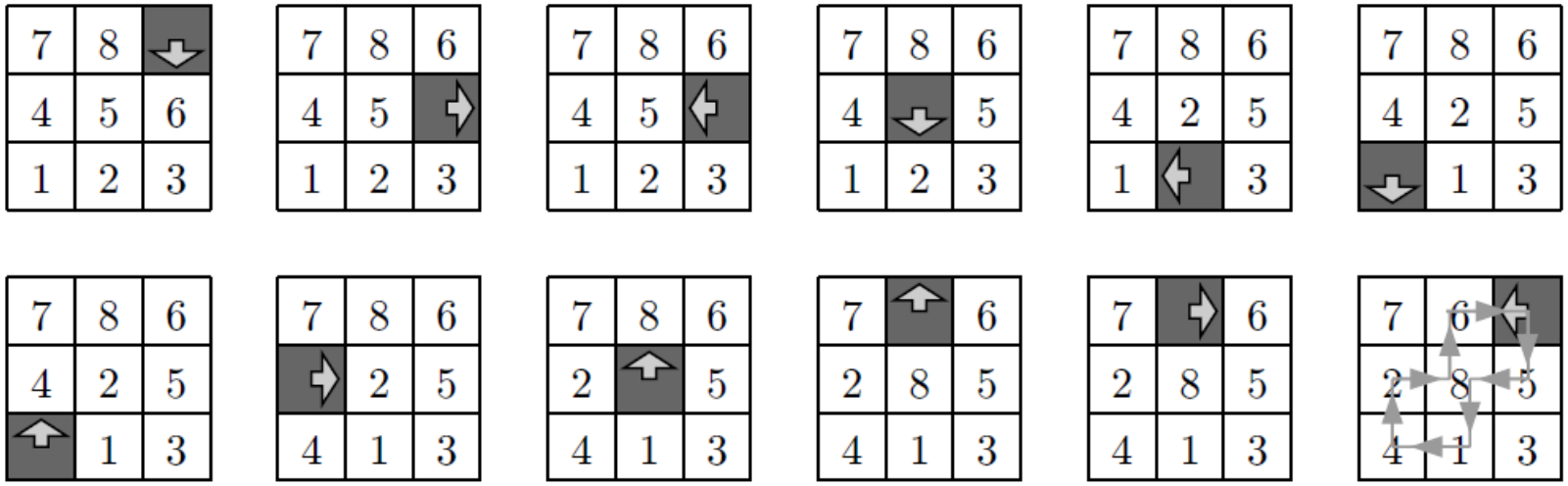
Podemos entonces analizar cómo evoluciona el error de acuerdo al tamaño del bunch

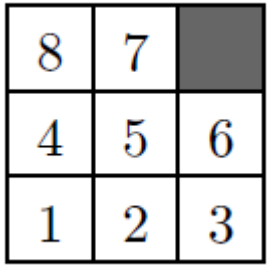


- Para bunching menores que el tiempo de correlación (aprox. 64), el error está bajo estimado.
- Para bunchings intermedios aparece una meseta característica, indicando el verdadero error de la simulación
- Finalmente, para bunches muy grandes el error es sobrestimado debido a que, aunque lidiamos con data no correlacionada, hay muy pocos datos para promediar, lo que agrega ruido a nuestra data.

Limitaciones del método de Montecarlo: Violaciones a la ergodicidad

- La limitación más común de una cadena de Markov es su lenta convergencia: el programa tiene grandes dificultades en trata de des correlacionarse de su condición inicial.
- En el peor de los casos, incluso en el limite de tiempo de computación infinito no se obtienen samples independientes.
- Cuando esto ocurre hablamos de un algoritmo no ergódico.





Dada la condición inicial, esta configuración nunca puede ser alcanzada.

Comentario sobre números aleatorios

- Un computador es incapaz de producir verdaderos números aleatorios.
- Lo que entrega, son números pseudo-aleatorios, los cuales se obtienen de seguir un algoritmo dada una semilla (o seed).
- La calidad de los números aleatorios depende entonces del tipo de algoritmo utilizado. En particular, es usual que los métodos estándar incluidos en los lenguajes de programación (e.g., rand() en C) sean rápidos de ejecutar, pero de baja calidad (e.g., fallan test estadísticos de independencia, pueden tener periodos cortos o pueden volverse predicibles).
- El método estándar en ciencia es Mersenne Twister. Es parte de la librería random de C++:

```
#include <random>
std::mt19937_64 dre;
std::uniform_int_distribution<> init_dist(0,1);
std::uniform_real_distribution<> real_dist(0,1);
```

← Generador de números aleatorios con Mersenne Twister

Funciones que ocupan el generador para obtener enteror o reales de una distribución uniforme

Bibliografía

Werner Krauth

Statistical Mechanics:
Algorithms and Computations

Capítulo 1