Verifica della Classe di Universalità del Modello Clock 2D a 4 Stati

Manuel Strazzullo
Dipartimento di Fisica, Università di Pisa

5 gennaio 2026

Sommario

In questo lavoro le proprietà critiche del modello Clock bidimensionale vengono studiate con i metodi del Markov Chain Monte Carlo, implementato con uno script Fortran; in particolare, dagli esponenti critici e dalla temperatura critica ricavati per il modello, viene dimostrata la sua appartenenza alla classe di universalità del modello di Ising 2D classico.

1 Introduzione

Il modello Clock in due dimensioni descrive una generalizzazione dei sistemi a spin 1/2 su reticolo in cui l'energia di una particolare configurazione è data da

$$E = -J \sum_{<x,y>} \cos\left(\frac{2\pi}{q}(s_x - s_y)\right) - h\sum_x \cos\left(\frac{2\pi}{q}s_x\right) = -J \sum_{<x,y>} \mathrm{Re}\left(C_x^* C_x\right) - h\sum_x \mathrm{Re}\left(C_x\right) \tag{1}$$

con $C_x = e^{-\frac{2\pi}{q}s_x}$ e dove ogni s_x assume uno tra i q valori $\{0, ..., q-1\}$ e $< x, y >$ indica la somma sui primi vicini nel reticolo.

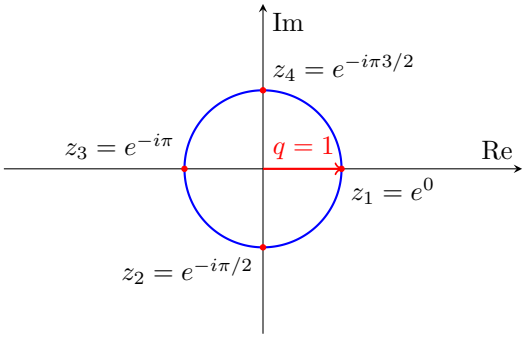

Figura 1: I quattro stati del modello in rappresentazione complessa $z_q = e^{-\pi(q-1)/2}$; passando da uno stato al successivo la freccia che "indica" lo stato si muove come la lancetta di un orologio.

Per $q = 2$ si ottiene il modello di Ising 2D, mentre per $q = 4$ si può dimostrare che

$$Z_{q=4}(\beta) = \left(Z_{q=2}(\beta/2)\right)^2 \tag{2}$$

che implica un punto critico per $\beta = 2\beta_{ising}$; lo scopo di questo lavoro è verificare nel caso 2D, per $q = 4$ la presenza di una transizione di fase per $\beta = 2\beta_{c,ising} = \log(1+\sqrt{2}) \approx 0.881374$ (avendo posto $J = 1$), appartenente alla classe di universalità Ising 2D.

2 Simulazione tramite MCMC

Simuliamo il modello Clock, su reticolo quadrato e con condizioni al contorno periodiche, utilizzando i metodi del Markov Chain Monte Carlo; il codice, riportato in appendice, implementa il seguente algoritmo:

1. `Inizializzo il reticolo come una matrice` $(L \times L)$`, e assegno un valore casuale tra` $\{0, 1, ..., q-1\}$ `a ciascun punto;`

* `Per` N `numero desiderato di iterazioni:`

2. `Seleziono un sito` (i,j) `del reticolo in modo deterministico;`
3. `Definisco uno stato di prova in cui, per il sito` (i,j), $s_{(i,j)} \longrightarrow s_{(i,j)} \pm 1$ `con distribuzione uniforme (lo stato` $q-1$ `precede lo stato 0 e viceversa);`
4. `Step Metropolis: accetto lo stato di prova con probabilità` $P_{acc} = min(1, e^{-\beta \Delta E})$;
5. `Se lo stato non viene accettato, tengo quello di partenza;`
6. `Dopo aver selezionato una volta ogni sito del reticolo, calcolo e salvo le osservabili rilevanti.`

L'algoritmo utilizzato per simulare l'evoluzione termodinamica del sistema è di tipo Metropolis a sito singolo; si può dimostrare che per questo modello i punti $\{2-5\}$ formano una catena di Markov irriducibile e aperiodica, dunque l'algoritmo campiona asintoticamente la distribuzione desiderata per le osservabili.

Per valori della lunghezza del reticolo che vanno da $L = 16$ a $L = 78$ sono state effettuate misure a vari β (uno script che automatizza una parte delle misure è riportato in appendice); per ciascun β sono state effettuate $N = 2 \cdot 10^6$ iterazioni, con un tempo di misura per ogni β che va da $T \sim 3$ min per il valore minimo di L a $T \sim 60$ min per il valore massimo. Le osservabili misurate in questa fase sono

1. $\epsilon = \frac{1}{L^2} \sum_{<x,y>} \cos\left(\frac{2\pi}{q}(s_x - s_y)\right)$ (energia per sito);
2. $m = \frac{1}{L^2} \sum_x e^{i\frac{2\pi}{q} s_x}$ (magnetizzazione per sito);

3 Analisi dati

Le osservabili rilevanti in questa fase sono (omettendo fattori β irrilevanti per l'andamento critico)

1. $C = L^2 \left(< \epsilon^2 > - < \epsilon >^2 \right)$ (calore specifico);
2. $\chi = L^2 \left(< |m|^2 > - < |m| >^2 \right)$ (suscettività magnetica);
3. $U = \frac{<|m|^4>}{(<|m|^2>)^2}$ (cumulante di Binder);

oltre alle già riportate $< \epsilon >$ e $< |m| >$.

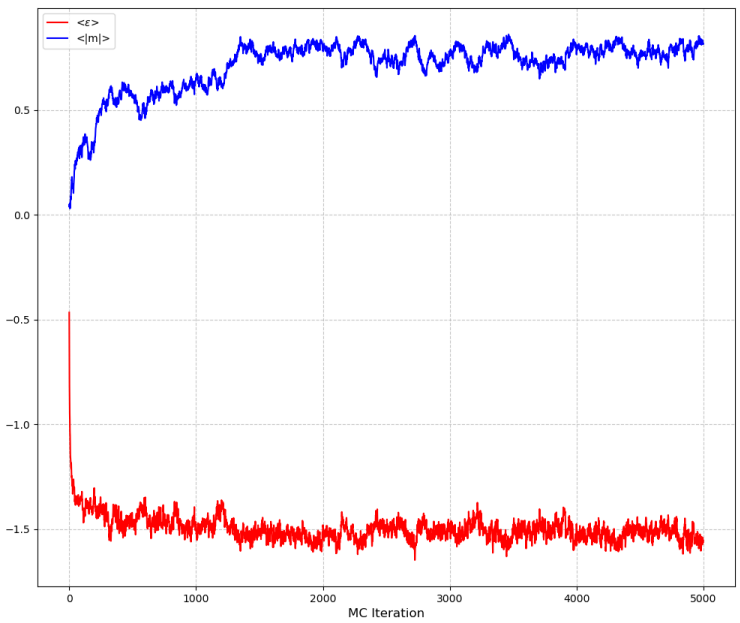

Figura 2: Termalizzazione al procedere del Monte Carlo per ϵ (rosso) e $|m|$ (blu) per il reticolo 64×64 vicino a $\beta = \beta_c$; il sistema, nelle taglie più grandi e vicino al punto critico, termalizza dopo $O(10^3)$ iterazioni del Montecarlo, dunque nell'analisi dati sono state scartate, per ogni misura di β, le prime 10000 misure di ϵ e $|m|$.

Il reticolo, una volta inizializzato, non è nello stato corrispondente a ciò che fisicamente sarebbe "equilibrio termodinamico", che nel linguaggio delle catene di Markov significa che un certo numero di iterazioni del Monte Carlo serve a far convergere l'algoritmo Metropolis alla distribuzione desiderata; ciò implica che vanno scartate misure delle osservabili fino a quando il sistema non raggiunge condizioni stazionarie, pena una stima delle incertezze completamente sbagliata. Il taglio può essere effettuato graficando l'evoluzione temporale delle osservabili misurate, come riportato in un esempio in Fig. 2; una volta scelta una soglia per i reticoli più grandi e vicino alla temperatura critica, condizioni nelle quali l'algoritmo impiega più iterazioni a convergere, utilizziamo la stessa soglia per tutte le condizioni di misura, perdendo comunque meno dell'1% delle misure.

Scartate le misure fuori equilibrio, procediamo a plottare tutte le osservabili rilevanti per verificarne gli andamenti (Fig. 3).

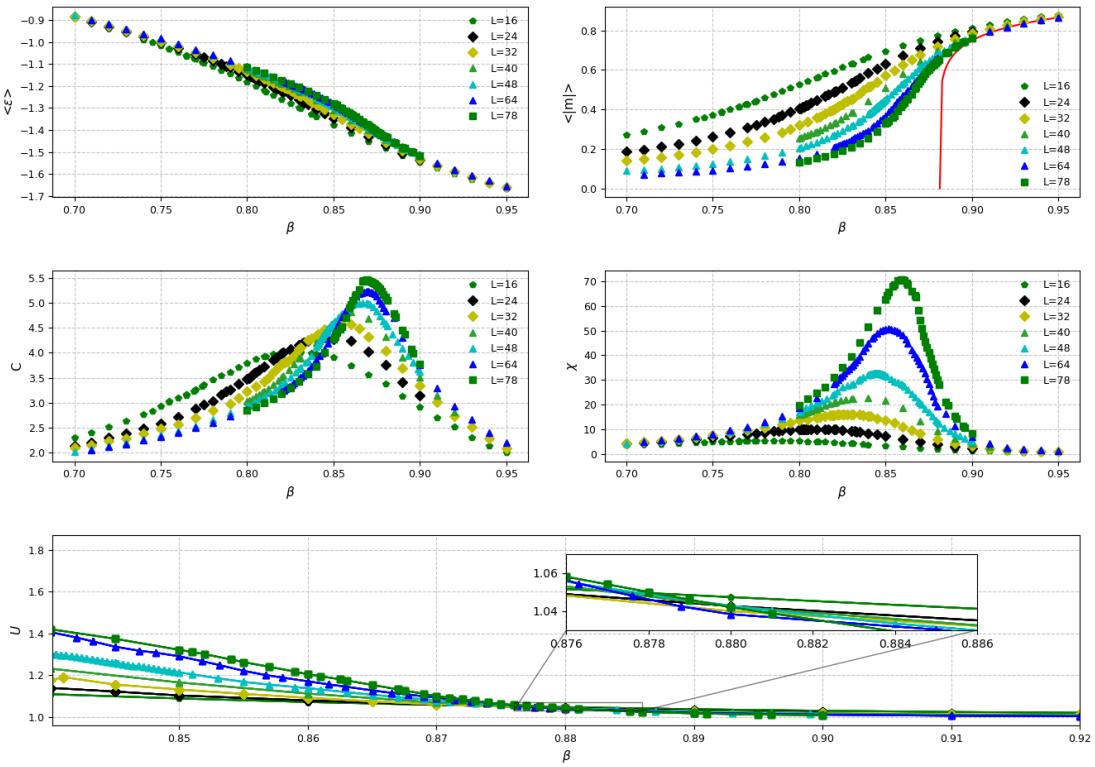

Figura 3: Risultati preliminari per le quantità rilevanti; nel plot del cumulante di Binder U i simboli per ciascun L sono gli stessi usati negli altri plot, mentre nel plot del parametro d'ordine $< |m| >$ è riportato l'andamento nel limite termodinamico per lo stesso parametro nel modello di Ising 2D, $m_0(\beta) = (1 - \sinh^{-4}(2\beta))^{1/8}$, con β opportunamente riscalato.

Gli andamenti sono quelli attesi per il modello di Ising 2D: i cumulanti di Binder U per diverse dimensioni del reticolo si incontrano intorno a $2\beta_{c,ising}$, la densità di energia $< \epsilon >$ non presenta divergenze, il calore specifico C e la suscettività χ tendono a divergere nel limite termodinamico e la magnetizzazione $< |m| >$, che con l'aggiunta del modulo diventa un buon parametro d'ordine per il sistema, tende allo stesso andamento asintotico del parametro d'ordine di Ising 2D.

In Fig. 4 sono riportate alcune misure della magnetizzazione senza l'aggiunta del modulo; è ben visibile che, come nel modello di Ising 2D, la magnetizzazione non riesce a quantificare un eventuale ordinamento del sistema (nonostante le simmetrie dei due modelli siano diverse).

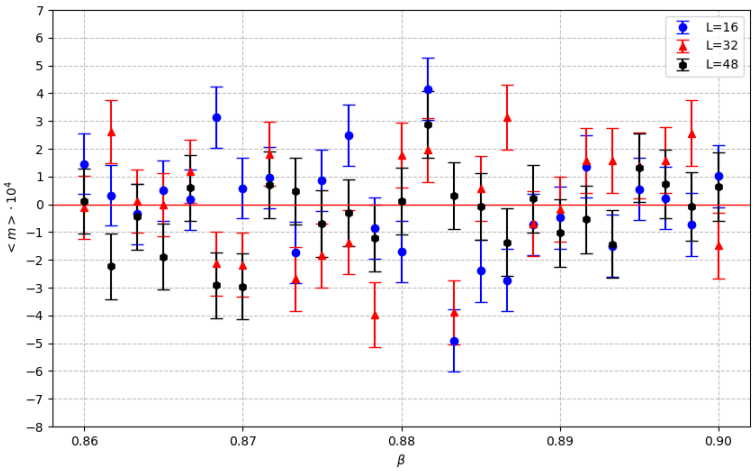

Figura 4: Media della magnetizzazione calcolata come $\left\langle \cos\left(\frac{2\pi}{q} s_x\right)\right\rangle$ su diverse taglie del reticolo; il numero di campionamenti in questo caso è stato ridotto rispetto al numero dei campionamenti usato per i dati da analizzare e gli errori sono calcolati con i metodi discussi in seguito.

Una ulteriore verifica visiva e preliminare che è possibile fare è basata sul *finite size scaling*: vicino al punto critico, gli andamenti di alcune osservabili sono descritti da funzioni di scala universali e riscalando opportunamente gli insiemi di dati è possibile osservare un "collasso" delle misure, in cui tutti i punti seguono lo stesso andamento (a meno di correzioni dipendenti dalla taglia del sistema).

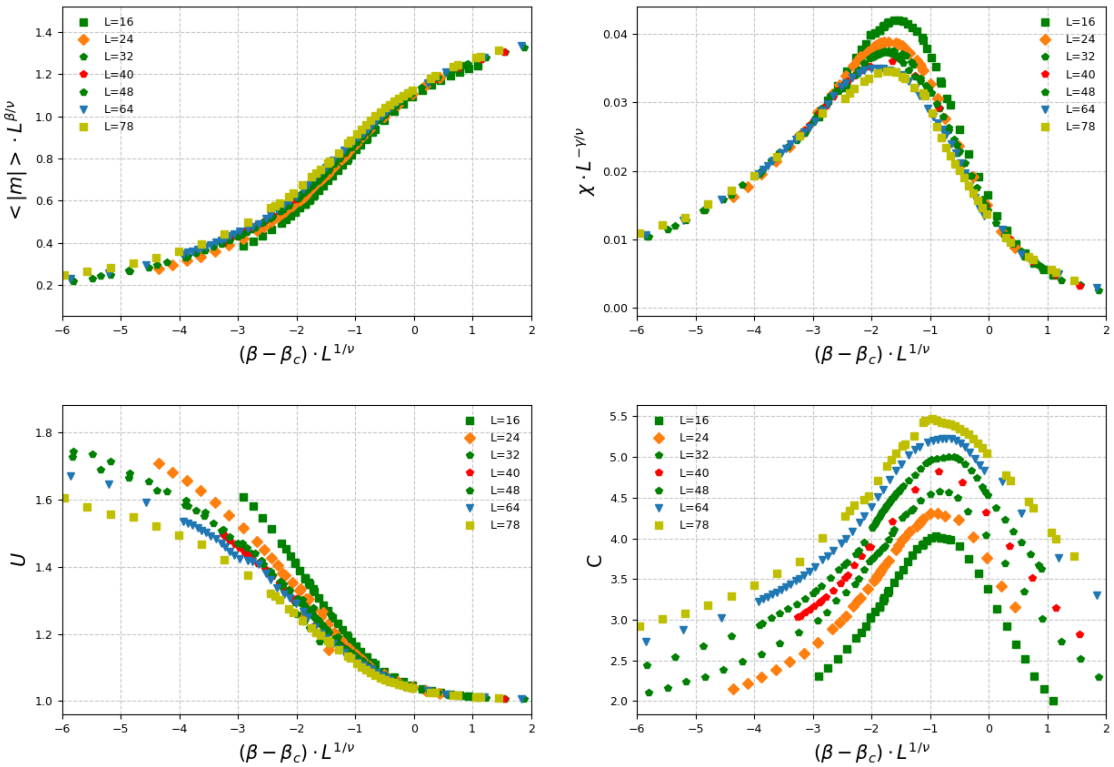

Figura 5: FSS per le quantità rilevanti; le correzioni per le taglie minori del reticolo sono ben visibili a occhio.

Gli andamenti critici rilevanti sono

- $< |m| > \simeq L^{-\beta/\nu} m_2[(\beta - \beta_c)L^{1/\nu}]$;
- $C \simeq L^{\alpha/\nu} C_2[(\beta - \beta_c)L^{1/\nu}]$;
- $\chi \simeq L^{\gamma/\nu} \chi_2[(\beta - \beta_c)L^{1/\nu}]$;
- $U \simeq U_2[(\beta - \beta_c)L^{1/\nu}]$;

dove gli esponenti $\alpha, \beta, \gamma, \nu$ sono gli esponenti critici della transizione di fase; se il modello Clock appartiene alla classe di universalità di Ising 2D, allora riscalando i dati con gli esponenti critici noti analiticamente per quel modello ($\alpha = 0$, $\beta = 1/8$, $\gamma = 7/4$, $\nu = 1$) dovremmo ottenere dei collassi. La suddetta verifica è riportata in Fig. 5, e i risultati sono quelli attesi, se non per il calore specifico C, sul quale è opportuno spendere due parole aggiuntive.

L'esponente critico che regola l'andamento del calore specifico al punto critico è $\alpha = 0$, il che implica una divergenza più lenta rispetto a quella lineare in L; è possibile dimostrare che la divergenza in questione è logaritmica, e noi verifichiamo l'ipotesi con un fit dei picchi in funzione della lunghezza L del reticolo (Fig. 6); i valori dei massimi sono stati a loro volta ricavati tramite fit, come spiegato in seguito. Il fit mostra chiaramente l'andamento logaritmico dei massimi di C.

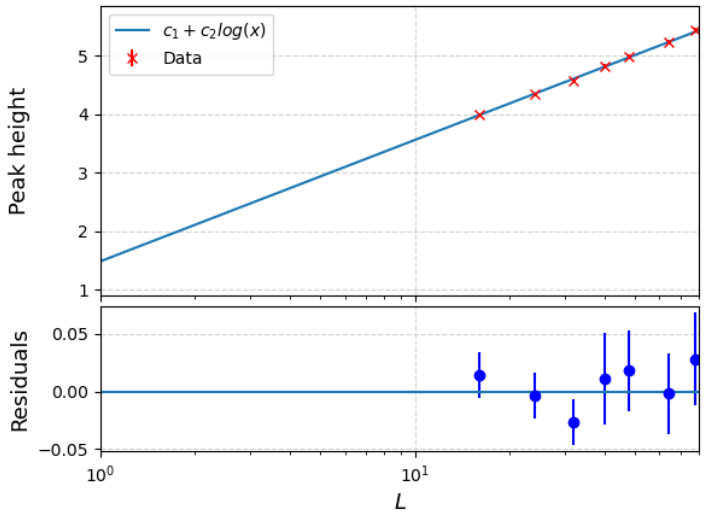

Figura 6: Fit per la divergenza logaritmica in L del calore specifico.

Discutiamo ora la procedura adottata per tenere conto delle correlazioni tra i dati generati tramite MCMC. I dati contengono una qualche correlazione dovuta alla natura pseudocasuale del generatore che, se non considerata, porta a una sottostima dell'errore sulle osservabili (calcolato come varianza della media sui campionamenti). Sia nel caso di osservabili primarie (ϵ, $|m|$) che di osservabili secondarie (C, χ, U) usiamo la tecnica del *blocking*, in cui dividiamo il campione di N misure in blocchi di $k \ll N$ elementi, sui quali poi calcoliamo medie e varianze di osservabili generiche O come

$$\overline{O} = \frac{1}{N/k} \sum_i^{N/k} O_i^{(k)} \tag{3}$$

$$\overline{\sigma}_{\overline{O}} = \frac{1}{N/k} \frac{1}{N/k - 1} \sum_i^{N/k} \left(O_i^{(k)} - \overline{O} \right)^2 \tag{4}$$

in cui $O_i^{(k)}$ è la media sull'i-esimo dei N/k blocchi, composto da k elementi; al variare di k, l'errore così stimato cresce fino a un valore asintotico di saturazione. Per ogni taglia del reticolo effettuiamo una visualizzazione dell'errore al variare di k, così da poter scegliere un valore da utilizzare nell'analisi corrispondente ai valori saturati dell'errore; un esempio è riportato in Fig. 7.

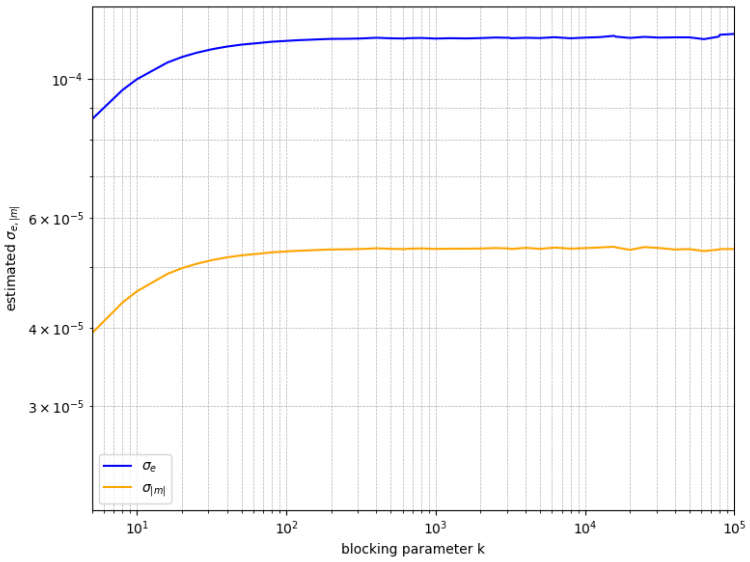

Figura 7: Esempio di analisi per il parametro k usato nel *blocking* delle variabili primarie, intorno a $\beta = \beta_c$, per un reticolo (4×4); in generale $k = 10^3$ è sufficiente in tutti i casi per ottenere una saturazione dell'errore.

Per quanto riguarda le osservabili secondarie, utilizziamo il metodo *bootstrap* per la stima degli errori: generato un insieme di N dati, lo campioniamo R volte con distribuzione uniforme, in modo da generare R campioni di N misure che speriamo approssimino la distribuzione di partenza (quella secondo cui abbiamo generato i dati all'inizio). Possiamo allora stimare l'errore di un'osservabile secondaria O come varianza dell'osservabile stessa su questi R campioni. Ora è possibile mettere insieme *bootstrap* e *blocking* per tenere conto delle correlazioni: in ciascuno degli R campioni generati con il *bootstrap* effettuiamo la procedura del *blocking*, prima di calcolare la media e la varianza come in Eq. 3 e 4.

Per determinare numericamente alcuni esponenti critici ci avvaliamo della suscettività χ e del FSS: dagli andamenti critici descritti in precedenza si ricava

1. $\beta_{pc} \approx \beta_c + aL^{-1/\nu}$
2. $\chi_{max} \approx b_0 + b_1 L^{\gamma/\nu}$

dove $\beta_{pc}(L)$ è il valore della temperatura inversa per cui si ha il massimo di χ. Possiamo ricavare i valori di χ_{max} e β_{pc} per ogni lunghezza del reticolo fittando intorno ai picchi della suscettività con una funzione del tipo

$$\chi(\beta) \approx d(\beta_{pc}(L) - \beta)^2 + \chi_{max}(L)$$

in cui i parametri di fit sono $\chi_{max}(L)$, $\beta_{pc}(L)$ e d; dopodichè, possiamo fittare i valori trovati per ciascun L con le leggi appena descritte per $\chi_{max}(L)$ e $\beta_{pc}(L)$ per ricavare gli esponenti critici ν e γ e la temperatura critica β_c.

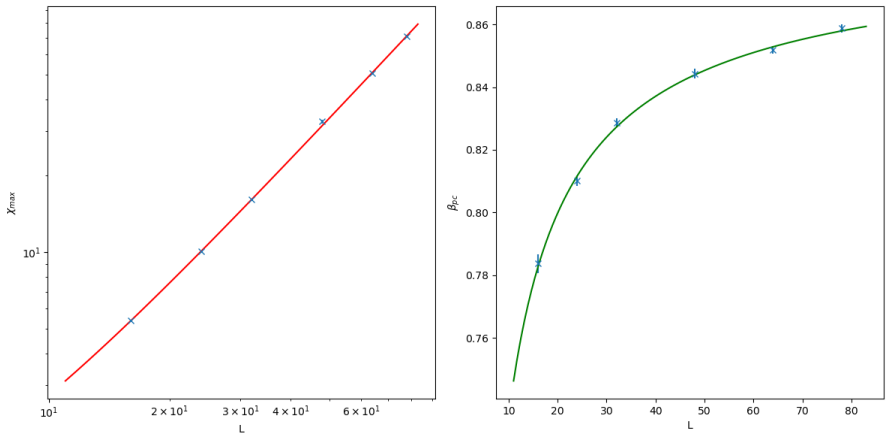

Figura 8: Esempio di fit per β_{pc}, χ_{max} considerando tutte le lunghezze del reticolo.

I fit intorno ai picchi di χ sono stabili fino a $6-7$ punti al massimo per tutte le dimensioni L (i dati intorno ai picchi sono stati generati con densità quasi uniformi per tutti gli L), che risulta essere l'intervallo in cui vale l'approssimazione parabolica usata; in questo intervallo, i $\chi^2/ndof$ risultano compresi tra 1 e 3. I successivi fit per $\chi_{max}(L)$ e $\beta_{pc}(L)$ danno risultati ragionevoli, con $\chi/ndof$ compresi tra 1 e 2.5; un esempio è riportato in Fig. 8.

Per tenere conto degli errori sistematici di questa procedura, ripetiamo i fit per $\chi_{max}(L)$ e $\beta_{pc}(L)$, eliminando i risultati ottenuti dalle taglie più basse del reticolo (in cui le correzioni alle approssimazioni utilizzate sono più rilevanti) fino a $L_{min} = 32$, oltre il quale il fit non converge.

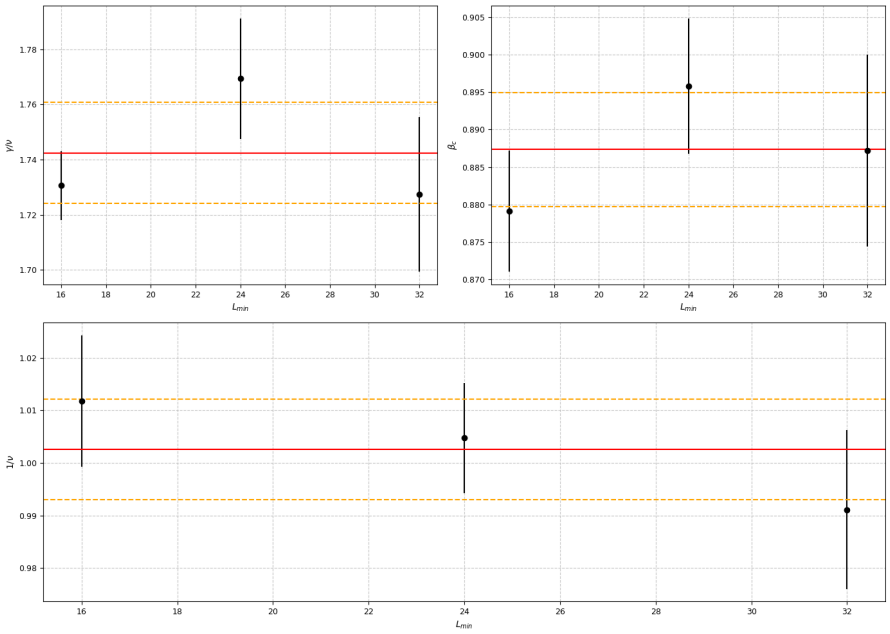

Figura 9: Risultati per β_c, γ/ν, $1/\nu$ al variare della lunghezza minima considerata nei fit; le linee continue indicano il valore centrale le linee tratteggiate rappresentano gli errori.

Mettiamo insieme i risultati per $L_{min} = 16, 24, 32$ e calcoliamo i valori centrali e gli errori per ν, γ e β_c, sommando gli errori statistici (nel senso di deviazione standard) e sistematici (dal fit); i risultati sono riportati in Fig. 9, e numericamente troviamo $\beta_c = 0.887(8)$, $\gamma/\nu = 1.742(18)$, $1/\nu = 1.002(9)$, che sono in accordo con l'ipotesi della classe di universalità Ising 2D.

Procediamo infine a una stima dell'esponente dinamico z', che determina l'andamento del tempo di autocorrelazione integrato (di un'osservabile O) tramite $\tau^{(O)} \sim \xi^{z'}$, con ξ lunghezza di correlazione del sistema. E' noto che in 2D, per un'osservabile primaria,

$$\sigma_{\overline{O}} \sim L^{(z'+\gamma/\nu-2)/2} = L^{(z'-0.25)/2} \tag{5}$$

nel nostro caso, e osservando l'andamento lineare dell'errore sulla magnetizzazione media $\sigma_{\overline{|m|}}$ al variare di L, possiamo stimare $z' \approx 2.25$, che corrisponde all'esponente dinamico per Ising 2D.

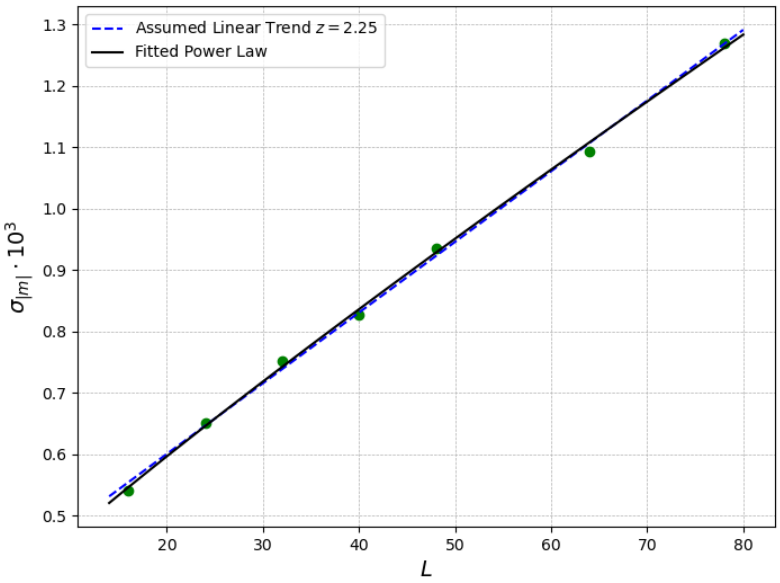

Figura 10: Andamento dell'errore $\sigma_{\overline{|m|}}$ al variare della taglia del sistema, per $\beta \simeq \beta_c$; per la curva fittata, $z' = 2.04(0.17)$.

In Fig. 10 è riportato l'andamento osservato per $\sigma_{\overline{|m|}}$, con i dati che sono stati fittati *a priori* con una retta e, inoltre, con una legge di potenza L^k generale; entrambi i fit non presentano errori in ingresso, ma dimostrano bene l'andamento lineare della variabile dipendente.

Appendice: Codici MCMC

```
program clock_model
    implicit none                           !!tutto in double
    integer, parameter :: L = 16            !!lunghezza del reticolo
    integer, parameter :: N = 2*10**6       !!iterazioni del Montecarlo
    integer, parameter :: q = 4             !!parametro del modello
    integer :: i, j, step, new_state, ix, iy, k
    real(8) :: theta(L, L), theta_new(L, L)
    real(8) :: energy, energy_change, magn, rand_temp
    real(8) :: beta
    real(8), parameter :: dpi = 2.0d0 * 3.141592653589793d0


    call random_seed()      !!inizializzo il generatore
                            !!(seed casuale se no argomento) e il reticolo
    read*, beta
    call random_angles(theta)
    print*, L*L, beta

    do step = 1, N
        do i = 1, L
            do j =1, L
                !!do k = 1, L              linee per debugging
                !!print*, theta(k,:)
                !!end do

                call random_number(rand_temp)
                if (rand_temp < 0.5) then
                    new_state = int(mod(theta(i, j) + 1.0, real(q, kind=8)))  !! +-1
                else
                     if (theta(i, j) > 0.9) then
                         new_state = int(theta(i, j) - 1.0)
                     else
                         new_state = int(q - 1.0)
                     end if
                end if

```

```
                !!print*, ''
        !!print*, i, j, new_state
        !!print*, ''

        energy_change = calc_energy_change(theta, i, j, new_state)

        if (energy_change < 0.0) then   !!metro
          theta(i, j) = real(new_state, kind=8)
        else
          call random_number(rand_temp)
          if (rand_temp < exp(-energy_change * beta)) then
            theta(i, j) = real(new_state, kind=8)
          end if
        end if
      end do
        end do
        energy = calc_energy(theta)
        magn = calc_magn(theta)
        print*, energy, magn
        !!print*, ''

    end do

contains

    subroutine random_angles(theta)    !!inizializza una matrice con valori casuali da 0 a q-1
        real(8), dimension(L, L) :: theta
        real(8) :: rand_temp
        integer :: i, j

        do i = 1, L
            do j = 1, L
                call random_number(rand_temp)
                theta(i, j) = int(rand_temp * q)
            end do
        end do
    end subroutine random_angles

    function calc_energy(theta) result(total_energy)
        real(8), dimension(L, L) :: theta
        real(8) :: total_energy
        integer :: i, j
        integer :: neighbor_x, neighbor_y
        real(8) :: diff, cos_diff

        total_energy = 0.0
        do i = 1, L
            do j = 1, L
                neighbor_x = mod(i, L) + 1
                neighbor_y = mod(j, L) + 1

                diff = mod(real(theta(i, j) - theta(neighbor_x, j), kind=8), real(q, kind=8))
                cos_diff = cos(dpi * diff / real(q, kind=8))
                total_energy = total_energy - cos_diff

                diff = mod(real(theta(i, j) - theta(i, neighbor_y), kind=8), real(q, kind=8))
                cos_diff = cos(dpi * diff / real(q, kind=8))
                total_energy = total_energy - cos_diff
            end do
        end do
        total_energy = total_energy / (L * L)
    end function calculate_total_energy

    function calc_energy_change(theta, ix, iy, new_state) result(energy_change)  !!funzione
    per il dE
        real(8), dimension(L, L) :: theta
        integer :: ix, iy, new_state
        real(8) :: energy_change
        real(8) :: diff, cos_diff
        integer :: neighbor_x, neighbor_y

        energy_change = 0.0

        neighbor_x = mod(ix, L) + 1
        neighbor_y = mod(iy, L) + 1

```

```
        diff = mod(real(new_state - theta(neighbor_x, iy), kind=8), real(q, kind=8))
        cos_diff = cos(dpi * diff / real(q, kind=8))
        energy_change = energy_change - cos_diff + cos(dpi * &
        & mod(real(theta(ix, iy) - theta(neighbor_x, iy), kind=8), real(q, kind=8)) / real(q,
    kind=8))

        diff = mod(real(new_state - theta(ix, neighbor_y), kind=8), real(q, kind=8))
        cos_diff = cos(dpi * diff / real(q, kind=8))
        energy_change = energy_change - cos_diff + cos(dpi * &
        & mod(real(theta(ix, iy) - theta(ix, neighbor_y), kind=8), real(q, kind=8)) / real(q,
    kind=8))

        if (ix > 1.9) then
            neighbor_x = mod(ix, L) - 1
        else
            neighbor_x = L
        end if

        diff = mod(real(new_state - theta(neighbor_x, iy), kind=8), real(q, kind=8))
        cos_diff = cos(dpi * diff / real(q, kind=8))
        energy_change = energy_change - cos_diff + cos(dpi * &
        & mod(real(theta(ix, iy) - theta(neighbor_x, iy), kind=8), real(q, kind=8)) / real(q,
    kind=8))

        if (iy > 1.9) then
            neighbor_y = mod(iy, L) - 1
        else
            neighbor_y = L
        end if

        diff = mod(real(new_state - theta(ix, neighbor_y), kind=8), real(q, kind=8))
        cos_diff = cos(dpi * diff / real(q, kind=8))
        energy_change = energy_change - cos_diff + cos(dpi * &
        & mod(real(theta(ix, iy) - theta(ix, neighbor_y), kind=8), real(q, kind=8)) / real(q,
    kind=8))

    end function calc_energy_change

    function calc_magn(theta) result(mag)
        real(8), dimension(L, L) :: theta
        real(8) :: mag, temp1, temp2
        integer :: i, j

        mag = 0.0
        temp1 = 0.0
        temp2 = 0.0
        do i = 1, L
            do j = 1, L
                temp1 = temp1 + &
                cos(dpi * real(theta(i, j), kind=8) / real(q, kind=8))
                temp2 = temp2 + &
                sin(dpi * real(theta(i, j), kind=8) / real(q, kind=8))
            end do
        end do
        temp1 = temp1 / (L * L)
        temp2 = temp2 / (L * L)
        mag = sqrt(temp1*temp1 + temp2*temp2)
    end function calc_magn

end program clock_model
```

Listing 1: Codice .f90 utilizzato per il Montecarlo.

```
#!/bin/bash
export LC_NUMERIC="en_US.UTF-8"      #sistema le virgole dei decimali

gfortran clock.f90 -O2 -o clock.x

for i in $(seq "beta_start" 0.01 "beta_end"); do   #mettere i valori desiderati
    output_file="clock_beta_${i}_L.txt"     #mettere L prima di ogni run
    echo "Sending input beta= $i to simulation"
    # pipe
    echo "$i" | ./clock.x > "$output_file"
done
echo "Done simulating"
```

Listing 2: Script .sh per automatizzare le simulazioni a dimensione fissata del reticolo.