Journal of Nano- and Electronic Physics
Vol. **13** No 2, 02009(4pp) (2021)

Журнал нано- та електронної фізики
Том **13** № 2, 02009(4cc) (2021)

Python – a Tool for Percolation Analysis in Triangular Lattice

N. Gupta[1], M. Nath[1,*], S. Chakraborty[2], A. Bandyopadhyay[3]

[1] *Department of Physics, Lovely Professional University, Phagwara 144411, Punjab, India*
[2] *Jagannath Kishore College, P.O. Purulia, Dist. Purulia, West Bengal 723101, India*
[3] *Department of Physics, University of Gour Banga, NH-34, P.O. Mokdumpur, Dist. Malda, West Bengal, India*

(Received 11 January 2021; revised manuscript received 28 March 2021; published online 09 April 2021)

Percolation theory, developed more than 60 years before to describe the behavior of flow phenomena in porous medium, has undergone an extensive area of applications in recent years, ranging from epidemiology, financial market, soil science, pharmaceutical technology to composite material structure. Here in this paper, percolation theory is applied to the triangular lattice and its characterization has been done using Monte-Carlo simulation. Python language has been used to develop the code. For this, we have used the inbuilt libraries of Python like NumPy, SciPy, Matplotlib etc. Hoshen-Kopelman (HK) algorithm is used to identify the cluster and its numbering procedure. This algorithm is being preferred over the other methods as it consumes low computer memory and less computation time. The prime point of interest in percolation is known as percolation threshold (p_c) which is computed for our case is 0.5. We have also characterized the percolation by finding the other quantities as: normalized mass of cluster (M), percolation probability (P_p), the density of the infinite cluster (P_∞) and ordered parameter $\Omega(L)$. We have extracted critical exponents from our data and found that they match exactly with their universal values. To the best of our knowledge, we are the first group to report percolation in triangular lattice by means of HK algorithm using Python language.

Keywords: Percolation threshold, Monte-Carlo simulation, HK algorithm.

DOI: 10.21272/jnep.13(2).02009 PACS numbers: 64.60.ah, 05.10.Ln

1. INTRODUCTION

The standard theory of percolation was developed by Broadbent and Hammersley in 1957 [1], but till date it has an attractive, open and challenging application in variety of physical problems like determining the landscape ecology influenced by habitat fragmentation [2] or differentiation between benign and malignant tumors in the breast and colorectal histological images [3] or analysis of forest fire in a realistic and scientific way to prevent its propagation in forest [4] or understanding interdependent networks from the point of view of epidemic spreading [5] or giving theoretical description of a composite material with random geometry [6] or critical phenomena in statistical mechanics. Also, Golovina et al. have studied ferromagnetic ordering in ferroelectric nanoparticle using magnetic percolation theory [7]. The percolation model is appealing to a wide variety of field because of its simplicity of description, ease of visualization yet able to explain or to predict the critical behavior of the system be it regular or distorted. The realization of percolation cluster in the system is done by Monte-Carlo simulation. The probability at which the system first time makes its connectivity throughout the lattice, is known as the percolation threshold (p_c) of the system and is considered to be the most important parameter of a percolating system. In the present work, we have developed a code using Python programing to represent site percolation in a random system of triangular lattice. Though FORTRAN language is mostly favored by the Computational Physics community for coding, but Python, being an open source programing language with may inbuilt libraries like Numerical Python (NumPy), Panda, Scientific Python (SciPy), Math-plot library (matplotlib) etc. can also be used to perform the identical mathematical approach. The percolation has been analyzed with the help of HK algorithm. To the best of our knowledge, we are the first group to report the algorithm for square lattices using Python programing [8]. Here, in this paper, we have studied the percolation in triangular lattices by means of HK algorithm.

2. THE MODEL

The site percolation in triangular lattice is simulated and fitted on a square network of size $L \times L$ (2-dimension) with suitable consideration of nearest neighbors as shown in Fig. 1. The nearest neighbors of a particular site (bold dot) is shown by bold lines in Fig. 1a. If the location of any site in the lattice is represented by (x, y) as shown in Fig. 1b, then the six nearest neighbors in triangular symmetry are $(x \pm 1, y)$, $(x, y \pm 1)$, $(x - 1, y - 1)$, $(x + 1, y + 1)$. The percolation through the triangular lattice has been studied by Monte-Carlo simulation. A random number generator is used to label each site with a probability. All the sites in the lattice must be either open or closed with probability p or $q = (1 - p)$, respectively. If p is close to zero, the probability of the open sites to be isolated is maximum; whereas when p tends to unity, majority of the open sites will be amalgamated to become a part of the infinite cluster (or spanning cluster).

It is considered that an open site will be a part of a cluster if in its nearest neighborhood, there is another open site. If it is not connected to any of its nearest neighborhood then, it is considered as a unit sized cluster. An infinite cluster is formed in the lattice if there is

* madhumitanath_21@yahoo.co.in
The results were presented at the International Conference on Multifunctional Nanomaterials (ICMN2020)

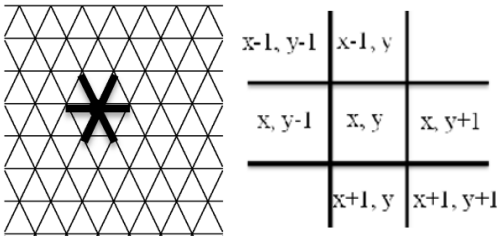


Fig. 1 – (a) The triangular lattice is fitted onto the square lattice with the suitable nearest neighbors as shown by the bold lines. (b) The nearest neighbors of any typical site (x, y) are labeled as follows: top and bottom by $(x \pm 1, y)$, left and right by $(x, y \pm 1)$, 2 diagonals by $(x - 1, y\varepsilon 1)$ and $(x + 1, y + 1)$

any connecting path between top and bottom row of the lattice or left and right side of the lattice. In a lattice, there will be always co-existence of percolating as well as non-percolating clusters. Therefore, it is always convenient to consider the normalization factor of the mass of the cluster. We have calculated the percolation probability (P_p), the density of the infinite cluster (P_∞) and the ordered parameter $\Omega(L)$. In this paper, the lattice size (L) given as 30, 70, 100 and 150. The same trend has also been observed by P. Dean [9] and he had shown that percolation condition for 48×48 lattice lie close to their limiting values for very large lattice sizes. The random matrix generator which is used as a key factor to characterize the percolation phenomenon guarantees the quality output of Monte-Carlo simulations. We have started by generating an empty square matrix. Then triangular lattice is realized on this square lattice after proper consideration of nearest neighborhood. Each empty site is assigned with a random number. This typical site then defined as open (assigned with the value – 1), if its value is less than or equal to the value of occupation probability p, which varies from 0 to 1, else it will be treated as closed site (assigned value is 0). The process of site identification and numbering was done with the help of Hoshen-Kopelman (HK) algorithm [10]. First open site (whose value is – 1) will start its numbering from 1. After that each open site will find its open nearest neighbor and equal numbering will be done for that site (in computer literature, the process of finding and uniting two sites is termed as "union-find" method) which implies that two nearest open sites will be tagged with the same number. If any open site does not find any nearest open neighbor, then a new number will be assigned for that site, which is incremented by 1 from its previous open site. The code for this algorithm was developed by Python programing and the major libraries used for this purpose are Numpy, Scipy and Matplotlib. The computational experiment was done for $N = 500$ times.

3. RESULTS AND DISCUSSION

In this section, we will discuss the variation of the following parameters for different values of p ranging from 0 to 1: percolation probability (P_p), normalized mass of cluster ($M(p)$), density of the infinite cluster (P_∞) and ordered parameter $\Omega(L)$. The open sites present in the matrix with probability p, must belong to any one of the following states: (i) it can be part of the infinite (or percolating) cluster or, (ii) it can be part of non-percolating cluster. To analyze this fact pictorially, we have 100 and $p = 0.3$ and 0.5 (see Fig. 2). The rest sites will be closed with probability $q = 1–p$. Fig. 2a shows that when $p = 0.3$, then most of the open sites belong to state (ii) as described above and as p increases to 0.5 (Fig. 2b), the probability that open sites (marked with black) will be amaulgumed and be a part of percolating cluster increases. The percolation probability is calculated by taking the ratio of the count for which the system generates percolating cluster to the total number (N) of run of computational experiment (we have taken $N = 500$).

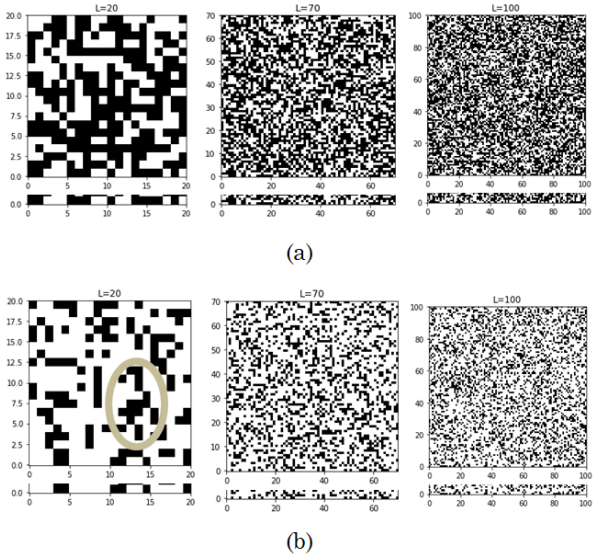


Fig. 2 – The open and closed sites in the triangular lattice of size $L = 20$, 70 and 100 are indicated by black and white respectively for (a) $p = 0.3$, (b) $p = 0.5$. The encircled region in (a) indicates that most of the open sites belong to non-percolating cluster for lower value of probability p

The percolation probability has been studied for different values of p, ranging from of 0 to 1, for different size lattices ($L = 30$, 70, 100, 150). In our simulation experiment, we have observed that for $L = 150$ lattice size, within a very narrow range of p values the system makes a phase transition. If we focus on the values of p from 0.47 to 0.52, it is observed that the system becomes percolating from its non-percolating phase. A detail investigation of p values in the range of 0.45 to 0.55 for different lattice sizes, show that at $p = 0.5$ the system becomes percolating (Fig. 3). The percolation threshold for triangular lattice (p_c) available in literature [9, 11] is in close proximity to our estimated value 0.5. This indicates the reliability of our code for percolation in triangular lattice using HK algorithm and written in Python programing.

We characterize another quantity for percolation, namely normalized mass of cluster ($M(p)$) which is a measure of occupied sites present in the lattice. It is defined by equation (1):

$$M(p) = \left\langle N_{occupied} \right\rangle / L^2 . \qquad (1)$$

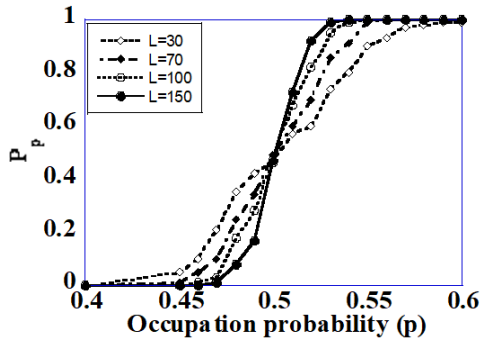


Fig. 3 – Variation of percolation probability (P_p) with occupation probability (p) is plotted for different values of p, ranging from 0 to 1

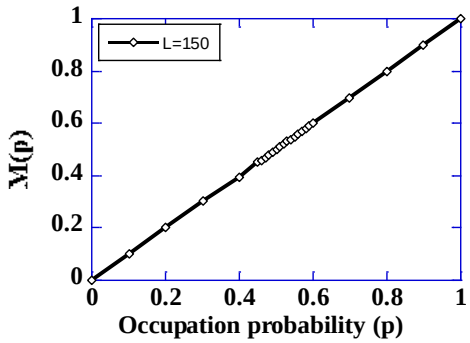


Fig. 4 – Normalized mass of cluster ($M(p)$) is plotted against occupation probability (p), ranging from 0 to 1 for $L = 150$

To calculate this for a system of size L, we have taken the average on the count of open sites present ($N_{occupied}$) in the system for N computational experiments and then normalize it by dividing it by L^2.

Fig. 4 shows that $M(p)$ is small for lower value of p, and it increases with increase in the value of p. For lower value of p, there are few sites that are open. When p value increases, more number of sites will become open which in turn will increase $M(p)$.

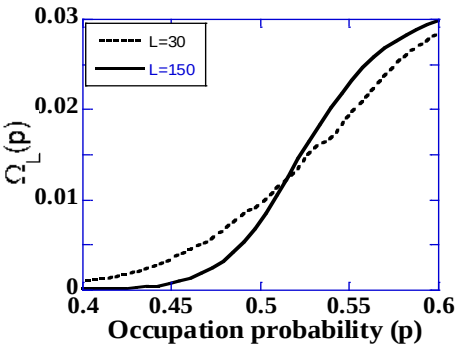
To explore more about percolation, two other parameters are discussed in this section. The first one is the order parameter $\Omega_L(p)$ which is defined as

$$\Omega_L(p) = \langle S_{\max} \rangle / L^2 . \qquad (2)$$

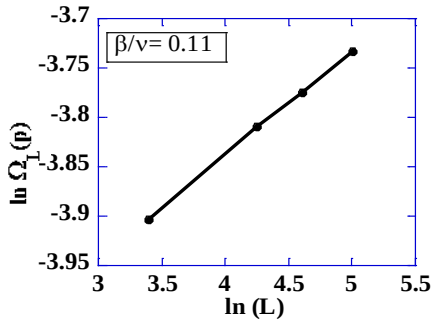
To calculate this parameter for a system of size L, we have taken the average on the count of number of sites in the largest cluster (S_{max}) in the system for N computational experiments and then divide it by L^2 to make it normalized.

Fig. 5a shows $\Omega_L(p)$ for lattice sizes $L = 30$ and 150. The other two sizes of lattice ($L = 70$ and 100) are not shown here to maintain the clarity in the image. When p is lower, then more number of sites belong to the smaller lattice size; whereas for larger p, the numbers of sites belong to the largest cluster will be more for bigger lattice.

So, we observe that the curve is steeper for the larger L and there is a cross-over in the value of $\Omega_L(p)$ occurs at $p_c = 0.5$ (percolation threshold obtained earlier also). The values of $\Omega_L(p)$ are extracted at a fixed value



a



b

Fig. 5 – (a) The variation of $\Omega_L(p)$ with occupation probability p is shown for $L = 30$ and $L = 150$. (b) Plot of $\ln\Omega_L(p)$ with $\ln L$ at a fixed value of $(p - p_c)L^{1/\nu}$ gives the value of critical exponent $\beta/\nu = 0.11$

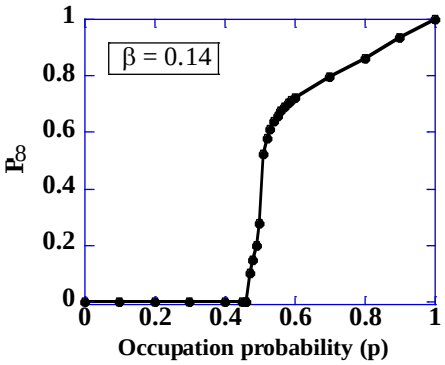


Fig. 6 – The density of infinite cluster (P_∞) with occupation probability p is plotted for $L = 150$. The critical exponent β is derived near percolation threshold (p_c) and it is found to be 0.14

of $(p - p_c)L^{1/\nu}$ and these are plotted with $\ln\Omega_L(p)$ vs. $\ln L$ (Fig. 5b). The slope of the straight line gives $\beta/\nu = 0.11$.

Another parameter discussed here is the density of infinite cluster which is defined as the probability that an occupied site belongs to the infinite cluster P_∞. Fig. 6 shows that for $p < p_c$, no infinite cluster belong to the lattice, so P_∞ value is zero. Near p_c, the probability that any randomly chosen site belong to infinite cluster becomes non-zero and increases with occupation probability.

This increase may characterized by the critical exponent β given by the standard relation in percolation as

$$P_\infty = 0 \quad \text{for} \quad p \le p_c; \qquad = \left(p - p_c\right)^\beta \quad \text{for} \quad p > p_c. \tag{3}$$

The critical exponent derived from our plot is $\beta = 0.14$. This value is exactly identical with the universal value for percolation in two dimensional lattices [12].

4. CONCLUSIONS

The site percolation in triangular lattice is presented through HK algorithm and the algorithm has been developed by using open-source software Python. The code has been developed with the help of Monte-Carlo simulation. In perspective to the occurrence of reliability in the outcome of percolation process in triangular lattice, the computational experiment has been performed for 500 times. The percolation threshold, the critical exponents for the above said lattice show that the code developed by our group using the open source software Python is highly accurate to discuss the percolation problem and ensures that Python can be an excellent tool for coding to Physics community as compared to the other conventional programming languages like FORTRAN or C. In addition, we have studied the characteristics of a percolating system through percolation probability (P_p), normalized mass of cluster ($M(p)$), density of the infinite cluster (P_∞) and ordered parameter $\Omega(L)$.

REFERENCES

1. S. Broadbent, J. Hammersley, *P. Camb. Philol. Soc.* **53**, 629 (1957).
2. M.M. Holland, P.G. Risser, Ecotones. *The role of landscape boundaries in the management and restoration of changing environments, MAB Series* (New York: Chapman and Hall: 1991).
3. G.F. Roberto, M.Z. Nascimento, A.S. Martins, T.A.A. Tosta, P.R. Faria, L.A. Neves, *Comput. Graph.* **84**, 134 (2019).
4. B. Porteriea, N. Zekri, J.P. Clerc, J.C. Loraud, *Combust. Flame* **149**, 63 (2007).
5. L. Jiang, Q. Xu, B. Ouyang, Y. Lang, Y. Dai, J. Tong, *Math. Probl. Eng.* **2018**, 1 (2018).
6. A. Bunde, W. Dieterich, *J. Electroceram.* **5** No 2, 81 (2000).
7. I.S. Golovina, S.V. Lemishko, A.N. Morozovska, *Nanoscale Res. Lett.* **12** No 1,382 (2017).
8. M. Nath, A. Bandyopadhyay, S. Chakraborty, *AIP Conf. Proc.* **2220**, 130005-1 (2020).
9. P. Dean, *Mathematical P. Camb. Philol. Soc.* **59**, 397 (1963).
10. J. Hoshen, R. Kopelman, *Phys. Rev. B* **14**, 3438 (1976).
11. P. Deckmyn, G. Davies, D. Bell, *Appl. Math. Model.* **19**, 258 (1995).
12. M.E. Levinshteln, B.I. Shklovskil, M.S. Shur, A.L. Efros, *J. Exp. Theor. Phys.* **42**, 197 (1975).

Python – інструмент для аналізу перколяції в трикутній ґратці

N. Gupta[1], M. Nath[1], S. Chakraborty[2], A. Bandyopadhyay[3]

[1] *Department of Physics, Lovely Professional University, Phagwara 144411, Punjab, India*
[2] *Jagannath Kishore College, P.O. Purulia, Dist. Purulia, West Bengal 723101, India*
[3] *Department of Physics, University of Gour Banga, NH-34, P.O. Mokdumpur, Dist. Malda, West Bengal, India*

Теорія перколяції, яка запропонована понад 60 років тому для опису поведінки явищ течії в пористому середовищі, в останні роки набула широкого застосування, починаючи від епідеміології, фінансового ринку, ґрунтознавства, фармацевтичних технологій і закінчуючи структурою композиційних матеріалів. У статті теорія перколяції застосовується до трикутної ґратки, а її дослідження було виконано з використанням моделювання Монте-Карло. Для розробки коду використовувалася мова програмування Python. Для цього ми застосували вбудовані бібліотеки Python, такі як NumPy, SciPy, Matplotlib тощо. Алгоритм Хошена-Копельмана використовується для ідентифікації кластера та процедури його нумерації. Цей алгоритм має перевагу над іншими методами, оскільки він потребує менше пам'яті і часу на обчислення. Об'єктом підвищеного інтересу в теорії перколяції є поріг перколяції (p_c), який у нашому випадку є рівним 0,5. Ми також охарактеризували перколяцію, знайшовши інші величини, такі як нормована маса кластера (M), ймовірність перколяції (P_p), густина нескінченного кластера (P_∞) і параметр порядку $\Omega(L)$. Ми отримали критичні показники з наших даних і виявили, що вони точно відповідають своїм універсальним значенням. Наскільки нам відомо, ми є першою групою, яка повідомила про перколяції в трикутній ґратці за допомогою алгоритму Хошена-Копельмана з використанням мови Python.

Ключові слова: Поріг перколяції, Моделювання Монте-Карло, Алгоритм Хошена-Копельмана.