Computational Physics Project: Many Body Physics

Week 1: Site Percolation on 2D Square Lattice

Math and Physics Club

Mentor: Arnav Jain

Mentee: Jayent Dev

Roll No.: 24B1232

Contents

1 Week 1: Site Percolation on 2D Square Lattice

1.1 Computational Process

1.1.1 Grid Initialization

We create an $n \times n$ square lattice where each site is randomly occupied with probability p. This is implemented by generating a matrix of random numbers uniformly distributed in $[0, 1)$ and marking sites as occupied wherever the random value is less than p.

Algorithm 1: Grid Initialization

```
function InitializeGrid(n, p)
    R ← random matrix of size n × n with values in [0,1)
    grid ← (R < p)
    return grid
end function
```

1.1.2 Cluster Identification: Hoshen–Kopelman Algorithm

The Hoshen–Kopelman algorithm efficiently identifies connected clusters of occupied sites using nearest-neighbor connectivity (up, down, left, right). A union–find data structure is used to merge clusters during a single pass through the lattice.

Algorithm 2: Hoshen–Kopelman Algorithm

```
function HoshenKopelman(grid)
    labels ← zero matrix
    parent ← empty dictionary
    currentLabel ← 0

    for each site (i,j) in grid:
        if site is occupied:
            neighbors ← labels of left and top occupied neighbors
            if neighbors empty:
                currentLabel++
                labels[i,j] ← currentLabel
                parent[currentLabel] ← currentLabel
            else:
                minLabel ← min(neighbors)
                labels[i,j] ← minLabel
                union all neighbor labels
    relabel all sites to root labels
    return labels
end function
```

1.1.3 Percolation Detection

A system is said to percolate if at least one cluster spans either vertically (top to bottom) or horizontally (left to right).

Algorithm 3: Percolation Detection

```
function CheckPercolation(labels)
    extract boundary labels
    if intersection exists:
        return True
    else:
        return False
end function
```

1.1.4 Calculated Quantities

We compute the following observables:

Weighted Average Cluster Size

$$S = \frac{\sum_s s^2 n_s}{\sum_s s n_s} \tag{1}$$

The percolating cluster, if present, is excluded.

Percolation Strength

$$P = \begin{cases} \dfrac{\text{size of percolating cluster}}{n^2}, & \text{if system percolates} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Percolation Probability

$$P_{\text{perc}}(n, p) = \frac{\text{number of percolating realizations}}{\text{total number of trials}} \tag{3}$$

1.2 Simulation Parameters

- System sizes: $L = 8, 16, 32$
- Occupation probabilities: 25 values uniformly spaced between $p = 0.3$ and $p = 0.8$
- Number of trials per (L, p): 300

1.3 Results

1.3.1 Graphs

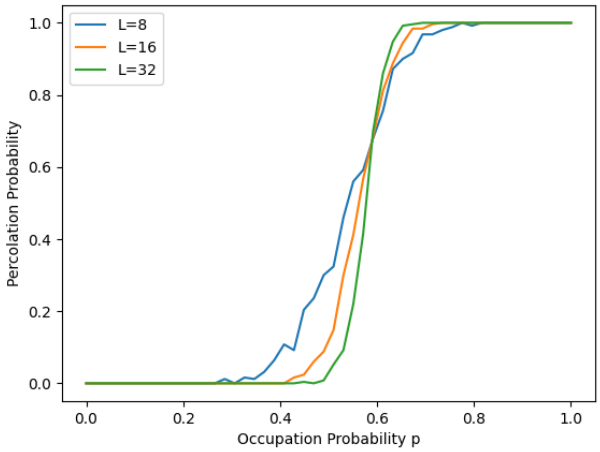

Figure 1: Percolation probability as a function of occupation probability p.

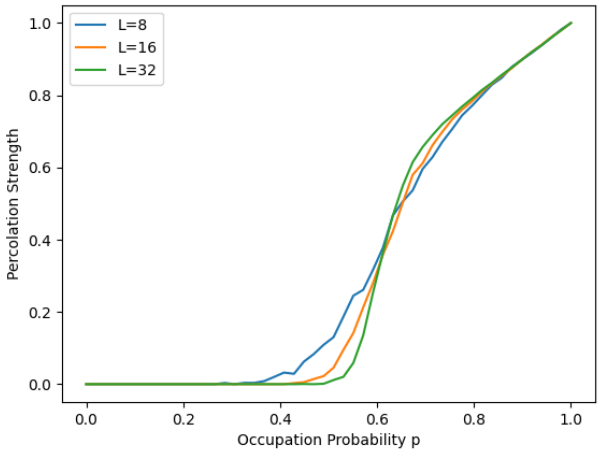

Figure 2: Percolation strength as a function of occupation probability p.

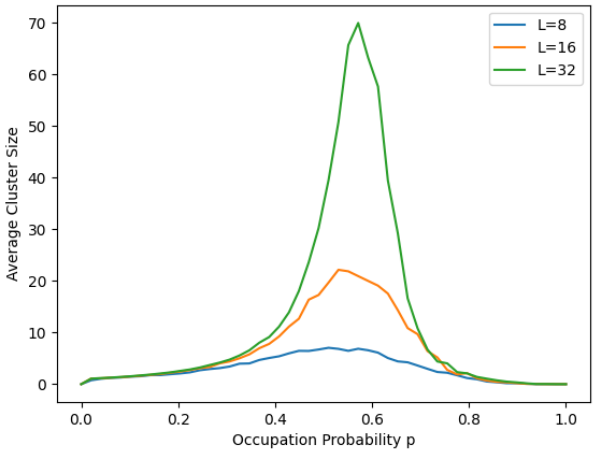

Figure 3: Weighted average cluster size as a function of occupation probability p.

1.3.2 Observations

- Percolation probability shows a sharp transition near $p \approx 0.59$.
- Percolation strength is zero below p_c and increases rapidly above it.
- Weighted average cluster size exhibits a pronounced peak near p_c.
- Peak height increases strongly with system size, indicating finite-size effects.

1.4 Conclusions

1. A clear phase transition is observed at $p_c \approx 0.55$–0.60.
2. The result agrees with the theoretical value $p_c = 0.592746\ldots$.
3. The Hoshen–Kopelman algorithm efficiently identifies clusters in a single pass.
4. Finite-size effects become more pronounced with increasing system size.

1.5 GitHub Repo Link

GitHub Repository link having the Week-1 code and report also - Many Body Physics