4. Introduzione alla percolazione

Abbiamo iniziato ad esplorare il concetto di percolazione, muovendo i primi passi in un argomento che ci accompagnerà a lungo. Fin da subito, però, è emerso un aspetto particolarmente interessante che merita di essere evidenziato: questo modello offre una chiave di lettura per molti fenomeni, come il filtraggio del percolato dai rifiuti, la preparazione del caffè, la diffusione di epidemie o incendi nelle foreste, e la transizione di un materiale granulare da isolante a conduttore variando la concentrazione di grani metallici.

L'approccio iniziale è stato prevalentemente intuitivo. Abbiamo cominciato definendo un **reticolo quadrato**: un insieme di punti in cui ogni punto ha una disposizione locale ordinata. Più precisamente, ciascun punto ha quattro vicini immediati (a sinistra, a destra, sopra e sotto) e quattro vicini diagonali (nord-est, sud-est, sud-ovest, nord-ovest). Questo schema si ripete per ogni punto, eccetto sui bordi del reticolo. La struttura così ottenuta è chiamata **reticolo**.

Tale struttura non è unica: ad esempio, passando da due a tre dimensioni, possiamo definire un reticolo cubico (aggiungendo due nuovi vicini, davanti e dietro). Sia il reticolo quadrato che quello cubico sono esempi di **reticoli ipercubici**, estendibili a un numero generico di dimensioni. Inoltre, la geometria della cella primitiva può variare, come nel caso dei reticoli triangolari o esagonali.

Una volta stabilito il reticolo, abbiamo introdotto una probabilità p che descrive la probabilità che ciascun sito del reticolo sia colorato. A questo punto, il reticolo risulta composto da siti colorati o non colorati.

Abbiamo poi definito i **cluster**: un cluster è un insieme di siti colorati tra loro collegati, ovvero due siti appartengono allo stesso cluster se è possibile muoversi dall'uno all'altro passando solo attraverso vicini immediati colorati. Questa definizione soddisfa i criteri di una relazione di equivalenza, per cui ogni sito colorato appartiene a un unico cluster.

Quando un cluster riesce a connettere due lati opposti del reticolo, si parla di percolazione.

Nel **limite termodinamico** (ovvero quando il reticolo è idealmente infinito), esiste una soglia critica per la probabilità di colorazione: al di sotto di questo valore, la percolazione non si verifica, mentre al di sopra sì.

Per studiare questo fenomeno, abbiamo bisogno di saper creare e colorare il reticolo e di identificare i cluster. Abbiamo brevemente discusso come effettuare una simulazione in MATLAB, utilizzando il comando `rand<p` per generare un reticolo in base alla probabilità specificata. Le simulazioni numeriche ci hanno portato a osservare un nuovo esempio di stabilità nel limite termodinamico: la colorazione del reticolo può essere considerata come

una serie di prove ripetute, e la distribuzione di probabilità coinvolta è quella binomiale. In media, il numero di siti colorati tende a stabilizzarsi all'aumentare del numero di siti.

Infine, abbiamo accennato agli algoritmi di **cluster-finding**, indispensabili per identificare i cluster e determinare se si è verificata la percolazione. Inoltre, abbiamo consultato una tabella di soglie di percolazione per vari tipi di reticoli e modelli, scoprendo che molti risultati non sono calcolabili analiticamente, aprendo così ulteriori possibilità per le nostre simulazioni.

Un primo algoritmo di cluster finding

La funzione `CercaCluster` implementa un algoritmo per identificare i cluster in un reticolo quadrato di dimensione $L \times L$ e determinare se si verifica la percolazione, ossia l'esistenza di un percorso continuo di siti occupati che collega i bordi opposti del reticolo. L'algoritmo considera la percolazione sia dal bordo superiore al bordo inferiore (top-bottom) sia dal bordo sinistro al bordo destro (left-right). Ecco una spiegazione dettagliata del funzionamento dell'algoritmo

Inizializzazione del reticolo e dei parametri

```
res.matrice = zeros(L + 2);
aux = rand(L) < p;
res.matrice(2:end-1, 2:end-1) = aux;
```

- **Reticolo esteso**: Si crea una matrice `res.matrice` di dimensione $(L+2) \times (L+2)$ inizializzata a zero. Questa matrice estesa serve per gestire facilmente i bordi senza dover gestire casi speciali per i siti al confine.
- **Popolazione del reticolo**: Si genera una matrice `aux` di dimensione $L \times L$ dove ogni sito ha una probabilità p di essere occupato (cioè, di essere "colorato"). Questo viene fatto confrontando valori casuali tra 0 e 1 con la probabilità p.
- **Inserimento nel reticolo esteso**: La matrice `aux` viene inserita al centro di `res.matrice`, lasciando un bordo di zeri attorno.

Inizializzazione delle variabili di percolazione e delle etichette

```
res.percolazioneTB = 0;
res.percolazioneLR = 0;

res.p = p;

res.label = zeros(L + 2);

labelC = 1;
```

- **Flag di percolazione**: Le variabili `res.percolazioneTB` e `res.percolazioneLR` sono inizializzate a 0 e serviranno per indicare se si verifica la percolazione verticale o orizzontale.
- **Matrice delle etichette**: Si crea una matrice `res.label` della stessa dimensione di `res.matrice` per assegnare un'etichetta univoca a ciascun cluster identificato.
- **Contatore delle etichette**: `labelC` è un contatore che assegna un nuovo numero a ogni nuovo cluster trovato.

Identificazione dei cluster tramite ricerca in profondità

```
valid = find(res.matrice);

for iter = 1:length(valid)
    ii = valid(iter);
    if (res.label(ii) == 0)
        pila = ii;
        res.label(ii) = labelC;

        j = 1;
        while (j <= length(pila))
            elemento = pila(j);

            % Esplora i vicini
            if (res.matrice(elemento - 1) && res.label(elemento - 1) == 0)
                pila(end + 1) = elemento - 1;
                res.label(elemento - 1) = labelC;
            end

            if (res.matrice(elemento + 1) && res.label(elemento + 1) == 0)
                pila(end + 1) = elemento + 1;
                res.label(elemento + 1) = labelC;
            end

            if (res.matrice(elemento - L - 2) && res.label(elemento - L - 2)
== 0)
                pila(end + 1) = elemento - L - 2;
                res.label(elemento - L - 2) = labelC;
            end

            if (res.matrice(elemento + L + 2) && res.label(elemento + L + 2)
== 0)
                pila(end + 1) = elemento + L + 2;
                res.label(elemento + L + 2) = labelC;
            end

            j = j + 1;
        end
```

```
            labelC = labelC + 1;
        end
    end
```

Identificazione dei siti occupati: `valid` contiene gli indici lineari dei siti occupati nella matrice `res.matrice`.

Ricerca dei cluster:

- **Nuovo cluster**: Se un sito occupato non è stato ancora etichettato (`res.label(ii) == 0`), si avvia una ricerca per identificare tutti i siti connessi ad esso.
- **Pila (stack)**: Si utilizza una pila `pila` per implementare una ricerca in profondità (Depth-First Search, DFS). Si inizializza la pila con l'indice del sito corrente.
- **Esplorazione dei vicini**:
  - Si esaminano i siti adiacenti (su, giù, sinistra, destra) controllando se sono occupati e non ancora etichettati.
    - Se un vicino soddisfa queste condizioni, viene aggiunto alla pila e gli viene assegnata l'etichetta corrente `labelC`.
- **Iterazione**: Si continua ad espandere la ricerca fino a quando la pila è vuota, garantendo che tutti i siti connessi siano etichettati correttamente.
- **Incremento dell'etichetta**: Una volta completata la ricerca per un cluster, si incrementa `labelC` per il prossimo cluster.

Rimozione dei bordi e preparazione per la verifica di percolazione

```
res.label = res.label(2 : end - 1, 2 : end - 1);
res.matrice = res.matrice(2 : end - 1, 2 : end - 1);
```

Rimozione dei bordi: Si eliminano le righe e le colonne aggiuntive utilizzate per il padding, riportando le matrici `res.label` e `res.matrice` alla dimensione originale $L \times L$.

Verifica della percolazione

Percolazione da sinistra a destra (left-right)

```
auxL = unique(res.label(1:L));
left = auxL(auxL > 0);

auxR = unique(res.label(L*(L-1) + 1:L*L));
right = auxR(auxR > 0);

if (~isempty(intersect(left, right)))
    res.percolazioneLR = 1;
end
```

Identificazione dei cluster ai bordi:

- **Bordo sinistro**: Si estraggono le etichette dei siti sul bordo sinistro del reticolo.
- **Bordo destro**: Si estraggono le etichette dei siti sul bordo destro del reticolo.

Verifica della connessione:

- Si cerca l'intersezione tra le etichette dei bordi sinistro e destro.
- Se esiste almeno un'etichetta comune, significa che c'è un cluster che collega il bordo sinistro al bordo destro, indicando percolazione orizzontale.

Aggiornamento del flag: Se la percolazione orizzontale si verifica, `res.percolazioneLR` viene impostato a 1.

Percolazione dall'alto verso il basso (top-bottom)

```
auxT = unique(res.label(1:L:L*(L-1) + 1));
top = auxT(auxT > 0);

auxB = unique(res.label(L:L:L*L));
bottom = auxB(auxB > 0);

if (~isempty(intersect(top, bottom)))
    res.percolazioneTB = 1;
end
```

Identificazione dei cluster ai bordi:

- **Bordo superiore**: Si estraggono le etichette dei siti sul bordo superiore del reticolo.
- **Bordo inferiore**: Si estraggono le etichette dei siti sul bordo inferiore del reticolo.

Verifica della connessione:

- Si cerca l'intersezione tra le etichette dei bordi superiore e inferiore.
- Se esiste almeno un'etichetta comune, significa che c'è un cluster che collega il bordo superiore al bordo inferiore, indicando percolazione verticale.
- **Aggiornamento del flag**: Se la percolazione verticale si verifica, `res.percolazioneTB` viene impostato a 1.

Restituzione dei risultati

La funzione restituisce una struttura `res` contenente:

- `res.p`: La probabilità utilizzata per occupare i siti.
- `res.matrice`: La matrice (L \times L) dei siti occupati (1) e vuoti (0).
- `res.label`: La matrice (L \times L) con le etichette dei cluster.

- `res.percolazioneTB`: Indica se si verifica la percolazione verticale (1) o meno (0).
- `res.percolazioneLR`: Indica se si verifica la percolazione orizzontale (1) o meno (0).

Riassunto del funzionamento

L'algoritmo simula il fenomeno della percolazione su un reticolo quadrato:

1. **Generazione del reticolo**: Si crea un reticolo dove ogni sito è occupato con probabilità (p).
2. **Identificazione dei cluster**: Si etichettano i cluster di siti occupati connessi tra loro utilizzando una ricerca in profondità.
3. **Verifica della percolazione**: Si controlla se esiste almeno un cluster che connette i bordi opposti del reticolo, sia in direzione orizzontale che verticale.
4. **Restituzione dei risultati**: Si forniscono le informazioni sul reticolo, le etichette dei cluster e l'esito della verifica di percolazione.

Questo algoritmo, pur non ottimale in termini di complessità computazionale, risulta robusto e fornisce tutte le informazioni necessarie per il nostro studio, come il numero di cluster, la loro dimensione (determinata dall'altezza della pila al termine del processo) e l'identificativo di ciascun sito colorato. Tuttavia, non è efficiente dal punto di vista del tempo di calcolo, in quanto è evidente che il medesimo sito viene considerato più volte.

Per l'implementazione dell'algoritmo, risulta essenziale l'accesso ai primi vicini di ciascun sito. Se ciò è relativamente semplice per i siti non posti ai bordi del reticolo, il caso dei siti di bordo richiede ulteriori attenzioni. La soluzione da noi adottata prevede l'aggiunta di una cornice di zeri attorno al reticolo e la limitazione dell'esplorazione esclusivamente ai siti interni, evitando quindi i siti di bordo.

Algoritmo completo

```
function res = CercaCluster(L, p)
% CercaCluster trova i cluster su un reticolo L x L con probabilità p di
% avere un sito occupato.
%
% Input:
%   L - lunghezza del lato della matrice
%   p - probabilità che un sito sia occupato
%
% Output:
%   res.p - probabilità utilizzata per occupare i siti
%   res.matrice - matrice dei siti occupati
%   res.label - matrice delle etichette dei cluster trovati
%   res.percolazioneTB - indica se esiste percolazione dall'alto verso il
basso (1) o no (0)
```

```
%   res.percolazioneLR - indica se esiste percolazione da sinistra a destra
(1) o no (0)

    % Inizializzazione della matrice con bordo per gestire i confini
    res.matrice = zeros(L + 2);  % Matrice estesa con bordo
    aux = rand(L) < p;           % Matrice dei siti occupati con probabilità
p
    res.matrice(2:end-1, 2:end-1) = aux;  % Inserisce aux al centro di
res.matrice

    % Inizializzazione dei parametri
    res.percolazioneTB = 0;  % Flag per percolazione dall'alto verso il
basso
    res.percolazioneLR = 0;  % Flag per percolazione da sinistra a destra
    res.p = p;               % Probabilità utilizzata
    res.label = zeros(L + 2);  % Matrice delle etichette dei cluster
    labelC = 1;               % Contatore per le etichette dei cluster

    % Trova gli indici lineari dei siti occupati
    valid = find(res.matrice);

    % Identificazione dei cluster tramite ricerca in profondità
    for iter = 1:length(valid)
        ii = valid(iter);
        if res.label(ii) == 0
            % Nuovo cluster trovato
            pila = ii;  % Inizializza la pila per DFS
            res.label(ii) = labelC;

            j = 1;
            while j <= length(pila)
                elemento = pila(j);

                % Esplora i vicini (sinistra, destra, sopra, sotto)
                vicini = [elemento - 1, elemento + 1, elemento - (L + 2),
elemento + (L + 2)];

                for v = vicini
                    if res.matrice(v) && res.label(v) == 0
                        pila(end + 1) = v;
                        res.label(v) = labelC;
                    end
                end
```

```
                j = j + 1;
            end

            labelC = labelC + 1;  % Incrementa l'etichetta per il prossimo
cluster
        end
    end

    % Rimuove i bordi dalla matrice e dalla matrice delle etichette
    res.label = res.label(2:end-1, 2:end-1);
    res.matrice = res.matrice(2:end-1, 2:end-1);

    % Verifica percolazione da sinistra a destra
    left = unique(res.label(:, 1));     % Etichette sul bordo sinistro
    left = left(left > 0);              % Rimuove zeri

    right = unique(res.label(:, end));  % Etichette sul bordo destro
    right = right(right > 0);           % Rimuove zeri

    if ~isempty(intersect(left, right))
        res.percolazioneLR = 1;  % Percolazione orizzontale esiste
    end

    % Verifica percolazione dall'alto verso il basso
    top = unique(res.label(1, :));      % Etichette sul bordo superiore
    top = top(top > 0);                 % Rimuove zeri

    bottom = unique(res.label(end, :)); % Etichette sul bordo inferiore
    bottom = bottom(bottom > 0);        % Rimuove zeri

    if ~isempty(intersect(top, bottom))
        res.percolazioneTB = 1;  % Percolazione verticale esiste
    end
end
```