

# Modelo de Ising

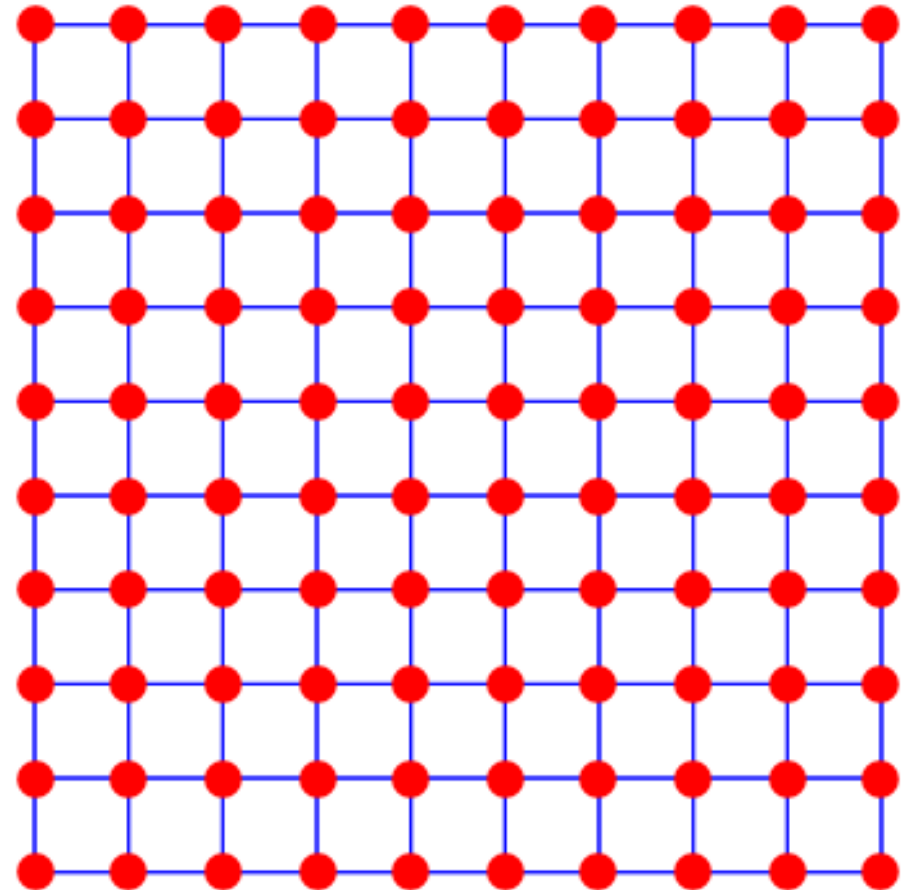


1D

$$-J \sum_{\langle i,j \rangle} S_z^i S_z^j - \mu B \sum_i S_z^i$$

Interacción a  
primeros vecinos

Campo externo



2D

# Ensamble canónico

$$Z_C = \sum_{\substack{\text{estados} \\ L \times L \text{ espines}}} e^{-\beta E_n}$$

$$U = \frac{1}{Z_C} \sum_{\substack{\text{estados} \\ L \times L \text{ espines}}} E_n e^{-\beta E_n}$$

$$U = \sum_{\substack{\text{estados} \\ L \times L \text{ espines}}} E_n p_n$$

$$p_n = \frac{1}{Z_C} e^{-\beta E_n}$$

**Probabilidad**

Idea: Si tengo configuraciones (o sea estados!) que tengan la distribución de probabilidad adecuada

$$\langle O \rangle = \sum_n p_n \langle n | O | n \rangle \simeq \frac{1}{n_\alpha} \sum_{\substack{n_\alpha \text{ estados} \\ \text{representativos}}} \langle \alpha | O | \alpha \rangle$$

**SAMPLING**

# Metropolis Monte Carlo


- Monte Carlo es un nombre genérico para hacer sampling.
- En los métodos Monte Carlo de cadena de Markov, como Metropolis!. Un estado sigue al otro, y depende solo del anterior. Las transiciones se contruyen de manera que en el estado estacionario (luego de muchas repeticiones) las configuraciones provienen de la distribucion deseada  $p_n$

# Algoritmo de Metropolis

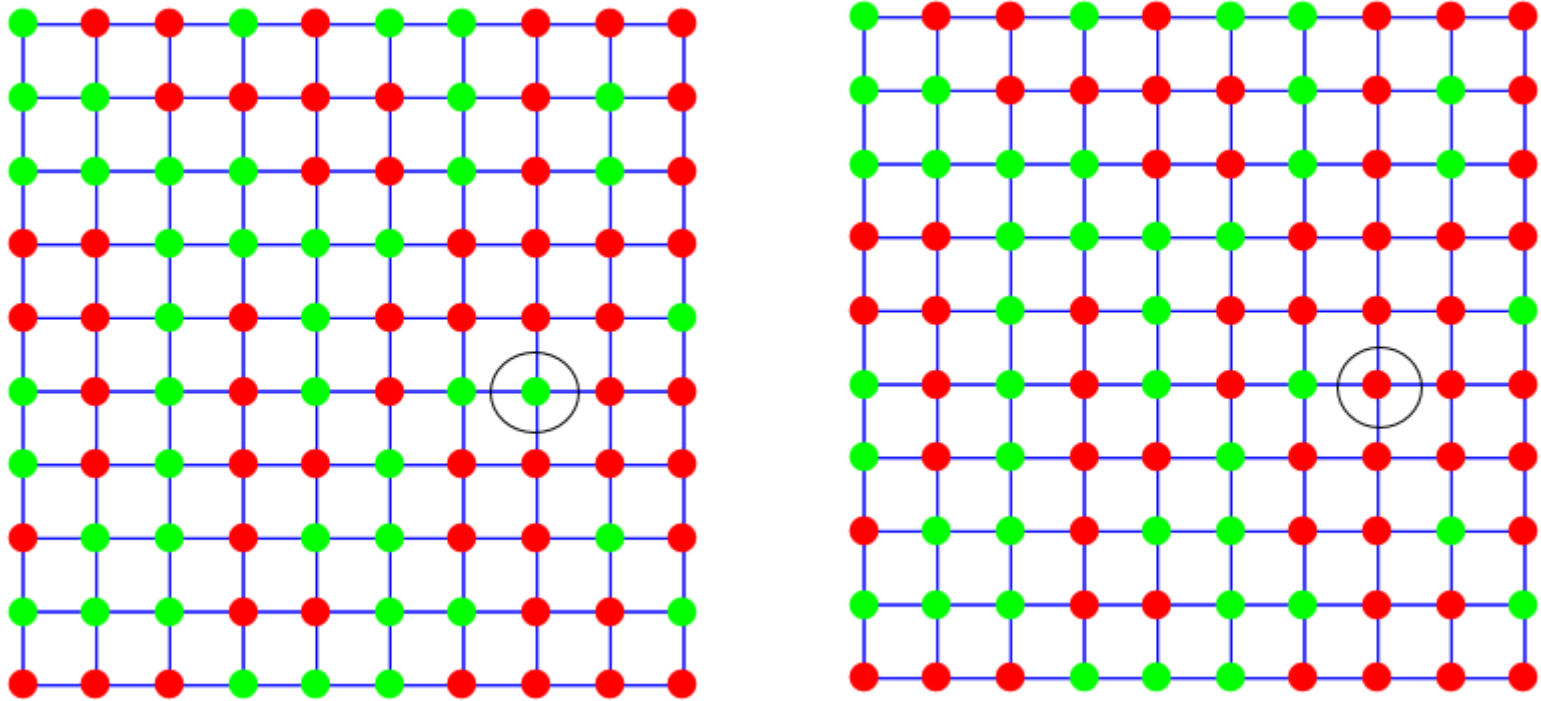
1. Elijo un estado posible  $x$ , para iniciar la cadena
2. Ciclo
  - i. Genero un estado distinto  $x'$  (con alguna dist.)
  - ii. Calculo  $a = \text{Min}\{ 1, p(x')/p(x)\}$
  - iii. Con probabilidad  $a$  acepto el cambio  $x \rightarrow x'$
  - iv. Guardo  $x'$  (si acepté) o repito  $x$

**NB:**  $p(x)$  no necesita estar normalizada!!!

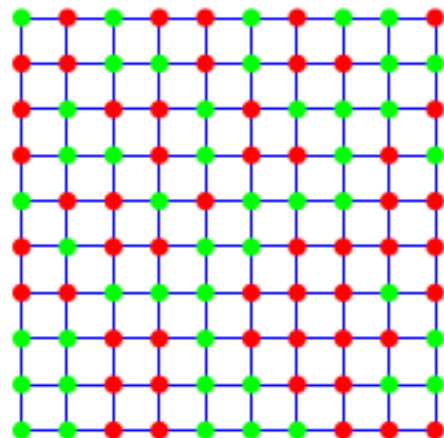
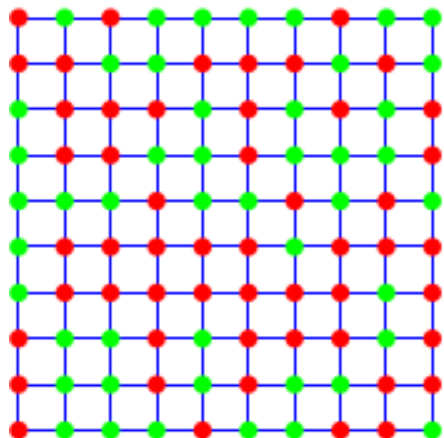
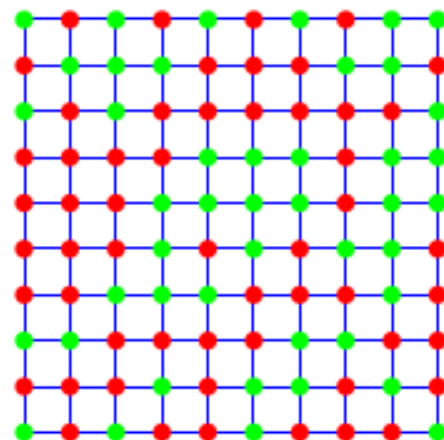
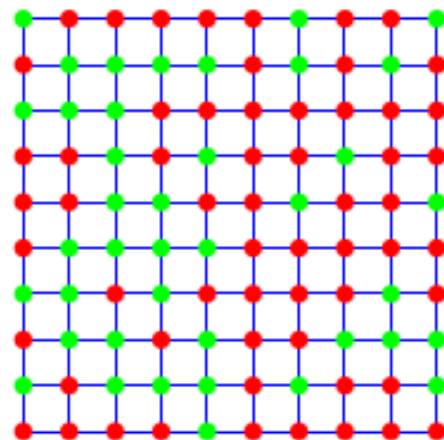
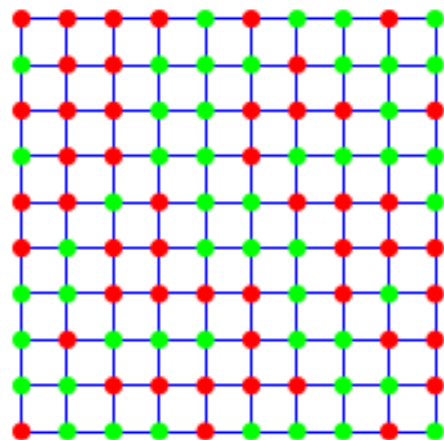
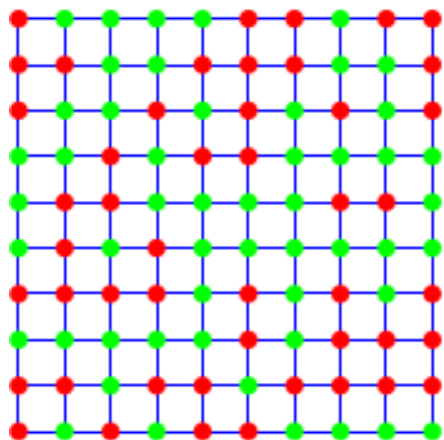
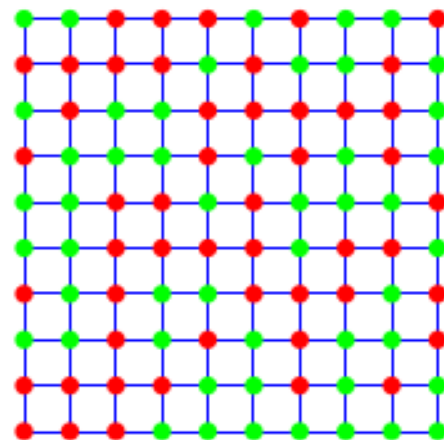
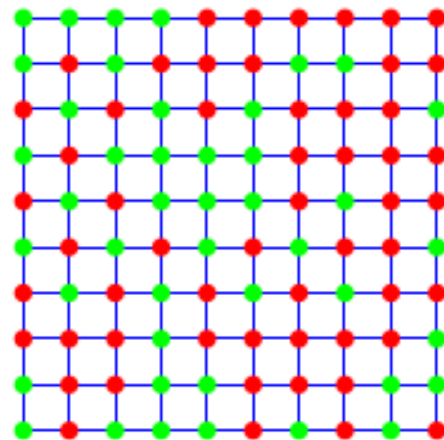
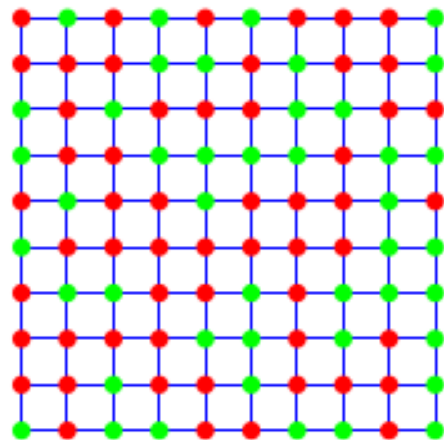
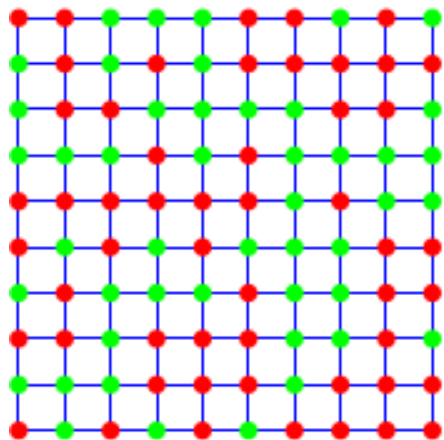
# Sampling ising: aceptamos y rechazamos

1. Partimos de una configuración inicial (como elegirla?)
  2. Proponemos un cambio de un espín de la red (flip)
    - i. La energía baja? Genial lo aceptamos.
    - ii. La energía sube? Lo aceptamos con probabilidad  $e^{-\beta \Delta E}$
    - iii. Repetimos para todos los espines de la red.
  3. Volvemos a recorrer toda la red
  4. Para los estados de la cadena, calculamos los valores de las magnitudes que estemos interesados.
- 

# Ejemplo



Flip



Cadena de Markov  
Estados representativos

# Como lo hacemos?

- Si sabes programar en C, C++, Fortran, Python o algun otro lenguaje, mejor es más rapido.
- Si no, (y no quieres aprender..) puedes usar entornos como matlab, octave, etc.
- Luego para presentar los resultados usas lo que preferis (word+origin, latex+octave/grace)
- Aca vamos a dar una brevisima introducción a Octave (o matlab).



# Octave / Matlab

**Algunas ayudas básicas.** (el comando help es tu amigo)

- Definimos variables (matrices en general)
- Las mayúsculas y minúsculas son distintas, e.g.  $A \neq a$  .
- ; al final de la instrucción hace que no se muestre el resultado (pero se calcula!).
- Rangos: e.g., vector  $a = 0, 0.1, 0.2, \dots, 1$   
 $a = 0:0.1:1$ ; genera un vector desde 0 a 1 con paso 0.1; Es lo mismo que decir  $a=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$
- Si A es una matriz de 2x2, los elementos son A(1,1), A(1,2), A(2,1),A(2,2)
- Un bucle/ciclo (do o for en otros lenguajes) y un condicional tienen una estructura común.

```
for i = 1:10
    Hago cosas...
end
```

```
if (condicion)
    Hago cosas...
end
```

Operadores:

`==` igual

`~=` distinto

`>`, `<`, `>=`, `<=`

- Un conjunto de instrucciones se pueden poner en un archivo tal cual como lo ejecutaría, eso es un script, el archivo debe llamarse “micalculo.m” y se ejecutan escribiendo “micalculo”
- En los scripts puedo poner comentarios que no se ejecutan, para eso utilizo el %
- También puedo definir funciones en archivos separados .m,

```
function A=mifuncion(a,b)
Hago cosas y asigno A
end %no es necesario siempre
```

Probemos Octave viendo los  
generadores de numeros aleatorios