Informe de Laboratorio N°04

Alumno: Méndez Martín
Docentes: Dra. Marconi Verónica I.; Dr. Banchio Adolfo
Universidad Nacional de Córdoba (UNC)
Facultad de Matemática, Astronomía, Física y Computación (FaMAF)
Curso de Física Computacional: Problema N°01

(*) Para mayor claridad se recomienda ver figuras en la versión digital.

I. Introducción

I-1. Resumen

En el presente trabajo se implementará una simulación de Monte Carlo para el modelo de Ising ferromagnético en 2D con campo externo nulo y condiciones de contorno periódicas y se analizaran sistemas de tamaño 10x10, 20x20 y 40x40, de tal forma de comparar con los resultados analíticos de Onsager para el modelo de Ising en dos dimensiones:

$$k_B T_c/J = \frac{2}{ln(\sqrt{2}+1)} = 2{,}2692 \qquad (1)$$

$\beta = 1/8$,$\gamma = 7/4$,$\big[M \sim (T_c - T)^{\beta}, \chi \sim (T - T_c)^{\beta}\big]$. Se estudiarán aspectos teóricos y numéricos. Analizando respuestas transitorias y estacionarias, condiciones de contorno periódicas, transiciones de fase, propiedades termodinámicas (energía, magnetización, calor específico y susceptibilidad magnética), configuraciones de orden/desorden, exponentes críticos, cumulantes de Binder y autocorrelaciones.

I-A. Modelo físico

El modelo de Ising es muy importante porque permite capturar la física de muchos sistemas con interacciones de corto alcance. El modelo estándar consiste en un sistema de spins en una red con interacciones de primeros vecinos. El hamiltoniano general del modelo es el siguiente,

$$\mathcal{H} - J\sum_{\langle i,j\rangle} s_i s_j - h\sum_i s_i; s_i = \pm 1; h = \mu B; J > 0 \qquad (2)$$

donde B es la inducción magnética aplicada, μ es el momento magnético asociada a cada núcleo, J es la constante de acople entre los spins. El primer término tiene en cuenta la energía de intercambio y, como $J > 0$ la energía es mínima para el caso en que tenemos spins paralelos y estamos en presencia de un material *ferromagnético* (que sería nuestro caso de estudio), en caso contrario tendremos que $J < 0$, la energía mínima corresponde cuando todos los spins son antiparalelos y tenemos un material *antiferromagnético*. El segundo término expresa la interacción del sistema con el campo magnético externo aplicado, donde el signo menos significa que la energía es mínima en aquellos casos en que el momento magnético asociado a cada núcleo $\overrightarrow{\mu}$ y $\overrightarrow{B}$ se alinean. Además, el valor de s_i depende del spin de la partícula, para spins $s_i = \pm 1/2$, la constante de acople J debe absorber un factor de $1/4$ y la constante h debe absorber un factor de $1/2$ para considerar los únicos valores relevantes de $s_i = \pm 1$ (sistema de dos niveles). Cabe aclarar que, si bien en nuestro caso de estudio consideramos a J como un factor constante, la realidad es que la física ocurre precisamente en cómo definimos dicha constante para explicar procesos más complejos, es decir, esta constante puede depender de los acoples particulares en cuyo caso dependerá del par i, j considerado, además podrá depender de la temperatura. Por otro lado, si bien nosotros estamos interesados en describir propiedades en el estado estable (idealmente en el límite termodinámico) consideraremos $h = 0$, es decir, en el estado estable las interacciones entre spins primeros vecinos dominaran frente a las interacciones de cada spin con el campo magnético externo. Las fases del sistema estarán caracterizadas por la magnetización promedio (parámetro de orden) (ver figura 1).

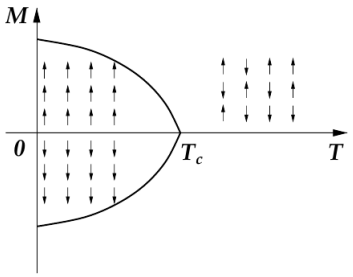

Fig. 1. Condiciones de contorno periódicas

I-A1. Condiciones de contorno periódicas (PBC)

Para implementar computacionalmente el modelo de Ising se requiere trabajar con matrices de spins finitas, por ello, se trabaja con PBC (por sus siglas en inglés). Para computar las auto energías del hamiltoniano de Ising es necesario que cada spin vea a sus cuatro primeros vecinos, por ello, las PBC se llevan a cabo de como se muestra en la figura 2 donde la primer y última fila de spins tienen como vecinos de izquierda a la última columna de spins y como vecinos de derecha a la primer columna de spins, sin embargo, se diferencian en que la primer fila de spins tiene como vecinos superiores a la última fila de spins y la última fila de spins tiene como vecinos inferiores a la primer fila de spins. Claramente, estas PBC serán relevantes cuando se estudien los efectos de tamaño y de superficie. Por ejemplo, con las PBC no existirán efectos de superficie, pues tendremos redes de spins infinitas, sin embargo, tendremos efectos de tamaño, es decir, al calcular propiedades magnéticas (o transiciones de fase) sabemos que los desarrollos multipolares decaen como $r^{-}n$,

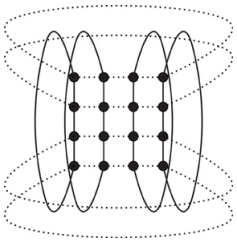

Fig. 2. Condiciones de contorno periódicas

es decir, son interacciones de largo alcance y como sólo consideramos primeros vecinos y, además, la red es finita tendremos diferencias significativas respecto a los resultados experimentales. Entonces, para mejorar los resultados la red de spins deberá ser mayor que la longitud característica de correlación. Cabe aclarar que existen otras condiciones de contorno posibles, por ejemplo, las condiciones de contorno libres (FBC) donde los vecinos de los bordes de la matriz de spins son nulos, y también, campos de contorno (campos efectivos). Dependiendo de los fenómenos y/o propiedades que se quieran estudiar se aplicarán uno u otras condiciones, incluso podrán aplicarse de forma simultánea.

II. Resultados y Discusiones

II-A. Problemas numéricos

Para realizar una relajación hacia la temperatura de equilibrio y explorar el espacio de configuraciones de forma aleatoria, fue necesario obtener números random enteros para ello se utilizó tal como se ilustra en el siguiente programa (haciendo uso de la función *floor(x)*):

```
! make clean
! make test_integer_rnd.o && ./test_integer_rnd.o
program test_integer_rnd
    use module_precision;use module_random_generator
    implicit none
    integer(sp), parameter :: m=1,n=10
    integer(sp)            :: seed,seed_val(8)
    integer(sp)            :: i,j,i_rnd,j_rnd
    real(dp)               :: nrand
    ! generate seed
    call date_and_time(values=seed_val)
    seed=seed_val(8)*seed_val(7)*seed_val(6)+&
         seed_val(5)
    ! floor(x,type) returns the greatest integer
    !  less than or equal to x
    do i=1,n;do j=1,n
        ! We want to choose one from m to n integers
        nrand=ran2(seed)
        i_rnd=m+floor((n+1-m)*nrand,sp) ! rows
        nrand=ran2(seed)
        j_rnd=m+floor((n+1-m)*nrand,sp) ! columns
        write(*,*) i_rnd,j_rnd
    end do;end do
end program test_integer_rnd
```

Por otra parte, como en algunas partes del programa es necesario computar la función exponencial decreciente de argumentos muy grandes, se corre el riesgo de producir *underflows* de valores, por ello, se utilizó la función *tiny(x)* tal como se muestra en el siguiente programa:

```
! make clean && make test_underflows_exp_function.o
! ./test_underflows_exp_function.o
program test_underflows_exp_function
    use module_precision
    implicit none
    real(dp)    :: x,result
    integer(sp) :: i
    ! tiny(x) returns the smallest positive
    !  (non zero) number in the model of the type of
         x
    write(*,*) 'abs(log(tiny(x)))=',&
               abs(log(tiny(1._dp)))
    do i=1,800
        x=real(i,dp)
        if (x<abs(log(tiny(result)))) then
            result=exp(-x)
        else; result=0._dp; end if
        write(*,*) x,result
    end do
end program test_underflows_exp_function
! Note: https://gcc.gnu.org/onlinedocs/gcc-4.5.4/
    gfortran/TINY.html
```

II-A1. ¿Cómo explorar el espacio de configuraciones?

Para explorar el espacio de configuraciones de forma ordenada, no hay dudas de que debemos consultar en cada lugar de la red si es posible *flipear* el spin, es decir, de esta forma nos aseguramos fielmente que estamos recorriendo todo el espacio de configuraciones y en consecuencia realizamos un MC_{step}. Ahora bien, al momento de explorar el espacio de configuraciones de forma random, no estamos seguros de que tirando nn índices de la matriz de spins visitaremos todos los spins, pues podría ocurrir que se repitieran ciertos índices, sin embargo, si tiramos más que nn índices, con toda seguridad, visitaríamos más de un índice y esto no sería un MC_{step}. Por ello, al explorar el espacio de configuraciones de forma aleatoria, si tiran, igual que en el caso ordenado, tantos índices como spins tengamos. La diferencia sustancial entre estos dos métodos (ambos igualmente válidos) es que al momento de agrandar la dimensión de la red el método aleatorio es mucho más rápido computacionalmente que el método ordenado.

II-A2. ¿Configuraciones con mínimos locales?

En algunas simulaciones se han obtenido resultados no esperados al relajar el sistema, por ejemplo, a bajas temperaturas se obtenían determinadas configuraciones de spins para la cual no podía ocurrir ningún flipeo, es decir, nunca ocurría que < 0 y para los casos en que > 0 el sistema no era capaz de relajar hacia el equilibrio real, pues como la relajación sigue la ley de Boltzmann, a bajas temperaturas el $\Delta U/T$ es muy grande y la exponencial decreciente es muy pequeña y el flip nunca puede ocurrir. Esto puede deberse a problemas en las PBC, las cuales si no están implementadas correctamente pueden inducir bandas de spins con el mismo valor (todos positivos o todos negativos) las cuales tienen la particular característica de que no admiten (a bajas temperaturas) relajamiento hacía algún otro mínimo de energía y tendremos un estado *metaestable*. Además, debemos tener en cuenta que los efectos de tamaño en las simulaciones también pueden influir en estas cuestiones, y a medida que disminuimos el tamaño más apreciables se hacen estos efectos, para ello se debe mencionar que existen métodos más eficientes para analizar el comportamiento del sistema a bajas temperaturas, donde se controla el tiempo de permanencia del sistema en un posible mínimo global, y en caso de tratarse

de un mínimo global se modifica el algoritmo (mejorando la implementación del importance sampling) se permite que el sistema relaje hacia el mínimo global.

II-A3. ¿Cómo determinamos el tiempo de termalización?

Es necesario aclarar que el tiempo de termalización no es independiente de la temperatura, ni de la configuración inicial de la red de spins. Por ello se utilizaron 3000 pasos de Monte Carlo (MC_{step}) luego de los cuales se considera estado estacionario y donde se empiezan a recolectar los observables. Sin embargo, para reducir el tiempo de CPU es necesario determinar con precisión la cantidad de pasos de Monte Carlo que se necesitan descartar de acuerdo a cada rango de temperatura que se esté simulando. Para determinar el tiempo de termalización de $MC_{step} = 3000$ se analizó la respuesta del sistema para distintas temperaturas (tanto lejos como cerca de la temperatura crítica), y usando un total de $MC_{step} = 50000$, en la figura 3 se muestran los resultados obtenidos para una red 40x40 (se simuló este tamaño de red, debido a que los sistemas pequeños al tener menor cantidad de configuraciones posibles, tienden a termalizar más rapido). En las gráficas se puede observar que el transitorio es muy pequeño, es decir, que los valores magnetización y energía convergen muy rápido a los valores estables entonces, a modo de asegurarnos de estar en régimen estacionario se propuso un transitorio de $MC_{step} = 3000$.

II-B. Cambios de flips

Teniendo en cuenta que el método de metrópolis es, en esencia, un caso de *importance sampling* donde evaluamos las energías de cada configuración flipeando un único spin de la red y donde se tienen dos casos, por un lado, si esta nueva energía es menor que la energía de la configuración anterior se acepta el flipeo, y el sistema evoluciona y, por otro lado, si la nueva energía es mayor a la energía anterior se acepta bajo el criterio de boltzmann que consiste en aceptar el flip bajo la siguiente hipótesis (ver ecuación 3), donde el rnd es un número random de distribución uniforme que varia de cero a uno (en el presente trabajo se trabajo con el generador de números random Mersenne Twister).

$$exp(-\Delta U/k_B T) >= rnd \tag{3}$$

Por otro lado, para no computar todas las veces la energía de la configuración nueva, sólo evaluamos la diferencia entre energías (actual y anterior) y en función de su valor (negativa, positiva o nula) aceptamos bajo el criterio anterior, el fragmento de código 1 que se muestra a continuación realiza el cálculo anterior, donde se tuvieron en cuenta, en caso de aceptar el flip, los cambios PBC.

Listing 1: Subrutine: Método de Metrópolis

```
do k=1,n*n ! exploramos el espacio de
    configuraciones de forma random
 ! floor(x,type) returns the greatest integer less
    than or equal to X
 ! random integer from 2 to n+1 (fila)
 nrand=real(grnd(),dp);i=2_sp+floor(n*nrand,sp)
 ! random integer from 2 to n+1 (columna)
 nrand=real(grnd(),dp);j=2_sp+floor(n*nrand,sp)
 deltaU_adim=U_delta_adim(aux_matrix_pbc(i,j),&
  aux_matrix_pbc(i,j+1),aux_matrix_pbc(i,j-1),&
  aux_matrix_pbc(i+1,j),aux_matrix_pbc(i-1,j))
 cond1:if (deltaU_adim<=0._dp) then
  U_adim=U_adim+deltaU_adim
  aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
  ! actualizamos condiciones de borde
  call pbc_ij(aux_matrix_pbc,n,i,j)
 else
  if (T_adim==0._dp) exit cond1
  nrand=real(grnd(),dp)
  if (deltaU_adim*(1._dp/T_adim)<&
      abs(log(tiny(1._dp)))) then
   if (exp(-deltaU_adim*(1._dp/T_adim))>=nrand) then
    U_adim=U_adim+deltaU_adim
    aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
    ! actualizamos condiciones de borde
    call pbc_ij(aux_matrix_pbc,n,i,j)
   end if
   else
    if (nrand==0._dp) then
     U_adim=U_adim+deltaU_adim
     aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
     ! actualizamos condiciones de borde
     call pbc_ij(aux_matrix_pbc,n,i,j)
    end if
   end if
 end if cond1
end do
```

II-C. Cálculo de propiedades termodinámicas

Los resultados obtenidos de magnetización media (en valor absoluto), energía media, calor específico (a volumen constante) y susceptibilidad magnética se muestran en las gráficas 4, para distintos tamaños de red. En estas gráficas se puede observar que en el caso de la magnetización a medida que aumenta el tamaño de red los resultados se aproximan más al valor exacto. Analizando la energía (por unidad de tamaño de red), por el contrario, dio muy similar en todos los casos. Además, al analizar el calor específico y la susceptibilidad magnética que sabemos son, en esencia, una medida directa de las fluctuaciones de energía y magnetización, respectivamente, vemos que al aumentar el tamaño de red el calor específico tiende a una delta de Dirac entorno a la temperatura crítica lo cual nos muestra que en tamaños de red finitos no existen divergencias sino que el sistema tiene la capacidad de flipear spins entre una fase y otra lo cual sabemos que en la realidad no ocurre; para el caso de la susceptibilidad vemos que tampoco existe una divergencia real, pero de igual forma a medida que aumenta el tamaño de red se tiende a una Delta de Dirac, también, notamos una desviación de los picos de las curvas acercándose ésta a la temperatura crítica a medida que aumentamos el tamaño de red. Por otro lado, en las corridas anteriores se tomo el tiempo de cpu para obtener una idea estimada de cuán costoso computacionalmente se vuelve el problema a medida que incrementamos el tamaño de red los resultados fueron $180{,}0229[s]$ para una red de 10x10, $771{,}0757[s]$ para una red de 20x20 y $2923{,}0955[s]$ para una red de 40x40, notando que en este último caso se incremento el tiempo unas 16 veces más aproximadamente que el primer caso y unas 4 veces más aproximadamente que el segundo caso.

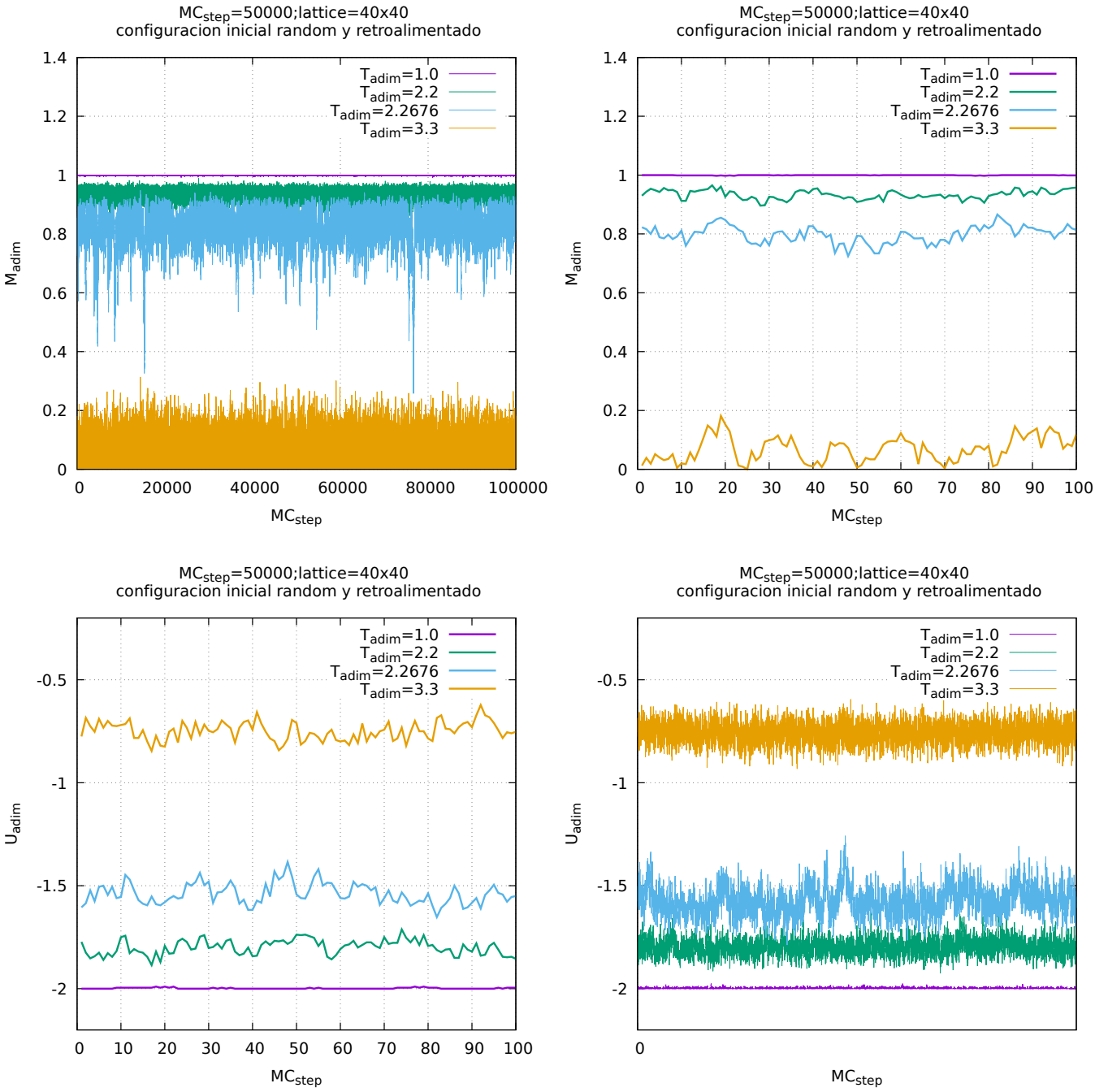

Fig. 3. Respuesta transitoria y estacionaria del sistema

II-D. Distribuciones de probabilidad

En la figura 5 se muestran los histogramas obtenidos representando las distribuciones de probabilidad para la magnetización, para ello se compraron histogramas a temperaturas por debajo y por encima de la temperatura crítica y para distintos tamaños de red. Notemos que para temperaturas bajas $T < Tc_{aprox}$ tenemos una distribución *doble pico*, con mayor probabilidad en los valores de magnetización $M_{adm} = \pm 1$ y a medida que aumentamos el tamaño de red los picos se hacen más angostos, por lo menos esto es claro de ver entre los tamaños 10x10 y 20x20, para el caso de 40x40 se ve, en lugar de más angosto, más ancho que el caso de 20x20, esto podría deberse a que no se simuló exactamente a la temperatura crítica del sistema según esta dimensión (Tc_{aprox}). Lo que esperaríamos ver es que a medida que aumenta la dimensión de red la probabilidad de transición entre magnetización $M_{adm} = 1$ y $M_{adm} = -1$ se hace nula, recordando que, experimentalmente se medirá ó $M_{adm} = 1$ ó $M_{adm} = -1$, no ambas, y el valor dependerá de la configuración inicial de spins que tenga la muestra. Por otro lado, para temperaturas altas $T > Tc_{aprox}$ observamos que tenemos una distribución *único pico*, donde la probabilidad es mayor para una magnetización nula, y aquí sí

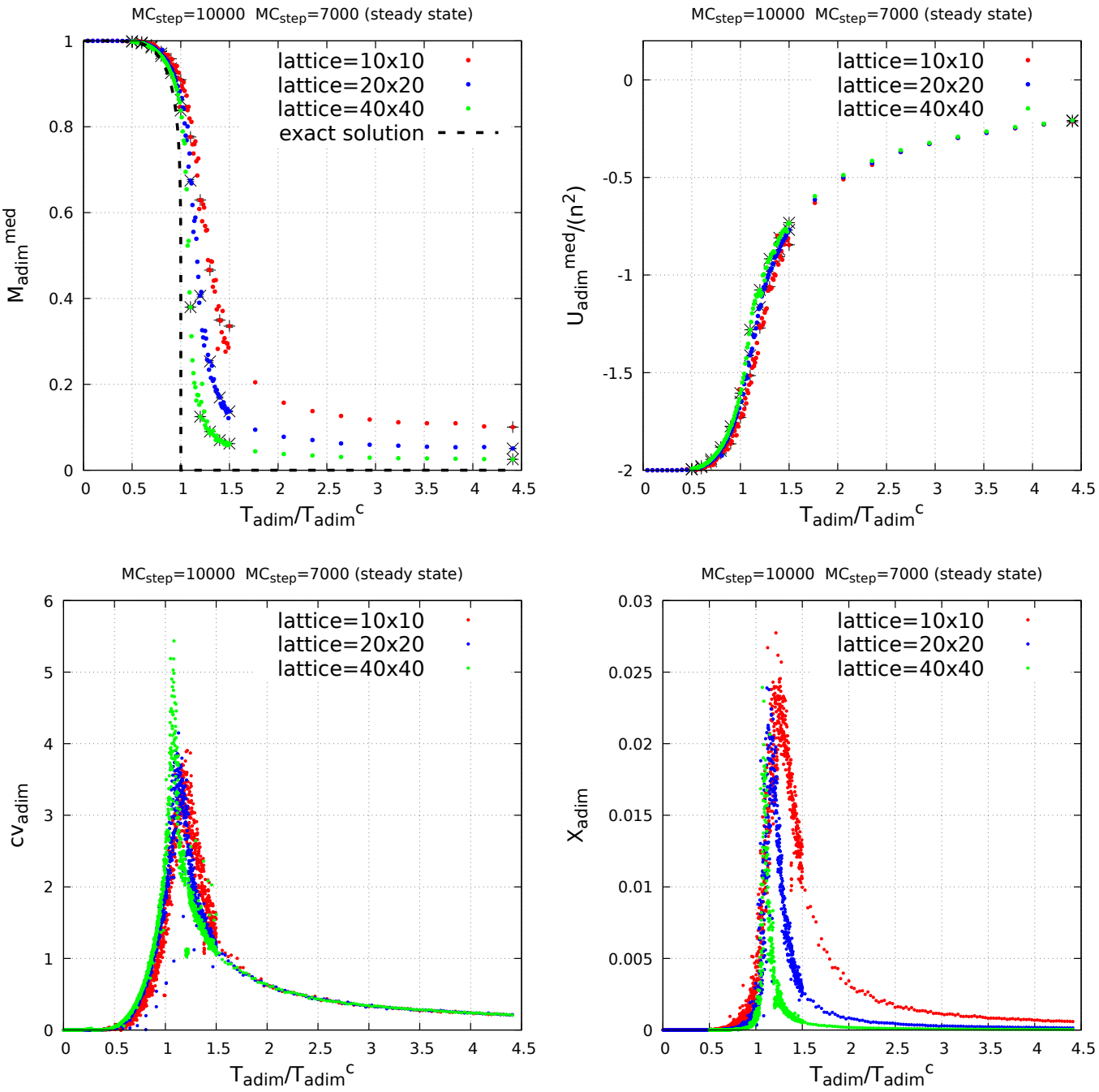

Fig. 4. Resultados de observables para distintos tamaños de red

se ve claramente que al aumentar la dimensión de la red el pico se vuelve angosto. Estas dos formas de distribución *doble pico* y *único pico* se puede explicar si recordamos que el modelo de Ising está gobernado por la energía libre de Hemholtz ($F = U - T \cdot S$) entonces a bajas temperaturas los spins tienden a ordenarse disminuyendo lo más posible la entropía del sistema (S) y el sistema estará gobernado por su energía interna obteniendo una magnetización media definida (qué en sistemas finitos, se observan efectos de tamaño que hacen que los spins puedan *flipear* entre los dos valores degenerados de magnetización), en cambio, a altas temperaturas los spins tienden a desalinearse aumentando lo más posible la entropía del sistema y, en consecuencia, es esta la que gobierna el sistema obteniendo una magnetización media nula. Entonces, vemos que el sistema está en continua lucha entre la energía interna y entropía, y el sistema evolucionará según cual de estas cantidades venza para una data temperatura. Para verificar que efectivamente se obtienen configuraciones distintas a bajas (ordenado) y a altas (desordenado) temperaturas se muestran en la figura 13, donde se imprimieron las configuraciones de spins para una red de tamaño 40x40 para temperaturas por debajo, igual y por encima de la temperatura crítica.

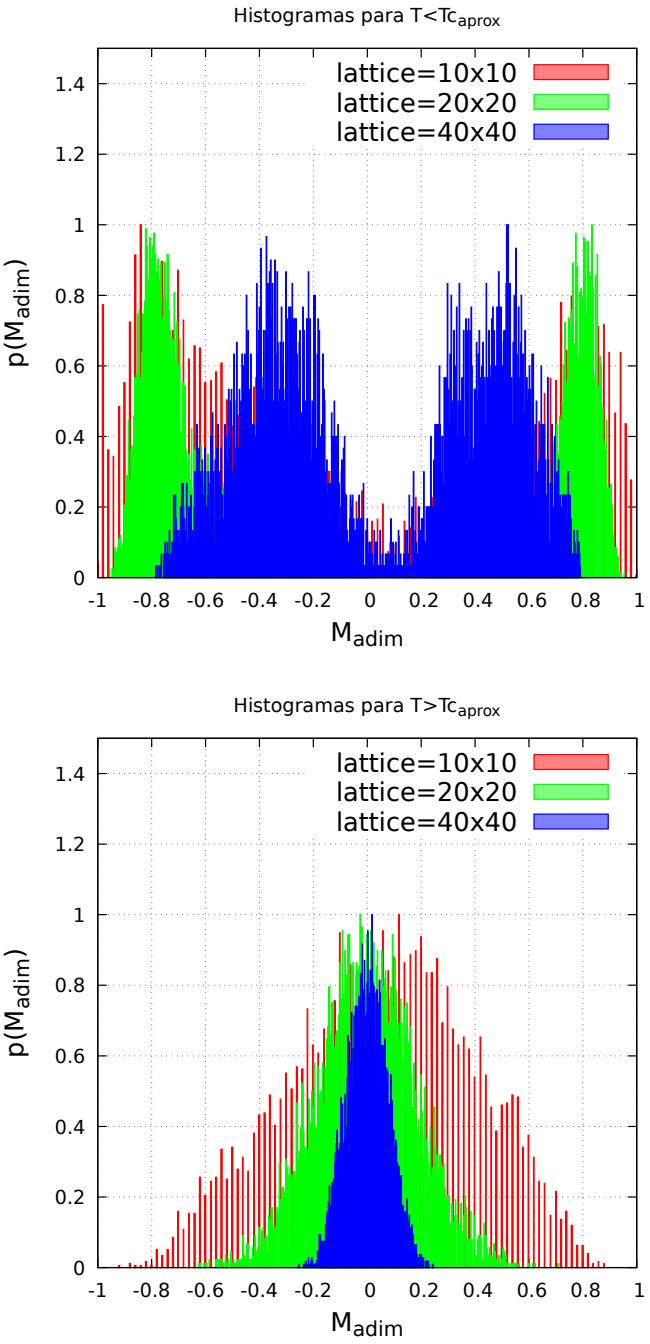

Fig. 5. Histogramas de magnetización para distintos tamaños de red y distintas temperaturas

Notemos además que se compararon las configuraciones en el equilibrio partiendo con una configuración inicial random (alta temperatura, desordenado, enfriamiento) y partiendo con una configuración inicial ordenada con todos los spins up (baja temperatura, ordenado, calentamiento), además se debe aclarar que la palabra *retroalimentado* en la figura se refiere a que todas las termalizaciones se hacen gradualmente a temperaturas intermedias con un delta de temperaturas no tan abrupto (evitando el *quenching*) y en consecuencia cada paso de temperatura comienza con una configuración inicial de spins según termalizó la temperatura inmediatamente anterior, de esta forma se reduce tiempo de computo y se simula de acuerdo a parámetros más realistas.

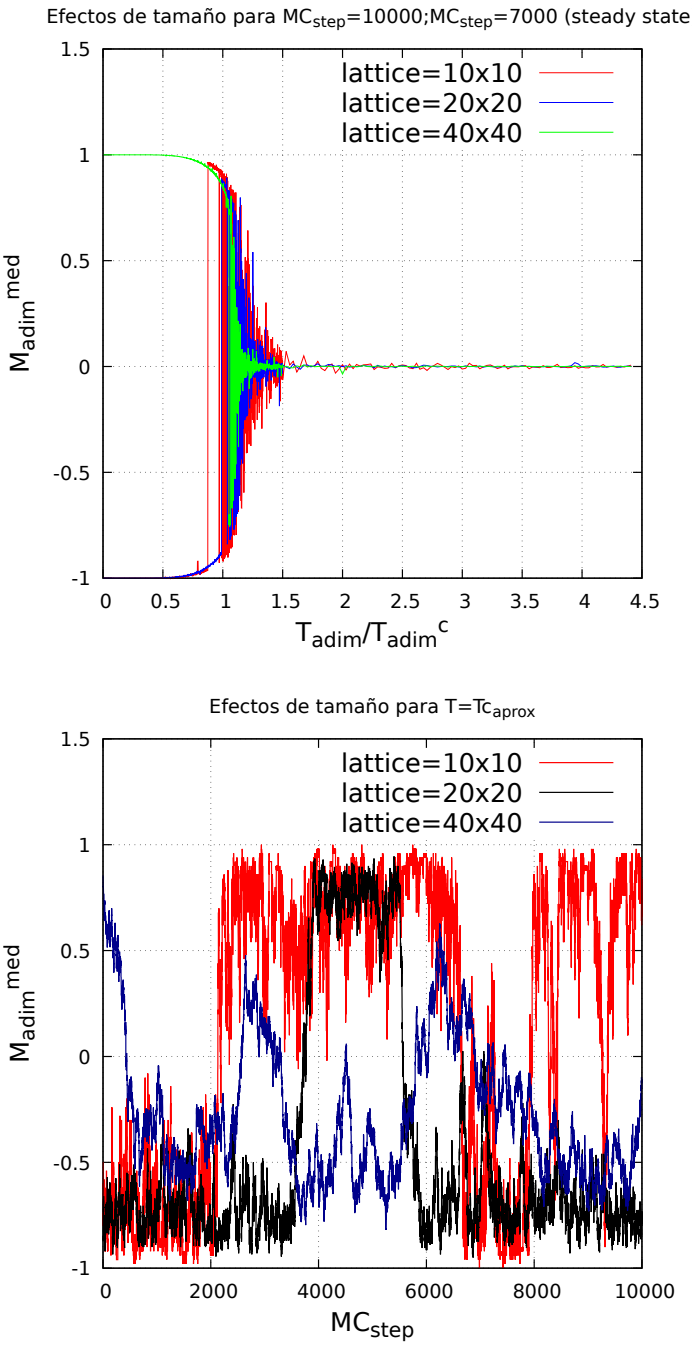

Fig. 6. Efectos de tamaño en oscilaciones de la magnetización

Por otro lado, para entender un poco más los efectos de tamaño en las simulaciones se graficaron las oscilaciones que presenta la magnetización comparando para distintos tamaños de red. En la figura 6 se observa a la izquierda que a medida que aumenta el tamaño de la red, las oscilaciones en $T = T_c$ disminuyen, lo que nos muestra que, en consecuencia, también disminuyen las fluctuaciones en la magnetización. Y, a la derecha, se observa que el tiempo ergódico aumenta a medida

que aumenta el tamaño de red aumenta, es decir, que el sistema tiene menos probabilidad de flipear entre valores de magnetización ± 1, además, este gráfico nos muestra también que, si trabajamos con el la expresión de la magnetización sin valor absoluto debemos tener presente este tiempo ergódico y realizar los promedios en tiempos de observación menores a estos, caso contrario corremos el riesgo de que los valores medios nos den nulos, cuando en la realidad no es así.

II-E. Análisis entorno a la transición de fase

II-E1. Cumulantes de Binder

Para estimar las temperaturas críticas según cada tamaño de red se calculó el cumulante de Binder (4) que caracteriza a distribuciones de probabilidad gaussianas,

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\left(\langle m^2 \rangle_L\right)^2} \tag{4}$$

que relaciona el cuarto momento con el segundo momento de la magnetización específica. Este parámetro es bueno porque presenta una cambio claro para la temperatura crítica, además, como el cumulante no depende de las fluctuaciones de (de energía o de magnetización) su empleo para determinar aproximadamente la temperatura es mejor, que al analizar únicamente la energía o la magnetización, cuyos resultados cerca de la temperatura crítica es muy imprecisa. Los resultados obtenidos se muestran en la figura 7, donde simularon distintos tamaños de red y dos versiones en las configuraciones, la versión 1 consiste en realizar un único experimento de mayor duración y la versión 2 consiste en realizar varios experimentos de menor duración. En consecuencia, al variar los tamaños de red y calculamos el cumulante de Binder para cada caso, esperamos que todos ellos se crucen en un único punto que determina la temperatura crítica. Entonces, analizando los resultados se puede observar que las gráficas, a pesar de usar extrapolación de orden cúbico para evidenciar los cruces, no se obtuvieron buenos resultados, sin embargo, si se evidencia que a medida que aumenta el tamaño de red el cumulante tiende a una función escalón, además, como vimos en la figura 4 a medida que aumentamos el tamaño de red, la temperatura crítica aproximada tiende a acercarse por derecha a la temperatura crítica exacta, esto se evidencia en los resultados del cumulante, donde vemos que .ª groso modo"podemos decir que los cruces ocurren a la derecha de la temperatura crítica exacta (a la izquierda del valor $T_{adm}/T^c_{adm} = 1$, donde podriamos estimar que $T_{adm}/T^c_{adm|exact} \sim 1{,}045 \Rightarrow T^c_{adm|aprox} \sim 2{,}3713$ contra un valor exacto de $T^c_{adm|exact} \sim 2{,}2692$. Las diferencias significativas podrían deberse a la utilización de pocos MC_{step} (se utilizaron 5000 en total y descartamos los primeros 3000 del transitorio para los cálculos).

II-E2. Exponentes críticos

Según la teoría de fenómenos críticos (que utiliza el concepto de *mean field theory*) se puede demostrar que entorno a la temperatura crítica (en las proximidades de las transiciones de fase) las propiedades de susceptibilidad magnética, calor específico, magnetización y correlaciones (longitud) están gobernadas por leyes universales con coeficientes (exponentes) específicos, los cuales no son independientes (están correlacionados), pero lo interesante es que el comportamiento de estas magnitudes no dependen de los detalles del sistema físico sino de características generales (para sistemas ferromagnéticos dependen de la dimensión de la red, del rango de interacción y de las magnitudes de spins). En particular, se analizaron los exponentes críticos de la magnetización (en valor absoluto) y de la susceptibilidad magnética, para ello se realizaron *fitteos* de los resultados numéricos, con las propuestas de fitteo según la ecuación 5, válidas en entornos cercanos a la temperatura crítica y donde los coeficientes a, b, γ, β son parámetros de ajustes a determinar.

$$\begin{aligned} \chi_{adm} &= a\left[T^c_{adm}\left(\frac{T_{adm}}{T^c_{adm}} - 1\right)\right]^{-\gamma} \\ M_{adm} &= b\left[T^c_{adm}\left(1 - \frac{T_{adm}}{T^c_{adm}}\right)\right]^{\beta} \end{aligned} \tag{5}$$

Los resultados obtenidos se muestran en la figura 8, donde se puede observar que el fitteo anduvo bien para determinar aproximadamente el exponente γ de la susceptibilidad magnética (obteniendo un valor $\gamma_{aprox} = 1{,}33$ contra un valor $\gamma_{exacto} = 1{,}75$), sin embargo, no fue bueno para calcular el exponente β de la magnetización (obteniendo un valor $\gamma_{aprox} = 0{,}0365$ contra un valor $\gamma_{exacto} = 0{,}125$), quizás esto pueda deberse a dos factores, por un lado al tamaño de red simulado, pues al acortar el tiempo de simulación se analizó una red pequeña de 10x10 spins, y por otro, podría deberse a los pocos MC_{step} utilizados (se utilizaron 10000 en total y descartamos los primeros 7000 del transitorio para los cálculos).

II-F. Autorrelaciones temporales

II-G. Función de autocorrelación temporal

La función de autocorrelación se define como se muestra en la ecuación 6. La misma tiene la propiedad de normalización ($A(k = 0) = 1$) y para tiempos de correlación grandes la función de autocorrelación decae exponencialmente ($\tau_{corr} \to \infty \Rightarrow A(k) \to a \cdot exp\left(-k/\tau_{O,exp}\right)$), donde $\tau_{O,exp}$ es el tiempo de autocorrelación exponencial y a es alguna constante de proporcionalidad. Básicamente, las autocorrelaciones nos permiten cuantificar qué tanta información se preserva de un observable en un tiempo dado tiempo inicial y el mismo observable en un tiempo dado final donde, si esta autocorrelación es no nula, decimos que que existe autocorrelación a un tiempo de correlación $\tau_{corr} = t_{final} - t_{inicial}$. Es muy importante analizar la autocorrelación debido a que el sistema no solo esta regido por correlaciones espaciales sino también por correlaciones temporales que si son significativas los errores estadísticos del observable (que está autocorrelacionado temporalmente) se verán incrementados en un factor $\sqrt{2\tau_{corr}}$ que, si el tiempo de correlación (τ_{corr}) es significativo los errores aumentan mucho, además nos sirve para cuantificar si existe alguna variable que esté relacionada dinámicamente a tiempos largos.

$$A(k) = \frac{\left[\langle O_i O_{i+k} \rangle - \langle O_i \rangle^2\right]}{\left[\langle (O_i)^2 \rangle - \langle O_i \rangle^2\right]} \tag{6}$$

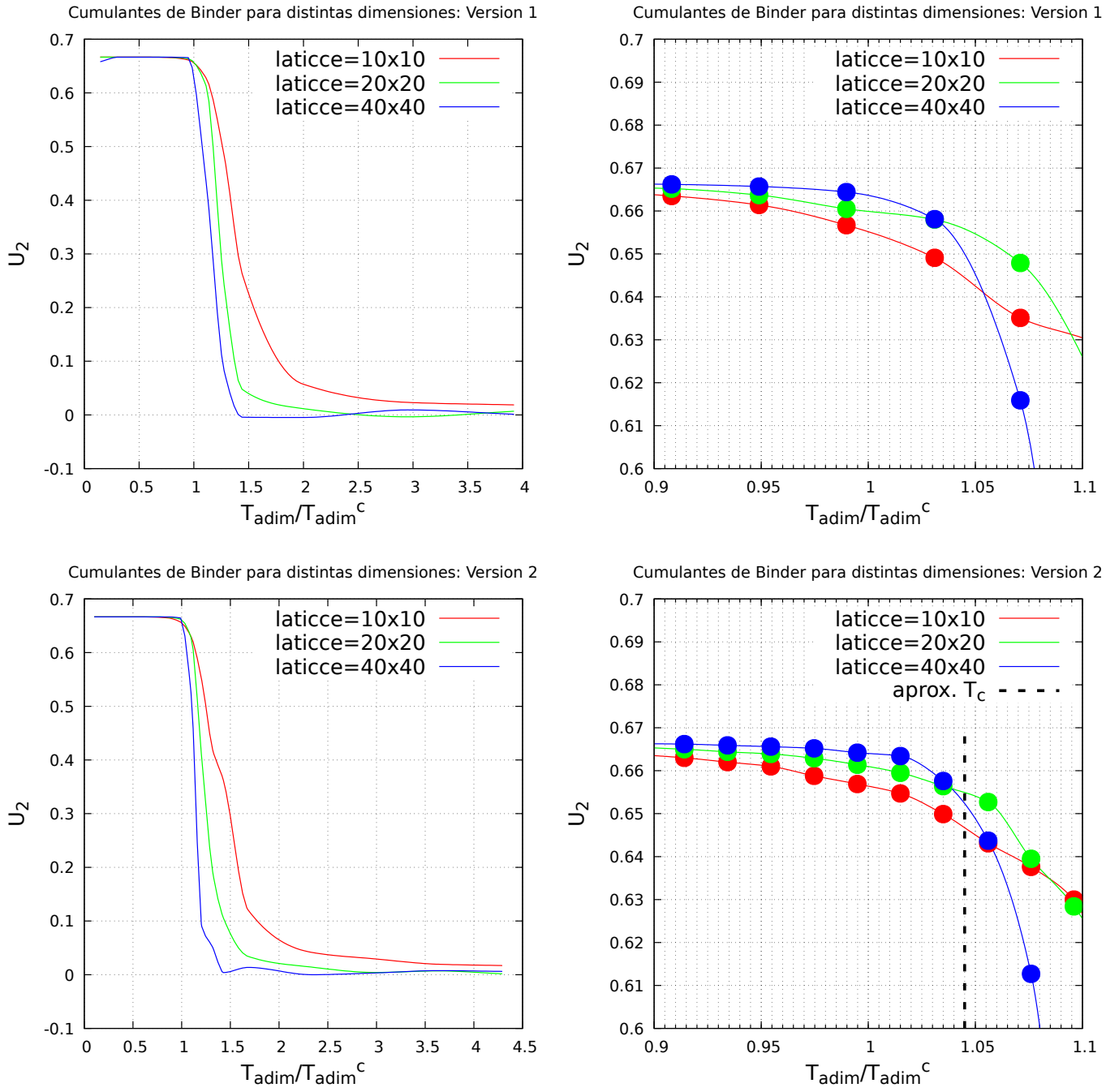

Fig. 7. Cumulante de Binder y estimación de la temperatura crítica

II-G1. ¿Cómo calculamos numéricamente la función de autocorrelación?

Listing 2: Subrutine: función autocorrelación

```
! subrutina para calcular la funci n de autocorrelaci n
subroutine func_autocor(obs,time_index,tau_corr,num_of_terms,&
autocor_vector,aux_vector1,mask_vector,aux_vector2,&
obs_med,var)
implicit none
integer(sp), intent(in) :: time_index ! indice del observable
integer(sp), intent(in) :: tau_corr ! tiempo de correlaci n
! menor cant de terminos admisibles (debe ser multiplo de tau_corr)
integer(sp), intent(in) :: num_of_terms
real(dp), intent(in)    :: obs ! observable
real(dp), intent(inout) :: autocor_vector(tau_corr) ! vector autocorrelaci n
! vectores auxiliares
real(dp), intent(inout) :: aux_vector1(tau_corr),aux_vector2(tau_corr)
```

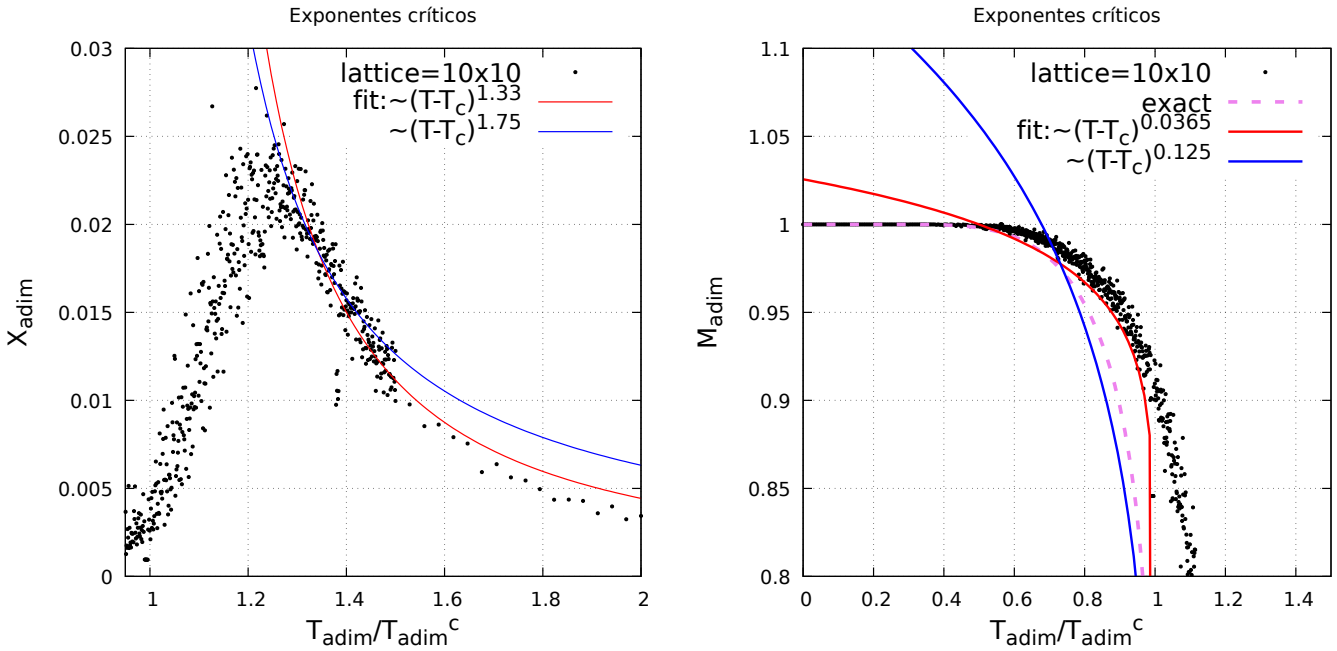

Fig. 8. Determinación aproximada de exponentes críticos

```
14        real(dp), intent(inout) :: mask_vector(
              tau_corr) ! vector m scara
15        real(dp), intent(inout) :: obs_med,var !
              valor medio y varianza
16        integer(sp) :: i,index ! loop indices
17        integer(sp) :: total_obs_num ! total de
              observables que deben
18        ! calculamos valores primeros y segundos
              momentos
19        obs_med=obs_med+obs ! primeros momentos (
              acumulaci n)
20        var=var+obs*obs     ! segundos momentos (
              acumulaci n)
21        ! llenamos por primera vez el vector de dim=
              tau_corr
22        if (time_index<=tau_corr) autocor_vector(
              time_index)=obs
23        if (time_index==tau_corr) then
24            aux_vector1(:)=autocor_vector(:)
25            aux_vector2(:)=autocor_vector(:)
26        end if
27        ! determinamos total de observables que van
              a ingresar
28        total_obs_num=num_of_terms+tau_corr
29        ! computamos las correlaciones a todo tiempo
30        if (time_index>tau_corr.and.time_index<=(
              total_obs_num)) then
31            ! definimos el indice dentro del rango
                  [1,tau_corr]
32            if (mod(time_index,tau_corr)==0_sp) then
                  ;index=tau_corr
33            else;index=mod(time_index,tau_corr);end
                  if
34            aux_vector2(:)=cshift(aux_vector2(:),
                  shift=1)
35            aux_vector2(tau_corr)=obs
36            autocor_vector(index)=autocor_vector(
                  index)+&
37                dot_product(aux_vector1(:),
                      aux_vector2(:))
38        end if
39        if (mod(time_index,tau_corr)==0_sp)
              aux_vector1(:)=aux_vector2(:)
40        ! computamos las correlaciones faltantes
41        if (time_index==total_obs_num) then
42            mask_vector(:)=1._dp
43            aux_vector1(:)=aux_vector2(:)
44            do index=1,tau_corr-1
45                aux_vector2(:)=cshift(aux_vector2(:)
                      ,shift=1)
46                mask_vector(tau_corr-(index-1))=0.
                      _dp
47                aux_vector2(:)=aux_vector2(:)*
                      mask_vector(:)
48                autocor_vector(index)=autocor_vector
                      (index)+&
49                    dot_product(aux_vector1(:),
                          aux_vector2(:))
50                ! promediamos en el ensamble
51                autocor_vector(index)=autocor_vector
                      (index)*&
52                    (1._dp/real(num_of_terms+
                          tau_corr-index,dp))
53            end do
54            ! promediamos en el ensamble (tau_corr
                  t rminos)
55            autocor_vector(tau_corr)=autocor_vector(
                  tau_corr)*&
56                (1._dp/real(num_of_terms,dp))
57            obs_med=obs_med*(1._dp/real(
                  total_obs_num,dp)) ! primeros
                  momentos
58            var=var*(1._dp/real(total_obs_num,dp)) !
                   segundos momentos
59            var=var-obs_med*obs_med ! varianza
60        end if
61    end subroutine func_autocor
```

Para calcular la función de autocorrelación numéricamente se hizo uso de la función *shift* para crear una subrutina (ver referencia 2), que consiste básicamente en llenar un vector con observables a un tiempo dado e ir computando en cada paso las correlaciones a todos los tiempos requeridos, de tal forma de utilizar la mayor cantidad de datos para mejorar la estadística,

aprovechándonos de la ergodicidad del sistema, para ello en el ejemplo III se muestra de forma simple cómo funciona la subrutina para un dado *tau-corr* (tiempo máximo para el cual se quiere calcular la correlación) y un dado *num-of-terms* (numero de términos que se admiten para computar la correlación a tiempo *tau-corr*) haciendo uso de la función shift y definiendo vectores "máscara"para computar. Esta implementación tiene la ventaja de que se aprovechan todos los datos para computar correlaciones y aumentamos datos para mejorar la estadística. Por otro lado, las autocorrelaciones se llevaron a cabo para tamaños de 10x10,20x20 y 40x40, para las temperaturas $T_{adim}/T^c_{adim} \sim \{0{,}88; 0{,}98; 0{,}99; 1{,}4\}$, con $tau_{corr} = 1000$, $MC_{step} = 1010999$ (que nos aseguran que tengamos un millón de términos para calcular las funciones de aucorrelación a tiempo mil, que sabemos, es la que menos términos tiene y en consecuencia menor precisión) y un transitorio de $MC^{trans}_{step} = 10000$ y los resultados obtenidos se muestran en la figuras 9,10,11 y 12 donde podemos que la función de autocorrelación crece a medida que nos acercamos a la temperatura crítica y a medida que aumentamos el tamaño de la red de spins. Además los gráficos aumentados en escala logarítmica nos permiten ver el rápido decaimiento de la función de correlación para τ_{corr} pequeños. Por otro lado, en las figuras se pueden observar lineas punteadas que corresponden a tiempos de correlación estimados para los cuales, si las mediciones de observables de magnetización (se estimó para estas ya que, en general son mayores que para la energía) se hacen en tiempos mayores al estimado, los resultados se pueden considerar estadísticamente independientes, claro está que estos tiempos de correlación aumentan a medida que nos acercamos a la temperatura crítica y a medida que aumentamos la dimensión del sistema (es más, para el caso en que $T = 2{,}2676$ vemos que el tiempo de correlación excede el tiempo máximo de mil), esto es claramente un problema porque para obtener resultados (mediciones) independientes debemos correr mayor cantidad de MC_{step} (ya que el tiempo entre mediciones debe ser mayor al tiempo de correlación) lo que incrementa el coste computacional, este fenómeno se denomina *critical slowing down* y para lidiar con este se crean algoritmos que flipean *clusters* (o grupos) de spins de forma conjunta, en lugar de un único spin.

III. Ejemplo para función autocorrelación

```
num_of_terms=tau_corr*i con i={1,2,3,...}
obs={A1,A2,...,A(num_of_terms+tau_corr)}
++++++++++++++++++++++++++++++++++++
si elegimos tau_corr=4 e i=2
=>num_of_terms=8;obs={A1,A2,...,A12}
++++++++++++++++++++++++++++++++++++
paso 1
 autocor_vector=[A1,A2,A3,A4]
 aux_vector1=[A1,A2,A3,A4]
 aux_vector2=[A1,A2,A3,A4]
++++++++++++++++++++++++++++++++++++
paso 2 (agrego observable A5)
 aux_vector2=shift[aux_vector2] y A5
            =[A2,A3,A4,A1] y A5
            =[A2,A3,A4,A5]
 =>autocor_vector(1)=
    =dot_product(aux_vector1*aux_vector2)
    =(A1*A2+A2*A3+A3*A4+A4*A5)
++++++++++++++++++++++++++++++++++++
paso 3 (agrego observable A6)
 idem a los pasos anteriores
 aux_vector2=[A3,A4,A5,A6]
 =>autocor_vector(2)=
    =dot_product(aux_vector1*aux_vector2)
```

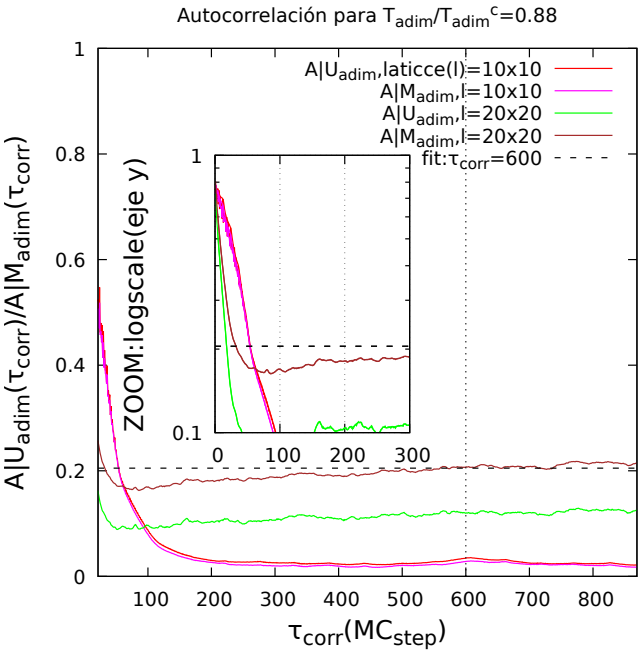

Fig. 9. Funciones de autocorrelación: $T_{adm} = 2{,}0$

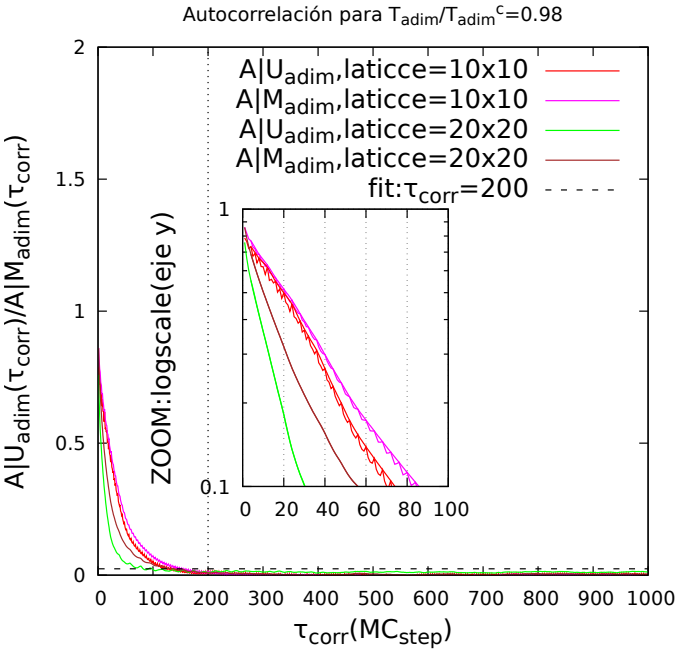

Fig. 10. Funciones de autocorrelación: $T_{adm} = 2{,}22$

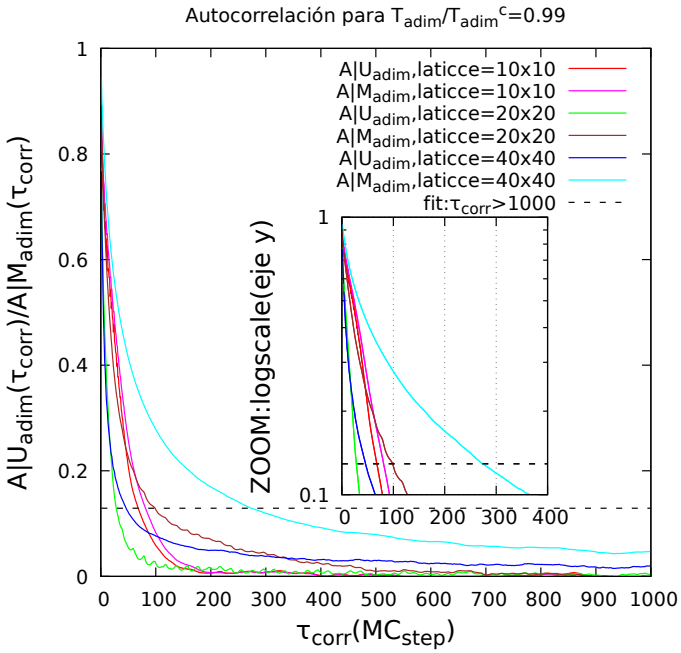

Fig. 11. Funciones de autocorrelación: $T_{adm} = 2{,}2676$

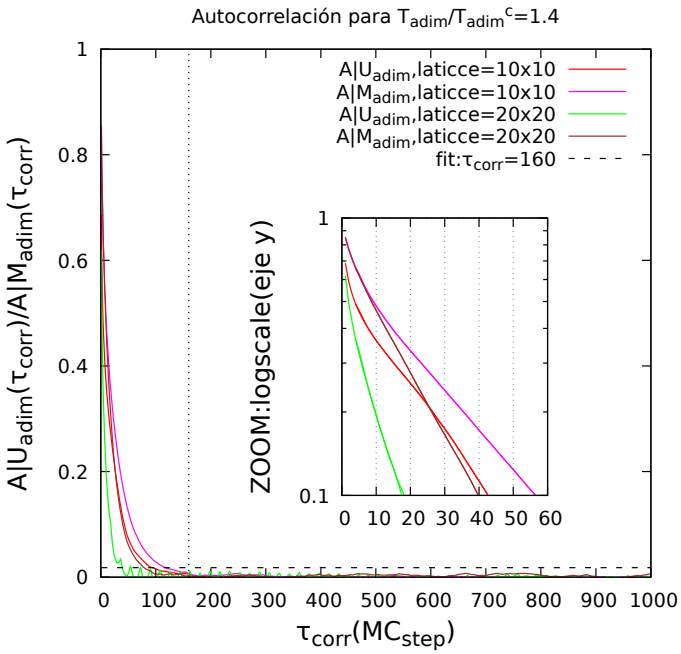

Fig. 12. Funciones de autocorrelación: $T_{adm} = 3{,}3$

```
    =(A1*A3+A2*A4+A3*A5+A4*A6)
++++++++++++++++++++++++++++++++++++
paso 4 (agrego observable A7)
 idem a los pasos anteriores
 aux_vector2=[A4,A5,A6,A7]
 =>autocor_vector(3)=
    =dot_product(aux_vector1*aux_vector2)
    =(A1*A4+A2*A5+A3*A6+A4*A7)
++++++++++++++++++++++++++++++++++++
paso 5 (agrego observable A8)
 idem a los pasos anteriores
 aux_vector2=[A5,A6,A7,A8]
 =>autocor_vector(4)
    =dot_product(aux_vector1*aux_vector2)
    =(A1*A5+A2*A6+A3*A7+A4*A8)
++++++++++++++++++++++++++++++++++++
Aquí, como hicimos una vuelta completa
debemos modificar los vectores auxiliares.
++++++++++++++++++++++++++++++++++++
aux_vector1=aux_vector2
 =[A5,A6,A7,A8]
++++++++++++++++++++++++++++++++++++
paso 6 (agrego observable A9)
 aux_vector2=shift[aux_vector2] y A9
             =[A6,A7,A8,A9]
 =>autocor_vector(1)=autocor_vector(1)
    +dot_product(aux_vector1*aux_vector2)+
    =(A1*A2+A2*A3+A3*A4+A4*A5+
      +A5*A6+A6*A7+A7*A8+A8*A9)
++++++++++++++++++++++++++++++++++++
paso 7 (agrego observable A10)
 idem a los pasos anteriores
 aux_vector2=[A7,A8,A9,A10]
 =>autocor_vector(2)=autocor_vector(2)+
    +dot_product(aux_vector1*aux_vector2)
    =(A1*A3+A2*A4+A3*A5+
      A4*A6+A5*A7+A6*A7+A8*A10)
...
++++++++++++++++++++++++++++++++++++
paso 9 (agrego obserbable A12)
 idem a los pasos anteriores
 aux_vector2=[A9,A10,A11,A12]
 =>autocor_vector(4)=autocor_vector(4)+
    +dot_product(aux_vector1*aux_vector2)
    =(A1*A5+A2*A6+A3*A7+A4*A8+
      +A5*A9+A6*A10+A7*A11+A8*A12)
++++++++++++++++++++++++++++++++++++
Finalmente computamos los términos
no directos para los términos
autocor_vector(i) con i<4
++++++++++++++++++++++++++++++++++++
paso 10 (recolecto datos faltantes)
 mask=[1,1,1,1]
 aux_vector1=aux_vector2
             =[A9,A10,A11,A12]
 paso 10.1
  aux_vector2=shift[aux_vector2]
              =[A10,A11,A12,A9]
  mask=[1,1,1,0]
  aux_vector2=aux_vector2*mask
              =[A10,A11,A12,0]
  =>autocor_vector(1)=autocor_vector(1)+
     +dot_product(aux_vector1*aux_vector2)
     =(A1*A2+A2*A3+A3*A4+A4*A5+A5*A6+A6*A7+
       +A7*A8+A8*A9+A9*A10+A10*A11+A11*A12)
```

```
paso 10.1
 idem a los pasos anteriores
 aux_vector2=[A11,A12,0,A10]
 mask=[1,1,0,0]
 aux_vector2=aux_vector2*mask
            =[A11,A12,0,0]
 =>autocor_vector(2)=autocor_vector(2)+
    +dot_product(aux_vector1*aux_vector2)
    =(A1*A3+A2*A4+A3*A5+A4*A6+A5*A7+
      +A6*A7+A8*A10+A9*A11+A10*A12)
  ...
```

IV. Códigos

Repositorio de GitHub

- https://github.com/mendzmartin/fiscomp2022.git

Repositorio GitHub del problema

- https://github.com/mendzmartin/fiscomp2022/tree/main/lab04

Códigos principales y Makefile

- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_01.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_02.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_03.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_04.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_05.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/ising_ferromagnetic_model_06.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab04/prob01/code/Makefile

Módulos utilizados

- https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_2D_ferromagnetic_ising.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_mt19937.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_mzran.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_random_generator.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_precision.f90

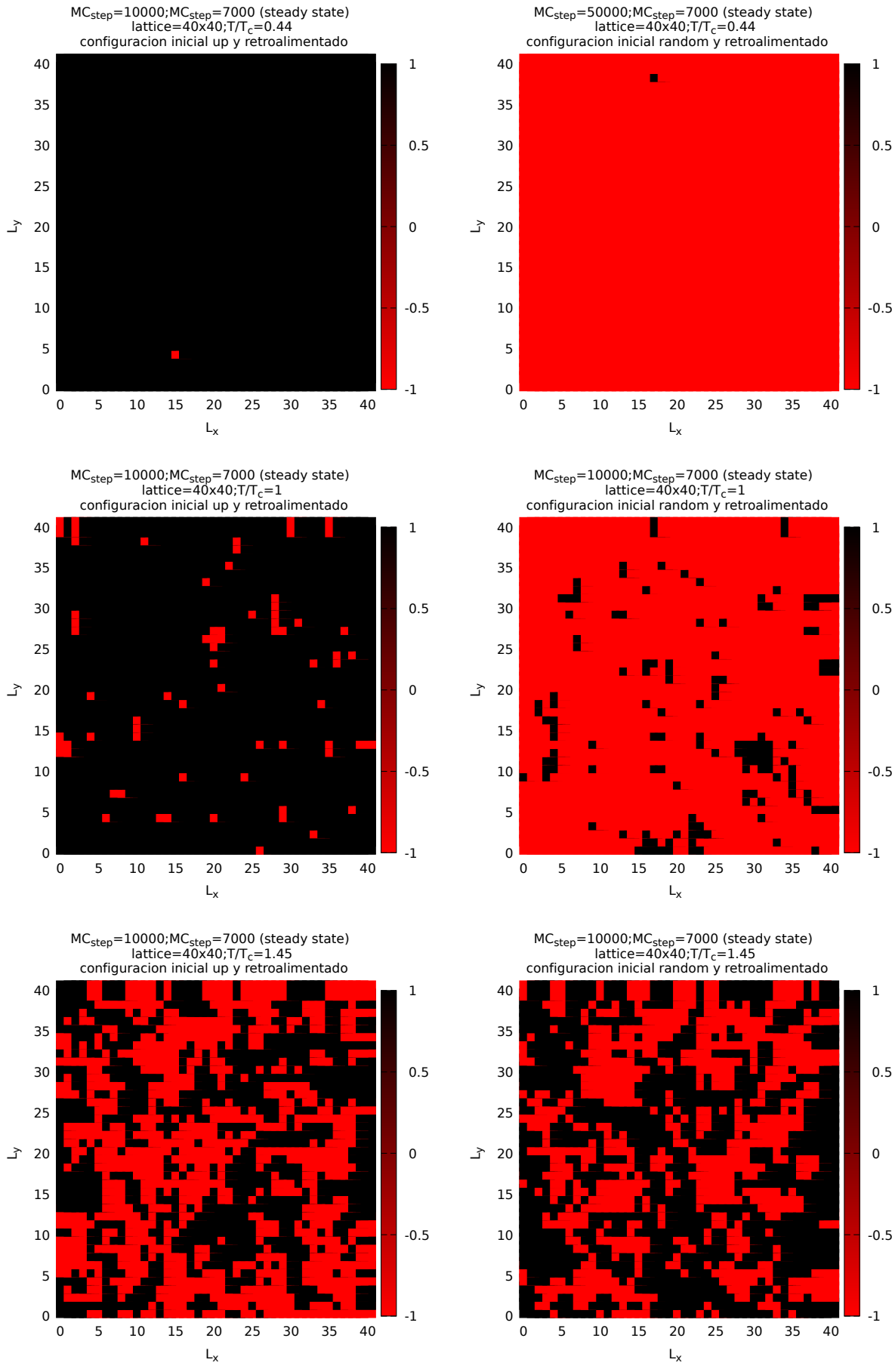

Fig. 13. Configuraciones de spins para distintas temperaturas

Laboratorio 04 - Problema 01 - Códigos

ising_ferromagnetic_model_01.f90

```
! Programa para resolver el inciso a)
! (primero se uso el ising_ferromagnetic_model_03.f90 para determinar MC_step_trans)
! make clean && make ising_ferromagnetic_model_01.o && ./ising_ferromagnetic_model_01.o
program ising_ferromagnetic_model_01
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none

    integer(sp), parameter   :: n=40_sp,m=10_sp                        ! sitios de red por dimension
    integer(sp), parameter   :: MC_step=10000_sp,MC_step_trans=3000_sp ! Monte Carlo step total and transitory
    real(dp),    parameter   :: Tc_adim=2.2676_dp                      ! temperatura de Curie adimensional
    integer(sp), allocatable :: aux_matrix_pbc(:,:)
    integer(sp)              :: i,j,istat
    real(dp)                 :: Madim,U_adim

    allocate(aux_matrix_pbc(n+2,n+2))

    !call initial_spins_configuration(2_sp,n,aux_matrix_pbc) ! genero configuracion inicial random
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc) ! genero configuracion inicial todos up

    ! calculamos energía interna (configuración inicial)
    call average_energy(aux_matrix_pbc,n,U_adim)
    ! calculamos magnetización inicial
    Madim=M_adim(aux_matrix_pbc,n)

    ! temperaturas de equilibrio
    !T0=1;T1=2.2;T2=2.2676;T3=3.3

    21 format(2(A12,x),A12)

    open(10,file='../results/result_01a_T=1_up.dat',status='replace',action='write',iostat=istat)  ! ordenado
    !open(10,file='../results/result_01a_T=1_rnd.dat',status='replace',action='write',iostat=istat) ! desordenado
    write(*,*) 'istat(10file) = ',istat
    write(10,21) 'MC_step','U_adim','M_adim'
    call ising_relax(n,MC_step,MC_step_trans,m,10,aux_matrix_pbc,0._dp,1._dp,Tc_adim,U_adim,Madim)
    close(10)

    ! mostramos mapa de spins antes de Tc
    open(50,file='../results/result_01a_spinmap_T=1.dat',status='replace',action='write',iostat=istat)
    write(*,*) 'istat(50file) = ',istat
    do j=2,n+1;do i=2,n+1
        write(50,'(2(I5,x),I5)') i-1,j-1,aux_matrix_pbc(i,j)
    end do;end do
    close(50)

    open(11,file='../results/result_01a_T=2_up.dat',status='replace',action='write',iostat=istat)  ! ordenado
    !open(11,file='../results/result_01a_T=2_rnd.dat',status='replace',action='write',iostat=istat) ! desordenado
    write(*,*) 'istat(11file) = ',istat
    write(11,21) 'MC_step','U_adim','M_adim'
    call ising_relax(n,MC_step,MC_step_trans,m,11,aux_matrix_pbc,1.1_dp,2._dp,Tc_adim,U_adim,Madim)
    close(11)

    open(12,file='../results/result_01a_T=Tc_up.dat',status='replace',action='write',iostat=istat)  ! ordenado
    !open(12,file='../results/result_01a_T=Tc_rnd.dat',status='replace',action='write',iostat=istat) ! desordenado
    write(*,*) 'istat(12file) = ',istat
    write(12,21) 'MC_step','U_adim','M_adim'
    call ising_relax(n,MC_step,MC_step_trans,m,12,aux_matrix_pbc,2.1_dp,Tc_adim,Tc_adim,U_adim,Madim)
    close(12)

    ! mostramos mapa de spins en Tc
    open(50,file='../results/result_01a_spinmap_T=Tc.dat',status='replace',action='write',iostat=istat)
    write(*,*) 'istat(50file) = ',istat
    do j=2,n+1;do i=2,n+1
        write(50,'(2(I5,x),I5)') i-1,j-1,aux_matrix_pbc(i,j)
    end do;end do
    close(50)


    open(13,file='../results/result_01a_T=3.3_up.dat',status='replace',action='write',iostat=istat)  ! ordenado
    !open(13,file='../results/result_01a_T=3.3_rnd.dat',status='replace',action='write',iostat=istat) ! desordenado
    write(*,*) 'istat(13file) = ',istat
    write(13,21) 'MC_step','U_adim','M_adim'
    call ising_relax(n,MC_step,MC_step_trans,m,13,aux_matrix_pbc,Tc_adim+0.1_dp,3.3_dp,Tc_adim,U_adim,Madim)
    close(13)

    ! mostramos mapa de spins luego de Tc
    open(50,file='../results/result_01a_spinmap_T=3.3.dat',status='replace',action='write',iostat=istat)
    write(*,*) 'istat(50file) = ',istat
    do j=2,n+1;do i=2,n+1
        write(50,'(2(I5,x),I5)') i-1,j-1,aux_matrix_pbc(i,j)
    end do;end do
    close(50)
```

```
82
83      deallocate(aux_matrix_pbc)
84  end program ising_ferromagnetic_model_01
85
86  subroutine ising_relax(n,MC_step,MC_step_trans,m,file_num,aux_matrix_pbc,T_start,T_end,Tc_adim,U_adim,Madim)
87      use module_precision;use module_2D_ferromagnetic_ising
88      implicit none
89      real(dp),    intent(in)     :: T_end,T_start,Tc_adim
90      integer(sp), intent(in)     :: n,m,file_num,MC_step,MC_step_trans
91      integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
92      real(dp),    intent(inout)  :: U_adim,Madim
93      integer(sp)                 :: i,j
94      real(dp)                    :: T_step,T_adim
95      real(dp)                    :: U_med_adim,sigma_U,error_U
96      real(dp)                    :: M_med_adim,sigma_M,error_M
97      real(dp)                    :: s0,s1_U,s2_U,s1_M,s2_M ! variables para hacer estadística
98
99      20 format(I12,x,E12.4,x,E12.4)
100     T_step=abs(T_end-T_start)*(1._dp/real(m-1_sp,dp))
101     ! termalizo hasta una temperatura anterior a la buscada
102     do j=1,m-1
103         T_adim=T_start+T_step*real(j-1_sp,dp)
104         write(*,'(A12,E12.4)') 'T_adim=',T_adim
105         do i=1,MC_step
106             call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
107         end do
108     end do
109     ! termalizo a la temperatura buscada
110     write(*,'(A12,E12.4)') 'T_adim=',T_end
111     s0=0._dp;s1_U=0._dp;s2_U=0._dp;s1_M=0._dp;s2_M=0._dp
112     do i=1,MC_step
113         call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_end,U_adim)
114         Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
115         ! datos para hacer estadística en steady state
116         if (i>=MC_step_trans) then
117             s0=s0+1._dp
118             s2_U=s2_U+U_adim*U_adim;s1_U=s1_U+U_adim ! energía
119             s2_M=s2_M+Madim*Madim;s1_M=s1_M+Madim    ! magnetización
120         end if
121         write(file_num,20) i,U_adim,Madim
122     end do
123
124     !calculamos media,desviacion estándar, varianza y error
125     U_med_adim=s1_U*(1._dp/s0)
126     sigma_U=sqrt((s2_U-s0*U_med_adim*U_med_adim)*(1._dp/(s0-1._dp)))
127     error_U=sigma_U*(1._dp/sqrt(s0-1._dp))
128     M_med_adim=s1_M*(1._dp/s0)
129     sigma_M=sqrt((s2_M-s0*M_med_adim*M_med_adim)*(1._dp/(s0-1._dp)))
130     error_M=sigma_M*(1._dp/sqrt(s0-1._dp))
131
132     write(*,'(A9,I2,A2,I2)')  'Lattice=',n,'x',n
133     write(*,'(A14,E12.4)')    'M_exact_adim=',M_exact_adim(n,T_end,Tc_adim)
134     write(*,'(2(A14,E12.4))') 'M_med_adim=',M_med_adim,'error_M',error_M
135     write(*,'(2(A14,E12.4))') 'U_med_adim=',U_med_adim,'error_U',error_U
136
137 end subroutine ising_relax
```

ising_ferromagnetic_model_02.f90

```
1   ! P01.b (TyMEII G04.P16;G04.P02)
2   ! make clean && make ising_ferromagnetic_model_02.o && ./ising_ferromagnetic_model_02.o
3   program ising_ferromagnetic_model_02
4       use module_precision;use module_2D_ferromagnetic_ising
5       implicit none
6
7       integer(sp), parameter   :: n=40_sp                                  ! sitios de red por dimension
8       integer(sp), parameter   :: MC_step=15000_sp,MC_step_trans=8000_sp  ! Monte Carlo step total and transitory
9       integer(sp), parameter   :: m1=50_sp,m2=100_sp                    ! puntos p/ mallado fino y grueso
10      real(dp),    parameter   :: Tmin_adim=0._dp,Tmax_adim=10._dp        ! temperatura adimensional
11      real(dp),    parameter   :: Tc_adim=2.2676_dp                      ! temperatura de Curie adimensional
12      real(dp),    parameter   :: deltaT_adim=2.2676_dp*0.5_dp           ! intervalo para incremetar ptos
13      integer(sp), allocatable :: aux_matrix_pbc(:,:)
14      integer(sp)              :: i,istat
15      real(dp)                 :: U_adim,U_med_adim,sigma_U,error_U       ! Energía interna
16      real(dp)                 :: Madim,M_med_adim,sigma_M,error_M,Mexact ! Magenitación
17      real(dp)                 :: susc_adim                               ! Susceptibilidad magnética
18      real(dp)                 :: cv                                      ! calor específico
19      real(dp)                 :: binder_cumulant                         ! cumulante de Binder
20      real(dp)                 :: T_adim,T_step1,T_step2
21      real(dp)                 :: time_start,time_end                     ! tiempos de CPU
22
23      open(10,file='../results/result_01b_07_40x40_v2.dat',status='replace',action='write',iostat=istat)
24      !open(10,file='../results/result_01b_07_40x40_osc_magne.dat',status='replace',action='write',iostat=istat)
25      write(*,*) 'istat(10file) = ',istat
26      21 format(8(A12,x),A12);write(10,21) 'T/Tc','U_med','U_error','M_med','M_error','Mexact','cv','susc','Binder'
27      20 format(8(E12.4,x),E12.4)
```

```
28
29      call cpu_time(time_start)
30
31      allocate(aux_matrix_pbc(n+2,n+2))
32      ! genero configuracion inicial (random,descorrelacionada)
33      call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
34      ! calculamos energía interna (configuración inicial)
35      call average_energy(aux_matrix_pbc,n,U_adim)
36      ! calculamos magnetización inicial
37      Madim=M_adim(aux_matrix_pbc,n)
38
39      ! mallado grueso
40      do i=1,m1
41          T_step1=abs(Tc_adim-deltaT_adim-Tmin_adim)*(1._dp/real(m1-1,dp))
42          T_adim=Tmin_adim+T_step1*real(i-1,dp)
43          call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
44                         U_adim,U_med_adim,sigma_U,error_U,&
45                         Madim,M_med_adim,sigma_M,error_M,Mexact,&
46                         susc_adim,cv,binder_cumulant)
47          write(10,20) T_adim*(1._dp/Tc_adim),U_med_adim,error_U,&
48                       M_med_adim,error_M,Mexact,cv,susc_adim,binder_cumulant
49          write(*,*) i
50      end do
51
52      ! mallado fino
53      do i=1,m2
54          T_step2=2_dp*deltaT_adim*(1._dp/real(m2-1,dp))
55          T_adim=(Tc_adim-deltaT_adim)+T_step2*real(i-1,dp)
56          call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
57                         U_adim,U_med_adim,sigma_U,error_U,&
58                         Madim,M_med_adim,sigma_M,error_M,Mexact,&
59                         susc_adim,cv,binder_cumulant)
60          write(10,20) T_adim*(1._dp/Tc_adim),U_med_adim,error_U,&
61                       M_med_adim,error_M,Mexact,cv,susc_adim,binder_cumulant
62          write(*,*) i+m1
63      end do
64
65      ! mallado grueso
66      do i=1,m1
67          T_step1=abs(Tmax_adim-(Tc_adim+deltaT_adim))*(1._dp/real(m1-1,dp))
68          T_adim=(Tc_adim+deltaT_adim)+T_step1*real(i-1,dp)
69          call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
70                         U_adim,U_med_adim,sigma_U,error_U,&
71                         Madim,M_med_adim,sigma_M,error_M,Mexact,&
72                         susc_adim,cv,binder_cumulant)
73          write(10,20) T_adim*(1._dp/Tc_adim),U_med_adim,error_U,&
74                       M_med_adim,error_M,Mexact,cv,susc_adim,binder_cumulant
75          write(*,*) i+m1+m2
76      end do
77
78      close(10)
79      deallocate(aux_matrix_pbc)
80
81      call cpu_time(time_end); write(*,*) 'elapsed time = ',(time_end-time_start)
82  end program ising_ferromagnetic_model_02
83
84  subroutine rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
85                       U_adim,U_med_adim_mom,sigma_U_mom,error_U_mom,&
86                       Madim,M_med_adim_mom,sigma_M_mom,error_M_mom,Mexact,&
87                       susc_adim,cv,binder_cumulant)
88      use module_precision;use module_2D_ferromagnetic_ising
89      implicit none
90
91      integer(sp), intent(in)     :: n,MC_step,MC_step_trans
92      real(dp),    intent(in)     :: T_adim,Tc_adim
93      integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
94      real(dp),    intent(inout)  :: U_adim,U_med_adim_mom,sigma_U_mom,error_U_mom       ! Energía interna
95      real(dp),    intent(inout)  :: Madim,M_med_adim_mom,sigma_M_mom,error_M_mom,Mexact ! Magenitación
96      real(dp),    intent(inout)  :: susc_adim                                          ! Susceptibilidad magnética
97      real(dp),    intent(inout)  :: cv                                                 ! calor específico
98      real(dp),    intent(inout)  :: binder_cumulant
99
100     integer(sp), parameter   :: m_exp=50_sp               ! numero de experimentos
101     real(dp)                 :: s0,s1_U,s2_U,s1_M,s2_M,s4_M   ! variables para hacer estadística
102     real(dp)                 :: s4_M_mom,s2_M_mom,s2_U_mom,sigma_U_mom_v2
103     integer(sp)              :: i,j
104     real(dp),    allocatable :: U_med_adim_vector(:),M_med_adim_vector(:)
105
106     allocate(U_med_adim_vector(m_exp),M_med_adim_vector(m_exp))
107     s0=real(MC_step-MC_step_trans,dp)
108
109     s4_M_mom=0._dp;s2_M_mom=0._dp;s2_U_mom=0._dp;sigma_U_mom_v2=0._dp
110     do j=1,m_exp
111         s1_U=0._dp;s2_U=0._dp;s1_M=0._dp;s2_M=0._dp;s4_M=0._dp
112         do i=1,MC_step
113             call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
114             Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
```

```
            ! datos para hacer estadística en steady state
            if (i>=MC_step_trans) then
                s2_U=s2_U+U_adim*U_adim;s1_U=s1_U+U_adim ! primer y segundo momento energía
                s2_M=s2_M+Madim*Madim;s1_M=s1_M+Madim    ! primer y segundo momento magnetización
                s4_M=s4_M+Madim*Madim*Madim*Madim        ! cuarto momento magnetización
            end if
        end do
        ! calculamos valores medios
        U_med_adim_vector(j)=s1_U*(1._dp/s0)
        M_med_adim_vector(j)=s1_M*(1._dp/s0)
        s4_M_mom=s4_M_mom+s4_M*(1._dp/s0);s2_M_mom=s2_M_mom+s2_M*(1._dp/s0)
        s2_U_mom=s2_U_mom+s2_U*(1._dp/s0)
    end do

    ! calculamos media de medias,desviacion estándar, varianza y error
    U_med_adim_mom=(1._dp/real(m_exp,dp))*sum(U_med_adim_vector(:))
    sigma_U_mom_v2=sqrt(real(m_exp,dp)*s2_U_mom-sum(U_med_adim_vector(:))*sum(U_med_adim_vector(:))*&
                   (1._dp/(real(m_exp,dp)*real(m_exp-1,dp))))
    U_med_adim_vector(:)=(U_med_adim_vector(:)-U_med_adim_mom)
    U_med_adim_vector(:)=U_med_adim_vector(:)*U_med_adim_vector(:)
    sigma_U_mom=sqrt((1._dp/real(m_exp,dp))*sum(U_med_adim_vector(:)))
    error_U_mom=sigma_U_mom*(1._dp/sqrt(real(m_exp,dp)))

    M_med_adim_mom=(1._dp/real(m_exp,dp))*sum(M_med_adim_vector(:))
    M_med_adim_vector(:)=(M_med_adim_vector(:)-M_med_adim_mom)
    M_med_adim_vector(:)=M_med_adim_vector(:)*M_med_adim_vector(:)
    sigma_M_mom=sqrt((1._dp/real(m_exp,dp))*sum(M_med_adim_vector(:)))
    error_M_mom=sigma_M_mom*(1._dp/sqrt(real(m_exp,dp)))
    if (T_adim==0._dp) then;cv=0._dp;susc_adim=0._dp
    else
        ! cv=(1._dp/T_adim)*sigma_U_mom*sigma_U_mom
        cv=(1._dp/T_adim)*sigma_U_mom_v2*sigma_U_mom_v2
        susc_adim=(1._dp/T_adim)*sigma_M_mom*sigma_M_mom
    end if
    Mexact=M_exact_adim(n,T_adim,Tc_adim)
    binder_cumulant=1._dp-real(m_exp,dp)*s4_M_mom*(1._dp/(3._dp*s2_M_mom*s2_M_mom))

    deallocate(U_med_adim_vector,M_med_adim_vector)

end subroutine rlx_ising
```

ising_ferromagnetic_model_03.f90

```
! Programa para determinar el tiempo de termalización
!  se realizan corridas del método de MC "metropolis" para distintas temperaturas
!  tanto cerca como lejos de la temperatura crítica.
! make clean && make ising_ferromagnetic_model_03.o && ./ising_ferromagnetic_model_03.o
program ising_ferromagnetic_model_03
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none

    integer(sp), parameter   :: MC_step=10000_sp     ! Monte Carlo step total and transitory
    integer(sp), parameter   :: type_analysis=1_sp
    real(dp),    parameter   :: Tc_adim=2.2676_dp    ! temperatura de Curie adimensional
    integer(sp), allocatable :: aux_matrix_pbc(:,:)
    integer(sp)              :: n=40_sp              ! sitios de red por dimension
    integer(sp)              :: istat,m              ! cantidad puntos para definir el paso de temperaturas
    real(dp)                 :: Madim,U_adim

    21 format(2(A12,x),A12)

    select case (type_analysis)
        case(1) ! termalizar hasta varias temperaturas (según pide el inciso b)
            ! temperaturas de equilibrio
            !T0=1;T1=2.2;T2=2.2676;T3=3.3
            n=40_sp;m=10_sp
            allocate(aux_matrix_pbc(n+2,n+2))
            ! genero configuracion inicial random
            call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
            ! calculamos energía interna (configuración inicial)
            call average_energy(aux_matrix_pbc,n,U_adim)
            ! calculamos magnetización inicial
            Madim=M_adim(aux_matrix_pbc,n)

            open(10,file='../results/result_01a_MCS_trans_T0.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(10file) = ',istat
            write(10,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,10,aux_matrix_pbc,0._dp,1._dp,U_adim,Madim)
            close(10)

            open(11,file='../results/result_01a_MCS_trans_T1.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(11file) = ',istat
            write(11,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,11,aux_matrix_pbc,1.1_dp,2._dp,U_adim,Madim)
            close(11)

```

```
            open(12,file='../results/result_01a_MCS_trans_T2.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(12file) = ',istat
            write(12,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,12,aux_matrix_pbc,2.1_dp,Tc_adim,U_adim,Madim)
            close(12)

            open(13,file='../results/result_01a_MCS_trans_T3.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(13file) = ',istat
            write(13,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,13,aux_matrix_pbc,Tc_adim+0.1_dp,3.3_dp,U_adim,Madim)
            close(13)

            deallocate(aux_matrix_pbc)
        case(2) ! termalizar hasta una temperatura fija (para analizar efectos de tamaño)
            ! Tc1=1.19Tc;Tc2=1.115Tc;Tc3=1.118Tc

            m=10_sp;n=10_sp;allocate(aux_matrix_pbc(n+2,n+2))
            call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
            call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
            open(14,file='../results/result_01b_T=Tcaprox_10x10_osc_magne.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(14file) = ',istat;write(14,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,14,aux_matrix_pbc,0._dp,Tc_adim*1.19_dp,U_adim,Madim)
            close(14);deallocate(aux_matrix_pbc)

            n=20_sp;allocate(aux_matrix_pbc(n+2,n+2))
            call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
            call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
            open(15,file='../results/result_01b_T=Tcaprox_20x20_osc_magne.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(15file) = ',istat;write(15,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,15,aux_matrix_pbc,0._dp,Tc_adim*1.115_dp,U_adim,Madim)
            close(15);deallocate(aux_matrix_pbc)

            n=40_sp;allocate(aux_matrix_pbc(n+2,n+2))
            call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
            call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
            open(16,file='../results/result_01b_T=Tcaprox_40x40_osc_magne.dat',status='replace',action='write',iostat=istat)
            write(*,*) 'istat(16file) = ',istat;write(16,21) 'MC_step','U_adim','M_adim'
            call ising_relax(n,MC_step,m,16,aux_matrix_pbc,0._dp,Tc_adim*1.115_dp,U_adim,Madim)
            close(16);deallocate(aux_matrix_pbc)
    end select
end program ising_ferromagnetic_model_03

subroutine ising_relax(n,MC_step,m,file_num,aux_matrix_pbc,T_start,T_end,U_adim,Madim)
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none
    real(dp),    intent(in)     :: T_end,T_start
    integer(sp), intent(in)     :: n,m,file_num,MC_step
    integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
    real(dp),    intent(inout)  :: U_adim,Madim
    integer(sp)                 :: i,j
    real(dp)                    :: T_step,T_adim

    20 format(I12,x,E12.4,x,E12.4)

    bucle_01: do j=1,m ! cantidad de Temperaturas diferentes
        T_step=abs(T_end-T_start)*(1._dp/real(m-1_sp,dp))
        T_adim=T_start+T_step*real(j-1_sp,dp)
        write(*,'(A12,E12.4)') 'T_adim=',T_adim
        do i=1,MC_step
            call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
            Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
            if (j==m) write(file_num,20) i,U_adim,Madim
        end do
    end do bucle_01

end subroutine ising_relax
```

ising_ferromagnetic_model_04.f90

```
! Programa para calcular histogramas (inciso c)
! make clean && make ising_ferromagnetic_model_04.o && ./ising_ferromagnetic_model_04.o
program ising_ferromagnetic_model_04
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none

    integer(sp), parameter   :: MC_step=10000_sp,MC_step_rlx=3000_sp     ! Monte Carlo step total and transitory
    real(dp),    parameter   :: Tc_adim=2.2692_dp    ! temperatura de Curie adimensional
    integer(sp), allocatable :: aux_matrix_pbc(:,:)
    real(dp),    allocatable :: Madim_vector(:)    ! vector de magnetización para histograma
    integer(sp)              :: n                    ! sitios de red por dimension
    integer(sp)              :: m              ! cantidad puntos para definir el paso de temperaturas
    real(dp)                 :: Madim,U_adim


    m=10_sp;n=10_sp;allocate(aux_matrix_pbc(n+2,n+2));allocate(Madim_vector(MC_step-MC_step_rlx))
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
```

```
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.1999_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_lessTc_10x10.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.5_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_greaterTc_10x10.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    deallocate(aux_matrix_pbc);deallocate(Madim_vector)

    m=10_sp;n=20_sp;allocate(aux_matrix_pbc(n+2,n+2));allocate(Madim_vector(MC_step-MC_step_rlx))
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.11_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_lessTc_20x20.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.5_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_greaterTc_20x20.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    deallocate(aux_matrix_pbc);deallocate(Madim_vector)

    m=10_sp;n=40_sp;allocate(aux_matrix_pbc(n+2,n+2));allocate(Madim_vector(MC_step-MC_step_rlx))
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.118_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_lessTc_40x40.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    call average_energy(aux_matrix_pbc,n,U_adim);Madim=M_adim(aux_matrix_pbc,n)
    call ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,0._dp,Tc_adim*1.5_dp,U_adim,Madim,Madim_vector)
    call histogram('../results/result_01c_historgram_greaterTc_40x40.dat',10_sp,Madim_vector,MC_step-MC_step_rlx)
    deallocate(aux_matrix_pbc);deallocate(Madim_vector)

end program ising_ferromagnetic_model_04

subroutine ising_relax(n,MC_step,MC_step_rlx,m,aux_matrix_pbc,T_start,T_end,U_adim,Madim,Madim_vector)
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none
    real(dp),    intent(in)     :: T_end,T_start
    integer(sp), intent(in)     :: n,m,MC_step,MC_step_rlx
    integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
    real(dp),    intent(inout)  :: U_adim,Madim
    real(dp),    intent(inout)    :: Madim_vector(MC_step-MC_step_rlx)
    integer(sp)                 :: i,j
    real(dp)                    :: T_step,T_adim

    bucle_01: do j=1,m ! cantidad de Temperaturas diferentes
        T_step=abs(T_end-T_start)*(1._dp/real(m-1_sp,dp))
        T_adim=T_start+T_step*real(j-1_sp,dp)
        do i=1,MC_step_rlx
            call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
            Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
        end do
        do i=MC_step_rlx+1,MC_step
            call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
            Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
            if (j==m) Madim_vector(i-MC_step_rlx)=Madim
        end do
    end do bucle_01

end subroutine ising_relax

! subrutina para crear e imprimir histograma
subroutine histogram(file_name,file_number,x_vector,x_dim)
    use module_precision

    implicit none
    character(len=*), intent(in) :: file_name
    integer(sp),      intent(in) :: file_number
    integer(sp),      intent(in) :: x_dim           ! dimension
    real(dp),         intent(in) :: x_vector(x_dim) ! normalized data

    integer(sp), parameter   :: n_bins=1000_sp   ! numbers of bins (JUGAR CON ESTE VALOR)
    real(dp),    allocatable :: bins_points(:)  ! points between bins vector
    integer(sp), allocatable :: counter(:)      ! counter vector of bins
    real(sp)                 :: max_value       ! maximun counter value
    real(dp)                 :: min_bin_point,max_bin_point
    real(dp)                 :: bins_step       ! step of points between bins
    integer(sp)              :: i,j,istat       ! loop and control variables

    ! armamos el vector de bins
    allocate(bins_points(n_bins+1),counter(n_bins))
    max_bin_point=1._dp;min_bin_point=-1_dp
    bins_step=abs(max_bin_point-min_bin_point)*(1._dp/n_bins)
    do i=1,n_bins+1;bins_points(i)=min_bin_point+bins_step*real(i-1,dp);end do

    ! llenamos el vector contador de bins
    do i=1,n_bins
        counter(i)=0
```

```
        do j=1,x_dim
            if ((bins_points(i)<=x_vector(j)).and.(bins_points(i+1)>=x_vector(j))) then
                counter(i)=counter(i)+1
            endif
    enddo;enddo

    max_value=real(maxval(counter(:)),dp)
    if (max_value==0._dp) write(*,*) 'math error'

    ! escribimos el histograma
    open(file_number,file=file_name,status='replace',action='write',iostat=istat)
    21 format(A12,x,A12); 20 format(E12.4,x,E12.4);write(*,*) 'istat=', istat
    write(file_number,21) 'bins points','counter'
    do i = 1,n_bins; write(file_number,20) bins_points(i),real(counter(i),dp)*(1._dp/max_value); enddo
    close(file_number)
end subroutine histogram
```

ising_ferromagnetic_model_05.f90

```
! P01.d (calculamos los cumulantes de binder)
! make clean && make ising_ferromagnetic_model_05.o && ./ising_ferromagnetic_model_05.o
program ising_ferromagnetic_model_05
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none

    integer(sp), parameter   :: n=10_sp                                  ! sitios de red por dimension
    integer(sp), parameter   :: MC_step=5000_sp,MC_step_trans=3000_sp    ! Monte Carlo step total and transitory
    integer(sp), parameter   :: m1=25_sp,m2=50_sp                        ! puntos p/ mallado fino y grueso
    real(dp),    parameter   :: Tmin_adim=0._dp,Tmax_adim=10._dp         ! temperatura adimensional
    real(dp),    parameter   :: Tc_adim=2.2692_dp                        ! temperatura de Curie adimensional
    real(dp),    parameter   :: deltaT_adim=Tc_adim*0.5_dp               ! intervalo para incremetar ptos
    integer(sp), allocatable :: aux_matrix_pbc(:,:)
    integer(sp)              :: i,istat
    real(dp)                 :: U_adim,U_med_adim,sigma_U,error_U        ! Energía interna
    real(dp)                 :: Madim,M_med_adim,sigma_M,error_M,Mexact  ! Magenitación
    real(dp)                 :: susc_adim                                ! Susceptibilidad magnética
    real(dp)                 :: cv                                       ! calor específico
    real(dp)                 :: T_adim,T_step1,T_step2
    real(dp)                 :: time_start,time_end                      ! tiempos de CPU
    real(dp)                 :: binder_cumulant

    open(10,file='../results/result_01d_binder_cumulant_10x10.dat',status='replace',action='write',iostat=istat)
    !open(10,file='../results/result_01d_binder_cumulant_20x20.dat',status='replace',action='write',iostat=istat)
    !open(10,file='../results/result_01d_binder_cumulant_40x40.dat',status='replace',action='write',iostat=istat)
    write(*,*) 'istat(10file) = ',istat
    21 format(A12,x,A12);write(10,21) 'T/Tc','binder_cumulant'
    20 format(E12.4,x,E12.4)

    call cpu_time(time_start)

    allocate(aux_matrix_pbc(n+2,n+2))
    ! genero configuracion inicial (random,descorrelacionada)
    call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
    ! calculamos energía interna (configuración inicial)
    call average_energy(aux_matrix_pbc,n,U_adim)
    ! calculamos magnetización inicial
    Madim=M_adim(aux_matrix_pbc,n)

    ! mallado grueso
    do i=1,m1
        T_step1=abs(Tc_adim-deltaT_adim-Tmin_adim)*(1._dp/real(m1-1,dp))
        T_adim=Tmin_adim+T_step1*real(i-1,dp)
        call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
                       U_adim,U_med_adim,sigma_U,error_U,&
                       Madim,M_med_adim,sigma_M,error_M,Mexact,&
                       susc_adim,cv,binder_cumulant)
        write(10,20) T_adim*(1._dp/Tc_adim),binder_cumulant
    end do
    write(*,*) 'primer bucle terminado...'

    ! mallado fino
    do i=1,m2
        T_step2=2_dp*deltaT_adim*(1._dp/real(m2-1,dp))
        T_adim=(Tc_adim-deltaT_adim)+T_step2*real(i-1,dp)
        call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
                       U_adim,U_med_adim,sigma_U,error_U,&
                       Madim,M_med_adim,sigma_M,error_M,Mexact,&
                       susc_adim,cv,binder_cumulant)
        write(10,20) T_adim*(1._dp/Tc_adim),binder_cumulant
    end do
    write(*,*) 'segundo bucle terminado...'

    ! mallado grueso
    do i=1,m1
        T_step1=abs(Tmax_adim-(Tc_adim+deltaT_adim))*(1._dp/real(m1-1,dp))
        T_adim=(Tc_adim+deltaT_adim)+T_step1*real(i-1,dp)
```

```
        call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
                       U_adim,U_med_adim,sigma_U,error_U,&
                       Madim,M_med_adim,sigma_M,error_M,Mexact,&
                       susc_adim,cv,binder_cumulant)
        write(10,20) T_adim*(1._dp/Tc_adim),binder_cumulant
    end do

    close(10)
    deallocate(aux_matrix_pbc)

    call cpu_time(time_end); write(*,*) 'elapsed time = ',(time_end-time_start)
end program ising_ferromagnetic_model_05

subroutine rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tc_adim,&
                     U_adim,U_med_adim_mom,sigma_U_mom,error_U_mom,&
                     Madim,M_med_adim_mom,sigma_M_mom,error_M_mom,Mexact,&
                     susc_adim,cv,binder_cumulant)
    use module_precision;use module_2D_ferromagnetic_ising
    implicit none

    integer(sp), intent(in)     :: n,MC_step,MC_step_trans
    real(dp),    intent(in)     :: T_adim,Tc_adim
    integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
    real(dp),    intent(inout)  :: U_adim,U_med_adim_mom,sigma_U_mom,error_U_mom       ! Energía interna
    real(dp),    intent(inout)  :: Madim,M_med_adim_mom,sigma_M_mom,error_M_mom,Mexact        ! Magenitación
    real(dp),    intent(inout)  :: susc_adim                                  ! Susceptibilidad magnética
    real(dp),    intent(inout)  :: cv                                         ! calor específico
    real(dp),    intent(inout)  :: binder_cumulant

    integer(sp), parameter   :: m_exp=50_sp                ! numero de experimentos
    real(dp)                 :: s0,s1_U,s2_U,s1_M,s2_M,s4_M   ! variables para hacer estadística
    real(dp)                 :: s4_M_mom,s2_M_mom
    integer(sp)              :: i,j
    real(dp),    allocatable :: U_med_adim_vector(:),M_med_adim_vector(:)

    allocate(U_med_adim_vector(m_exp),M_med_adim_vector(m_exp))
    s0=real(MC_step-MC_step_trans,dp)

    s4_M_mom=0._dp;s2_M_mom=0._dp
    do j=1,m_exp
        s1_U=0._dp;s2_U=0._dp;s1_M=0._dp;s2_M=0._dp;s4_M=0._dp
        do i=1,MC_step
            call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
            Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
            ! datos para hacer estadística en steady state
            if (i>=MC_step_trans) then
                s2_U=s2_U+U_adim*U_adim;s1_U=s1_U+U_adim ! primer y segundo momento energía
                s2_M=s2_M+Madim*Madim;s1_M=s1_M+Madim    ! primer y segundo momento magnetización
                s4_M=s4_M+Madim*Madim*Madim*Madim        ! cuarto momento magnetización
            end if
        end do
        ! calculamos valores medios
        U_med_adim_vector(j)=s1_U*(1._dp/s0)
        M_med_adim_vector(j)=s1_M*(1._dp/s0)
        s4_M_mom=s4_M_mom+s4_M*(1._dp/s0);s2_M_mom=s2_M_mom+s2_M*(1._dp/s0)
    end do

    ! calculamos media de medias,desviacion estándar, varianza y error
    U_med_adim_mom=(1._dp/real(m_exp))*sum(U_med_adim_vector(:))
    U_med_adim_vector(:)=(U_med_adim_vector(:)-U_med_adim_mom)
    U_med_adim_vector(:)=U_med_adim_vector(:)*U_med_adim_vector(:)
    sigma_U_mom=sqrt((1._dp/real(m_exp))*sum(U_med_adim_vector(:)))
    error_U_mom=sigma_U_mom*(1._dp/sqrt(real(m_exp)))

    M_med_adim_mom=(1._dp/real(m_exp))*sum(M_med_adim_vector(:))
    M_med_adim_vector(:)=(M_med_adim_vector(:)-M_med_adim_mom)
    M_med_adim_vector(:)=M_med_adim_vector(:)*M_med_adim_vector(:)
    sigma_M_mom=sqrt((1._dp/real(m_exp))*sum(M_med_adim_vector(:)))
    error_M_mom=sigma_M_mom*(1._dp/sqrt(real(m_exp)))
    if (T_adim==0._dp) then;cv=0._dp;susc_adim=0._dp
    else
        cv=(1._dp/T_adim)*sigma_U_mom*sigma_U_mom
        susc_adim=(1._dp/T_adim)*sigma_M_mom*sigma_M_mom
    end if
    Mexact=M_exact_adim(n,T_adim,Tc_adim)
    binder_cumulant=1._dp-real(m_exp,dp)*s4_M_mom*(1._dp/(3._dp*s2_M_mom*s2_M_mom))

    deallocate(U_med_adim_vector,M_med_adim_vector)

end subroutine rlx_ising
```

ising_ferromagnetic_model_06.f90

```
! P01.e (calculamos función de autocorrelación de la energía y magnetización)
! make clean && make ising_ferromagnetic_model_06.o && ./ising_ferromagnetic_model_06.o
program ising_ferromagnetic_model_06
```

```
4       use module_precision;use module_2D_ferromagnetic_ising
5       implicit none
6
7       integer(sp), parameter   :: n=10_sp                ! sitios de red por dimension
8       integer(sp), parameter   :: MC_step_trans=10000_sp  ! Monte Carlo step transitory
9       integer(sp), parameter   :: m=30_sp                ! puntos p/ para deltas de temperaturas
10      integer(sp), parameter   :: tau_corr=1000_sp       ! tiempo máximo de correlación
11      integer(sp), parameter   :: num_of_terms=1000000_sp ! cantidad de terminos para la última autocorr
12      integer(sp), parameter   :: MC_step=num_of_terms+tau_corr+MC_step_trans-1 ! total Monte Carlo step
13      real(dp),    parameter   :: Tmin_adim=0._dp,Tmax_adim=2.22_dp ! temperatura adimensional
14      integer(sp), allocatable :: aux_matrix_pbc(:,:)
15      real(dp),    allocatable :: autocor_vector_mom_U(:),autocor_vector_mom_M(:)   ! autocorrelación
16      integer(sp)              :: i,istat
17      real(dp)                 :: U_adim,Madim    ! Energía interna y Magenitación
18      real(dp)                 :: T_adim,T_step
19
20      !open(10,file='../results/result_01e_20x20_autocorr_T2.0.dat',status='replace',action='write',iostat=istat)
21      open(10,file='../results/result_01e_10x10_autocorr_T2.22.dat',status='replace',action='write',iostat=istat)
22      !open(10,file='../results/result_01e_40x40_autocorr_T2.2676.dat',status='replace',action='write',iostat=istat)
23      ! open(10,file='../results/result_01e_10x10_autocorr_T2.5.dat',status='replace',action='write',iostat=istat)
24      ! open(10,file='../results/result_01e_10x10_autocorr_T3.3.dat',status='replace',action='write',iostat=istat)
25      write(*,*) 'istat(10file) = ',istat
26      21 format(2(A12,x),A12);write(10,21) 'tau_corr','autocorr_U','autocorr_M'
27      20 format(I12,x,E12.4,x,E12.4)
28
29      allocate(aux_matrix_pbc(n+2,n+2))
30      allocate(autocor_vector_mom_U(tau_corr),autocor_vector_mom_M(tau_corr))
31      ! genero configuracion inicial (random,descorrelacionada)
32      call initial_spins_configuration(1_sp,n,aux_matrix_pbc)
33      ! calculamos energía interna (configuración inicial)
34      call average_energy(aux_matrix_pbc,n,U_adim)
35      ! calculamos magnetización inicial
36      Madim=M_adim(aux_matrix_pbc,n)
37
38      do i=1,m
39          T_step=abs(Tmax_adim-Tmin_adim)*(1._dp/real(m-1,dp))
40          T_adim=Tmin_adim+T_step*real(i-1,dp)
41          if (i==m) T_adim=Tmax_adim
42          call rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tmax_adim,&
43                         U_adim,Madim,autocor_vector_mom_U,autocor_vector_mom_M,&
44                         tau_corr,num_of_terms)
45          write(*,'(I3,x,A3,x,I3)') i,'de',m
46      end do
47
48      do i=1,tau_corr
49          write(10,20) i,autocor_vector_mom_U(i),autocor_vector_mom_M(i)
50      end do
51
52      close(10)
53      deallocate(aux_matrix_pbc)
54      deallocate(autocor_vector_mom_U);deallocate(autocor_vector_mom_M)
55  end program ising_ferromagnetic_model_06
56
57  subroutine rlx_ising(n,aux_matrix_pbc,MC_step,MC_step_trans,T_adim,Tmax_adim,&
58                       U_adim,Madim,autocor_vector_mom_U,autocor_vector_mom_M,&
59                       tau_corr,num_of_terms)
60      use module_precision;use module_2D_ferromagnetic_ising
61      implicit none
62
63      integer(sp), intent(in)     :: n,MC_step,MC_step_trans,tau_corr,num_of_terms
64      real(dp),    intent(in)     :: T_adim,Tmax_adim
65      integer(sp), intent(inout)  :: aux_matrix_pbc(n+2,n+2)
66      real(dp),    intent(inout)  :: U_adim       ! Energía interna
67      real(dp),    intent(inout)  :: Madim ! Magenitación
68      real(dp),    intent(inout)  :: autocor_vector_mom_U(tau_corr),autocor_vector_mom_M(tau_corr)
69
70      integer(sp), parameter   :: m_exp=3_sp ! numero de experimentos
71      integer(sp)              :: i,j
72      real(dp),    allocatable :: autocor_vector_U(:),aux_vector1_U(:),aux_vector2_U(:),mask_vector_U(:)
73      real(dp),    allocatable :: autocor_vector_M(:),aux_vector1_M(:),aux_vector2_M(:),mask_vector_M(:)
74      real(dp)                 :: U_med,var_U,M_med,var_M ! valor media y varianza (autocorrelación)
75
76      allocate(autocor_vector_U(tau_corr),aux_vector1_U(tau_corr),&
77               aux_vector2_U(tau_corr),mask_vector_U(tau_corr))
78      allocate(autocor_vector_M(tau_corr),aux_vector1_M(tau_corr),&
79               aux_vector2_M(tau_corr),mask_vector_M(tau_corr))
80      if (T_adim==Tmax_adim) then
81          autocor_vector_mom_U(:)=0._dp;U_med=0._dp;var_U=0._dp
82          autocor_vector_mom_M(:)=0._dp;M_med=0._dp;var_M=0._dp
83      end if
84      do j=1,m_exp
85          do i=1,MC_step
86              call MC_step_relaxation(1_sp,n,aux_matrix_pbc,T_adim,U_adim)
87              Madim=M_adim(aux_matrix_pbc,n) ! Magnetización
88              ! datos para hacer estadística en steady state
89              if (i>=MC_step_trans.and.T_adim==Tmax_adim) then! calcular autocorr para la Temp requerida
90                  call func_autocor(U_adim,i-MC_step_trans+1,tau_corr,num_of_terms,&
```

```
                                autocor_vector_U,aux_vector1_U,mask_vector_U,aux_vector2_U,&
                                U_med,var_U)
                call func_autocor(Madim,i-MC_step_trans+1,tau_corr,num_of_terms,&
                                autocor_vector_M,aux_vector1_M,mask_vector_M,aux_vector2_M,&
                                M_med,var_M)
            end if
        end do
        !autocorrelación
        if (T_adim==Tmax_adim) then
            autocor_vector_U(:)=(autocor_vector_U(:)-U_med*U_med)*(1._dp/var_U)
            autocor_vector_mom_U(:)=autocor_vector_mom_U(:)+autocor_vector_U(:)
            autocor_vector_M(:)=(autocor_vector_M(:)-M_med*M_med)*(1._dp/var_M)
            autocor_vector_mom_M(:)=autocor_vector_mom_M(:)+autocor_vector_M(:)
        end if
    end do
    deallocate(aux_vector1_U,aux_vector2_U,mask_vector_U)
    deallocate(aux_vector1_M,aux_vector2_M,mask_vector_M)
    if (T_adim==Tmax_adim) then
        autocor_vector_mom_U(:)=(1._dp/real(m_exp,dp))*autocor_vector_mom_U(:)
        autocor_vector_mom_M(:)=(1._dp/real(m_exp,dp))*autocor_vector_mom_M(:)
    end if
    deallocate(autocor_vector_U,autocor_vector_M)
end subroutine rlx_ising
```

module_2D_ferromagnetic_ising.f90

```
module module_2D_ferromagnetic_ising
    use module_precision;use module_mt19937, only: sgrnd,grnd
    implicit none
    contains

    subroutine initial_spins_configuration(T_init_type,n,aux_matrix_pbc)
        integer(sp), intent(in)    :: T_init_type,n
        integer(sp), intent(inout) :: aux_matrix_pbc(n+2,n+2)

        integer(sp) :: seed,seed_val(8),i,j
        real(dp)    :: nrand

        call date_and_time(values=seed_val)
        seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5);call sgrnd(seed)

        ! creamos matriz de spin con PBC
        select case (T_init_type)
            case(1) ! genero configuracion inicial (random ó T(inicial)=Infinity)
                do j=2,n+1;do i=2,n+1
                    nrand=real(grnd(),dp)
                    if (nrand<0.5_dp) then;aux_matrix_pbc(i,j)=-1_sp
                    else; aux_matrix_pbc(i,j)=1_sp;end if
                end do;end do
                call pbc(aux_matrix_pbc,n,5_sp) ! cambio de bordes
            case(2) ! genero configuracion inicial (ordenada ó T(inicial)=0)
                aux_matrix_pbc(:,:) = 1_sp ! todos los spins "up"
            case(3) ! genero configuracion inicial (ordenada ó T(inicial)=0)
                aux_matrix_pbc(:,:) = -1_sp ! todos los spins "down"
        end select

    end subroutine initial_spins_configuration

    ! Energia en unidades de J_int (coeficiente de interacción >0 => ferromagnetic)
    subroutine average_energy(aux_matrix_pbc,n,U)
        integer(sp), intent(in)    :: n
        integer(sp), intent(in)    :: aux_matrix_pbc(1:n+2,1:n+2)
        real(dp),    intent(inout) :: U
        integer(sp) :: i,j

        U=0._dp
        do j=2,n+1
            do i=2,n+1
                U=U-real(aux_matrix_pbc(i,j)*(aux_matrix_pbc(i,j+1)+aux_matrix_pbc(i,j-1)&
                    & +aux_matrix_pbc(i+1,j)+aux_matrix_pbc(i-1,j)),dp)
            end do
        end do
        U=U*0.5_dp ! suponemos J_int >0 (to compensate over-counting)
    end subroutine average_energy

    ! Delta energético por "flipear" un spin, en unidades de J_int
    function U_delta_adim(spin_value,spin1,spin2,spin3,spin4)
        integer(sp), intent(in) :: spin_value,spin1,spin2,spin3,spin4
        integer(dp)             :: U_delta_adim
        U_delta_adim=2_dp*real(spin_value*(spin1+spin2+spin3+spin4),dp)
    end function U_delta_adim

    ! MAGNETIZACIÓN APROXIMADA (específica ó por unidad de partícula)
    function M_adim(aux_matrix_pbc,n)
        integer(sp), intent(in) :: n,aux_matrix_pbc(n+2,n+2)
        integer(sp)             :: i,j
```

```
61          real(dp)                :: M_adim
62          M_adim=0._dp
63          do j=2,n+1;do i=2,n+1
64              M_adim=M_adim+real(aux_matrix_pbc(i,j),dp)
65          end do;end do
66          M_adim=(1._dp/(n*n))*abs(M_adim) ! con valor absoluto
67          !M_adim=(1._dp/(n*n))*M_adim ! sin valor absoluto
68      end function M_adim
69
70      ! MAGNETIZACIÓN EXACTA
71      function M_exact_adim(n,T_adim,Tc_adim)
72          integer(sp), intent(in) :: n
73          real(dp),    intent(in) :: T_adim,Tc_adim
74          real(dp)                :: zz,M_exact_adim
75          cond1: if (T_adim<Tc_adim) then
76              if (T_adim==0._dp) then;M_exact_adim=1._dp;exit cond1;end if
77              if (4*(1._dp/T_adim)<abs(log(tiny(1._dp)))) then
78                  zz=exp(-4*(1._dp/T_adim)) ! zz=z**2
79              else;zz=0._dp;end if
80              M_exact_adim=((1+zz)**0.25_dp)*((1-6*zz+zz*zz)**0.125_dp)*(1._dp/sqrt(1._dp-zz))
81          else;M_exact_adim=0._dp;end if cond1
82      end function M_exact_adim
83
84      ! CALCULO DE RELAJACIÓN CON MÉTODO MONTE CARLO METRÓPOLIS
85      subroutine MC_step_relaxation(MC_step_type,n,aux_matrix_pbc,T_adim,U_adim)
86
87          integer(sp), intent(in)    :: MC_step_type,n            ! Tipo de MC relaxation y dimension
88          integer(sp), intent(inout) :: aux_matrix_pbc(n+2,n+2)   ! Matriz de spins
89          real(dp),    intent(inout) :: U_adim                    ! Energía interna
90          real(dp),    intent(in)    :: T_adim                    ! Temperatura de equilibrio
91
92          real(dp)    :: deltaU_adim
93          real(dp)    :: nrand
94          integer(sp) :: seed,seed_val(8),i,j,k
95
96          call date_and_time(values=seed_val)
97          seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5);call sgrnd(seed)
98
99          ! hago un paso de Monte Carlo (MC_step)
100         select case(MC_step_type)
101             case(1) ! exploramos el espacio de configuraciones de forma random
102                 do k=1,n*n
103                     ! floor(x,type) returns the greatest integer less than or equal to X
104                     nrand=real(grnd(),dp);i=2_sp+floor(n*nrand,sp) ! random integer from 2 to n+1 (fila)
105                     nrand=real(grnd(),dp);j=2_sp+floor(n*nrand,sp) ! random integer from 2 to n+1 (columna)
106
107                     deltaU_adim=U_delta_adim(aux_matrix_pbc(i,j),&
108                                              aux_matrix_pbc(i,j+1),aux_matrix_pbc(i,j-1),&
109                                              aux_matrix_pbc(i+1,j),aux_matrix_pbc(i-1,j))
110                     cond1:if (deltaU_adim<=0._dp) then
111                         U_adim=U_adim+deltaU_adim
112                         aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
113                         call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
114                     else
115                         if (T_adim==0._dp) exit cond1
116                         nrand=real(grnd(),dp)
117                         if (deltaU_adim*(1._dp/T_adim)<abs(log(tiny(1._dp)))) then
118                             if (exp(-deltaU_adim*(1._dp/T_adim))>=nrand) then
119                                 U_adim=U_adim+deltaU_adim
120                                 aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
121                                 call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
122                             end if
123                         else
124                             if (nrand==0._dp) then
125                                 U_adim=U_adim+deltaU_adim
126                                 aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
127                                 call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
128                             end if
129                         end if
130                     end if cond1
131                 end do
132             case(2) ! exploramos el espacio de configuraciones de forma ordenada
133                 do j=2,n+1;do i=2,n+1
134                     deltaU_adim=U_delta_adim(aux_matrix_pbc(i,j),&
135                                              aux_matrix_pbc(i,j+1),aux_matrix_pbc(i,j-1),&
136                                              aux_matrix_pbc(i+1,j),aux_matrix_pbc(i-1,j))
137                     cond2: if (deltaU_adim<=0._dp) then
138                         U_adim=U_adim+deltaU_adim
139                         aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
140                         call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
141                     else
142                         if (T_adim==0._dp) exit cond2
143                         nrand=real(grnd(),dp)
144                         if (deltaU_adim*(1._dp/T_adim)<abs(log(tiny(1._dp)))) then
145                             if (exp(-deltaU_adim*(1._dp/T_adim))>=nrand) then
146                                 U_adim=U_adim+deltaU_adim
147                                 aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
```

```
148                             call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
149                         end if
150                     else
151                         if (nrand==0._dp) then
152                             U_adim=U_adim+deltaU_adim
153                             aux_matrix_pbc(i,j)=-aux_matrix_pbc(i,j)
154                             call pbc_ij(aux_matrix_pbc,n,i,j) ! actualizamos condiciones de borde
155                         end if
156                     end if
157                 end if cond2
158             end do;end do
159         end select
160     end subroutine MC_step_relaxation
161
162     subroutine pbc(aux_matrix_pbc,n,change_type)
163         implicit none
164         integer(sp), intent(inout) :: aux_matrix_pbc(n+2,n+2)
165         integer(sp), intent(in)    :: n,change_type
166         select case(change_type) ! tipo de cambio
167             case(1) ! cambiamos primer fila     (borde superior)
168                 aux_matrix_pbc(1,2:n+1)=aux_matrix_pbc(n+1,2:n+1)
169             case(2) ! cambiamos última fila     (borde inferior)
170                 aux_matrix_pbc(n+2,2:n+1)=aux_matrix_pbc(2,2:n+1)
171             case(3) ! cambiamos primer columna  (borde izquierdo)
172                 aux_matrix_pbc(2:n+1,1)=aux_matrix_pbc(2:n+1,n+1)
173             case(4) ! cambiamos última columna  (borde derecho)
174                 aux_matrix_pbc(2:n+1,n+2)=aux_matrix_pbc(2:n+1,2)
175             case(5) ! cambiamos todo            (todos los bordes)
176                 aux_matrix_pbc(n+2,2:n+1)=aux_matrix_pbc(2,2:n+1)
177                 aux_matrix_pbc(2:n+1,n+2)=aux_matrix_pbc(2:n+1,2)
178                 aux_matrix_pbc(1,2:n+1)=aux_matrix_pbc(n+1,2:n+1)
179                 aux_matrix_pbc(2:n+1,1)=aux_matrix_pbc(2:n+1,n+1)
180         end select
181     end subroutine pbc
182
183     subroutine pbc_ij(aux_matrix_pbc,n,i,j)
184         implicit none
185         integer(sp), intent(in)    :: n,i,j
186         integer(sp), intent(inout) :: aux_matrix_pbc(n+2,n+2)
187
188         if (i==2.and.j==2) then          ! change right & lower edges
189             call pbc(aux_matrix_pbc,n,4_sp);call pbc(aux_matrix_pbc,n,2_sp)
190         else if (i==2.and.j==n+1) then   ! change left & lower edges
191             call pbc(aux_matrix_pbc,n,3_sp);call pbc(aux_matrix_pbc,n,2_sp)
192         else if (i==n+1.and.j==2) then   ! change right & upper edges
193             call pbc(aux_matrix_pbc,n,4_sp);call pbc(aux_matrix_pbc,n,1_sp)
194         else if (i==n+1.and.j==n+1) then ! change left & upper edges
195             call pbc(aux_matrix_pbc,n,3_sp);call pbc(aux_matrix_pbc,n,1_sp)
196         end if
197
198         if (i==2.and.j/=2.and.j/=n+1)   call pbc(aux_matrix_pbc,n,2_sp) ! cambio borde inferior
199         if (j==2.and.i/=2.and.i/=n+1)   call pbc(aux_matrix_pbc,n,4_sp) ! cambio borde derecho
200         if (i==n+1.and.j/=2.and.j/=n+1) call pbc(aux_matrix_pbc,n,1_sp) ! cambio borde superior
201         if (j==n+1.and.i/=2.and.i/=n+1) call pbc(aux_matrix_pbc,n,3_sp) ! cambio borde izquierdo
202
203     end subroutine pbc_ij
204
205     ! subrutina para calcular la función de autocorrelación
206     subroutine func_autocor(obs,time_index,tau_corr,num_of_terms,&
207                         autocor_vector,aux_vector1,mask_vector,aux_vector2,&
208                         obs_med,var)
209         implicit none
210
211         integer(sp), intent(in)    :: time_index    ! indice del observable
212         integer(sp), intent(in)    :: tau_corr      ! tiempo de correlación
213         integer(sp), intent(in)    :: num_of_terms  ! menor cant de terminos admisibles (debe ser multiplo de tau_corr)
214         real(dp),    intent(in)    :: obs           ! observable
215         real(dp),    intent(inout) :: autocor_vector(tau_corr)                     ! vector autocorrelación
216         real(dp),    intent(inout) :: aux_vector1(tau_corr),aux_vector2(tau_corr)  ! vectores auxiliares
217         real(dp),    intent(inout) :: mask_vector(tau_corr)                        ! vector máscara
218         real(dp),    intent(inout) :: obs_med,var                                  ! valor medio y varianza
219         integer(sp)                :: i,index                                      ! loop indices
220         integer(sp)                :: total_obs_num ! total de observables que deben
221
222         ! calculamos valores primeros y segundos momentos
223         obs_med=obs_med+obs ! primeros momentos (acumulación)
224         var=var+obs*obs     ! segundos momentos (acumulación)
225
226         ! llenamos por primera vez el vector de dim=tau_corr
227         if (time_index<=tau_corr) autocor_vector(time_index)=obs
228         if (time_index==tau_corr) then
229             aux_vector1(:)=autocor_vector(:)
230             aux_vector2(:)=autocor_vector(:)
231         end if
232
233         ! determinamos total de observables que van a ingresar
234         total_obs_num=num_of_terms+tau_corr
```

```
235
236         ! computamos las correlaciones a todo tiempo
237         if (time_index>tau_corr.and.time_index<=(total_obs_num)) then
238             ! definimos el indice dentro del rango [1,tau_corr]
239             if (mod(time_index,tau_corr)==0_sp) then;index=tau_corr
240             else;index=mod(time_index,tau_corr);end if
241             aux_vector2(:)=cshift(aux_vector2(:),shift=1)
242             aux_vector2(tau_corr)=obs
243             autocor_vector(index)=autocor_vector(index)+dot_product(aux_vector1(:),aux_vector2(:))
244         end if
245         if (mod(time_index,tau_corr)==0_sp) aux_vector1(:)=aux_vector2(:)
246
247         ! computamos las correlaciones faltantes
248         if (time_index==total_obs_num) then
249             mask_vector(:)=1._dp
250             aux_vector1(:)=aux_vector2(:)
251             do index=1,tau_corr-1
252                 aux_vector2(:)=cshift(aux_vector2(:),shift=1)
253                 mask_vector(tau_corr-(index-1))=0._dp
254                 aux_vector2(:)=aux_vector2(:)*mask_vector(:)
255                 autocor_vector(index)=autocor_vector(index)+dot_product(aux_vector1(:),aux_vector2(:))
256                 ! promediamos en el ensamble
257                 autocor_vector(index)=autocor_vector(index)*(1._dp/real(num_of_terms+tau_corr-index,dp))
258             end do
259             ! promediamos en el ensamble (tau_corr términos)
260             autocor_vector(tau_corr)=autocor_vector(tau_corr)*(1._dp/real(num_of_terms,dp))
261             obs_med=obs_med*(1._dp/real(total_obs_num,dp)) ! primeros momentos
262             var=var*(1._dp/real(total_obs_num,dp)) ! segundos momentos
263             var=var-obs_med*obs_med ! varianza
264         end if
265     end subroutine func_autocor
266
267 end module module_2D_ferromagnetic_ising
```