COMPUTATIONAL STUDIES OF PERCOLATION: DETERMINATION OF THE CLUSTER NUMBER SCALING FUNCTION FOR LATTICES IN 2 DIMENSIONS

By

Léonard NDUWAYO

BSc (Licencié en Sciences Physiques), University of Burundi

THESIS

Submitted in partial fulfillment of the requirements for the degree of Masters in Science in Physics at the University of Natal

Pietermaritzburg

December, 2002

Dedication

To my wife

Ida Ntawundora,

To my daughters

Ella Monia Ndemesha, Ornella Marina Girimbere, Snella Sabrina Akimana and Diella Marthe Gateka.

Their moral support and constant patience kept me going through my years of studies and contributed so much to my success.

Acknowledgements

I would like to thank my supervisors Dr Nithaya Chetty and Dr Robert Lindebaum for suggesting the topic of this thesis and for their support, which enabled me to progress with confidence in the field of Computational Physics.

I would also like to thank Professor Robert Ziff (University of Michigan) for his invaluable suggestions and for helping us develop some of the algorithms we have used.

I thank Mitchell de Bruyn for his efficient random number generating program. Also, I appreciate the help received from all the staff and students of the Physics Department. I am indebted to the numerous friends I have met in South Africa for their tireless assistance in various aspects of my student life.

Lastly, but not least, I acknowledge the Burundi Government for the financial support of my studies.

Declaration

I declare that this thesis is a result of my own research, except where specially indicated, and has not been submitted for any other degree or examination to any other university.

Signed:  Date: 18/03/2003

We hereby certify that this statement is correct.

Dr Nithaya Chetty

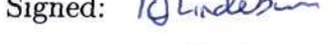
Supervisor

Signed: 

Date: 19/03/2003

Dr Robert Lindebaum

Supervisor

Signed: 

Date: 14/03/2003

Pass with distinction

Abstract

The main aim of this work is to study percolation theory on regular two dimensional (2D) lattices in order to determine numerically the cluster number scaling function. As an introduction to the topic, a brief overview of this theory is given by answering a two-fold question: *What is percolation and why is it important?* Some applications of percolation theory are also discussed.

The algorithms are written in the Fortran 90 programming Language and the programs are run primairly on Linux-based systems. Firstly, the simulation for both site and bond percolation is perfomed to determine the critical concentration p_c. The value of p_c for infinite lattices is deduced by means of a linear correlation and regression analysis. Secondly, the cluster number scaling function $f(z)$ is numerically determined by assuming its expression as proposed by Stauffer (Nakanishi and Stanley, 1980). Some of the relevant critical exponents are also calculated.

The numerical values of p_c obtained with our algorithm are close to the theoretical values found in the literature. For all values of the occupation probability p from zero to one, the number of clusters of size s per lattice site or bond $n_s(p)$ is recorded as a

function of both s and p. The computations demand a high processing speed and large memory. Graphs of $f(z)$ are plotted and verify the universal behaviour in the limits of infinite sizes for the lattices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of this introduction is to answer the two-fold question: *What is percolation and why is it important?* Some applications of the percolation theory are given.

In nature, it is possible to distinguish between two main kinds of systems (Giordano, 1997). On one hand there are systems called *deterministic* systems which are described by some mathematical rules (equations with boundary conditions and determined solutions). For example, the solution of the Schrödinger equation ($\nabla^2\Psi + \mathbf{k}^2\Psi = 0$) for a free particle in a box yields the particular values of its wave vector $\mathbf{k}$ compatible with the annihilation of the wave function on the edges of the box. On the other hand, there is a different class of systems known as *random* or *stochastic* systems, for example, the ideal gas. The gas has an internal energy independent of its volume and satisfies Joule's law for internal energy. These two requirements are, from the point of view of kinetic theory, both equivalent to saying that intermolecular attractions are negligible and the only interactions between particles are elastic shocks. Molecules should be of negligible

volume. Boyle's law, Joule's law, Dalton's law for partial pressures and Avogadro's hypothesis express the general state function of an ideal gas, namely $PV = NRT$. Relating to real gases, these laws are obeyed asymptotically only when their pressure tends to zero: the case of perfect order (Dictionary of Physics, 1991).

Disorder is a fundamental process which describes many real systems. The knowledge of disordered systems enables us to interpret experimental observations and predict properties of such systems. The term "disordered structures" induces variation in shape and constitution of randomness in the morphology of a system. By morphology, it is useful to understand two major aspects: topology and geometry. The allotropy varieties are powerful illustrations. An example from mineralogy is carbon, which may exist in the familiar black amorphous form (graphite) or in the beautiful crystalline form of diamond. From our human point of consideration, graphite and diamond are materials of very different economic and esthetic values. While both are allotropes of carbon, graphite and diamond have different atomic arrangements and bonding form. These two differences serve as an important basis to distinguish them[1].

Typical examples of randomness in systems are diffusion of a fluid and Brownian motion. It appears as a random process that involves disordered dynamics. Thus the statistical physics of disordered systems has to take into account the effect of both morphology and constitution.

[1]http://www.geo.ucalgary.ca/~tmenard/crystal/diamond.html

It is not possible to resolve all equations of motion concerning all particles of a disordered system in order to determine its macroscopic properties – neither is it necessary. An approach relating to average values of microscopic quantities is necessary to describe disordered systems; percolation theory is helpful in these situations.

In general, percolation may be defined as a *transition associated with the formation of a continuous path spanning an arbitrarily large range* (Sahimi, 1994). Literally, percolation also means an invasion of a wet fluid through a porous medium such as hot water flowing in coffee in a percolator. The study of percolation as a branch of statistical physics began in 1954 when Broadbent, a British physicist, suggested masks for protecting coal mine workers from breathing toxins (Chabot, 1994). Those masks had a tube in which small particles of coal were compacted together. By exerting pressure on the particles, the porosity of that medium would change and even become zero with obvious serious consequence; workers would breathe easily when a certain proportion p of pores were opened. $p = 0$ corresponds to the mask being blocked and $p = 1$ corresponds to all pores being open. The purpose of percolation theory is to identify what happens between these two extreme values of porosity.

According to the Dictionary of Science and Technology (1992), the word percolation has many meanings and applications according to the domain of interest. In **Physical Chemistry**, it is applied to the gradual movement of a liquid through a porous medium. Thus, percolation filtration is known as a refinery process that percolates through an adsorbent material to remove impurities. In **Food Technology**, it is the

process in which coffee is brewed in a percolator. The application used in **Hydrology** is the movement of water under hydrostatic pressure or by the force of gravity through interstices in rock, soil or other porous materials. Applied to a glacier or an ice sheet, the percolation zone is the area where some degree of surface melting occurs but where the snow layer is not completely soaked through or brought up to melting temperature and the melt water refreezes within the same layer.

In **Computer Programming**, three areas of interest may be considered for percolation. The first deals with the transfer of data from secondary storage to the main storage or from slower devices to faster devices. The second application relates to error recovery, which means the passing of control from a low level routine to a high level recovery routine. Thirdly, in bubble sorting, percolation may be considered as the rising of lower valued elements toward the top of the list.

Otherwise, percolation is defined in the leaching treatment of minerals such as a gentle flow of a solution through an ore bed to extract minerals. This notion is used in **Mining Engineering**. In this field, the percolation test is a method used for ascertaining the rate at which soil can absorb waste fluids.

Mostly, the word percolation is used as a qualification word eg. *invasion percolation model, percolation limit, percolation network, percolation transitions,* etc. (Encyclopedia of Applied Physics, 1994). Invasion percolation has been used to study low capillary number creeping flows dominated by surface tension. In the opposite limit of high

capillary numbers, surface tension can be neglected and diffusion-limited fingering is observed. Invasion percolation is used for describing the process of filling the active sites. An invasion edge is chosen where an external pressure is assumed to be applied and the most active bond available at each point in time, that is the bond with the largest radius, is sequentially occupied.

Two examples of percolation in **Solid State Physics** are the invasion of magnetic domains close to the magnetic transition temperature, and the infusion of impurities in an insulating material that establishes a continuous conduction path. In **Theoretical Physics**, the percolation network is a (usually infinite) network whose edges have been randomly assigned values of 1 or zero (called conducting or insulating links respectively). The problem for a given network is to determine the number or concentration of conducting links required for the network to become conducting.

The percolation transitions are geometric phase transitions that involve the clusters formed from sites (or bonds) on lattices that are occupied randomly with a probability p. For a very small value of p only isolated clusters of occupied sites exist. But as p tends to p_c the threshold concentration, the size of typical clusters diverges and an incipient infinite cluster that spans the lattice appears (Weiss, 1994)

The above definitions show how percolation is associated with no predictable and dynamic processes (randomness effect). Intuitively, the idea of disorder is related to the percolation process. Finally, one of the most important developments in modern physics

has been the realisation that certain properties of percolation as well as other phase transitions are universal and therefore independent of the micro-structure of the lattice. Percolation theory is a useful theoretical description, mainly a statistical description of behaviour which uses statistical arguments. It is in this sense that many disordered systems are simulated by the model of a random walker.

Concepts of percolation play an important role in the modeling of complex and disordered systems. The critical value of the percolation probability depends on the dimensionality and the connectivity of a system. Percolation describes phenomena that essentially involve a binary mixture of mutually exclusive states. Some applications of percolation theory are given in Table 1.1 (Deutscher, Zallen and Adler, 1983).

As mentioned by Sahimi (1994), percolation theory is useful as a model of disordered systems. It allows for the understanding of certain aspects of liquid-glass transitions, describes how small branching molecules react and form very large macromolecules and how fluid particles spread through a random medium.

This fluid could be liquid, vapor, heat flux, electric current, inflection of a solar system, ... The fluid particles decide where to go in the medium and the medium dictates the paths of particles. This process resembles the flow of coffee in a percolator (Sahimi, 1994).

Percolation theory may be used to determine whether a system is macroscopically connected or not. Connectivity plays a fundamental role in many phenomena involving disordered media. These phenomena occur as macroscopic effects such as superconduc-

Table 1.1: Examples of systems and phenomena involving critical transitions.

Phenomenon or system	Transition
Flow of liquid in a porous medium	Local/extend wetting
Spread of disease in a population	Containment/epidemic
Communication or resistor networks	Disconnected/connected
Composite material	Insulator/conductor
Super-conductor material	Normal/super-conducting
Melting	Solid/liquid
Dilute magnets	Paramagnetic/ferromagnetic
Polymer gelation, vulcanization	Liquid/gel
Mobility edge in amorphous semiconductors	Localized/extended states

tivity and superfluidity. They are observed in experiments under extreme conditions such as strong magnetic fields and low temperatures. These effects are indeed indicative of the quantum nature of particles. Above a well defined temperature, the flow properties of helium-4 are qualitatively no different from those of any other liquid. Immediately below the critical temperature, liquid helium acquires frictionless super-flow properties, and its difference in the behavior compared to normal helium increases sharply as the temperature drops. The specific heat has a near logarithmic singularity at the superfluid phase transition. Similarly, the superconducting materials are capable of carrying resistanceless current once they are cooled below a certain critical temperature and show a discontinuity in the heat capacity at the transition temperature (Lerner and Trigg,

1991). The penetration of the superfluid in the medium was viewed as a percolation system.

Furthermore, percolation theory is a tool for interpreting experimental data and to gain insight into the structure of media. During World War II, Flory and Stockmayer used percolation theory to describe how small branching molecules form larger macromolecules if more and more chemical bonds are formed between the original molecules. Flory went on to get the Nobel prize in 1974 for his fundamental achievements in the physical chemistry of macromolecules[2]. In 1982, the Nobel prize was awarded to Wilson for his contributions to the *scaling theory of phase transitions* which are aspects of percolation theory (Sahimi, 1994). Nearly one thousand articles are produced on the topic of percolation each year. Percolation theory remains a domain of active research in this third millennium.

The above dicussion constitutes the basis of my motivation for research into the percolation theory. In Chapter 2 we discuss random processes. Some properties of these systems have fractional dimensions and need to be simulated by means of appropriated Monte Carlo Methods. Chapters 3 and 4 contain numerical methods and numerical results for both site and bond percolation respectively . Chapter 5 includes the numerical determination of the cluster number scaling function for lattices in two dimensions. At the end we draw our concluding remarks.

[2]http://www.nobel.se/chemistry/laureates/1974/flory-autobio.html

Chapter 2

Random Processes

This chapter introduces the concepts of fractals and random processes such as random walks and self-avoiding walks. Also, we discuss how random numbers are useful in the simulation of random systems. These concepts are central in the studies of percolating systems, for instance a spanning cluster is a fractal.

2.1 Fractals

By looking at our surroundings we find that clouds, mountains, coast-lines, etc. are complex in shape as their description is the domain of "geometrically chaotic figures"[1]. It becomes difficult to describe the elements of nature by the simple use of mathematical rules. The following notes give the history and the background of fractional dimensions: *The roots of fractal geometry can be traced to the late 19th century, when mathematicians*

[1]Hassan,M. K. in http://www.swin.edu.au/chem/complex/vp/vp07/vp07.html/

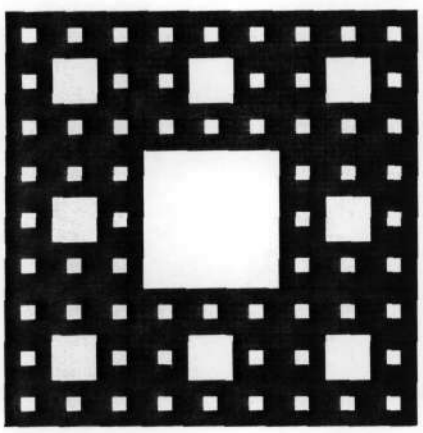
started to challenge Euclidian principles. Fractional dimensions were not discussed until 1919, however, when the German mathematician Felix Hausdorff put forward the idea in connection with the small scale structure of mathematical shapes. As completed by the Russian mathematician A.S. Besicovitch, Hausdorff dimensionality was a forerunner of fractal dimensionality. Other mathematicians of the time considered such strange shapes as pathologies that had no significance. This attitude persisted until the mid-20th century and the work of Mandelbrot, a Polish-born French mathematician who moved to the United States in 1958. His 1961 study of similarities in large and small scale fluctuations of the stock market was followed by work on phenomena involving nonstandard scaling, including the turbulent motion of fluids and the distribution of galaxies in the universe.[2]

The concept of fractals introduced by Mandelbrot characterizes those irregular patterns of nature. Therefore, a fractal can be defined as a shape made of similar parts to the whole. The simple way to understand the construction of a fractal is to repeat a deterministic operation at different length scales such as is found in Koch curves (Giordano, 1997) or the Sierpinski carpet[3] represented in Figure 2.1. From a 3×3 grid of squares, the interior central square is removed. Eight squares are then left and form a ring with sharp corners. Each of the eight squares is divided into a 3×3 grid of squares, and the new central squares are removed, and so on... The operation is iterative at different scales. In Figure 2.1 the operation is limited to level 3.

[2]http://www.astroa.physics.metu.edu.tr/MANUELS/cgi_perl.tut/Esbsams/tractal.html/

[3]http://www.swin.edu.au/chem/complex/vp/vp07/vp07.html/

Figure 2.1: Sierpinski Carpet



Stanley was the first to find that the structure of percolation clusters can be described by the fractal concept (Nakanishi and Stanley, 1980). Thus fractal dimensions of percolation and of substructures composing percolation clusters are used in the study of geometrical properties for a full characterization (of percolation clusters). The fact that a percolation phenomenon can be described by the concept of a fractal implies that percolation is a stochastic process. In the following section, a definition of what is meant by a random process or stochastic process is given.

2.2 Random processes

A fundamental notion in probability theory is a random process: an experiment whose outcome is not determined in advance. The ensemble of all possible outcomes of the experiment may be known but each event occurs with a certain probability. To illustrate this notion, random walks and self avoiding walks are considered below.

2.2.1 Random walks (RW)

A classical random walker is well known and defined as a simple discrete model of Brownian motion with applications to gambling, heat flow, stock market and diffusion. In 2D, the simulation of RW is based on the following rule:
It is equally likely to pick any of four directions to head forward, each with a probability equal to 1/4. Consequently, the motion will generally turn around the origin.

The question that may be asked is: "What is the average location for many random walkers?" Since the walkers are independent of each and other and all paths are equally likely, for every walker away from the origin there is another which travels in the opposite direction. Therefore, the expected final average position is zero. In this situation, it becomes more interesting to evaluate the mean square displacement $< \mathbf{r}^2 >$ after n steps which is,

$$< \mathbf{r}^2 >= n. \tag{2.1}$$

The proof of the equation 2.1 is based on elementary statistical induction. For each random walker, after n steps, the position vector $\mathbf{r}$ is the sum of n elementary displacement vectors $\mathbf{r}_i$, i = 1,...,n. By calculating the square of that sum and its average, the result involves scalar products of the form $\mathbf{r}_i \cdot \mathbf{r}_j$. For $i = j$, the scalar product is the square of the nearest-neighbour distance assumed to be unity. On the other hand, for i and j different, the scalar product may be -1 or +1 with equal probability. Sometimes $\mathbf{r}_i \cdot \mathbf{r}_j$ is zero when $\mathbf{r}_i$ is perpendicular to $\mathbf{r}_j$. On average, the sum of the products $\mathbf{r}_i \cdot \mathbf{r}_j$ cancel out except for $i = j$ where they are equal to the unity. Thus the mean squared sum equals n.

$$
\begin{aligned}
&\mathbf{r} = \textstyle\sum_{i=1}^{n} \mathbf{r}_i \\
&< \mathbf{r}^2 >= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{r}_i \cdot \mathbf{r}_j \\
&=< \textstyle\sum_{i=1}^{n} \mathbf{r}_i^2 + \sum_{i \neq j}^{n} \mathbf{r}_i \cdot \mathbf{r}_j > \\
&=< n + 0 > \\
&= n
\end{aligned}
$$

The above relation is also valid for a random walker in 3D when the motion is made up of unit steps in directions chosen randomly at each step.

2.2.2 Self avoiding walks (SAW)

A self-avoiding walker is a random walker that does not intersect itself. It is an intermediate case between a free particle and random walker. The mean square displacement $< \mathbf{r}^2 >$ after n steps is proportional to t^{β} where t is proportional to n; the exponent β is a real number between 1 and 2.

2.3 Random numbers

Random numbers have a variety of applications. Apart from their usual use in specifying a random sample from a large population, random numbers can also be used for integrating complicated functions of one or more variables. The estimation of the real number π and the approximation of the irrational number e (the Naperian number) can be done by means of random numbers using to appropriate Monte Carlo methods (Ellis *et al.*, 1994).

Thus the simulation of a random system on a computer needs a generator of random numbers. For example, a linear congruential random number generator produces a set of integers from 0 to another integer by using a recurrence relation (Giordano, 1997). From this set, it is possible to generate another set of real numbers evenly distributed in a range of any interval of interest.

Computer languages generally have a built-in function for producing random numbers. In Appendix A.1, we give a copy of the codes for the random number generator used in this project.

Chapter 3

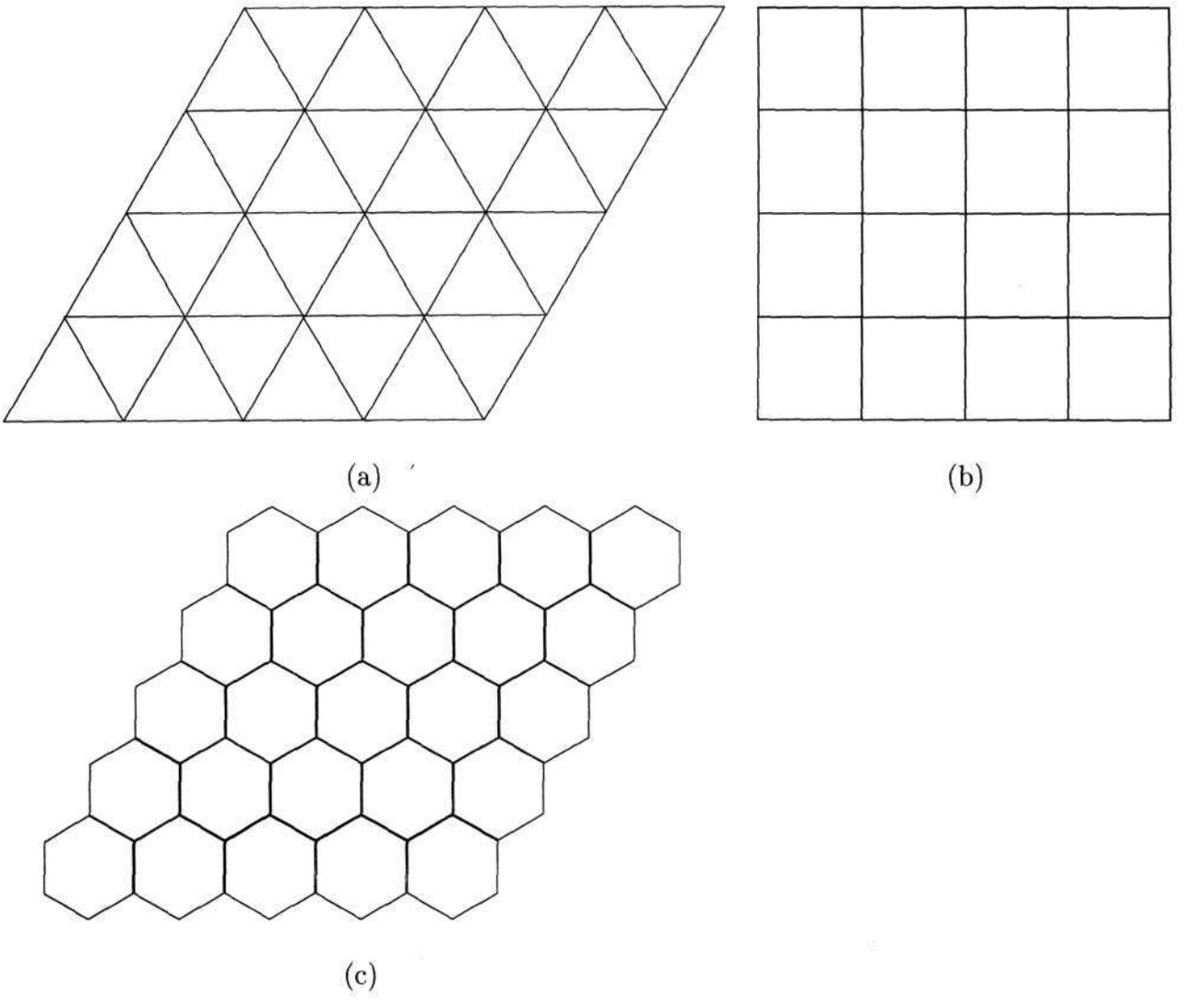
Site Percolation and Bond Percolation - Numerical Methods

In the present chapter, both site and bond percolation are investigated by following the steps: the labelling of sites or bonds, the determination of nearest neighbours of a site or bond and the determination of the onset of the spanning cluster.

Our studies focus on regular lattices in 2D: triangular, square and honeycomb lattices. These lattices may be viewed as grids where the bars are called bonds. The junction points are called sites. Percolation models involve the random occupation of sites or bonds on these lattices. Site percolation corresponds to when the sites are occupied with probability p, and bond percolation when the bonds are occupied with probability p. Figure 3.1 shows the lattices on which our work is based.

Percolation states are created by occupying sites or bonds one by one randomly chosen

Figure 3.1: Triangular lattice (a), Square lattice (b) and Honeycomb lattice (c).



on the lattice, starting with an empty lattice (Newmann and Ziff, 2001). This may be done as follows:

1) Sites or bonds are listed in some meaningful order.

2) At the j^{th} step in the algorithm, a site (bond) is chosen randomly from the list of unoccupied sites (bonds) – from j to total number of sites (bonds). This site (bond) is

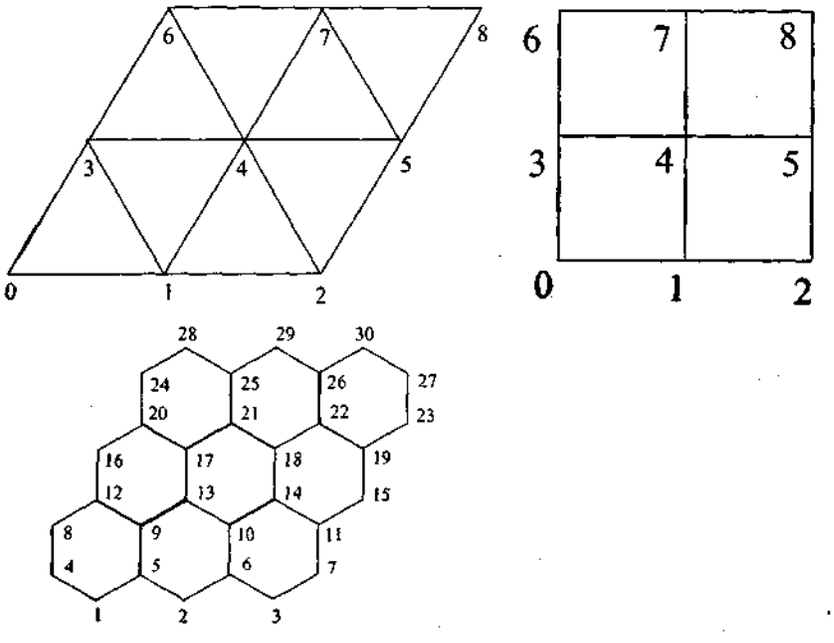
then occupied and its label is switched with that of the j^{th} site (bond).

3.1 Site percolation

3.1.1 Labelling of sites

In our procedure, each site on a given lattice is labelled in an unique manner as is shown in Figure 3.2 for $N = 3$.

Figure 3.2: Diagrams showing labels of sites on lattices ($N = 3$)



The total number of sites in the lattice is related to the value of N where N is the number of sites along the bottom edge. It is N^2 for the triangular and square lattices.

For a honeycomb lattice, the total number of sites is $2(N+1)^2-2$. The total number of sites in the lattice is also called the system size. Table 3.1 summarizes the properties of the lattices.

Table 3.1: Lattice properties

	Kind of lattices		
	Triangular	Square	Honeycomb
Number of sites along the bottom edge of the lattice	N	N	N
Number of sites in entire lattice	N^2	N^2	$2(N+1)^2-2$
Number of bonds in entire lattice	$(3N-1)(N-1)$	$2N(N-1)$	$3N^2+4N-1$
Generic number of nearest neighbours of a site	6	4	3
Generic number of nearest neighbours of a bond	10	6	4
Nearest neighbours of site L	$L-N$; $L-N+1$ $L-1$; $L+1$; $L+N-1$; $L+N$	$L-N$; $L-1$; $L+1$; $L+N$	$L-N-1$; $L+N+1$ $L+N$ or $L-N$[1]

3.1.2 Determination of the nearest neighbours of a given site

The nearest neighbours of a site are those sites that are closest to the given site. The number of nearest neighbours is also called the coordination number of the lattice (Stauffer and Aharony, 1994). The number of nearest neighbours of each (infinite) lattice is listed in Table 3.1. The finite size of the lattices induces edge effects that we have to take into account in our computations. This is especially important when we define the nearest neighbours of a site at an edge of the sample.

Triangular and square lattices

Edge sites and the corner sites of the triangular lattice or square lattice may be easily recognized from their labels. There are in four categories. The first category (bottom edge) of these sites has a label which is between 0 and $N-1$. The second category (right edge) has a label which is a multiple of N minus one. The third category (left edge) is characterized by a label which is a multiple of N. Lastly, the fourth category (top edge) of sites has a label in the range $N(N-1)$ to N^2-1. With the exception of the corner sites with the labels 0, $N-1$, $N(N-1)$ and N^2-1, every edge site has four

[1] In the honeycomb lattice, a site of label L has in general three nearest neighbours. The two first neighbours have respectively labels $L-N-1$ and $L+N+1$. The third nearest neighbour has a label $L-N$ if that site labelled by L is the origin of the bond of orientation 3 as mentioned in section 3.2. Otherwise, the third nearest neighbour has a label $L+N$

nearest neighbours in the case of the triangular lattice and three nearest neighbours in the case of the square lattice. Table 3.2 gives labels of the nearest neighbours for the edge sites.

Table 3.2: Nearest neighbours of edge and corner sites in triangular and square lattices.

		Labels of nearest neighbours of site	
Site label	Localization of the site on the lattice sample	Triangular lattice	Square lattice
L	Bottom edge	$L-1$, $L+1$, $L+N-1$, $L+N$	$L-1$,$L+1$,$L+N$
L	Left edge	$L-N$, $L-1$, $L+N-1$, $L+N$	$L-N$,$L-1$,$L+N$
L	Right edge	$L-N$, $L-N+1$, $L+1$, $L+N$	$L-N$,$L+1$,$L+N$
L	Top edge	$L-N$, $L-N+1$, $L-1$, $L+1$	$L-N$,$L-1$,$L+1$
0	Bottom left corner	1, N	1, N
$N-1$	Bottom right corner	$N-2$, $2N-2$, $2N-1$	$N-2$, $2N-1$
N^2-N	Top left corner	$N(N-2)$, $N(N-2)+1$ $N(N-1)+1$	$N(N-2)$, $N(N-1)+1$
N^2-1	Top right corner	N^2-1, $N(N-1)-1$	N^2-2, $N(N-1)-1$

Honeycomb lattices

In general, the number of nearest neighbours of a site labelled by L in the honeycomb lattice sample is three and have been listed in Table 3.1. The exceptional sites are localized on the four edges of the system (bottom, right, left and top). These exceptional sites are easily recognized by studying their labels. These labels are less than or equal to N (bottom edge), even multiples of $(N+1)$ (left edge), even multiples of $(N+1)$ minus one (right edge), and greater than or equal to the total number of sites minus N (top edge). The sites with labels $N+1$ and $2(N+1)^2-2$ are also amongst the exceptional sites since they have only two nearest neighbours.

Table 3.3: Nearest neighbours of exceptional sites in the honeycomb lattice

Site label	Localization of the site on the lattice sample	Labels of nearest neighbours
L	Bottom edge	$L+N$, $L+N+1$
L	Left edge	$L-N-1$, $L+N+1$
L	Right edge	$L-N-1$, $L+N+1$
L	Top side	$L-N-1$, $L-N$
$N+1$	Left edge	1, $2(N+1)$
$2(N+1)^2-2$	Top edge	$2(N+1)^2-N-3$, $2(N+1)^2-N-2$

3.1.3 Labelling of clusters

The salient feature of our simulation of percolation on the lattices is to fill the system systematically. Each site is chosen randomly among the total number of unoccupied sites in the lattice. At each step of filling a site, the occupation probability p of sites in the system is defined as the fraction of occupied sites to the total number of sites. Connected occupied sites form clusters (Pang, 1997). Two occupied sites are connected when there is at least one path between them. That path consists of one or more bonds joining nearest neighbour occupied sites. Otherwise, two nearest neighbour sites are connected if they are both occupied (Sahimi, 1994). The size of a cluster is the number of occupied sites that it contains.

During the occupation of a site, three situations can arise. Firstly, a cluster of size one is created. The new occupied site is isolated: none of its nearest neighbours is occupied. In this case, we say that a new cluster is formed and needs to be assigned a new cluster label commencing from one. Secondly, the occupied site has one of its neighbours which is occupied and the size of the cluster is increased by one. In this situation, the label of the cluster remains unchanged. Lastly, the new occupied site establishes a bridge between two clusters with different labels. In this case, the total number of clusters is reduced by one and the new cluster formed has a size which is the sum of the sizes of the joined clusters plus one. The smallest of the two clusters is relabelled for computational convenience.

3.1.4 Determination of the spanning cluster

During our process of filling sites, a change in the system state occurs. Its topological structure evolves from a disconnected system to a connected one. As p increases, the sizes of clusters increase and some clusters merge into large clusters. Above a threshold probability p_c, one cluster reaches all boundaries of the lattice. This particular cluster is called the spanning or percolating cluster (Stauffer and Aharony, 1994).

The determination of the numerical value of p_c is done by checking when for the first time in our occupation process a cluster having occupied sites on all four edges of the lattice emerges. The value of p_c is then the ratio between the number of occupied sites and the total number of sites at this instant in time.

3.2 Bond percolation

As mentioned above, a bond is limited at its extremities by two sites which we refer to as its origin site (bottom left) and its end site.

In the triangular lattice, three orientations of bonds are possible. Orientation 1 corresponds to a bond which makes an angle of 0° with the horizontal; the second orientation labelled 2 is at 60° to the horizontal and the third orientation labelled 3 is at 120° to the horizontal direction. For the square lattice, only two orientations are possible: the horizontal and at a right angle with the horizontal. In our codes, these orientations are

labelled 1 and 2.

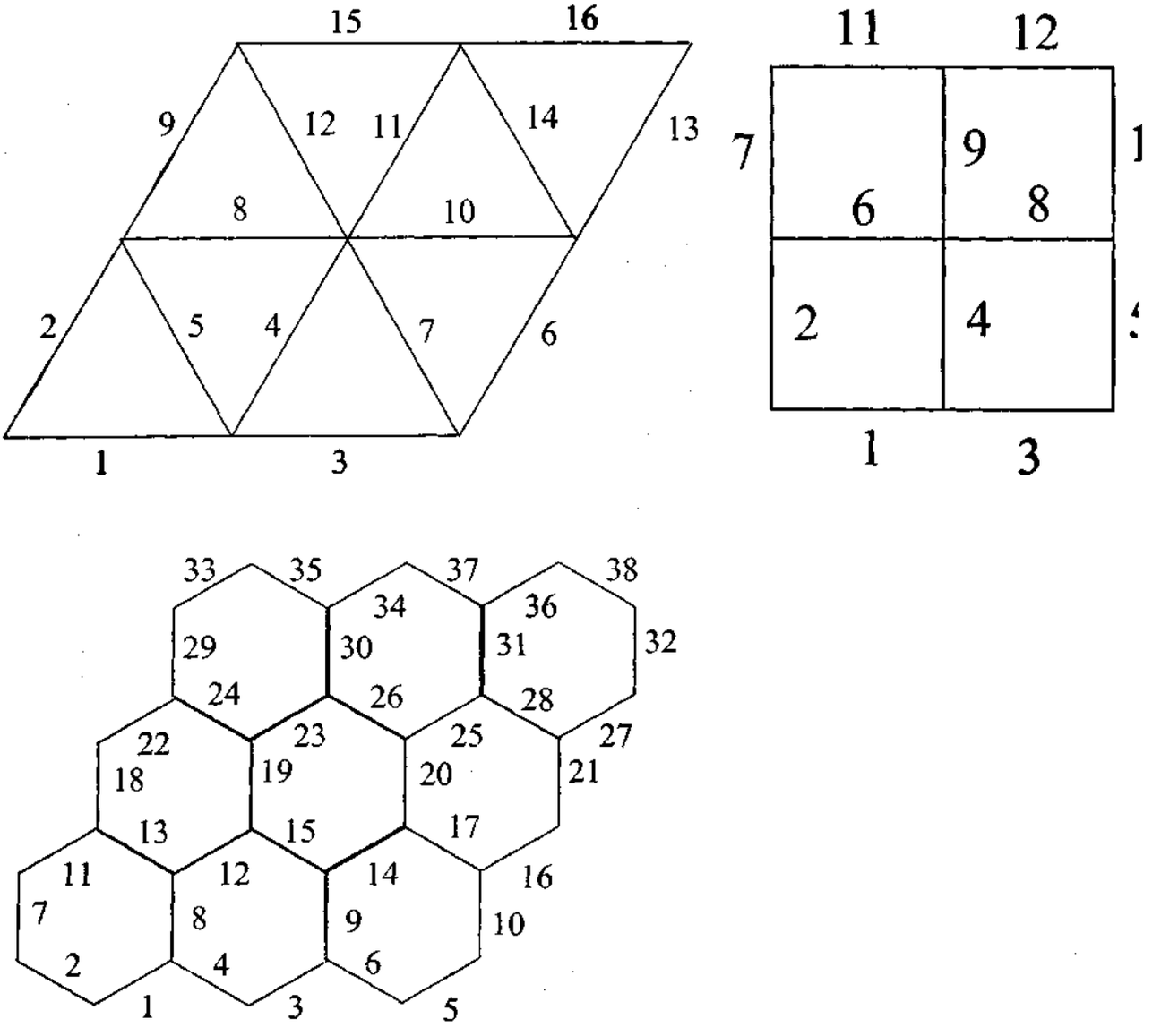
In the case of the honeycomb lattice, three orientations are possible. The orientation 1 corresponds to a bond which makes an angle of 60° with the horizontal; the second orientation labelled 2 is at 120° to the horizontal and the third orientation labelled 3 makes a right angle with the horizontal.

3.2.1 Labelling of bonds

The labelling of bonds in the different lattices refers to a coordinate pair of the origin site together with the bond orientation label. With the exception of the site labelled N^2-1 in the square and triangular lattices and all sites of labels from $2(N+1)^2-2-N$ to $2(N+1)^2-2$ in the honeycomb lattice, every site in the lattice is the origin of at least a single bond. This assertion is illustrated in Figures 3.2 and 3.3.

In general, all sites in the triangular lattices are origins of bonds with three orientations as defined above. In the square lattices, only 2 orientations are possible from any interior site. A honeycomb lattice presents some peculiarities: some sites are origins of orientations 1 and 2 but others are only origins of bonds with orientation 3. According to our algorithm, when an origin of a bond has a label greater than or equal to an odd multiple of $(N+1)$ and less than or equal to an even multiple of $(N+1)$ minus one, this site is the origin of a bond of orientation 3. Other sites are origins of bonds with

Figure 3.3: Diagrams showing labels of bonds for $N = 3$



orientations 1 and 2. In Table 3.4, we show some sites which constitute exceptions to the assertion given above concerning origins of bonds and their orientations.

Table 3.4: Bond orientation labels for exceptional cases

		Bond Orientations		
Label of the bond origin	Localization of the origin on the lattice sample	Triangular lattice	Square lattice	Honeycomb lattice
L	Bottom edge	1, 2, 3	1, 2	1, 2
L	Left edge	1, 2	1, 2	1 or 3[2]
L	Right edge	2, 3	2	1, 2 or 3[3]
L	Top edge	1	1	-
0	Bottom left corner	1, 2	1, 2	-
N^2-1	Top right corner for triangular and square lattices	-	-	1 or 3[4]

In the following section, we need to know the origin site of a bond and its end site. The end site of a bond is related to its orientation. In the triangular lattice, a bond with an origin labelled by L has an end site labelled by $L+1$, $L+N-1$ or $L+N$ depending on whether its bond orientation is 1, 2 or 3 respectively. A bond with an origin site

[2]The orientation is 1 when the site label L is an even multiple of $(N+1)$. Otherwise L is an odd multiple of $(N+1)$ and the bond orientation is 3.

[3]From the site of label L, there are bonds with orientations 1 and 2 if L is an odd multiple of $(N+1)$ minus one. But if L is an even multiple of $(N+1)$ minus one, the site labeled by L is origin of a bond with an orientation 3.

[4]For honeycomb lattice, the site labeled by N^2-1 is localized on the left side of the lattice sample. That site is the origin of bond with orientation 1 if N is an odd number else it is origin of a bond with orientation 3.

labelled by L in the square lattice has an end site labelled by $L+1$ or $L+N$ if the bond orientation is 1 or 2 respectively. Lastly a bond with an origin site labelled by L in the honeycomb lattice, has an end site labelled by $L+N+1$ for bonds of orientation 1 and 3, and $L+N$ for bonds of orientation 2.

3.2.2 Determination of the nearest neighbours of a bond

For a given bond, its nearest neighbours are all bonds that share origin sites or end sites (see Figure 3.3).

3.2.3 Labelling of clusters

When a bond chosen randomly is occupied in our algorithm, we say that its origin and end sites are also occupied, and we give these sites the same site label. Therefore, in bond percolation, two sites are connected if there is at least a path of occupied bonds between them.

If a bond is occupied, four situations can occur. Firstly, a new cluster of size one is formed for bond percolation. For computational convenience, we keep track of the corresponding origin sites and end sites which we occupy and label accordingly. The total number of occupied sites is increased by two and the number of occupied bonds increases by one. Secondly, before the occupation of the bond, we have the situation that either the origin site or the end site of the new bond is occupied and the other is

empty. Therefore, the total numbers of occupied bonds and sites are incremented by one. The new site label is the one of the first occupied site. Thirdly, both the origin site and the end site of the bond are occupied and have different site labels. The new bond establishes a bridge between the two clusters. The total number of occupied bonds is increased by one but the total number of occupied sites remains the same. The two clusters merge into one. The newly formed cluster takes the label of the progenitor cluster with larger size. The cluster relabelling is done for the smaller cluster by using a recursive subroutine. Lastly, the origin site and the end site of the new occupied bond are occupied and belong to the same cluster. This cluster has its size increased by one but the number of occupied sites remains the same.

3.2.4 Determination of spanning cluster

For bond percolation, the occupation probability is defined as the ratio between the number of occupied bonds and the total number of bonds on the lattice. The spanning cluster has the same meaning as in site percolation. The critical concentration p_c is determined when the spanning cluster appears for the first time. For all lattices considered, p_c for bond percolation is always less than its value for site percolation. In the next chapter we discuss the results obtained using our algorithms.

Chapter 4

Site Percolation and Bond Percolation - Numerical Results

In this chapter, we look at the numerical results obtained for the critical probability. We show how to deduce the critical probability for infinite lattices from data derived from finite systems by considering the lattice sizes or their linear dimension (N).

The algorithms (see Appendices) are written in Fortran 90 and are based on calls to recursive subroutines for the relabelling of the clusters. The Intel Fortran Compiler on a Linux Operating System and a 1.9 GHz AMD dual processor system are used in our computational scheme. The numerical values of the percolation threshold p_c are determined and compared to the known values of p_c for different lattices given in Table 4.1 for both site and bond percolation (Stauffer and Aharony, 1994).

The values for p_c depend on the lattice type. Because of finite size effects, the value of

p_c also depends on the size of the lattice and the number of iterations used to compute the average value. Within statistical errors, the numerical values for p_c constitute a good approximation to the exact values of p_c for the different lattices. Tables 4.2 to 4.4 give an idea of how the runtime per calculation changes depending on the lattice size. For example, in the case of bond percolation on the honeycomb lattice, when the size of the lattice is increased by a factor of nearly 100, the runtime is increased by a factor of 283040.

We apply a linear correlation and regression analysis (Goebner *et al.*, 2001) to extract results in the infinite system size limit. This is useful especially when we are averaging quantities; we assume that p_c is in linear relationship with the inverse of N or the inverse of the lattice size. According to our data, the lattice parameters $\frac{1}{N}$ and $\frac{1}{size}$ are determined easily and this makes simple the extrapolation for infinite lattices. In the analysis, our interest is focused on the correlation coefficient (r), the standard deviation (σ_m), the regression slope coefficient (a) and the intercept (b).

Two variables x and y are in linear relationship when they satisfy:

$$y = ax + b$$

where a is the slope and b the intercept.

y and x are called the dependent variable and the independent variable respectively. The correlation coefficient measures the strength of the linear relationship between the two variables x and y. Its values are in the range of -1 and +1. There is a perfect linearity when the value of r is ± 1. Poor linearity exists when the value of r is 0.

The regression slope coefficient is the average change in the dependent variable for a unit change in the independent variable. The standard deviation gives an idea of how averaged quantities are distributed around their mean value.

Mathematical expressions of the above quantities are shown below.

$r = \frac{\sum_{i=1}^{N}(x_i-\bar{x})\sum_{i=1}^{N}(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2 \sum_{i=1}^{N}(y_i-\bar{y})^2}}$ where N is the number of samples or observations.

$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$

$a = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2}$

$b = \bar{y} - a\bar{x}$

$\sigma^2 = \sum_{i=1}^{N} \frac{(x_i-\bar{x})^2}{N-1}$

$\sigma_m = \frac{\sigma}{\sqrt{N}}$

In Tables 4.2 to 4.4, p_{c_10} and p_{c_100} are respectively the numerical averages of the critical concentration taken over 10 and 100 iterations respectively. The standard deviation of these quantities are of order of 10^{-3}. Using either 10 or 100 calculations does not substantially affect the accuracy of the numerical values only making the standard deviation about 3 times smaller. For example, the numerical averages of p_{c_10} for the triangular lattice in site percolation are close to the theoretical value of p_c.

The general tendency is that as the size increases, the correspondence between the numerical and theoretical values of p_c becomes better. The only difficulty is that the runtimes increase dramatically with the system size. The runtimes given in Tables 4.2

to 4.4 are expressed in seconds(s) and correspond to a single iteration for a given lattice size. The determination of p_c for the honeycomb lattice in bond percolation for $N = 500$ corresponding to 100 iterations took more than 39 hours!

Table 4.1: Known values of p_c as published in literature (Stauffer and Aharony, 1994).

	Site Percolation	Bond Percolation
Triangular lattices	0.50000	0.34729
Square lattice	0.592746	0.50000
Honeycomb lattice	0.6962	0.65271

In Figures 4.1 and 4.2, we plot different values of p_c as a function of the inverse of N and lattice size respectively. The graphs describe site percolation for the triangular lattice with data given in Table 4.2 when the number of calculations is 100. Using linear regression, we extrapolate the numerical value of p_c for the infinite lattice which corresponds to the intercept of the graph. The correlation coefficients for Figures 4.1 and 4.2 are 0.98 and 0.97 respectively. The intercept is 0.5004 ± 0.004 in Figure 4.1 and it is 0.5028 ± 0.0045 in Figure 4.2. Table 4.5 gives the extrapolated values of p_c by considering its linearity with the inverse of the linear dimension (N) of the lattice and with the inverse of its size ($\propto N^2$ for infinite lattices). By looking at Table 4.5, we see that our calculations predict within statistical errors the values of p_c as given in Table 4.1. Evidently, we derive more accurate results by assuming linearity of p_c with $\frac{1}{N}$ rather than linearity of p_c with $\frac{1}{size}$.

Table 4.2: Numerical values of p_c determined on the triangular lattice

	Site Percolation				Bond Percolation			
N	Size	p_{c_10}	p_{c_100}	Runtime	Size	p_{c_10}	p_{c_100}	Runtime
50	2500	0.51672	0.51732	0.00s	7301	0.35202	0.35327	0.03s
100	10000	0.50080	0.50655	0.13s	29601	0.35413	0.35324	0.40s
150	22500	0.49924	0.50598	0.55s	68391	0.35298	0.35108	2.88s
200	40000	0.49901	0.50427	1.55s	119201	0.34796	0.35006	6.82s
250	62500	0.50477	0.50459	3.60s	186501	0.35385	0.35059	23.04s
300	90000	0.49965	0.50254	5.93s	268801	0.34930	0.34946	30.22s
350	122500	0.50220	0.50377	27.60s	366101	0.35000	0.34861	74.71s
400	160000	0.50340	0.50259	46.40s	478401	0.34869	0.34876	134.20s
450	202500	0.50115	0.50139	70.30s	605701	0.34803	0.34860	185.08s
500	250000	0.50075	0.50243	129.13s	748001	0.34840	0.34869	352.26s

Table 4.3: Numerical values of p_c determined on the square lattice

	Site Percolation				Bond Percolation			
N	Size	p_{c_10}	p_{c_100}	Runtime	Size	p_{c_10}	p_{c_100}	Runtime
50	2500	0.60776	0.60618	0.05s	4900	0.50255	0.50864	0.05s
100	10000	0.60270	0.59977	0.17s	19800	0.50421	0.50400	0.55s
150	22500	0.59665	0.59968	1.10s	44700	0.50192	0.50426	3.83s
200	40000	0.59775	0.59694	2.18s	79600	0.49862	0.50277	6.65s
250	62500	0.59703	0.59717	6.33s	124500	0.49927	0.50240	17.83s
300	90000	0.59414	0.59528	12.59s	179400	0.50238	0.50281	78.37s
350	122500	0.59171	0.59636	16.64s	244300	0.50117	0.50152	87.59s
400	160000	0.59512	0.59490	59.53s	319200	0.50266	0.50183	105.29s
450	202500	0.59436	0.59521	76.09s	404100	0.50176	0.50161	207.27s
500	250000	0.59220	0.59409	121.13s	499000	0.50170	0.50166	402.00s

Table 4.4: Numerical values of p_c determined on the honeycomb lattice

	Site Percolation				Bond Percolation			
N	Size	p_{c_10}	p_{c_100}	Runtime	Size	p_{c_10}	p_{c_100}	Runtime
50	5200	0.69952	0.69522	0.05s	7699	0.65580	0.66003	0.05s
100	20400	0.70014	0.69683	0.55s	30399	0.65827	0.65738	0.73s
150	45600	0.69594	0.69811	3.31s	68099	0.65751	0.65590	3.36s
200	80800	0.69903	0.69810	10.63s	120799	0.65628	0.65453	10.01s
250	126000	0.69942	0.69722	23.48s	188499	0.65249	0.65530	24.91s
300	181200	0.69662	0.69757	44.90s	271199	0.65321	0.65476	53.23s
350	246400	0.69847	0.69767	144.36s	368899	0.65508	0.65420	160.55s
400	321600	0.69567	0.69686	288.82s	481599	0.65474	0.65471	467.19s
450	406800	0.69720	0.69726	483.53s	609299	0.65426	0.65386	748.58s
500	502000	0.69680	0.69692	826.71s	751999	0.65270	0.65369	1414.52s

Table 4.5: Numerical values of p_c extrapolated to infinite lattices.

	Site percolation		Bond percolation	
	$p_c \propto 1/N$	$p_c \propto 1/size$	$p_c \propto 1/N$	$p_c \propto 1/size$
Triangular lattices	0.5004±0.0040	0.5028±0.0045	0.3486±0.0010	0.3496±0.001
Square lattices	0.5939±0.0030	0.5959±0.0035	0.5009±0.0020	0.5021±0.0000
Honeycomb lattice	0.6978±0.0000	0.6975±0.000	0.6534±0.0010	0.6545±0.0010

Figure 4.1: p_c versus $\frac{1}{N}$

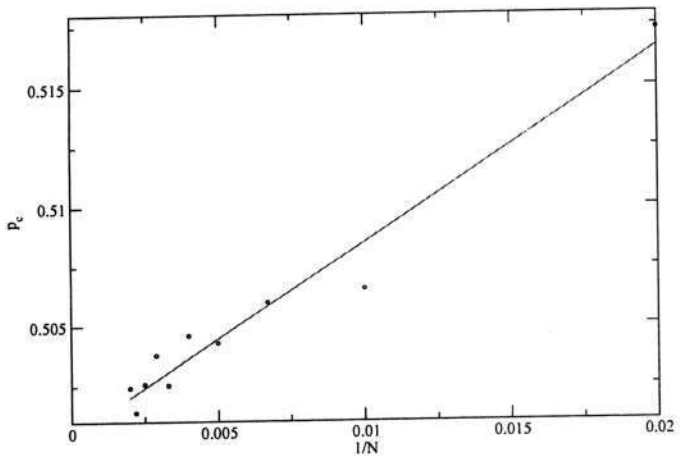
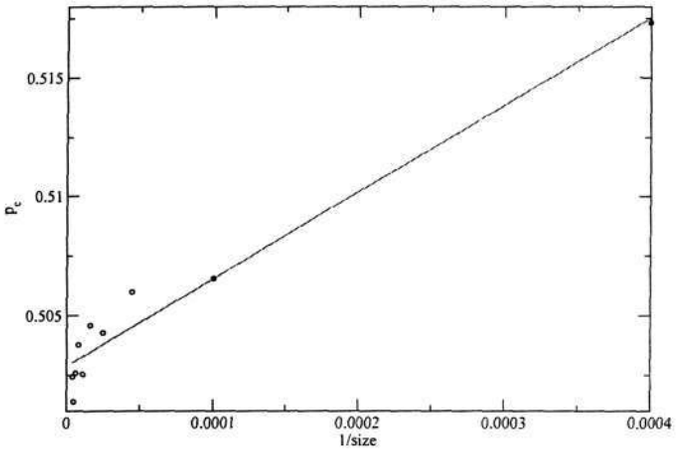


Figure 4.2: p_c versus $\frac{1}{size}$



Chapter 5

Cluster number scaling function $f(z)$ for lattices in 2 dimensions

In this chapter, an overview of the known results of the cluster number scaling function in one dimension and infinite dimensions (Bethe lattice) is discussed. The determination of the critical exponents in 2d related to $f(z)$ follows. Lastly, $f(z)$ is determined numerically when cluster sizes are binned and also when not binned.

5.1 Introduction

The change of a lattice state from having finite isolated clusters to having a spanning cluster (a cluster which has impinges on all edges in the case of finite lattices), is tantamount to a phase transition. In some cases in the literature, the percolation threshold is measured when a cluster crosses one directiononly, it means either vertical

or horizontal(Newman and Ziff, 2001). The order parameter is the probability p of a site or bond to be occupied. The transition appears at a critical concentration p_c. p_c plays the role of the critical point of a phase transition. Some properties of the system that are useful for describing the clusters on the lattice are affected by the transition. Amongst these properties of the percolation system are the total number of clusters M_0, the correlation length ξ, the probability that any occupied site or bond belongs to the percolating cluster P and the mean cluster size S. The correlation length, which is the root mean square distance between two sites belonging to the same finite cluster, reaches a maximum at p_c. It decays exponentially above and below p_c. ξ becomes infinite when a spanning cluster for an infinite lattice appears (Stauffer and Aharony, 1994).

S shows how large on average a cluster is when one points randomly to the clusters of the system. The critical behaviour observed in ξ at p_c is also observed in M_0, P and S. These quantities follow a power law behaviour near p_c with corresponding critical exponents. General results for these quantities have been given by Stauffer and Aharony (1994):

$$M_0 \propto |p - p_c|^{2-\alpha}$$
$$P \propto |p - p_c|^{\beta}$$
$$S \propto |p - p_c|^{-\gamma}$$

These authors presented discussions on the number of clusters of size s per lattice site at occupation probability p, $n_s(p)$ in a one-dimensional lattice and a Bethe lattice (ef-

fectively infinite dimensions). Exact solutions of $n_s(p)$ in one dimension and infinite dimensions suggest a general functional form of $n_s(p)$ in d dimensions:

$$n_s(p) = s^{-\tau} f(z), \text{ where}$$
$$z = (p - p_c)s^{\sigma}$$

$f(z)$ is called the cluster number scaling function. This is valid for large s and p near the critical concentration p_c.

The exact results for this function in a one dimensional lattice and the Bethe lattice are summarized in Table 5.1.

The exponents α, β, γ, σ and τ are universal and depend on the dimensionality of the

Table 5.1: Values of p_c and critical exponents in one dimension and the Bethe lattice related to $f(z)$

	p_c	α	β	γ	σ	τ
One dimension	1	0	0	1	1	2
Bethe lattice	$\frac{1}{z-1}$	-1	1	1	1/2	5/2

percolation system. There are relationships amongst these exponents (Weiss, 1994):

$$\tau = 2 + \frac{\beta}{\beta+\gamma}$$
$$\sigma = \frac{1}{\beta+\gamma}$$
$$2 - \alpha = 2\beta + \gamma$$

These exponents describe geometrical properties related to the percolation transition. The main object of this thesis is to verify numerically that $n_s(p)$ has the predicted form in two dimensions by considering the honeycomb, square and triangular lattices for both site and bond percolation – thus showing the universality of $f(z)$. In the literature [Nakanishi and Stanley (1980), Stauffer and Aharony (1994), Tiggemann (2001)] $n_s(p)$ was studied by using a procedure of combining the cluster sizes in groups of bins. In this work, $f(z)$ is analyzed when s is set in bins and results are compared when s is not set in bins. This is a quite stringent test of universality of $f(z)$. From the results of $n_s(p)$, the numerical values of the exponents σ and τ are deduced and listed in Table 5.4.

5.2 Discussion of numerical data relating to $n_s(p)$ in 2 dimensions

In the present work, all data related to $n_s(p)$ for all values p and s are stored. In the determination of $f(z)$, results are compared when s is set in bins or not. Bins are defined as follows: the k^{th} bin comprises all sizes s such as

$$2^{k-1} \leq s \leq 2^k - 1.$$

The size of the k^{th} bin is assumed to be the geometrical average of the its extreme values: $\frac{2^k}{\sqrt{2}}$ which is exact if $\tau = 2$.

Data relating to $n_s(p)$ demand a high processing speed and a large amount of memory

for storage. Consequently the lattices that we were able to study are limited to relatively small sizes. In Tables 5.2 and 5.3 we give an indication of the resources needed for our computations. The size of the system means the total number of sites or bonds in the lattice (see Chapter 3 (Figure 3.1)). The file size (expressed in megabytes(M) or in gigabytes(G)) refers to the data files used to store different distinct clusters, their sizes and their distribution. Data given in Table 5.2 correspond to $N = 100$ and for 100 calculations but data in Table 5.3 correspond to just a single calculation for $N = 500$. The runtime (expressed in seconds(s)) is the time taken for the calculations.

Table 5.2: Dependence of the computational runtime on the lattice sizes for $N = 100$

	Site percolation			Bond percolation		
	Size	File size	Runtime	Size	File size	Runtime
Triangular lattice	10000	240M	103.00s	29601	712M	576.25s
Square lattice	10000	278M	111.50s	19800	597M	372.63s
Honeycomb lattice	20400	799M	463.00s	30399	1.3G	895.00s

5.2.1 Determination of the numerical value of τ

By assuming in 2d that $n_s(p)$ follows the functional form,

$$n_s(p) = s^{-\tau} f((p - p_c)s^{\sigma}), \tag{5.1}$$

Table 5.3: Dependence of the computational runtime on the lattice sizes for $N = 500$

	Site percolation			bond Site percolation		
	Size	File size	Runtime	Size	File size	Runtime
Triangular lattice	250000	173M	1192.88s	748001	515M	11575.25s
Square lattice	250000	200M	1264.13s	499000	424M	7012.50s
Honeycomb lattice	502000	533M	5681.38s	751999	889M	11927.50s

we can calculate the value of τ by setting $p \approx p_c$:

$$n_s(p_c) = s^{-\tau} f(0) \tag{5.2}$$

$$\Rightarrow \log(n_s(p_c)) = \log(f(0)) - \tau \log(s) \tag{5.3}$$

Theoretically, the graph of $\log(n_s(p_c))$ versus $\log(s)$ (Equation 5.3) is a straight line with slope $-\tau$ and intercept $\log(f(0))$. In Figures 5.1 to 5.2, a set of graphs of that kind are plotted when s is set in bins. Otherwise, Figures 5.3 and 5.4 are related to graphs of $\log(n_s(p_c))$ versus $\log(s)$ when s is not binned. Data plotted correspond to $N = 100$ averaged over 100 calculations. The different numerical values of p_c obtained during our iterations are used. In Table 5.4 we list the numerical values of τ evaluated in this manner. It is important to mention that by fitting our points to a best straight line, we only consider the intermediate values of s to minimize the finite size effects. For example, the numerical value of τ relating to site percolation for the triangular lattice is obtained by ignoring the three first and the last three points in Figure 5.1. Within

the limits of finite sizes, the universality of τ for all lattices for both site and bond percolation is demonstrated.

Furthermore, the numerical values of $\ln(f(0))$ converge to the same value. Within the limits of the numerical simulation, $f(0)$ displays a constant value for our three types of lattices.

5.2.2 Determination of the numerical value of σ

In the data relating to $n_s(p)$, by varying s from 1 to the total number of sites or bonds, there is only one value of p for which $n_s(p)$ has a maximal value (See graphs of $f(z)$ - Figures 5.9 to 5.12). This means that $f(z)$ is a constant for that particular value of $p = p_{max}$. Let z_{max} be the value of z corresponding to p_{max}, then

$$z_{max} = (p_{max} - p_c)s^{\sigma} = C, \quad C \text{ is a constant}$$

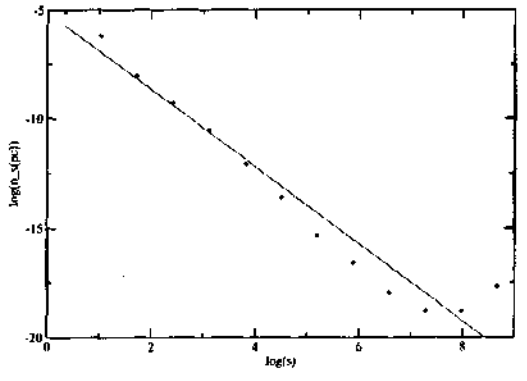
$$|p_c - p_{max}| \propto s^{-\sigma}$$

In principle, the graph of $\log|p_c - p_{max}|$ versus $\log(s)$ is a straight line where $-\sigma$ is the slope. A set of graphs in Figures 5.5 to 5.8 is plotted in order to determine the numerical value of σ by considering the case where p_c is greater than p_{max}. In Table 5.4, we list the numerical values of σ evaluated in this manner. The procedure used to determine the numerical value of τ is also applied in the determination of σ. Within the limits of finite sizes, the universality of σ for all lattices for both site and bond percolation is relatively demonstrated.

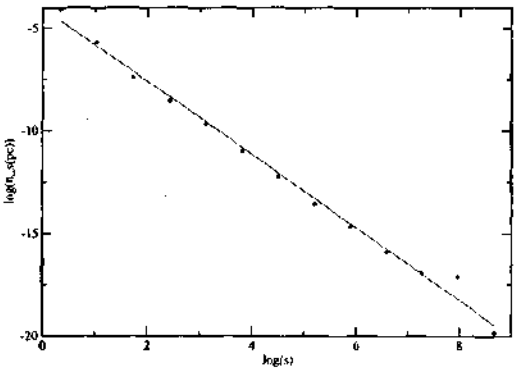
Figure 5.1: $\log(n_s(p_c))$ versus $\log(s)$: Site percolation, s is set in bins

$N = 100$

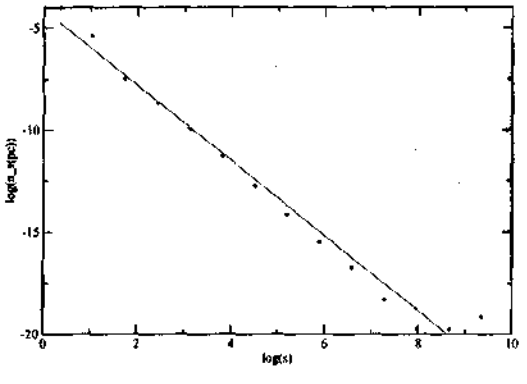
Triangular lattice



Square lattice



Honeycomb lattice



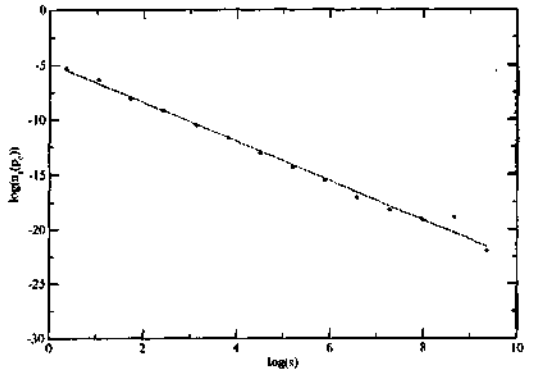
5.2.3 Numerical determination of the cluster number scaling function $f(z)$

In this subsection, we plot and discuss the graphs relating to the lattices for both site and bond percolation and involving the behaviour of $f(z)$. In Figures 5.9 to 5.12, $\frac{n_s(p)}{n_s(p_c)}$

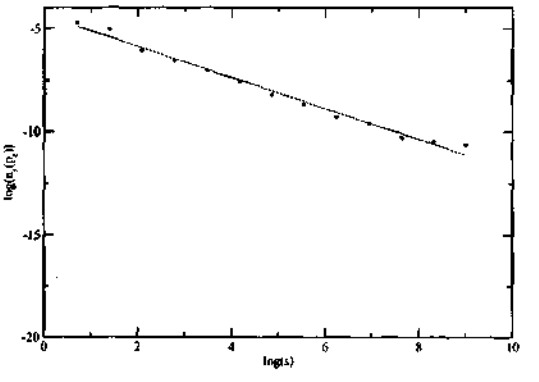
Figure 5.2: $\log(n_s(p_c))$ versus $\log(s)$: Bond percolation, s is set in bins

$N = 100$

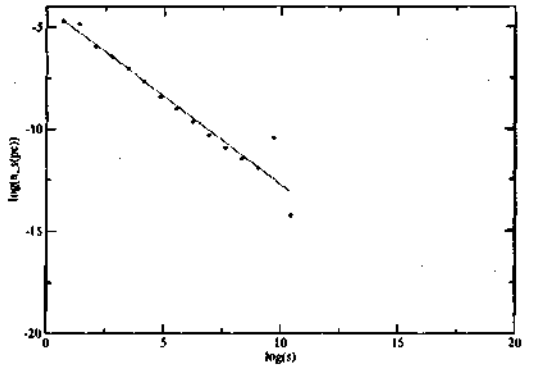
Triangular lattice



Square lattice



Honeycomb lattice

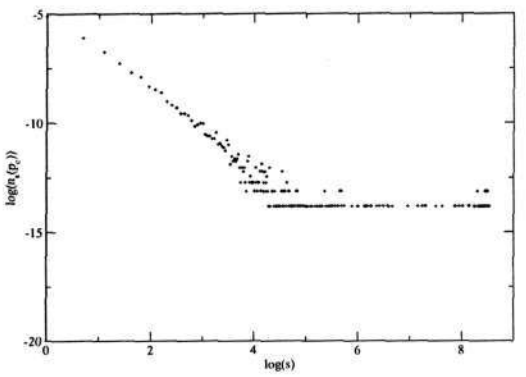


is plotted as a function of z where $z = (p - p_c)$. Theoretically, this procedure gives the function $\frac{f(z)}{f(0)}$. Theoretical values of σ and p_c are used in this case. σ takes the value of $\frac{36}{91}$ and the theoretical values of p_c used are listed in Table 4.1 (Stauffer and Aharony,

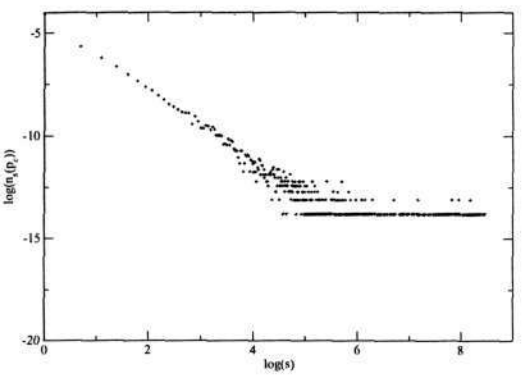
Figure 5.3: $\log(n_s(p_c))$ versus $\log(s)$: Site percolation, s is not set in bins

$N = 100$

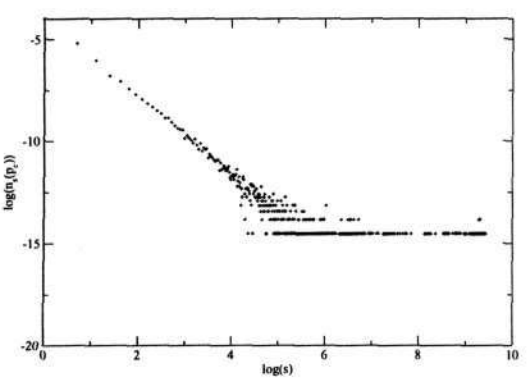
Triangular lattice



Square lattice



Honeycomb lattice



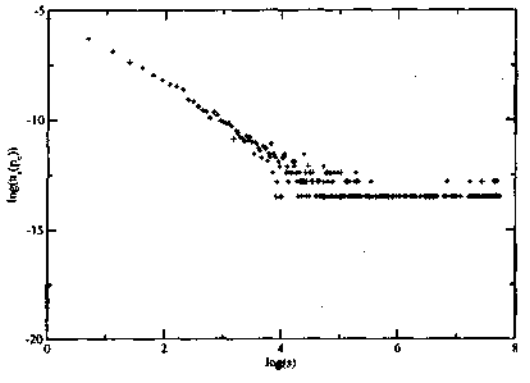
1994).

The expression $n_s(p) = s^{-\tau} f((p - p_c)s^{\sigma})$ is assumed to be valid for large values of s.

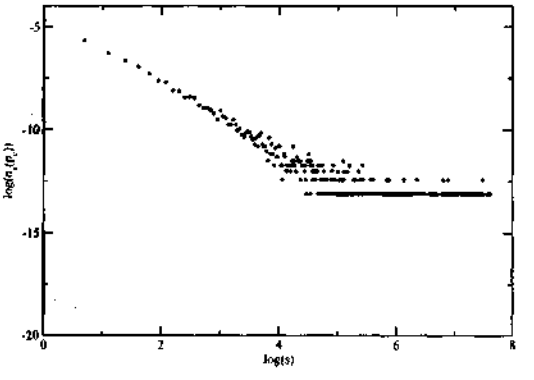
Figure 5.4: $\log(n_s(p_c))$ versus $\log(s)$: Bond percolation, s is not in bins

$N = 50$

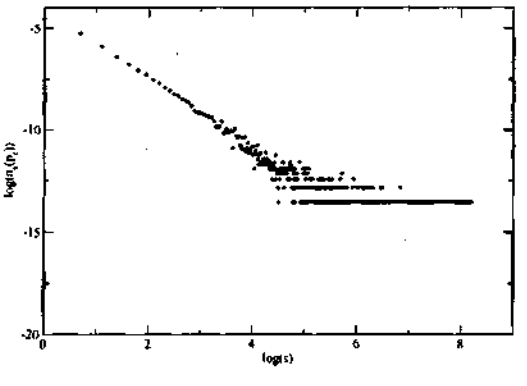
Triangular lattice



Square lattice



Honeycomb lattice

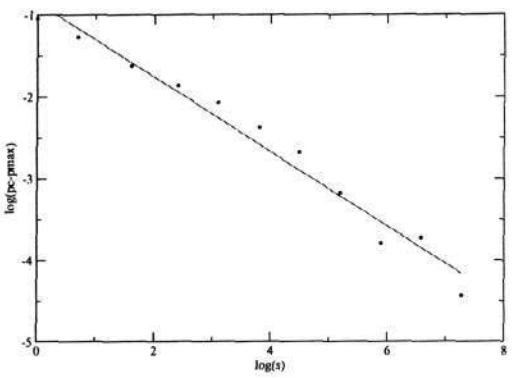


For finite lattices, it is not accurate to include large values of s because of edge effects. That is why the determination of $f(z)$ is limited to values of s in the middle range. This was also applied in the determination of τ and σ as well.

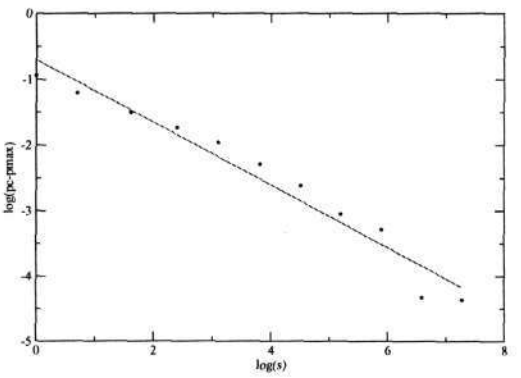
Figure 5.5: $\log(p_c - p_{max})$ versus $\log(s)$: Site percolation, s is set in bins

$N = 100$

Triangular lattice



Square lattice



Honeycomb lattice

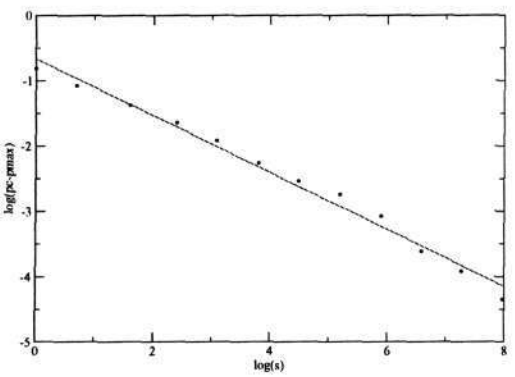
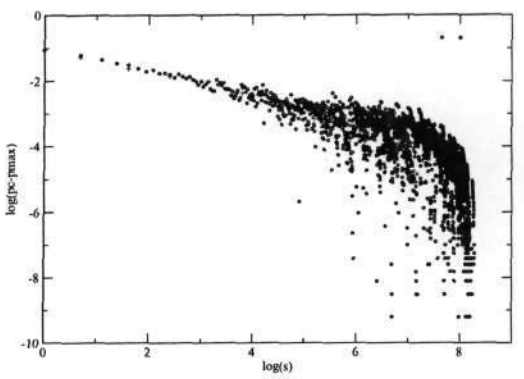


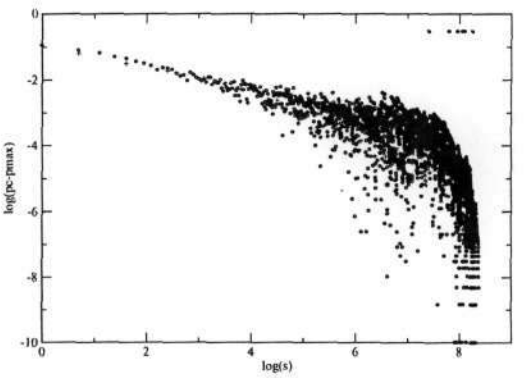
Figure 5.6: $\log(p_c - p_{max})$ versus $\log(s)$: Site percolation, s is not in bins

$N = 100$

Triangular lattice



Square lattice



Honeycomb lattice

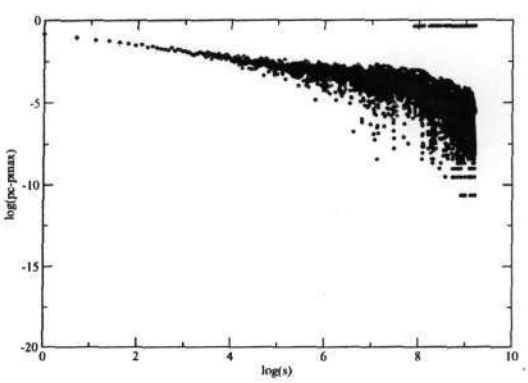
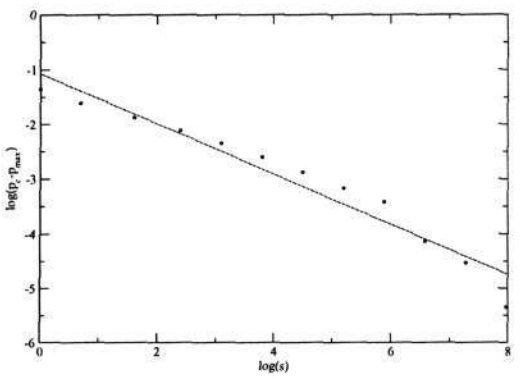


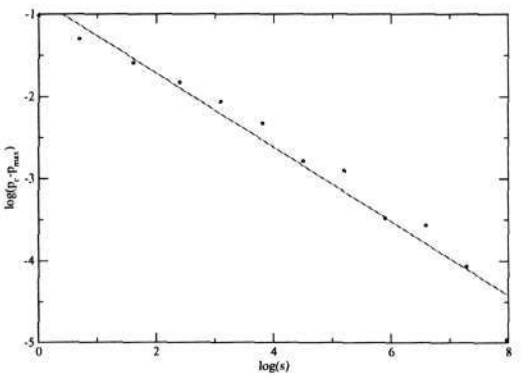
Figure 5.7: $\log(p_c - p_{max})$ versus $\log(s)$: Bond percolation, s is in bins

$N = 100$

Triangular lattice



Square lattice



Honeycomb lattice

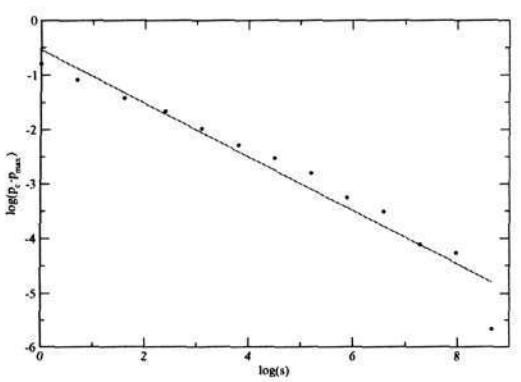
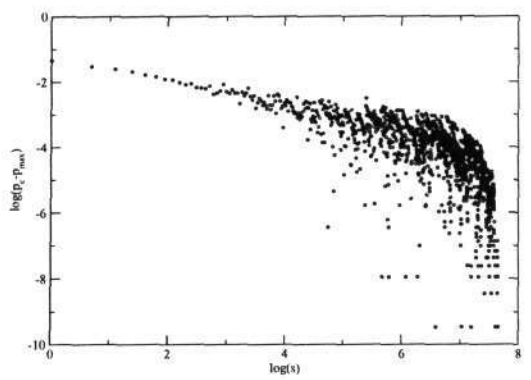


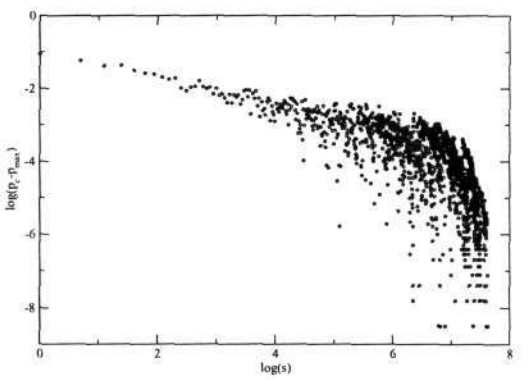
Figure 5.8: $\log(p_c - p_{max})$ versus $\log(s)$: Bond percolation, s is not in bins

$N = 50$

Triangular lattice



Square lattice



Honeycomb lattice

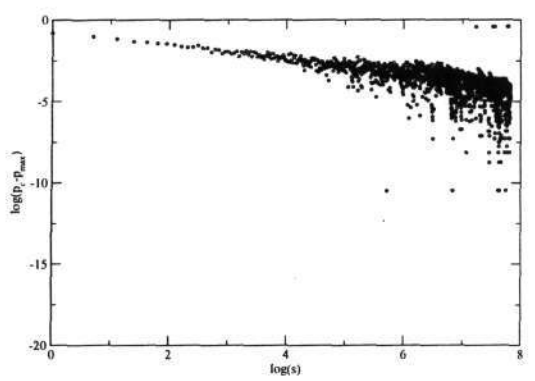


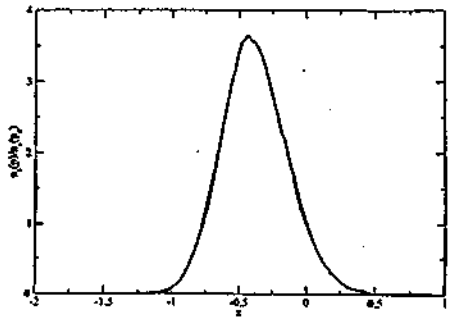
Table 5.4: Numerical values of τ, $\ln(f(0))$ and σ determined for $N = 100$ when cluster sizes are set in bins

	Site percolation			Bond percolation		
	τ	$\ln f(0)$	σ	τ	$\ln f(0)$	σ
Triangular lattice	2.13±0.03	-4.01±0.17	0.46±0.04	1.87±0.01	-3.87±0.08	0.45±0.00
Square lattice	1.78±0.04	-4.03±0.23	0.43±0.08	1.84±0.040	-4.38 ±0.013	0.39±0.02
Honeycomb lattice	1.95±0.01	-4.13±0.01	0.41±0.01	2.00±0.02	-4.07±0.04	0.41±0.02

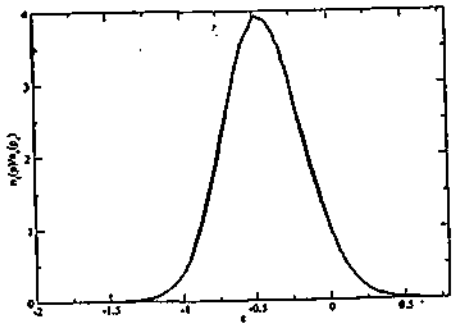
Figure 5.9: $\frac{n_s(p)}{n_s(p_c)}$ versus $(p-p_c)s^{\sigma}$: Site percolation, s in bin 5

N=100

Triangular lattice



Square lattice



Honeycomb lattice

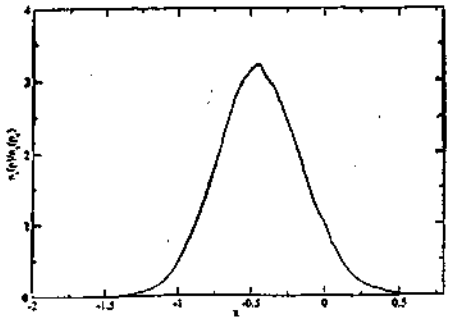
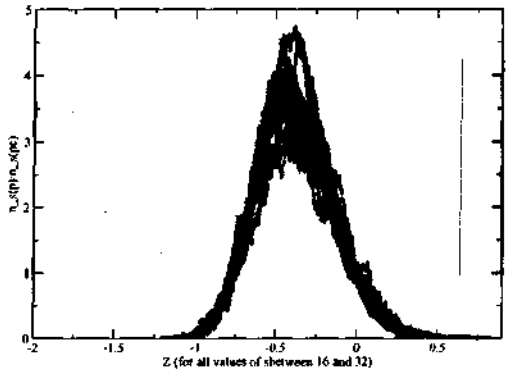


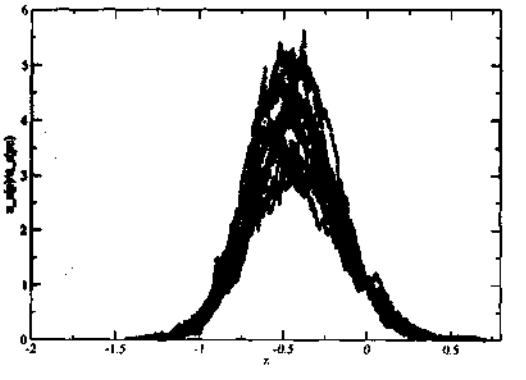
Figure 5.10: $\frac{n_s(p)}{n_s(p_c)}$ versus $(p - p_c)s^\sigma$: Site percolation with values of s from 16 to 31

$N = 100$

Triangular lattice



Square lattice



Honeycomb lattice

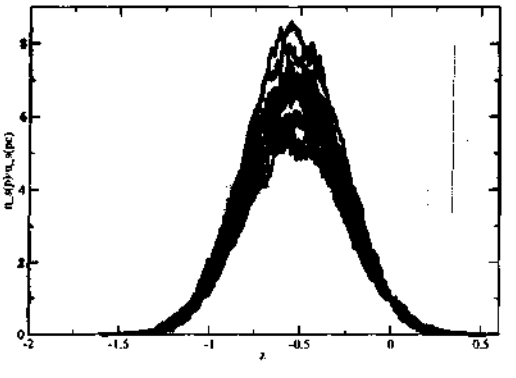
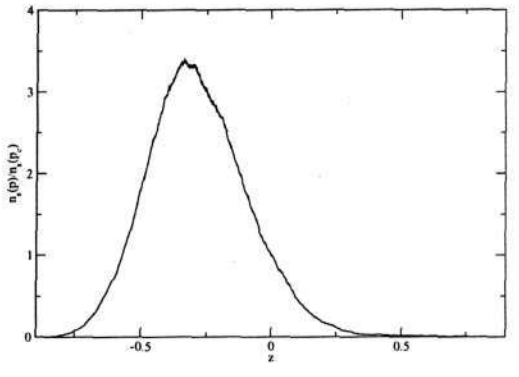


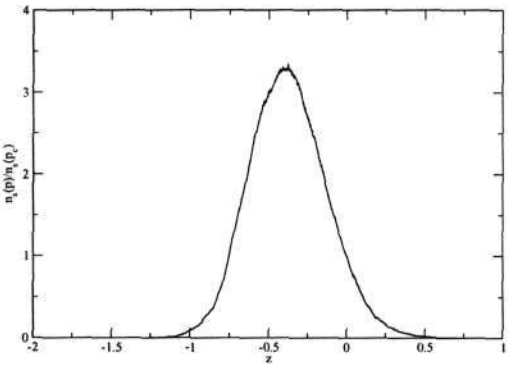
Figure 5.11: $\frac{n_s(p)}{n_s(p_c)}$ versus $(p - p_c)s^{\sigma}$: Bond percolation, s in bin 5

$N = 50$

Triangular lattice



Square lattice



Honeycomb lattice

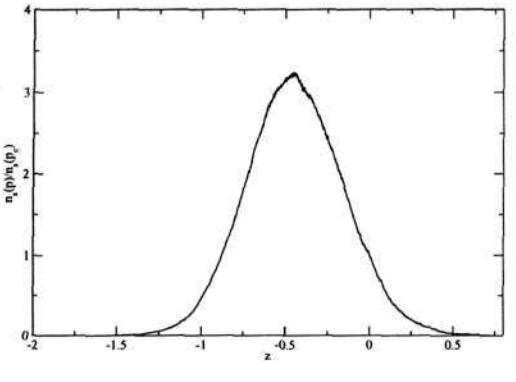
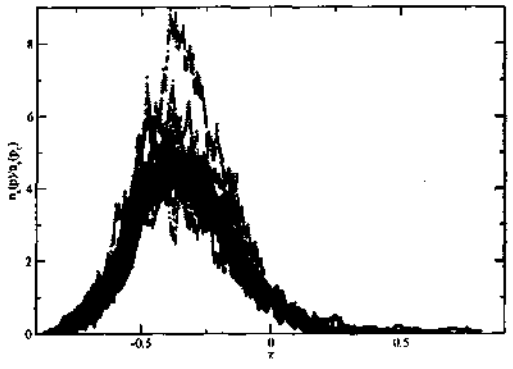


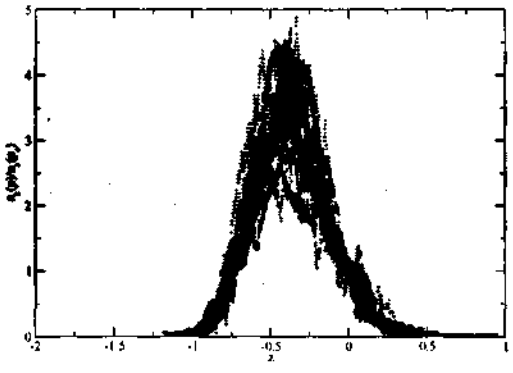
Figure 5.12: $\frac{n_s(p)}{n_s(p_c)}$ versus $(p - p_c)s^\sigma$: Bond percolation, with values of s from 16 to 31

$N = 50$

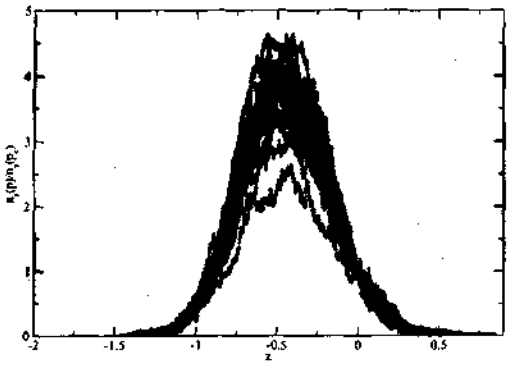
Triangular lattice



Square lattice



Honeycomb lattice



Conclusions

In analyzing our results, it is important to note that they are affected by statistical errors, finite size effects and systematic errors (Press *et al.*, 1992).

Statistical errors are due to the randomness of the Monte Carlo methods which induce deviations from the statistically most probable values. These errors are distributed normally. It is possible to diminish their impact by doing an even larger number of calculations and then averaging over the results. In Table 4.4, we note that the larger the lattice is, the smaller are the fluctuations from the theoretical values. We conclude that it is better to do a smaller number of calculations on larger lattices than a large number of calculations on smaller lattices.

Finite size errors are simply due to the fact that all computers have finite memory and processor speed. We only have the means of modelling finite system sizes. To deal with finite size effects, it is practical to study finite systems of different sizes and then extrapolate to the infinite system limit.

It is very difficult to deal with systematic errors. In the study of the percolation theory,

the general source of systematic errors is related to the use of the so called random numbers. The distribution of numbers is not truly random - the numbers may therefore be deemed pseudo-random. The control of systematic errors is not easy because their detection is also hard. It is a challenge to analyze the real impact of different sources of errors when testing theoretical assumptions by means of numerical methods.

The values of p_c for different lattices constitute a good approximation of the known values to within 10^{-3} (Table 4.1).

In Table 5.4, we list the numerical values of τ and σ by using the intermediate values of s thus taking care of the finite size effects. Some of our numerical values are in good agreement with the known values of τ and σ: $\frac{187}{91}$ and $\frac{36}{91}$ respectively (Stauffer and Aharony, 1994).

For a fixed value of s, $f(z)$ has a maximum value for a value of p less than p_c. This fact is sometimes masked by the finite size effects. For the values of p close to 0 and 1, $f(z)$ vanishes. In our simulation, we use the theoretical values of p_c for plotting $f(z)$.

By considering the graphs relating to $\frac{n_s(p)}{n_s(p_c)}$, we see that the maximum value is approximately 3.5 especially when s is set in bins for both site and bond percolation (Figures 5.9 and 5.11). In limits of the numerical determination, $\frac{f(z)}{f(o)}$ displays a universal behaviour.

For the graphs relating to $\frac{n_s(p)}{n_s(p_c)}$ when s is not set in bins (Figures 5.10 and 5.12), there are fluctuations about the maximum value. This is due to the fact that we plot the function $\frac{f(z)}{f(0)}$ in a range of values of s and related to 100 calculations.

References

Aharony, A. and Stauffer, D, *J. Phys. A* **30**, L301 (1997).

Cardy, *J.Phys. A* **25** L201(1992).

Chabot, N, *Le journal de maths des élèves*, Volume 1, No2, pp26-29 (1994).

Deutscher, G., Zallen, R. & Adler, J., Percolation structures and process, Annals of the Israel Physical Society, 504pp. (1983).

DeVries, P.L., A first course in Computational Physics, John Willey&Sons, New York, pp29-32. (1994).

Dictionary of effects and phenomena in physics, VCH Publishers, New York (1987).

Dictionary of Physics, Penguin Reference Books, second edition (1991).

Dictionary of science and technology, Academic Press (1992).

Ellis, T.M.R., Philips, I.R. and Lahey, T.M., Fortran 90 Programming, Addison-Wesley, USA 825pp (1994).

Encyclopedia of Applied Physics, Volume 2 & Volume 6, (1994).

Encyclopedia of Physics, 2nd edition, VCH Publishers, New York (1991).

Giordano, N.J., Computational Physics, Pentice Hall, pp157-201 (1997).

Grimmett, G., Percolation, 2nd edition, Springer (1999).

Groebner, D.F., Shannon, P.W., Fry, P.C. and Smith, K.D., Business: A Decision-Making Appr
5^{th} edition, Pentice Hall, New Jersy pp379-468 (2001).

Hahn, B.D., Fortran 90 for Scientists and Engineers, Arnold, London, 351pp (1998).

Hermann, H.J., Statistical Models for the Fracture of Disordered Materials, Amsterdam (1990).

Landis, G.A., *Journal of The British Interplanetary Society*, London, Vol.51 pp163-166 (1998).

Leath, P.L, *Phys. Rev. B* **14** pp5046-5056 (1976).

Lyons, L., A practical guide to data analysis for physical science students, Cambridge University Press (1992).

Margolina, A., Nakanishini, H., Stauffer, D., and Stanley, H.E., *J.Phys.A* **17** pp1683-1701 (1984).

Nakanishini, H., *J.Phys.A* **20** pp6075-6083 (1987).

Nakanishini, H., and Stanley, H.E., *Phys. Rev. B* **22**, pp2466 . (1980).

Newman, M.E.J., and Ziff, R.M., *Phys. Rev. E* **64** pp16706_1-16706_16 (2001).

Pang, T., An Introduction to Computational Physics, Cambridge University Press, pp51-52 (1997).

Press, W.H. et al., Numerical Recipes in Fortran: The Art of Scientific Computing, 2d edition, Cambridge, pp652-653 (1992).

Quintanilla, J., Torquato, S. and Ziff, R.M., *J. Phys. A: Gen.* **33**, L399-L407 (2000).

Ross, S., Stochastic Processes, second edition, John Willey & Sons, Inc. 551pp (1996).

Sahimi, M., Applications of percolation theory, Taylor & Francis, 258pp (1994).

Stauffer, D. and Aharony, A., Introduction to the percolation theory, revised 2nd edition, Taylor & Francis, 181pp (1994).

Tiggemann, D., Simulation of Percolation on Parallel Computers, Diploma thesis at the Institute for Theoretical Physics, University of Cologne (2001).

Weiss, G.H., Contemporary Problems in Statistical Physics, Siam, Philadelphia, pp103-146 (1994).

Glossary

Correlation length

The correlation length gives a measure of the typical length scale over which fluctuations of one microscopic variable are correlated with the fluctuations of another. In percolation theory, it is the typical cluster diameter of the clusters which gives the main contribution to the divergence of the second (and higher) moments of the cluster distribution.

Critical concentration

In site or bond percolation on an infinite lattice there is a critical value of the occupation probability p_c below which there is no spanning cluster and above which there is always one spanning cluster. The quantity p_c is also called the critical concentration.

Critical exponents

Around a critical point, physical quantities are often expressed as a power law, such as the distance from the critical point. The power is named a critical exponent or index. Thus, for percolation theory, the total number of clusters, the probability that a site or bond belongs to the spanning cluster and the mean cluster size displays critical behaviour at the critical concentration.

Critical phenomenon

This phenomenon occurs in the neighbourhood of a continuous phase transition. It is a drastic change due to small variations of one or more parameters.

Critical point

This is a point in a phase diagram, where a correction length associated with a physical system is infinite.

Dimension, Dimensionality

The word "dimension" is known through its conventional use. A line is a one dimensional object and a square is two-dimensional, a sphere is three-dimensional. How can we define exactly the dimensionality of an object? A rough definition is given by considering two similar objects. They are similar in the sense that they have exactly the same shape but different sizes (eg. a small and a large object). The dimensionality of the object is the power to which the ratio of corresponding lengths has to be raised to give the ratio of the sizes of the object. So the size of the line is its length, that of a disc is its area and that of a sphere is its volume. Corresponding lengths are, for example, the circumferences of two circles, or their radii, but not the radius of one and the circumference of another. Integer dimensionalities are called "Euclidean", non-integer dimensionalities according to the above definition are called "fractal".

Fractals

A fractal is an object of non-integer dimensionality. This means that the mass of such an object does not increase with its length raised to the Euclidean dimensions of the space it occupies. The mass of a plane fractal, one that lies on a surface, does not increase with its length squared, but with its length raised to some other power less than two. Surprisingly, many structures arising in nature are of fractal dimensionality.

Order parameter

It is a variable such as the strength of the infinite cluster in percolation theory used to describe the degree of order in a phase above or below its critical point.

Percolation theory

Consider an array whose sites can be occupied or empty. This array can be one, two or multi-dimensional. Sites are occupied randomly with a certain probability. Neighbouring occupied sites form a cluster. Percolation theory deals with properties of clusters of occupied sites in lattices. The word "percolation" is related to the possibility of having a cluster which spans the whole lattice: a cluster that percolates all directions from one side to the other.

Phase transition

It is a change of state such as occurs in the boiling or freezing of water, or in a change between ferromagnetic and paramagnetic states of a solid magnet. An abrupt change characterized by a jump in an order parameter is observed.

Power law

A power law establishes a relation of one quantity to another through exponentiation with a constant exponent. The exponents can be integers or non-integers. Power laws are abundant in nature. The origin of their omni-presence is unclear and may be thought of as the main subject of research into complex systems. Power laws are closely related to the concept of fractals.

Random process

It is a process which describes a quantity or outcome that cannot be predicted beforehand but only expressed as a probability.

Universality

Microscopically different physical systems exhibit critical point behavior with quantitatively identical features like critical exponents. Such exponents are said to be universal. In percolation theory, critical exponents do not depend on the structural details of a lattice but on the Euclidean dimension d.

Appendices

A.1 Random number generator codes

module RandomPhys

!!module RandomPhys

!! module that contains a linear congruential random number generator

!! use kinds

implicit none

integer, parameter :: i64=**selected_int_kind**(18)

integer, parameter :: r64=**selected_real_kind**(15)

integer (**kind**=i64), private :: I_seed=1_i64

integer (**kind**=i64), private, **dimension**(1:97) :: RLIST

integer (**kind**=i64), private, **parameter** :: a = 16807

integer (**kind**=i64), private, **parameter** :: c = 0

integer (**kind**=i64), private, **parameter** :: m = 2147483647

contains

function Random()

!function Random()

```
! returns a 64 bit pheudo random integer
! generated by: I_n+1 = (a*I_n + c)mod(m)
! where a, c and m are global constants in this module
implicit none
      integer (kind=r64) :: Random, tmp
      tmp = a*I_seed + c
      Random = mod(tmp,m)
      I_seed = Random
return
end function Random


function RandomNumber()
!!function RandomNumber()
!! returns a 64 bit pheudo random real number
! uses the function Random() to pick a random
! integer in the random list RLIST
! returns (integer/m) as a real
! replace the integer with a new one given by Random()
implicit none
      integer :: index
      real (kind=r64) :: RandomNumber
      index = 1+int(ubound(RLIST,1)*(real(Random())/real(m)))
      RandomNumber = real(RLIST(index))/real(m)
```

```
        RLIST(index) = Random()
    return
    end function RandomNumber


    subroutine RandomSetSeed(seed)
    !!subroutine RandomSetSeed(seed)
    !! takes a 64 bit integer - seed
    !! sets the initial random seed to seed
    ! sets up a list of random integers RLIST
    implicit none
        integer (kind=i64), intent(in) :: seed
        integer :: i
        I_seed=abs(seed)          if (I_seed==0) I_seed=174392
          do i=1, ubound(RLIST,1)
            RLIST(i) = Random()
          end do
      return
    end subroutine RandomSetSeed


    function RandomGetSeed()
    !!function RandomGetSeed()
    !! returns a 64 bit integer representing
    !! the current value of the random seed
```

```
implicit none
        integer (kind=i64) :: RandomGetSeed
        RandomGetSeed = I_seed
return
end function RandomGetSeed


end module RandomPhys
```

A.2 Site percolation codes: Triangular lattice.

```
program site_main
use site_module
implicit none
        integer,parameter::ntot=500,tot_cal=100
        integer::i,sum
        integer::jc,n
        real::runtime(2) open(unit=20,file='output.dat')
open(unit=21,file='process.dat')
          do n=50,ntot,50
             call cpu_time(runtime(1))
             call randomsetseed(123_i64)
             sum=0
```

```
            do i=1,tot_cal
               call site_percolation(n,jc)
               sum=sum+jc
            end do
          write(20,*)1./real(N**2),real(sum)/real(tot_cal*N**2)
           write(21,*) 'Average value of pc = ',real(sum)/real(tot_cal*N**2)
          call cpu_time(runtime(2))
          write(21,*)'Number of calculations = ',tot_cal
          write(21,*) 'size and cpu time (seconds) ', n, runtime(2)-runtime(1)
        end do
end program site_main

module site_module
use randomphys
implicit none
contains

subroutine site_percolation(N,jc)
implicit none
     integer, parameter::i64=selected_int_kind(18)
     integer,intent(in)::N
     integer::M(0:N**2-1), site(0:N**2-1)
     integer,dimension(6)::P
```

```
      integer::cluster,nn,i,j,k,q,r,temp
      integer,intent(inout)::jc
      integer::number_sites(0:N**2)
      integer::max_number_sites,max_M,occupied_neighbours
      integer::total_number_sites
      integer::old_label,percolation
      real(kind=r64)::rd
! total_number_sites ..... N*N = total number of sites
! M ..... cluster label of each site
! site ..... the 1 dimensional label of the sites
! cluster ..... increments by 1 whenever a new cluster is created
! nn ..... number of nearest neighbours for the present site
! P ..... site label for the nn nearest neighbours
! jc ..... jc/N**2 is the percolation threshold
! occupied_neighbours ..... number of occupied neighbouring sites for a particular site
! max_number_sites..... size of the largest neighbouring cluster
! max_M ..... cluster label of the largest neighbouring cluster
! percolation ..... =0 when no spanning cluster, =1 when spanning cluster exists
! number_of_clusters ..... total number of clusters for a particular size s for particular p
! distinct_sizes ..... total number of distinct cluster sizes for a particular p
      total_number_sites=N*N
      jc=0
      M=0
```

```
      cluster=0
      percolation=0
      number_sites=0
! labelling the sites
   do i=0,total_number_sites-1
      site(i)=i
   end do
! percolation
   do j=0,total_number_sites-1
      rd = randomnumber()
      r= int(Rd*(N**2-j)+j)
      q=site(r)
      call find_nn(q,N,P,nn)
      max_number_sites=-1
      occupied_neighbours=0
        do k=1,nn
          if (M(P(k)).gt.0) then
            occupied_neighbours=occupied_neighbours+1
              if(number_sites(M(P(k))).gt.max_number_sites) then
                max_number_sites=number_sites(M(P(k)))
                max_M=M(P(k))
              end if
            end if
```

```
        end do
      if(occupied_neighbours.eq.0) then
        cluster=cluster+1
        M(q)=cluster
        number_sites(cluster)=1
      else
        M(q)=max_M          number_sites(M(q))=number_sites(M(q))+1
          do k=1,nn
            if (M(P(k)).gt.0.and.M(P(k)).ne.M(q)) then
              number_sites(M(q))=number_sites(M(q))+number_sites(M(P(k)))
              number_sites(M(P(k)))=0
              old_label=M(P(k))
              call relabel_cluster(q,old_label,M,N)
            end if
          end do
      end if
! Here is the step for determining the spanning cluster and therefore the critical con-
centration.
      if(percolation.eq.0) then
        call check_percolation(N,M,percolation)
        if (percolation.eq.1)jc=j+1
      end if
    temp=site(j)
```

```
        site(j)=site(r)
        site(r)=temp
    end do
 return
 end subroutine site_percolation


 subroutine find_nn(q,N,P,nn)
 implicit none
        integer,intent(in)::q,N
        integer::x,y,xprime,yprime
        integer,intent(out)::P(6),nn
        y=int(q/N)
        x=q-N*y
! Here begins the identification of the nearest neighbours of a site
        nn=0
        P=0
        xprime=x+1
        yprime=y
if(xprime.LE.N-1) then
        nn=nn+1
        P(nn)=yprime*N+xprime
end if
        xprime=x-1
```

```
        yprime=y
 if(xprime.GE.0) then
        nn=nn+1
        P(nn)=yprime*N+xprime
 end if
        xprime=x
        yprime=y+1
 if(yprime.LE.N-1) then
        nn=nn+1
        P(nn)=yprime*N+xprime
 end if
        xprime=x
        yprime=y-1
 if(yprime.GE.0) then
        nn=nn+1
        P(nn)=yprime*N+xprime
 end if
        xprime=x-1
        yprime=y+1
 if(xprime.GE.0.and.yprime.LE.N-1) then
        nn=nn+1
        P(nn)=yprime*N+xprime
 end if
```

```
        xprime=x+1
        yprime=y-1
 if(xprime.LE.N-1.and.yprime¿=0) then
        nn=nn+1
        P(nn)=yprime*N+xprime
 end if
 return
 end subroutine find_nn


 recursive subroutine relabel_cluster(q,old_label,M,N)
 implicit none
        integer,intent(in)::q,old_label,N
        integer::P(6),i,nn
        integer,intent(inout)::M(0:N**2-1)
        call find_nn(q,N,P,nn)
    do i=1,nn
       if (M(P(i)).eq.old_label) then
          M(P(i))=M(q)
          call relabel_cluster(P(i),old_label,M,N)
       end if
   end do
return
end subroutine relabel_cluster
```

```
subroutine check_percolation(N,M,percolation)
implicit none
    integer,intent(in)::N
    integer,intent(in)::M(0:N**2-1)
    integer::b1,b2,b3,b4
    integer,intent(inout)::percolation
  do b1=0,N-1
    if(M(b1).gt.0) then
      do b2=N-1,N**2-1,N
        if(M(b2).eq.M(b1)) then
          do b3=0,N*(N-1),N
            if(M(b3).eq.M(b1)) then
              do b4=N*(N-1),N**2-1
                if(M(b4).eq.M(b1)) then
                  percolation=1
                  return
                end if
              end do
            end if
          end do
        end if
      end do
```

```
        end if
    end do
return
end subroutine check_percolation
end module site_module
```

A.3 Bond percolation codes: Square lattice.

```
program bond_main
use bond_module
implicit none
        integer,parameter:: ntot=500,tot_cal=100
        integer::i,N
        integer::jc,sum
        real::runtime(2)
open(unit=20,file='output.dat')
open(unit=21,file='process.dat')
do N=50,ntot,50
        call cpu_time(runtime(1))
        call randomsetseed(123_i64)
        sum=0
    do i=1,tot_cal
```

```
      call bond_percolation(N,jc)
      sum =sum+jc
    end do
      write(20,*)1./real(2*N*(N-1)),real(sum)/real(tot_cal*N*(N-1)*2)
      write(21,*) 'Average value of pc = ',real(sum)/real(tot_cal*N*(N-1)*2)
      call cpu_time(runtime(2))
      write(21,*)'Number of calculation = ',tot_cal
      write(21,*) 'size and cpu_ time (seconds) ', 2*N*(N-1), runtime(2)-runtime(1)
end do
end program bond_main

module bond_module
use randomphys
implicit none
contains

subroutine bond_percolation(N,jc)
implicit none
      integer, parameter::i64=selected_int_kind(18)
      integer,intent(in)::N
      integer::M(0:N**2-1)
      integer::number_bonds(0:2*N*(N-1))
      integer::number_sites(0:N*N)
```

```
type bond_label
    integer::site
    integer::orientation
    integer::label
end type bond_label
    type(bond_label)::bond(2*N*(N-1)),temp
    integerinteger::i,j,bond_count,r,total_number_bonds,old_label
    integer::jmin,jmax,endpt1,endpt2,cluster
    integer,intent(out)::jc
    integer::percolation
    real(kind=r64)::rd
! N*N ..... total number of sites
! M ..... cluster label of each site
! number_sites ..... number of sites in a particular cluster
! number_bonds ..... number of bonds in a particular cluster
! bond%site ..... site label of the origin for the bond
! bond%orientation ..... 1 for bond at 0 degrees, 2 for bond at 60 de-
grees, 3 for bond at 120 degrees
! bond%label ..... has values from 1 to 2*N*(N-1), is the 1-dimensional
label of the bond
! total_number_bonds ..... 2*N*(N-1)
! endpt1, endpt2 ..... site labels for endpoints of the occupied bond
! cluster ..... increments by 1 whenever a new cluster is created
```

! percolation ..... =0 when no spanning cluster, =1 when spanning cluster exists

! sizes ..... number of clusters of size s for a particular value of p

! distinct_sizes ..... total number of distinct cluster sizes for a particular p

! initialization

```
total_number_bonds= 2*N*(N-1)
M=0
jc=0
cluster=0
percolation=0
number_bonds=0
number_sites=0
```

! labelling the bonds

```
bond_count=0
do i=0,N**2-N-1
  if (i+1.eq.int((i+1)/N)*N) then
    jmin=2; jmax=2
  else
    jmin=1;jmax=2
  end if
```

```
      do j=jmin,jmax
         bond_count=bond_count+1
         bond(bond_count)%site = i
         bond(bond_count)%orientation = j
         bond(bond_count)%label=bond_count
      end do
   end do
   do i=N**2-N,N**2-2
      bond_count=bond_count+1
      bond(bond_count)%site=i
     bond(bond_count)%orientation=1
     bond(bond_count)%label=bond_count
  end do
! percolation
   do j=1,total_number_bonds
      rd = randomnumber()
      r= int(rd*(total_number_bonds+1-j))+j
      endpt1=bond(r)%site
        if (bond(r)%orientation.eq.1) then
          endpt2=endpt1+1
        else
          endpt2=endpt1+N
```

```
end if
if (M(endpt1).eq.0.and.M(endpt2).eq.0) then
   cluster=cluster+1
   number_sites(cluster)=2
   number_bonds(cluster)=1
   M(endpt1)=cluster
   M(endpt2)=cluster
else if (M(endpt1).eq.0) then
   number_sites(M(endpt2))=number_sites(M(endpt2))+1
   number_bonds(M(endpt2))=number_bonds(M(endpt2))+1
   M(endpt1)=M(endpt2)
else if (M(endpt2).eq.0) then
   number_sites(M(endpt1))=number_sites(M(endpt1))+1
   number_bonds(M(endpt1))=number_bonds(M(endpt1))+1
   M(endpt2)=M(endpt1)
else if (M(endpt1).eq.M(endpt2)) then
   number_bonds(M(endpt1))=number_bonds(M(endpt1)) + 1
else if (M(endpt1).ne.M(endpt2)) then
   if (number_sites(M(endpt1)).le.number_sites(M(endpt2))) then
      number_sites(M(endpt2))=number_sites(M(endpt2))+number_sites(M(endpt1))
         number_bonds(M(endpt2))=number_bonds(M(endpt2))+number_bonds(M(er
         number_sites(M(endpt1))=0
         number_bonds(M(endpt1))=0
```

```
! update cluster labels
                    old_label=M(endpt1)
call relabel_cluster(endpt2,old_label,M,N)
                else
                  number_sites(M(endpt1))=number_sites(M(endpt1))+number_sites(M(endpt2)
                  number_bonds(M(endpt1))=number_bonds(M(endpt1))+number_bonds(M(en
                  number_sites(M(endpt2))=0
                  number_bonds(M(endpt2))=0
! update cluster labels
                  old_label=M(endpt2)
                  call relabel_cluster(endpt1,old_label,M,N)
                end if
            end if
! Here is the step for determining the spanning cluster
            if (percolation.eq.0) then
              call check_percolation(N,M,percolation)
              if (percolation.eq.1) jc=j
            end if
          temp=bond(j)
          bond(j)=bond(r)
          bond(r)=temp
        end do
      return
```

```
end subroutine bond_percolation


subroutine find_nn(q,N,P,nn)
implicit none
      integer,intent(in)::q,N
      integer::x,y,xprime,yprime
      integer, intent(out)::P(4),nn
      y=int(q/N)
      x=q-N*y
! Here begins the identification of the nearest neighbors of a site
      nn=0
      P=0
      xprime=x+1
      yprime=y
if(xprime.LE.N-1) then
      nn=nn+1
      P(nn)=yprime*N+xprime
end if
      xprime=x-1
      yprime=y
if(xprime.GE.0) then
      nn=nn+1
      P(nn)=yprime*N+xprime
```

```
end if
        xprime=x
        yprime=y+1
if(yprime.LE.N-1) then
        nn=nn+1
        P(nn)=yprime*N+xprime
end if
        xprime=x
        yprime=y-1
if(yprime.GE.0) then
        nn=nn+1
        P(nn)=yprime*N+xprime
end if
return
end subroutine find_nn

recursive subroutine relabel_cluster(q,old_label,M,N)
implicit none
        integer, intent(in)::q,old_label,N
        integer::P(4),i,nn
        integer ,intent(inout)::M(0:N**2-1)
        call find_nn(q,N,P,nn)
do i=1,nn
```

```
      if (M(P(i)).eq.old_label) then
         M(P(i))=M(q)
         call relabel_cluster(P(i),old_label,M,N)
      end if
end do
return
end subroutine relabel_cluster

subroutine check_percolation(N,M,percolation)
implicit none
      integer,intent(in)::N
      integer,intent(in)::M(0:N**2-1)
      integer::b1,b2,b3,b4
      integer,intent(inout)::percolation
do b1=0,N-1
      if(M(b1).gt.0) then
         do b2=N-1,N**2-1,N
            if(M(b2).eq.M(b1)) then
               do b3=0,N*(N-1),N
                  if(M(b3).eq.M(b1)) then
                     do b4=N*(N-1),N**2-1
                        if(M(b4).eq.M(b1)) then
                           percolation=1
```

```
                    return
                  end if
                end do
              end if
            end do
          end if
        end do
      end if
end do
return
end subroutine check_percolation
end module bond_module
```

A.4 Scaling function codes: Honeycomb lattices - Case of Bond Percolation

```
program bond_drive
use bond_drive_module
implicit none
    integer,parameter:: N=100,tot_cal=100
    integer::i,sum
```

```
      integer::jc
      real::runtime(2)
open(unit=20,file='number_clusters_100_100.dat',form='unformatted')
open(unit=21,file='jc_100_100.dat',form='unformatted')
open(unit=22,file='process_100_100.dat')
      write(20) n
      write(20) tot_cal
      call cpu_time(runtime(1))
      call randomsetseed(123_i64)
sum=0
do i=1,tot_cal
      call bond_percolation(N,jc)
      write(21) jc
      sum=sum+jc
end do
      write(22,*) 'pc= ',real(sum)/real(tot_cal*(3*n**2+4*n-1))
      call cpu_time(runtime(2))
      write(22,*)'Number of calculations = ',tot_cal
      write(22,*) 'size and cpu time (seconds) ', n, runtime(2)-runtime(1)
end program bond_drive

module bond_drive_module
use randomphys
```

```
implicit none
contains

subroutine bond_percolation(N,jc)
implicit none
      integer, parameter::i64=selected_int_kind(18)
      integer,intent(in)::N
      integer::M(1:2*(N+1)**2-2)
      integer::sizes(0:3*N**2+4*N-1)
      integer::number_bonds(0:3*N**2+4*N-1)
     integer::number_sites(0:2*(N+1)**2-2)
type bond_label
      integer::site
      integer::orientation
      integer::label
end type bond_label
      type(bond_label)::bond((3*N**2+4*N-1)),temp
      integer::i,j,bond_count,r,total_number_bonds,old_label
      integer::jmin,jmax,endpt1,endpt2,cluster
      integer,intent(out)::jc
      integer::percolation
      integer::distinct_sizes,s,c
     real(kind=r64)::rd
```

! 2(N+1)**2-2..... total number of sites*

! M ..... cluster label of each site

! number_sites ..... number of sites in a particular cluster

! number_bonds ..... number of bonds in a particular cluster

! bond%site..... site label of the origin for the bond

! bond%orientation..... 1 for bond at 60 degrees, 2 for bond at 120 degrees, 3 for bond at 90 degrees

*! bond%label ..... has values from 1 to 3*N**2+4*N-1), is the 1-dimensional label of the bond*

*! total_number_bonds ..... 3*N**2+4*N-1*

! endpt1, endpt2 ..... site labels for endpoints of the occupied bond

! cluster ..... increments by 1 whenever a new cluster is created

! percolation ..... =0 when no spanning cluster, =1 when spanning cluster exists

! sizes ..... number of clusters of size s for a particular value of p

! distinct_sizes ..... total number of distinct cluster sizes for a particular p

! initialization

total_number_bonds=3*N**2+4*N-1

M=0

```
      jc=0
      cluster=0
      percolation=0
      number_bonds=0
      number_sites=0
! labelling the bonds
      bond_count=0
do i=1,2*(N+1)**2-N-2
      if(i.le.N)then
        jmin=1;jmax=2
      else if(i.ge.N+1.and.i.le.2*N+1)then
        jmin=3;jmax=3
      else if(i.eq.int(i/(2*(N+1)))*2*(N+1))then
        jmin=1;jmax=1
      else if(i.eq.2*(N+1)**2-2-N)then
        jmin=2;jmax=2
      else if(i.lt.2*int(i/(2*(N+1)))*2*(N+1).and.i.ge.(int(i/(2*(N+1)))*2+1)*(N+1))then
        jmin=3;jmax=3
     else
        jmin=1;jmax=2
      end if
   do j=jmin,jmax
     bond_count=bond_count+1
```

```
      bond(bond_count)%site = i          bond(bond_count)%orientation = j
      bond(bond_count)%label=bond_count
   end do
end do
! percolation
do j=1,total_number_bonds
      rd = randomnumber()          r= int(rd*(total_number_bonds+1-j))+j
      endpt1=bond(r)%site
      if (bond(r)%orientation.eq.1.or.bond(r)%orientation.eq.3) then
         endpt2=endpt1+N+1
      else
         endpt2=endpt1+N
      end if
      if (M(endpt1).eq.0.and.M(endpt2).eq.0) then
         cluster=cluster+1
         number_sites(cluster)=2
         number_bonds(cluster)=1
         M(endpt1)=cluster
         M(endpt2)=cluster
      else if (M(endpt1).eq.0) then          number_sites(M(endpt2))=number_sites(M(endpt
         number_bonds(M(endpt2))=number_bonds(M(endpt2))+1
         M(endpt1)=M(endpt2)
      else if (M(endpt2).eq.0) then
```

```
        number_sites(M(endpt1))=number_sites(M(endpt1))+1
        number_bonds(M(endpt1))=number_bonds(M(endpt1))+1
        M(endpt2)=M(endpt1)
      else if (M(endpt1).eq.M(endpt2)) then
        number_bonds(M(endpt1))=number_bonds(M(endpt1)) + 1
      else if (M(endpt1).ne.M(endpt2)) then
        if (number_sites(M(endpt1)).le.number_sites(M(endpt2))) then
          number_sites(M(endpt2))=number_sites(M(endpt2))+number_sites(M(endpt1))
          number_bonds(M(endpt2))=number_bonds(M(endpt2))+number_bonds(M(endpt1)
          number_sites(M(endpt1))=0
          number_bonds(M(endpt1))=0
! update cluster labels
          old_label=M(endpt1)
          call relabel_cluster(endpt2,old_label,M,N)
        else
          number_sites(M(endpt1))=number_sites(M(endpt1))+number_          sites(M(e
          number_bonds(M(endpt1))=number_bonds(M(endpt1))+number_bonds(M(endpt2)
          number_sites(M(endpt2))=0
          number_bonds(M(endpt2))=0
! update cluster labels
          old_label=M(endpt2)
          call relabel_cluster(endpt1,old_label,M,N)
        end if
```

```
      end if
      sizes=0
      distinct_sizes=0
    do c=1,cluster
      s=number_bonds(c)
        if (s.gt.0) then          if (sizes(s).eq.0) distinct_sizes=distinct_sizes+1
        sizes(s)=sizes(s)+1
      end if
    end do
      write(20) distinct_sizes
   do s=1,total_number_bonds
      if (sizes(s)¿0) then
        write(20) s,sizes(s)
      end if
   end do
! Here is the step for determining the spanning cluster
      if (percolation.eq.0) then
        call check_percolation(N,M,percolation)
        if (percolation.eq.1) jc=j
      end if
      temp=bond(j)
      bond(j)=bond(r)
      bond(r)=temp
```

```
end do
return
end subroutine bond_percolation

subroutine find_nn(q,N,P,nn)
implicit none
      integer,intent(in)::q,N        integer,intent(out)::P(3),nn
! Here begins the identification of the nearest neighbors of a site
      nn=0
      P=0
if(q.le.N) then
      nn=2
      P(1)=q+N+1
      P(2)=q+N
else if(q.ge.2*(N+1)**2-N-1.and.q.le.2*(N+1)**2-2) then
      nn=2
      P(1)=q-N
      P(2)=q-N-1
else if(q.eq.N+1)then
      nn=2; P(1)=q+N+1; P(2)=q-N
else if(q.eq.2*(N+1)**2-2-N)then
      nn=2;P(1)=q+N;P(2)=q-N-1
else if(q.eq.int(q/(2*(N+1)))*2*(N+1).or.q.eq.int(q/(N+1))*2*(N+1)-1)then
```

```
      nn=2;P(1)=q+N+1;P(2)=q-N-1
 else if(q.gt.N+1.and.q.lt.2*N+1)then
      nn=3; P(1)=q+N+1; P(2)=q-N; P(3)=q-N-1
 else if(q.gt.int(q/(2*(N+1)))*2*(N+1).and.q.le.(2*int(q/(2*(N+1)))+1)*(N+1)-1)then
      nn=3
      P(1)=q+N+1
      P(2)=q+N
      P(3)=q-N-1
 else if(q.ge.(int(q/(2*(N+1)))*2+1)*(N+1).and.q.lt.(int(q/(2*(N+1)))*2+2)*(N+1)-
 1)then
      nn=3
      P(1)=q+N+1
      P(2)=q-N
      P(3)=q-N-1
 else if(q.gt.(int(q/(N+1))*2-1)*(N+1).and.q.lt.2*int(q/(N+1))*(N+1)-1)then
      nn=3
      P(1)=q+N+1
      P(2)=q+N
      P(3)=q-N-1
 else if(q.gt.2*(N+1)**2-2-2*N.and.q.lt.2*(N+1)**2-2-N)then
      nn=3
      P(1)=q+N+1
      P(2)=q+N
```

```
        P(3)=q-N-1
    end if
    return
    end subroutine find_nn


    recursive subroutine relabel_cluster(q,old_label,M,N)
    implicit none
        integer,intent(in)::q,old_label,N
        integer::P(3),i,nn
        integer,intent(inout)::M(1:2*(N+1)**2-2)
        call find_nn(q,N,P,nn)
    do i=1,nn
        if (M(P(i)).eq.old_label) then
          M(P(i))=M(q)
          call relabel_cluster(P(i),old_label,M,N)
        end if
    end do
    return
    end subroutine relabel_cluster


    subroutine check_percolation(N,M,percolation)
    implicit none
        integer,intent(in)::N
```

```
      integer,intent(in)::M(1:2*(N+1)**2-2)
      integer::b1,b2,b3,b4
      integer,intent(inout)::percolation
! Here is the step for determining the spanning cluster do b1=1,N
      if(M(b1).gt.0) then
        do b2=2*N+1,2*(N+1)**2-2*(N+1)-1,2*(N+1)
          if(M(b2).eq.M(b1)) then
            do b3=2*(N+1),2*(N+1)**2-2*(N+1),2*(N+1)
              if(M(b3).eq.M(b1)) then
                do b4=2*(N+1)**2-1-N,2*(N+1)**2-2
                  if(M(b4).eq.M(b1)) then
                    percolation=1
                    return
                  end if
                end do
              end if
            end do
          end if
        end do
      end if
end do
end subroutine check_percolation
end module bond_drive_module
```

```
program data_analysis
implicit none integer::n,tot_cal
      integer::i,j
      integer::total_number_bonds integer::jc
      integer::s,k,distinct_sizes,number_of_clusters
      real::sigma,tau,p,pc
      integer::bin_number,max_bin_number
      integer,allocatable::bin(:,:)
      real::geo_avg_s,var
! n ..... size of lattice
! total_number_bonds .....3* n**2 +4*n-1
! tot_cal ..... total number of calculations
! jc ..... pc = jc/(3* n**2 +4*n-1) is the percolation threshold
! distinct_sizes ..... number of distinct cluster sizes for a particular value
of p
! number_of_clusters ..... number of clusters of a particular size s for a par-
ticular value of p
! sigma, tau ..... critical exponents in 2d
! bin_number ..... bin index for cluster sizes in the range from 2**i to
2**(i+1)-1
! max_bin_number ..... total number of bins
! bin ..... number of clusters in each bin for a particular value of p
```

```
! geo_avg_s ..... geometric average of the cluster sizes
open(unit=20,file='number_clusters_100_100.dat',form='unformatted') open(unit=21,fi
open(unit=31,file='01.dat')
      sigma=36./91.
      tau=187./91.
read(20) n
read(20) tot_cal
      total_number_bonds = 3* n**2 +4*n-1
      max_bin_number=int(log(real(total_number_bonds))/log(2.))+1
      allocate(bin(max_bin_number,total_number_bonds))
      write(*,*) 'n =',n
      write(*,*) 'total number of calculations',tot_cal
      bin=0
do i=1,tot_cal
      read(21) jc
      do j=1,total_number_bonds
        read(20) distinct_sizes
        do k=1,distinct_sizes
          read(20) s,sizes(s)
          number_of_clusters=sizes(s)
          bin_number=int(log(real(s))/log(2.))+1
          bin(bin_number,j)=bin(bin_number,j)+number_of_clusters
        end do
```

```
      end do
end do
do k=1,max_bin_number          var=bin(k,jc)/100./total_number_bonds
      write(31,*) k*log(2.),log(var)
end do
end program data_analysis
```

Résumé

Born at Kiyange, Burundi in November 12, 1961, I graduated from high school in 1980 and entered directly into the University of Burundi where I graduated with a Bachelor of Science degree in Physics (Licencié en Sciences Physiques) in September 1985.

From that period until June 1994, I was employed by the Burundi Government as a high school teacher of Math and Physics, a high school headmaster and a pedagogical adviser of the National Curriculum Development Office for high schools. In 1990-1991, I was on service training at Leuven Catholic University (Louvain-La-Neuve) in Belgium. In June 1994, I was appointed by the University of Burundi as an assistant lecturer.

Since August 2000, I have been a postgraduate student at the University of Natal - Pietermaritzburg, South Africa.

December 2002

Léonard NDUWAYO

Translation of the Abstract in French

L'objet principal de ce travail a été d'étudier la théorie de percolation sur des réseaux réguliers en deux dimensions en vue de déterminer avec des méthodes numériques la fonction pondérée du nombre d'amas par site ou lien du réseau $f(z)$. Une brève introduction à ce sujet est faite dans l'optique de répondre à une question double: *Qu'est-ce que la percolation et pourquoi est-elle importante ?* Aussi sont mentionnées dans cette partie introductive des applications de cette théorie.

Les algorithmes employés dans nos méthodes sont rédigés dans un langage de programmation en Fortan 90 sur un Système Linux. Dans un premier temps, nous simulons le phénomène de percolation des sites et des liens pour déterminer les valeurs seuils de la probabilité de percolation sur des réseaux finis, p_c. On extrapole ensuite ces valeurs seuils sur des réseaux infinis par l'application des méthodes statistiques (régression et corrélation linéaires). Dans une seconde phase, nous déterminons de façon numérique la fonction pondérée du nombre d'amas par site ou lien des réseaux en assumant son expression telle qu'elle a été proposée par Stauffer (Nakanishi et Stanley, 1986). Les

exposants critiques liés à cette fonction, τ et σ, sont aussi calculés.

Les valeurs obtenues pour p_c sont proches des valeurs connues de la littérature scientifique. Pour toute valeur de la probabilité d'occupation des sites ou des liens p, allant de 0 à 1, le nombre d'amas de taille s par site ou lien du résau $n_s(p)$ est systématiquement enregistré comme étant une fonction à deux variables s et p. La détermination et le stockage des données relatives à $n_s(p)$ exigent un temps important de compilation et une grande capacité de l'outil informatique utilisé.

Indépendamment de la nature du réseau utilisé (triangulaire, carré et en nid d'abeille) et bien que la percolation de sites et la percolation des liens soient différentes, les graphes tracés et relatifs à $f(z)$ affichent un comportement similaire. Ainsi le caractère universel de $f(z)$ se retrouve vérifié dans les limites d'une simulation numérique recoulant surtout à l'usage des méthodes de Monté Carlo.