Lab 9

Percolation

**Purpose of the experiment:**

In this lab we will use the Hoshen - Kopelman method to label the clusters in a square lattice

**Problem Description:**

In this lab, we will declare a lattice of 10x10 called n. The lattice is occupied at random based on a probability p. All lattice sites (matrix elements) are initialized to 0. Once a site becomes occupied, the value of n[i,j] at that point is changed to 1 or true. The matrix will be all zeros and ones

For the implementation of the Hoshen – Kopelman method, the script starts with a partially filled lattice, picks a site randomly and defines it to be occupied as show in figure 1 below

i = int((lattice\_size-2)\*rng())+1

j = int((lattice\_size-2)\*rng())+1

while n[i,j] > 0:

i = int((lattice\_size-2)\*rng())+1

j = int((lattice\_size-2)\*rng())+1

n[i,j]=1

*(Figure 1 – implementation of Hoshen – Kopelman method for lab 9)*

Variable x is a counter to store cluster labels. When a new site is added, the counter will increase by 1. These labels for each each site, are stored in a 10x10 matrix called d as show in figure 2.

x = x+1

d[i,j] = x

(Figure 2 – storing cluster labels and creating a 10 by 10 matrix

This label is called the proper label of the site and is a positive integer number.

clust is a 100 element delay to keep track matrix connectivity. It will be updated based on the connectivity of the clusters. The new site that was added connect two clusters called m1

and m2. clust(m1) and clust(m2) to point to the new cluster number x as follows, indicating that m1 and m2 clusters are now part of cluster x.

clust(m1)=-x, clust(m2)= -x, clust(x)=x

The first two clusters now have additional label and have preserved its original labeling. x, will be the “proper” label of the new larger cluster. To find out if it’s cluster label for a given site

the cluster label, d[i,j] is analyzed. In this case, the cluster label could be e.g. m1. We look at clust(m1). If clust(m1) is positive, m1 is a proper cluster label and if it is negative, it means that m1 is now a larger cluster. To find it, we iteratively look at the other labels, until the proper label is found and associated with that cluster.

A matrix is generated in the script. When a new occupied site is added to the matrix, we add a new cluster label to the new site, and define the proper cluster label for that site to be x as well

d[i,j] = x

clust[x]=x

Then there is a search for four neighbors of the new site to find the occupied

sites. If the sites are occupied such as n[i-1,j], we will look for its cluster

number:

ind = d[i-1,j]

and determine if it is a number.

When clust[ind] > 0, ind is the right cluster number for this cluster. Otherwise, the script will keep looking for cluster number by looking at ind=clust[-1\*ind]. It will continue this until we find the ind > 0 or the right cluster. The cluster will then be needed to be updated by change to the new value x;

clust[ind] = -1 \* x

Where x is a proper label, and we know that from the fact that clust[x]=x a

positive number.

I then ran the script to create the matrix n and identify the clusters in the matrix and printed it.

I also printed the connectivity matrix for your output. The connectivity matrix should show the proper cluster label for each point.

For the second part of the lab, the percolation probability of the clusters was caclculated as shown in figure \_\_\_. I ran the script for different values of p (occupancy probability) and determine whether you have percolation. To do so, once the matrix n is generated, the script run the script to label the clusters. I then analyzed the edges, and see whether there are points on the four edges that have the same “proper” cluster number. If they did, a percolating network existed, if not your network is not percolated. I then calculated the percolation probability of the lattice as a function of lattice size (L=5,10, 15) and plotted it.

In the script, the edges of the cluster at i,j=0,lattice\_size-1 are left

empty intentionally. This is to simplify the search for nearest neighbors. So, the

actual edges of the network are located at i,j=1,lattice\_size-2

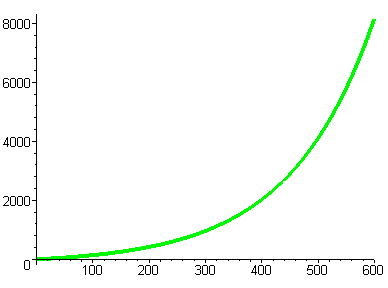
.**Equations solved and algorithms used:**

In this lab, I tried to solve using the Hoshen – Kopelman to solve the lab.

**Results & Analysis**

I was unable to finish the lab. While I was trying to organize, and create temporary clusters for organization, I couldn’t recreate positive data for the elements because the loops were infinite loops which stopped me from proceeding with the code. With help and direction of others, I created a large nested for loop that repeated for 80 times.

Based off the readings and research I did, I know that the graph of the of probability vs the occupation of each matrix will have increase exponentially until when probability is 1 where the whole matrix will be filled. The curve would be like figure 1.



*Figure 1*