Tarea # 3. Parte 2.
2D Ising Model: Monte Carlo Metropolis.

Johans Restrepo Cárdenas

Instituto de Física. Universidad de Antioquia.

27 de abril de 2020

Local Metropolis Algorithm

Corte y pegue el siguiente algoritmo para el modelo de Ising que se muestra a continuación explicando en su informe las partes del programa:

```
import random, math

def energy(S, N, nbr):
    E = 0.0
    for k in range(N):
        E -=  S[k] * sum(S[nn] for nn in nbr[k])
    return 0.5 * E

L = 6
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                    for i in range(N)}

T = 2.0
S = [random.choice([1, -1]) for k in range(N)]
nsteps = N * 100
beta = 1.0 / T
Energy = energy(S, N, nbr)
E = []
for step in xrange(nsteps):
    k = random.randint(0, N - 1)
    delta_E = 2.0 * S[k] * sum(S[nn] for nn in nbr[k])
    if random.uniform(0.0, 1.0) < math.exp(-beta * delta_E):
        S[k] *= -1
        Energy += delta_E
    E.append(Energy)
print 'mean energy per spin:', sum(E) / float(len(E) * N)
```

Local Metropolis Algorithm

Corra el programa unas 5 veces para $L = 6$ y $T = 2{,}0$ cambiando en cada una de ellas el número total de pasos de Monte Carlo (nsteps) en múltiplos de 10.

- Obtenga por **enumeración exacta** (averiguar lo que significa) el valor de la energía media por espín: $\langle E \rangle / N = -1{,}7473$ en unidades de $J = 1$.
- A partir del programa obtenga los respectivos valores de $\langle E \rangle / N$ y saque sus propias conclusiones al comparar con el resultado exacto.
- Modifique luego su programa para incluir a) lectura de una configuración inicial a partir de un archivo, b) escritura de la configuración final en un archivo y c) gráfica de la configuración final de espines en el espacio xy.

En la siguientes diapositivas se dan los retazos de programa para leer, escribir y graficar.

Local Metropolis Algorithm

Lectura y escritura de archivos:

```
filename = 'data_local_'+ str(L) + '_' + str(T) + '.txt'
if os.path.isfile(filename):
    f = open(filename, 'r')
    S = []
    for line in f:
        S.append(int(line))
    f.close()
    print 'Starting from file', filename
else:
    S = [random.choice([1, -1]) for k in range(N)]
    print 'Starting from a random configuration'
```

```
f = open(filename, 'w')
for a in S:
   f.write(str(a) + '\n')
f.close()
```

Local Metropolis Algorithm

Graficación de archivos:

```
def x_y(k, L):
    y = k // L
    x = k - y * L
    return x, y

conf = [[0 for x in range(L)] for y in range(L)]
for k in range(N):
    x, y = x_y(k, L)
    conf[x][y] = S[k]

pylab.imshow(conf, extent=[0, L, 0, L], interpolation='nearest')
pylab.set_cmap('hot')
pylab.title('Local_'+ str(T) + '_' + str(L))
pylab.savefig('plot_A2_local_'+ str(T) + '_' + str(L)+ '.png')
pylab.show()
```

Después de haber implementado los tres aspectos anteriores, corra su programa varias veces con $L = 128$ y $T = 3{,}0$ a partir de una configuración inicial aleatoria y obtenga una gráfica típica de una configuración final. Considere N*4000 iteraciones. Haga los respectivos comentarios acerca de lo que observa.

Local Metropolis Algorithm

- Repita el proceso anterior con varias corridas a la temperatura crítica $T = 2{,}27$ y obtenga una gráfica de una configuración típica.
  Qué observa? Puede ensayar aumentar el número de iteraciones a $N * 10000$ si su máquina se lo permite.
- Finalmente, haga lo mismo a una temperatura baja con $T = 1{,}0$. Realice esta parte con $L = 32$, luego con $L = 128$ y obtenga diferentes gráficas para diferentes iteraciones. Usted debería observar que el sistema evoluciona desde un estado con paredes de dominio (stripes) a uno puramente ferromagnético (esencialmente todo con espines $+1$ o -1) para un tiempo suficientemente largo.
- Haga los respectivos análisis y comentarios.

Local Metropolis Algorithm

Implemente dentro de su programa las siguientes líneas de cálculo del valor medio de la energía y su cuadrado para obtener la capacidad calorífica con base en el teorema de fluctuación-disipación, para diferentes tamaños de sistema.

```
E_mean = sum(E) / len(E)
E2_mean = sum(a ** 2 for a in E) / len(E)
cv = (E2_mean - E_mean ** 2 ) / N / T ** 2
```

Con base en dicha implementación, obtenga curvas de calor específico en función de la temperatura para diferentes tamaños de sistema. Debería obtener curvas similares a las que se muestran en la siguiente diapositiva. Describa lo que físicamente ocurre con la configuración de espínes para $T << T_C$, $T \approx T_C$ y $T >> T_C$. Compare sus resultados con los obtenidos en la primera parte de la tarea 3 en la que se hicieron los cálculos exactos de la función partición por enumeración exacta para sistemas pequeños.

Local Metropolis Algorithm

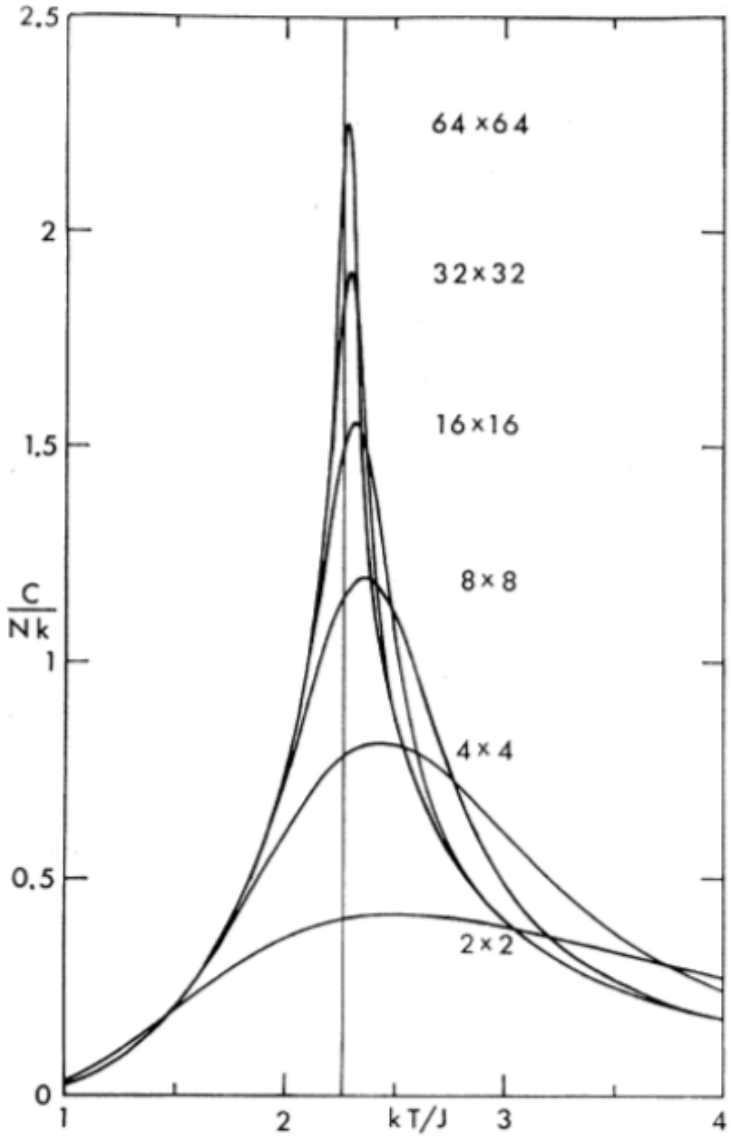

FIG. 1. The specific heat per spin for small Ising lattices; exact results for the $m \times n$ square lattice with periodic boundary conditions are displayed for $m = n = 2, 4, 8, 16, 32$, and 64 ($N = mn$). The limiting critical point is marked by a vertical line.