# PHYC40970- Advanced Laboratory II


# Percolation Theory

Nathan Power

19311361

# Abstract

This experiment investigates the percolation theory. Percolation theory is a branch of physics that studies the behaviour of systems at the threshold of a phase transition, specifically the formation of clusters in random networks. This report presents a theoretical and experimental analysis of percolation phenomena, including the critical exponents and percolation probability that govern the behaviour of systems near the percolation threshold. The results of numerical simulations are also discussed, demonstrating the validity of the theoretical and experimental predictions. The paper also includes a discussion of the applications of percolation theory in a variety of fields, including condensed matter physics, statistical mechanics, and network science.

# Introduction

If a container is filled with small glass beads and a battery is connected to the ends of the container, no current would pass and the system would be an insulator. Suppose that we choose a glass bead at random and replace it by a small steel ball. Clearly, the system would still be an insulator. If we continue randomly replacing glass beads with steel balls, eventually a current would pass.[1] What percentage of steel balls is needed for the container to become a conductor? The change from the insulating to the conducting state that occurs as the percentage of steel balls is increased is an example of a percolation phase transition.[2]

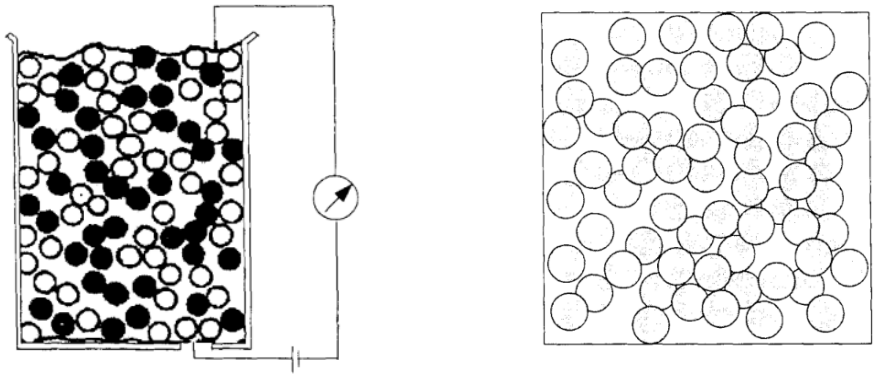


*Fig 1. (A) schematic diagram for a three dimensional percolation experiment with steel and glass balls [1] (B) Cookies placed at random on a large sheet. See that there is a path of overlapping cookies that connects the bottom and top edges of the cookie sheet. [2]*

Another example of percolation is from the kitchen. Imagine a large metal sheet on which we randomly place drops of cookie dough. Assume that each drop of cookie dough spreads while the cookies are baking in an oven. If two cookies touch, they coalesce to form one cookie. In time, we may have one very large cookie that spans from one edge of the sheet to the opposite edge. [2] If such a spanning cookie exists, we say that there has been a

percolation transition and this cookie is a spanning cluster. As we will discuss in more detail, percolation has to do with connectivity. [2]
The applications of percolation, go beyond metal-insulator transitions and the conductivity of wire mesh, and include the spread of disease in a population, the behaviour of magnets diluted by nonmagnetic impurities, the flow of liquid through porous material, the microstructure of fibre-reinforced concrete, and the characterization of gels. Percolation ideas also have been used to understand clusters in such diverse systems as granular matter and social networks. [2]

# Theory

## Site Percolation

As previously stated, percolation is the study of connectivity. The simplest question we can ask is when does a path form from one side of the sample to the other? By when, we mean at what value of p.[3] We discussed a simple model of the cookie tray, we represented the cookie sheet by a lattice where each site can be in one of two states, occupied or empty. Each site is occupied independently of its neighbours with probability p. This model of percolation is called site percolation. If we want to make a path along the occupied sites from one side to another, we must decide on when two sites are connected. Here, we will typically use nearest neighbour connectivity (Two sites in a square lattice are connected if they are nearest neighbours and their faces meet). In the square lattice in Fig. 2, each site has 4 nearest neighbours and 8 next-nearest neighbours. We call the value $p_c$, when we first get a path from one side to another, the percolation threshold. For a given version of the matrix, there is a well-defined value for $p_c$, but for another realisation, there would be another $p_c$. [1]

## Percolation Threshold, $p_c$

An easy way to study site percolation is to generate a uniform random number, X, in the unit interval $0 < X \leq 1$ for each site in the lattice. A site is occupied if its random number satisfies the condition $X \leq p$.[3] If p is small, we expect that only small isolated clusters will be present, Fig 2.(A). If p is near unity, we expect that most of the lattice will be occupied, and the occupied sites will form a large cluster that extends from one end of the lattice to the other, Fig 2.(C).

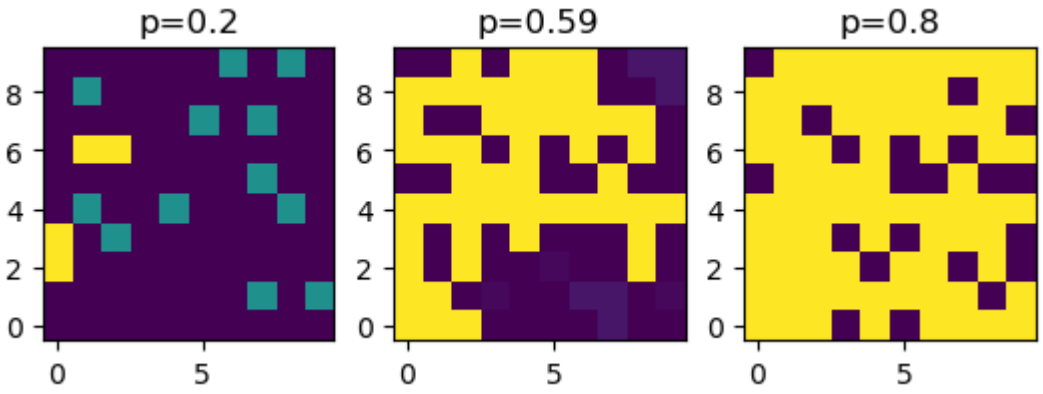



*Fig.2 Examples of site percolation clusters on a square lattice of linear dimension L = 10 for (A) p = 0.2, (B) p = 0.59, and (C) p = 0.8.*

Such a cluster is said to be a spanning cluster. Because there is no spanning cluster for small p and there is a spanning cluster for p near unity, there must be a value of p at which a spanning cluster first appears, Fig. 2(B).
We shall see that in the limit of an infinite lattice, there exists a well defined threshold probability pc such that:[3]

$For\ p\ <\ p_c\ ;\ no\ spanning\ cluster\ exists.$

$For\ p\ >\ p_c\ ;\ a\ spanning\ cluster\ exists.$

$For\ p\ =\ p_c\ ;\ a\ spanning\ cluster\ exists\ with\ a\ probability\ greater\ than\ zero\ and\ less\ than\ unity.$

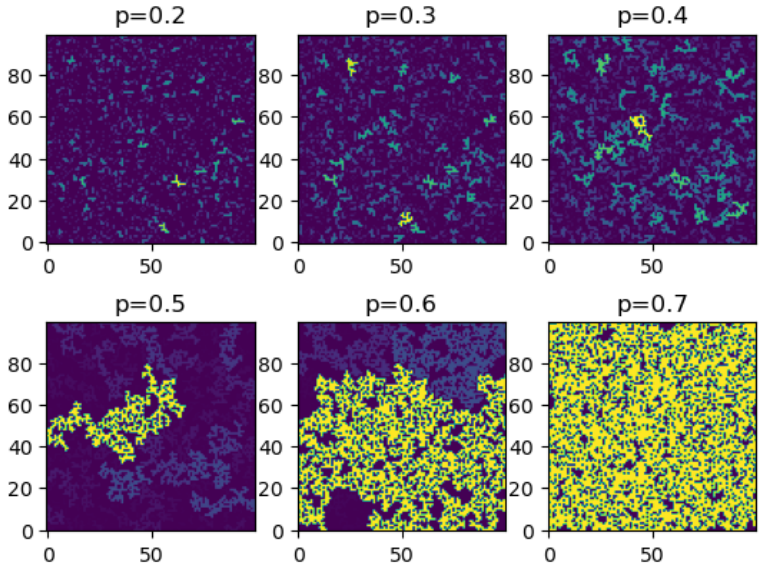



*Fig. 3 Plot of the clusters in a 100 × 100 system for various values of p.*

We emphasise that the defining characteristic of percolation is connectedness. Because the connectedness exhibits a qualitative change at a well defined value of a continuous parameter, we shall see that the transition from a state with no spanning cluster to a state with a spanning cluster is an example of a phase transition. Fig. 3 shows the clusters for a 100 × 100 system for p ranging from 0.2 to 0.7 in steps of 0.1. We see that the clusters

increase in size as p increases, but at p = 0.6, there is just one large cluster spanning the entire region. We have a percolating cluster, and we call this cluster that spans the system the spanning cluster. However, the transition is very rapid from p = 0.5 to p = 0.6. [3]We therefore look at this region in more detail in Fig. 4.

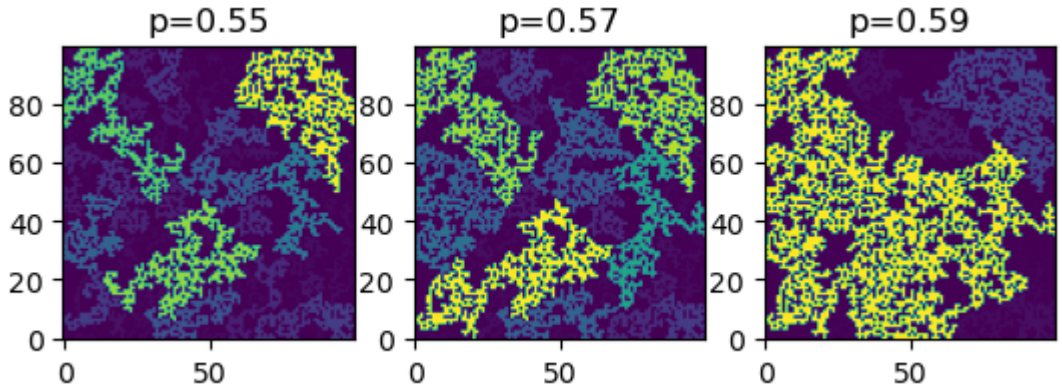



*Fig. 4 Plot of the clusters in a 100 × 100 system for values of p = 0.55, 0.57, 0.59.*

We see that the size of the largest cluster increases rapidly as p reaches a value around 0.6, which corresponds to pc for this system. At this point, the largest cluster spans the entire system. For the two-dimensional system illustrated here we know that in an infinite lattice the percolation threshold is $p_c \approx 0.5927$.

We are familiar with different phases of matter from our everyday experience. The most familiar example is water which can exist as a gas, liquid, or solid. It is well known that water changes from one phase to another at a well defined temperature and pressure, for example, the transition from ice to liquid water occurs at 0 ◦C at atmospheric pressure. Such a change of phase is an example of a thermodynamic phase transition which occurs at a critical point.[3]

The behaviour of a percolation system is different in one, two and three dimensions. However, the most important differences occur between one and two dimensions, where the difference is dramatic, whereas the difference between two and three dimensions is more of a degree that we can easily handle. Actually, the percolation problem becomes simpler again in higher dimensions. In two dimensions and three dimensions, it is possible to go around a hole, and still have connectivity which is not possible in one dimension. Here in this report, we will therefore focus on two and three-dimensional systems.[3]

## Percolation probability

When does the system percolate? When there exists a path connecting one side to another. This occurs at some value p = pc. However, in a finite system, the value of pc for a given grid system will vary with each run. It may be slightly above or slightly below the pc we would expect to find in an infinite sample. We can characterise this behaviour by introducing a probability Π(p, L): the probability for there to be a connected path from one side to another side as a function of p in a system of size L.[3]

We can measure Π(p, L) in a finite sample of size L×L, by generating many random lattices. For each matrix, we perform a cluster analysis for a sequence of p values. For each p we find all the clusters and we count up how many times a system percolates for a given p , Ni, and then divide by the total number of experimental runs, N, in order to estimate the probability for percolation for a given p; Π(p , L) = Ni/N [3][4]

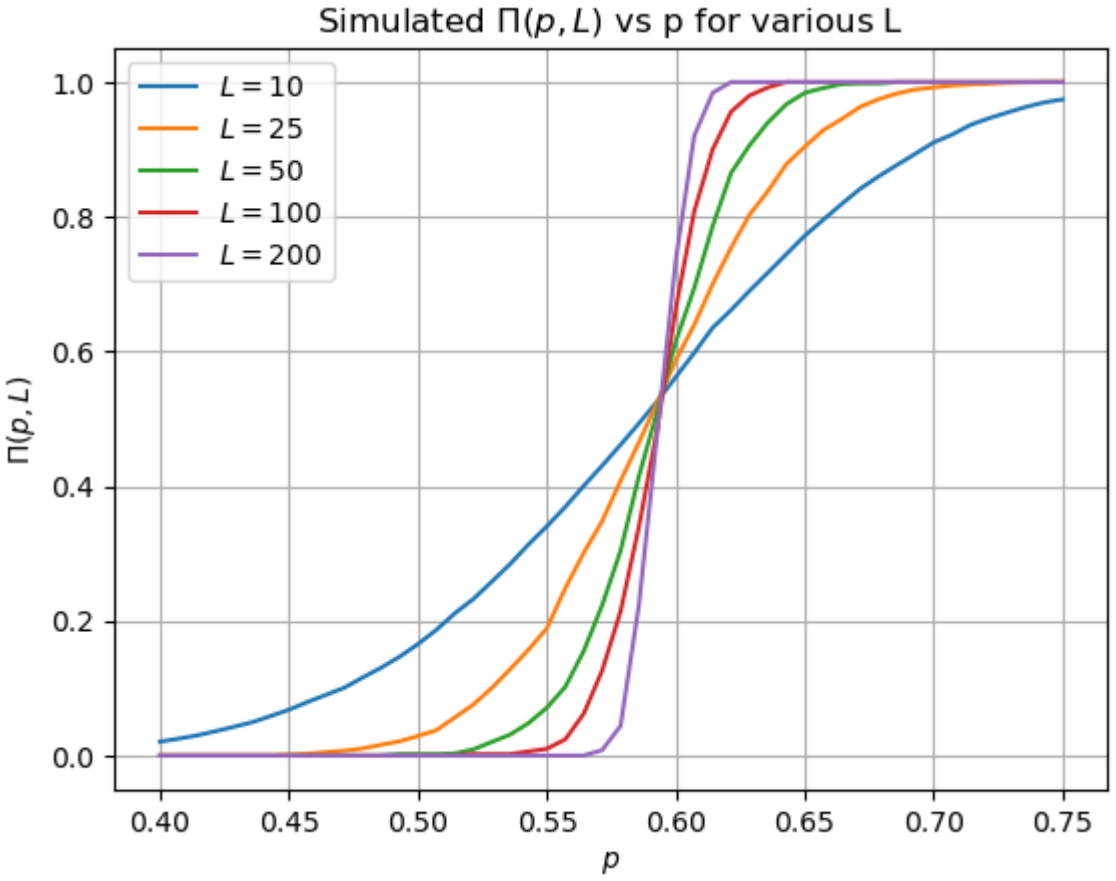



*Fig. 5 Plot of Π(p, L), the probability for there to be a spanning cluster.*

The probability Π(p, L) describes the probability for there to be a spanning cluster, but what about the spanning cluster itself, how can we analyse it? We see from Fig. 4 that the spanning cluster grows quickly around p = pc. Let us therefore characterise the cluster by its density, P(p, L) , which corresponds to the probability for a site to belong to the spanning cluster.[3]

We can measure P(p, L) by counting the size of the spanning cluster as a function of p for various values of p. We can find the mass of the spanning cluster, by finding a cluster that spans the system and then measuring the number of sites in the cluster

The resulting plot of P(p, L) is shown in the bottom of Fig.6. We see that P(p, L) changes rapidly around p = pc and that it grows slowly as p → 1. When p is near 1 all the set sites are connected and part of the spanning cluster. The density of the spanning cluster is therefore proportional to p in this limit. We can now see that both Π(p, L) and P(p, L) can be fit with a power law proportional to $P(p,\ L) \propto (p\ -\ p_c)^{\ v}$ and $\Pi(p,\ L) \propto (p\ -\ p_c)$ .[3][4]

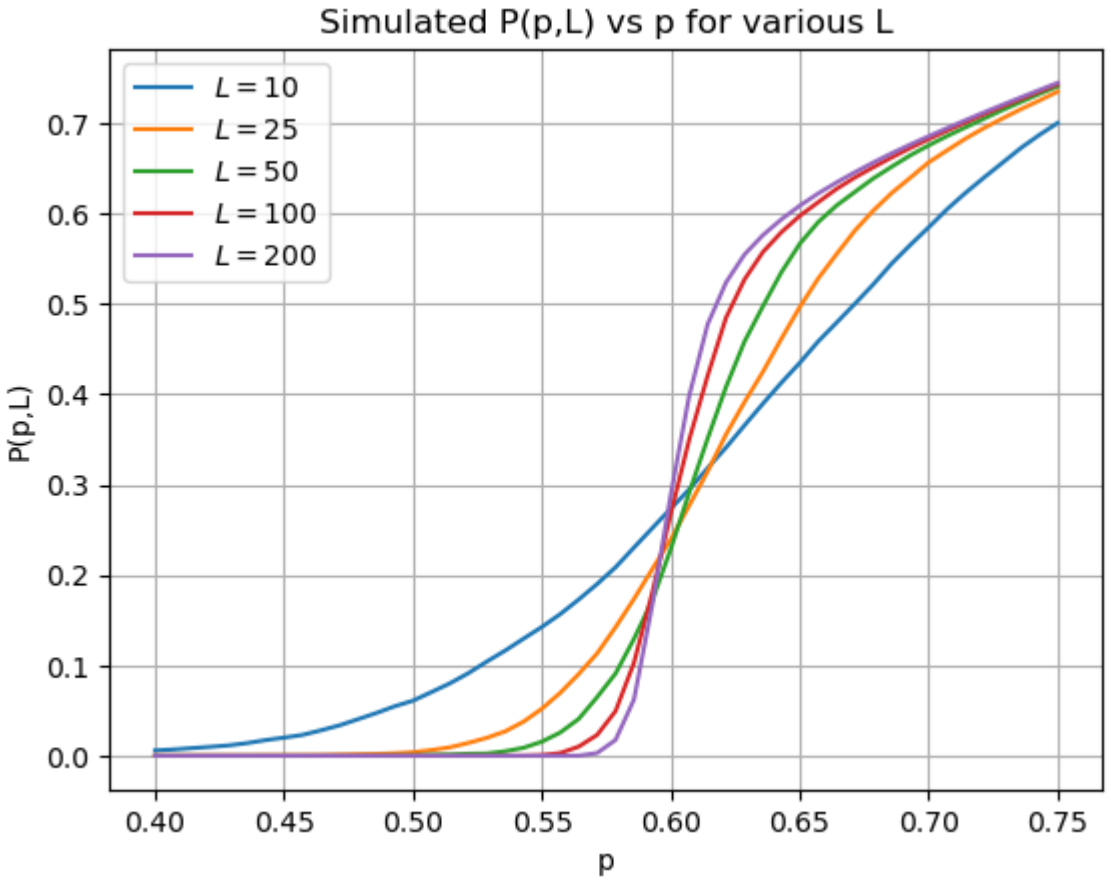
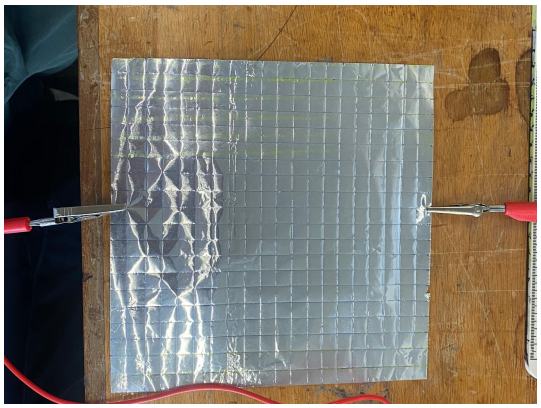



*Fig. 6 Plot of P(p, L), the probability for there to be a spanning cluster.*

The resulting plot of P(p, L) is shown in Fig. 6. We see that as L increases, P(p, L) approaches the shape expected in the limit when $L \to \infty$, this can also be seen in Fig. 5 with $\Pi(p, L)$.

Instead of trying to simulate one single simulation with as large L as possible, it would be more effective to instead vary L and then use this to estimate the relevant exponent ν and β.

# Experimental Procedure



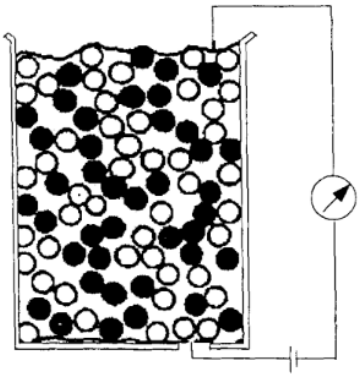


*Fig.7(A) and Fig.2(B)  shows the apparatus used in this experiment [1]*

Figure 7(A) and 7(B) show the apparatus used in this experiment. Two dimensional percolation phenomena can be observed with a piece of aluminium foil. In the first part of this experiment I measured the electrical conductivity of a large piece of uniform 20 x 20cm aluminium foil, which I divided into 100 equal square sites, as a function of the fraction of the sites that were occupied. I did this by measuring the resistance across the sheet with all sites occupied and then began removing a site corresponding to a decrease of 0.01 in p. The resistance was measured using crocodile clips at opposite sides of the sheet connected to an ohmmeter. The coordinates of the sites that were removed were given by a random number generator. The measured electrical conductivity was found to be a rapidly decreasing function of the fraction of sites still present and vanishes below a critical threshold, pc. The same idea was used in the second part of the experiment, where I used glass balls and steel balls to simulate the nonconducting and conducting elements, respectively, as schematically shown in Fig.7(A). In each measurement, steel balls were mixed up with glass balls and the mixture was poured into a tube. Two crumpled iron foils were pressed to connect the bottom and top of the system and they were linked to an ohmmeter. For each fraction of steel balls, p, I measured the total resistance of the mixture system. The above procedure was repeated again and again for different values of p.

# Results and Discussion

## Percolation Threshold in 2D and 3D

From the experimentally measured resistance, the corresponding values of the conductance and p as a function of p were taken at concentration intervals of 1%.The value of conductance decreased gradually from 0.0125 /Ω to 0 at percolation threshold = 0.5733 ± 0.1677.

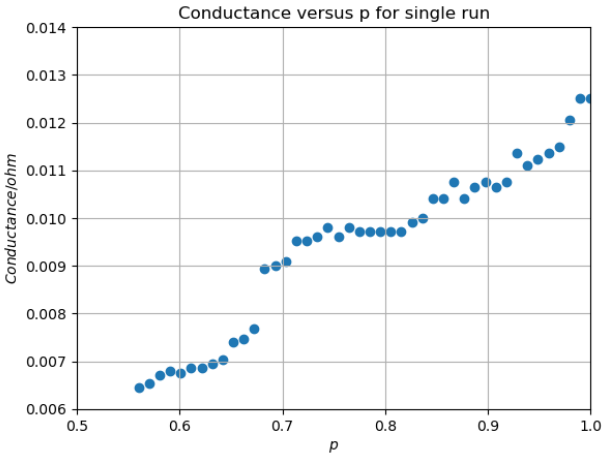
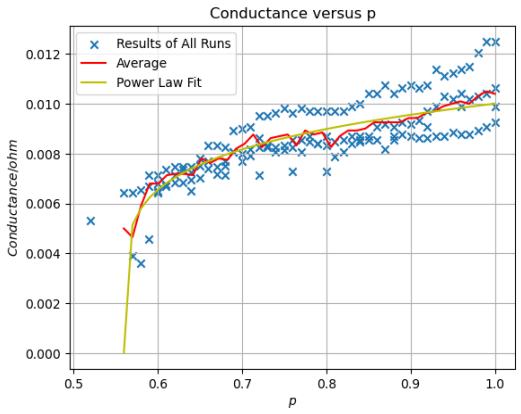



*Fig.8 shows the dependence of the conductance on p in a single experiment.*

Conductance reached the value of zero at the point when no links existed between the opposite sides of the foil. The measured density p at this point was p = 0.5733 ± 0.1677. This experimental value is to be compared with the theoretical value of ≃ 0.59 obtained by various methods and estimated value of 0.59277 [4].
For concentrations close to 1, the conductance was found to be nearly linear with p. This is in good agreement with effective-medium theory.[4]

The experimentally measured values of the concentration dependent conductance in a single typical experiment are shown in Fig. 8 The large fluctuations in the results of different experiments are due to the fact that the reported data were obtained from a relatively small sample.
For p>pc, the spanning cluster contained weak links that connected different parts of the cluster. Cutting out a site on a weak link caused a sharp increase in the measured resistance. On the contrary, there was no change when a site that belonged to a dead end of the cluster was cut out. This explains the steps and jumps near pc in Fig. 8. In percolation experiments, conductance follows the relationship; $\sigma(p, L) \propto (p - p_c)^t$
By calculating t from the points we obtained in Fig.9(A) t = 0.177 ∓ 0.021. This result is not agreement with the theoretical values between 1.1 and 1.4 for two dimensional systems.[4] This may be due to the fact that I conducted a small sample size and also due to the fact I used an L value of just 10. More accurate range of results would be found by repeating the experiment at various sizes of L for a large number of runs.



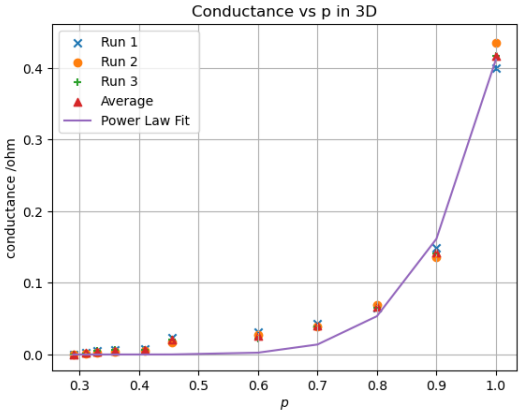




*Fig.9(A) and (B) show conductance vs p over all experimental runs, for 2D and 3D respectively*.

Looking at Fig.9(B) we can see the relationship between conductance in 3D and the conductive concentration of steel balls. We see it follows an exponential relationship and fits the power law of $\sigma(p, L) \propto (p - p_c)^t$ with a pc value of 0.29 ∓ 0.01 and a t value of 6.16 ∓ 0.55.

| Table 1 | | | | |
|---|---|---|---|---|
| # of Dimensions | Expected $p_c$ | Measured $p_c$ | v | |
| 2 | 0.59277 | 0.5733 ± 0.1677. | 1.577 ± 0.358 | 0.229 ± 0.098 |
| 3 | 0.27 | 0.2616 ± 0.01 | | |

Percolation probability in 2-dimensions

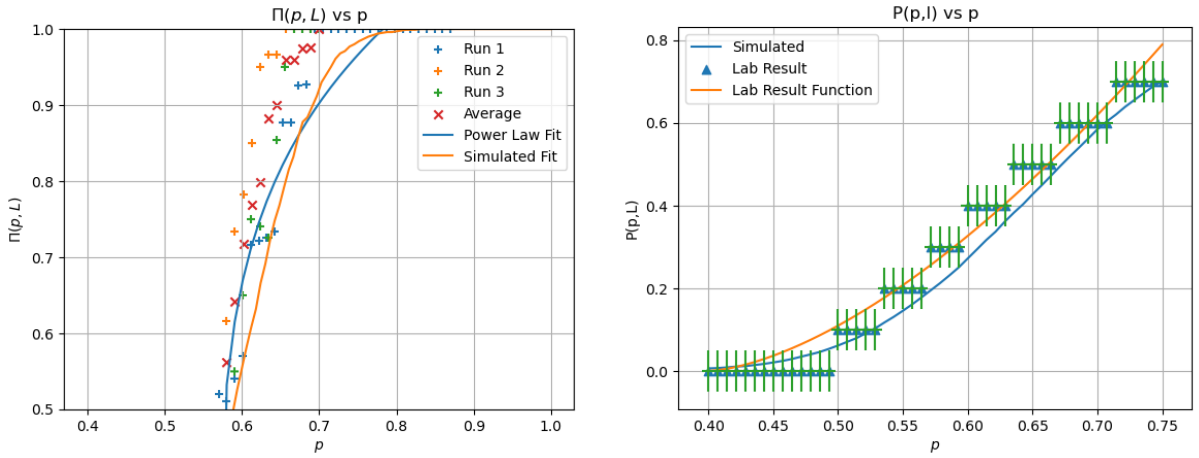



*Fig.10(A) and (B) show Π(p, L) vs p and P(p,l) vs p*

We can measure Π(p, L) in a finite lattice by finding all the clusters for each p and count up how many times a system percolates for a given p and then divide by the total number of experimental runs. In Fig.10(A) we see the percolation probability results roughly agree with the simulation curve for L=10. For increasing values of L this curve will become sharper just as in Fig.5. In Fig.10(B) the experimental values of P(p,l) are given as a function of p.The decrease of P(p,l) as p approaches pc agrees with the behaviour as predicted by the simulation and theory, that is smaller than 1. The decrease of P(p,l) becomes sharper around pc as the size of L increases as seen in simulations in Fig.6.

# Conclusion

From our results, it can be clearly seen that the majority of our data analysis agrees with the expected results found through simulation and through theory. [1][2][3][4] We can clearly see that we were able to replicate the ideas found in some of the references below [1][2][3][4] in our results for the percolation threshold values and percolation probability calculations. However, our conducting of this experiment was not without error. If I was to conduct this experiment again, I would make sure to run the experiment for many values of L and for a larger sample size. This way we could find more accurate results for the percolation threshold and for the critical values such as t, v and . I would also change the amount of points taken during this stage. I would take measurement in smaller increments closer to the predicted percolation threshold in order to examine it in further detail. Although we did not succeed in all areas in this experiment, such as fitting the conduction curve in 2D, I believe that it is important that undergraduates develop a strong understanding of percolation theory as it has relevance in many areas in everyday life such as disease control, social networks and thermodynamic phase transitions.

# References

[1]储少军, 牛强, 刘新宇 *and* 王欣, *2003. Simulation on the conductivity of charging stock with percolation structure in the submerged arc furnace.* 过程工程学报, *3(5), pp.413-418.*

[2] *An Introduction to Computer Simulation Methods Third Edition (revised),Harvey Gould, Jan Tobochnik, and Wolfgang Christian, Chapter 12 Percolation*

[3]*Malthe-Sørenssen, A. (2020). Percolation theory using Python. [online] Available at: https://www.uio.no/studier/emner/matnat/fys/FYS4460/v20/notes/book.pdf.*

[4] *Mehr, R., Grossman, T., Kristianpoller, N. and Gefen, Y., 1986. Simple percolation experiment in two dimensions. American Journal of Physics, 54(3), pp.271-273..*

# Appendix

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

In [2]:
```
from random import randint #RNG used to pick sites in 2D lattice
# generate some integers
for _ in range(2):
    value = randint(1, 5)
    print(value)
```

```
4
2
```

In [60]:
```
run1 = np.loadtxt("run1_2D.txt") #load in data
run2 = np.loadtxt("run2_2D.txt")
run3 = np.loadtxt("run3_2D.txt")
run5x5 = np.loadtxt("run1_2D_5x5.txt")
avg = np.loadtxt("avg_2D.txt")
prob = np.loadtxt("per_prob.txt")
```

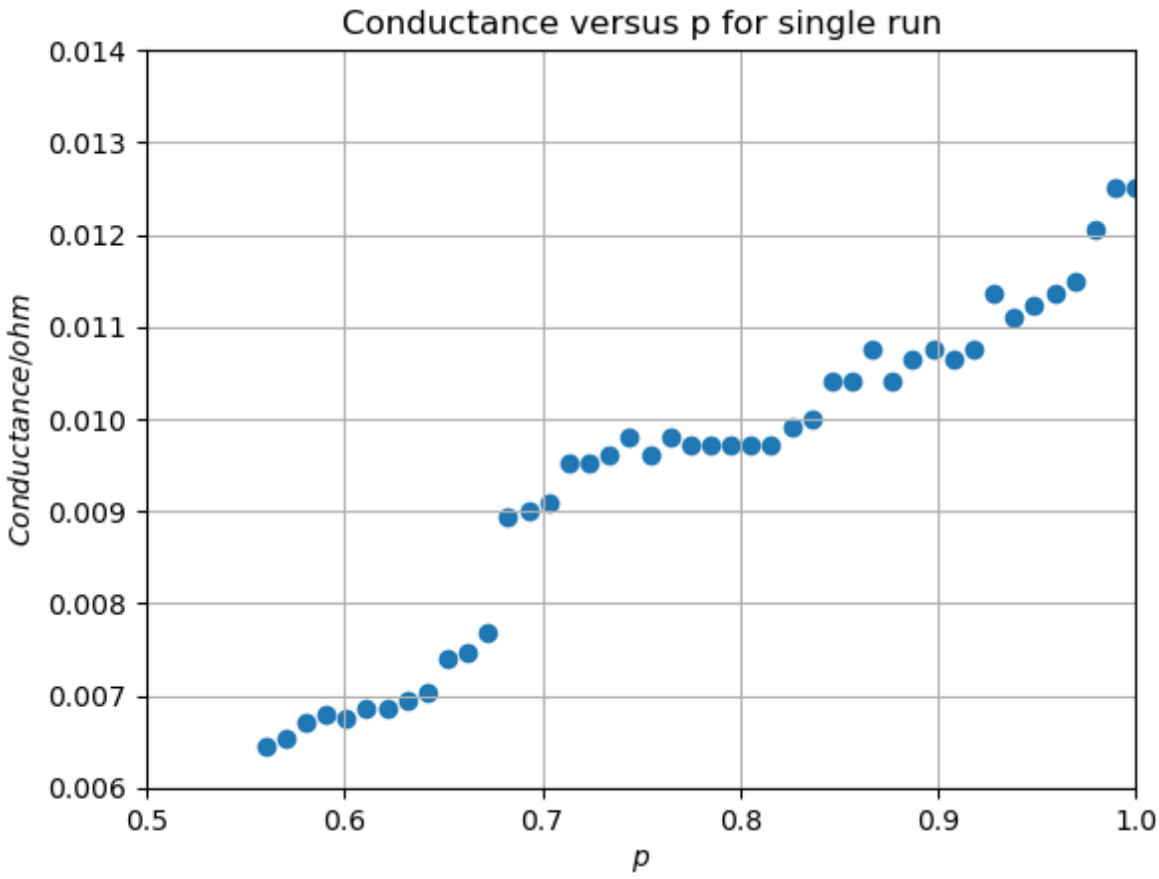
In [7]:
```
run1_1 = np.loadtxt("run1_2D_1.txt") #load in data
run2_1 = np.loadtxt("run2_2D_1.txt")
run3_1 = np.loadtxt("run3_2D_1.txt")
run5x5_1 = np.loadtxt("run1_2D_5x5_1.txt")
```

In [5]:
```
x=np.linspace(1.00,.56,44) #p values for different sets of data
x1=np.linspace(1.00,.58,42)
x2=np.linspace(1.00,.52,13)
x3=np.linspace(1.00,.55,45)
```

# Conduction in 2D

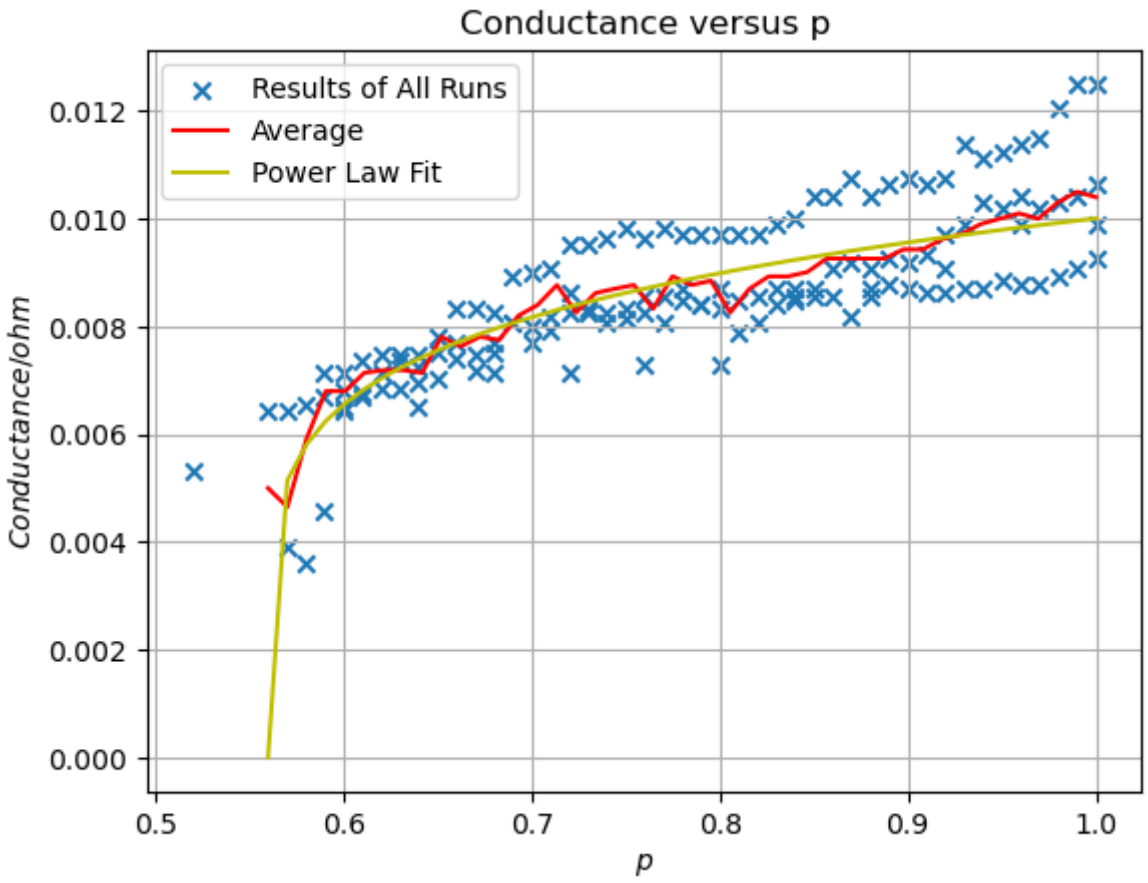
In [727…
```
plt.scatter(x,1/run1)
plt.ylim(0.006,0.014)
plt.xlim(0.5,1)
plt.grid(True)
plt.ylabel("$Conductance /ohm$")
plt.title("Conductance versus p for single run")
plt.xlabel("$p$")
```

Out[727]:
```
Text(0.5, 0, '$p$')
```




In [8]:
```
arr1 = np.concatenate((run1_1, run2_1,run3_1,run5x5_1))
```

In [20]:
```
plt.scatter(arr1[:,0]/100,1/arr1[:,1],marker = "x")
plt.plot(x,avg/1000, color="r")
plt.grid(True)
plt.plot(x,func2(x,0.0115707 , 0.17700229),color ="y")
plt.ylabel("$Conductance /ohm$")
plt.title("Conductance versus p")
plt.xlabel("$p$")
plt.legend(["Results of All Runs","Average","Power Law Fit "])
plt.grid(True)
```

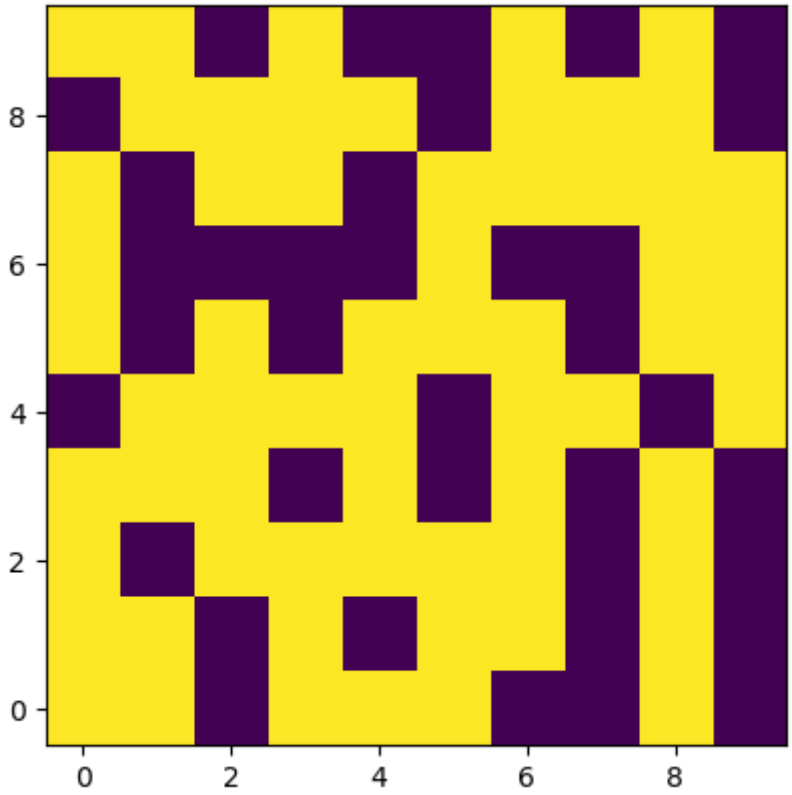



In [61]:
```
def func2(p,A, t): #conduction power law
    return A * np.power(p-.56,t)
popt, pcov = curve_fit(func2,x,avg/1000)
popt, pcov
```

Out[61]:
```
(array([0.0115707 , 0.17700229]),
 array([[1.61212492e-07, 7.48859395e-06],
        [7.48859395e-06, 4.20931935e-04]]))
```

# Simualtion of Square Lattice Site Percolation

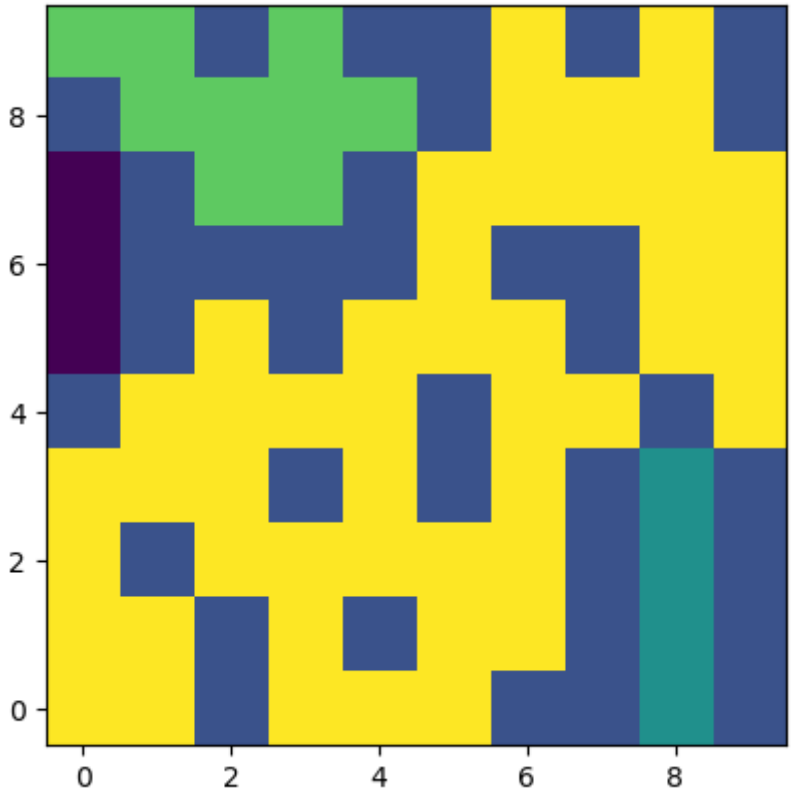
In [25]:
```
from pylab import *
L = 10
p = 0.6
z = rand(L,L)
m = z<p
imshow(m, origin="lower")
show()
```




In [28]:
```
from scipy.ndimage import measurements #different clusters have different colours
lw, num = measurements.label(m)
b = arange(lw.max() + 1)
shuffle(b)
shuffledLw = b[lw]
imshow(shuffledLw, origin="lower")
```

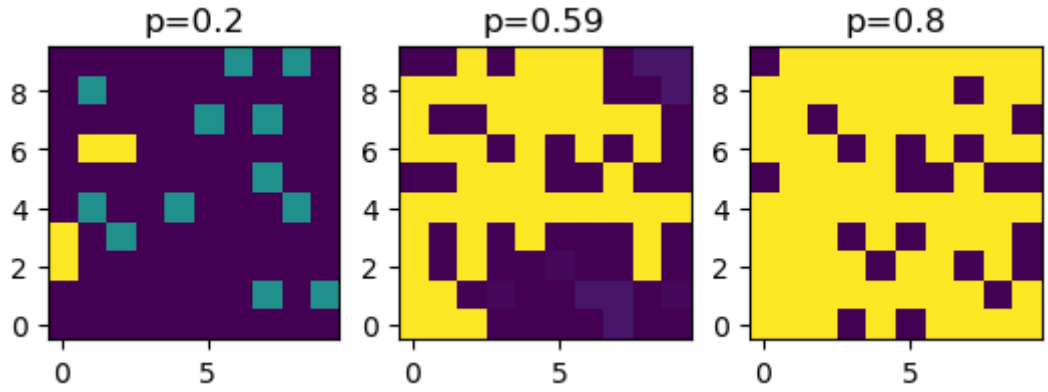
```
C:\Users\natha\AppData\Local\Temp\ipykernel_14924\2459514226.py:2: DeprecationWarn
ing: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.mea
surements` namespace is deprecated.
  lw, num = measurements.label(m)
```

Out[28]:
```
<matplotlib.image.AxesImage at 0x22dfac0ae80>
```



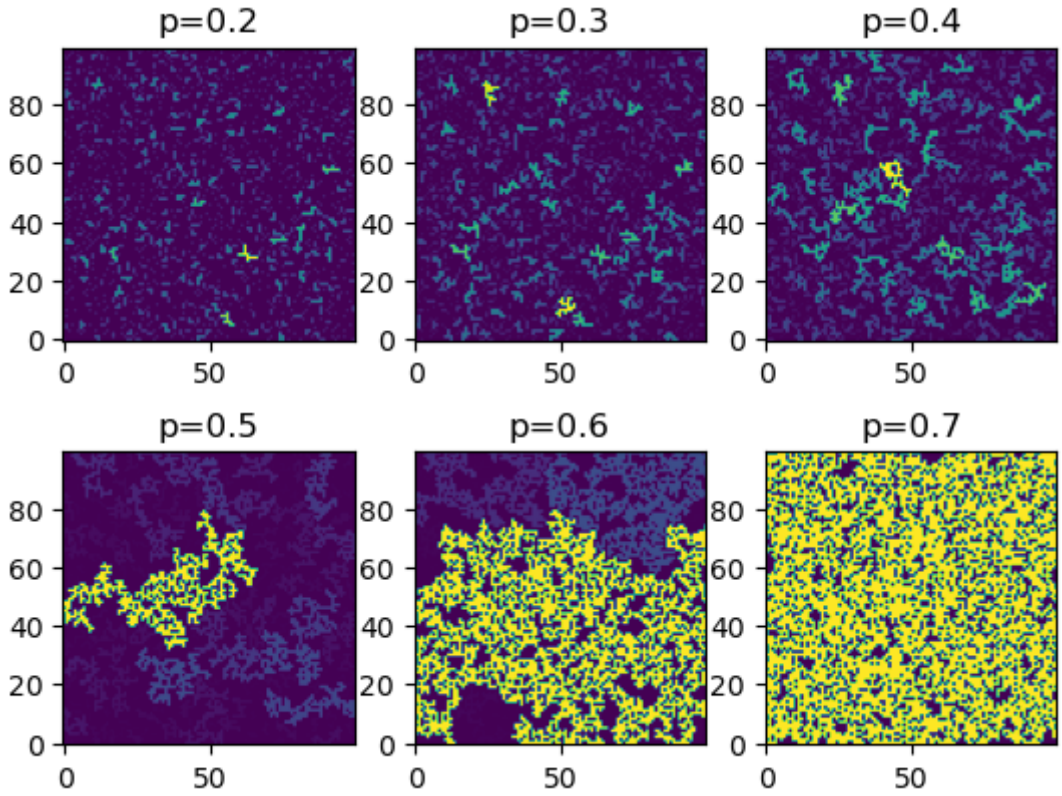

In [7]:
```
from pylab import *
from scipy.ndimage import measurements
L = 10
pv = [0.2,0.59,0.8]
z = rand(L,L)
for i in range(len(pv)):
    p = pv[i]
    m = z<p
    lw, num = measurements.label(m)
    area = measurements.sum(m, lw, index=arange(lw.max() + 1))
    areaImg = area[lw]
    subplot(1,3,i+1)
    tit = "p="+str(p)
    imshow(areaImg, origin="lower")
    title(tit)
    axis()
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_7528\1771305380.py:9: DeprecationWarni
ng: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.meas
urements` namespace is deprecated.
  lw, num = measurements.label(m)
C:\Users\natha\AppData\Local\Temp\ipykernel_7528\1771305380.py:10: DeprecationWarn
ing: Please use `sum` from the `scipy.ndimage` namespace, the `scipy.ndimage.measu
rements` namespace is deprecated.
  area = measurements.sum(m, lw, index=arange(lw.max() + 1))
```



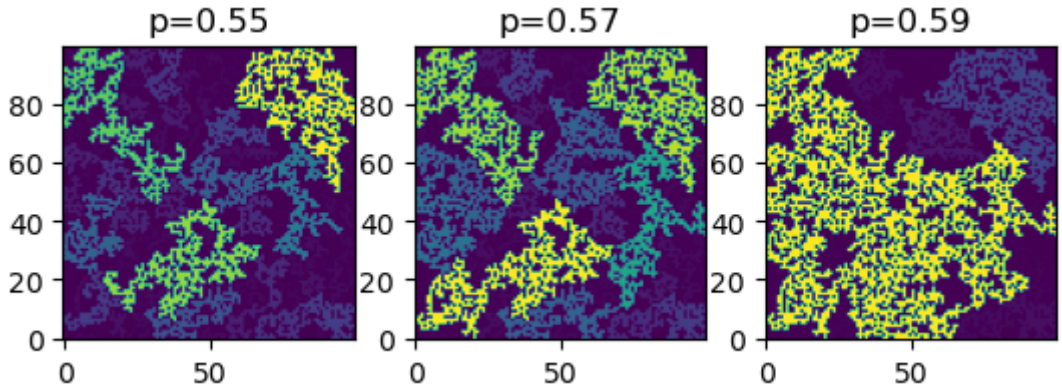

In [30]:
```
from pylab import *
from scipy.ndimage import measurements
L = 100
pv = [0.2,0.3,0.4,0.5,0.6,0.7]
z = rand(L,L)
for i in range(len(pv)):
    p = pv[i]
    m = z<p
    lw, num = measurements.label(m)
    area = measurements.sum(m, lw, index=arange(lw.max() + 1))
    areaImg = area[lw]
    subplot(2,3,i+1)
    tit = "p="+str(p)
    imshow(areaImg, origin="lower")
    title(tit)
    axis()
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_14924\3831240684.py:9: DeprecationWarn
ing: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.mea
surements` namespace is deprecated.
  lw, num = measurements.label(m)
C:\Users\natha\AppData\Local\Temp\ipykernel_14924\3831240684.py:10: DeprecationWar
ning: Please use `sum` from the `scipy.ndimage` namespace, the `scipy.ndimage.meas
urements` namespace is deprecated.
  area = measurements.sum(m, lw, index=arange(lw.max() + 1))
```




In [38]:
```
from pylab import * #examine the percolation critical point
from scipy.ndimage import measurements
L = 100
pv = [0.55,0.57,0.59]
z = rand(L,L)
for i in range(len(pv)):
    p = pv[i]
    m = z<p
    lw, num = measurements.label(m)
    area = measurements.sum(m, lw, index=arange(lw.max() + 1))
    areaImg = area[lw]
    subplot(2,3,i+1)
    tit = "p="+str(p)
    imshow(areaImg, origin="lower")
    title(tit)
    axis()
```

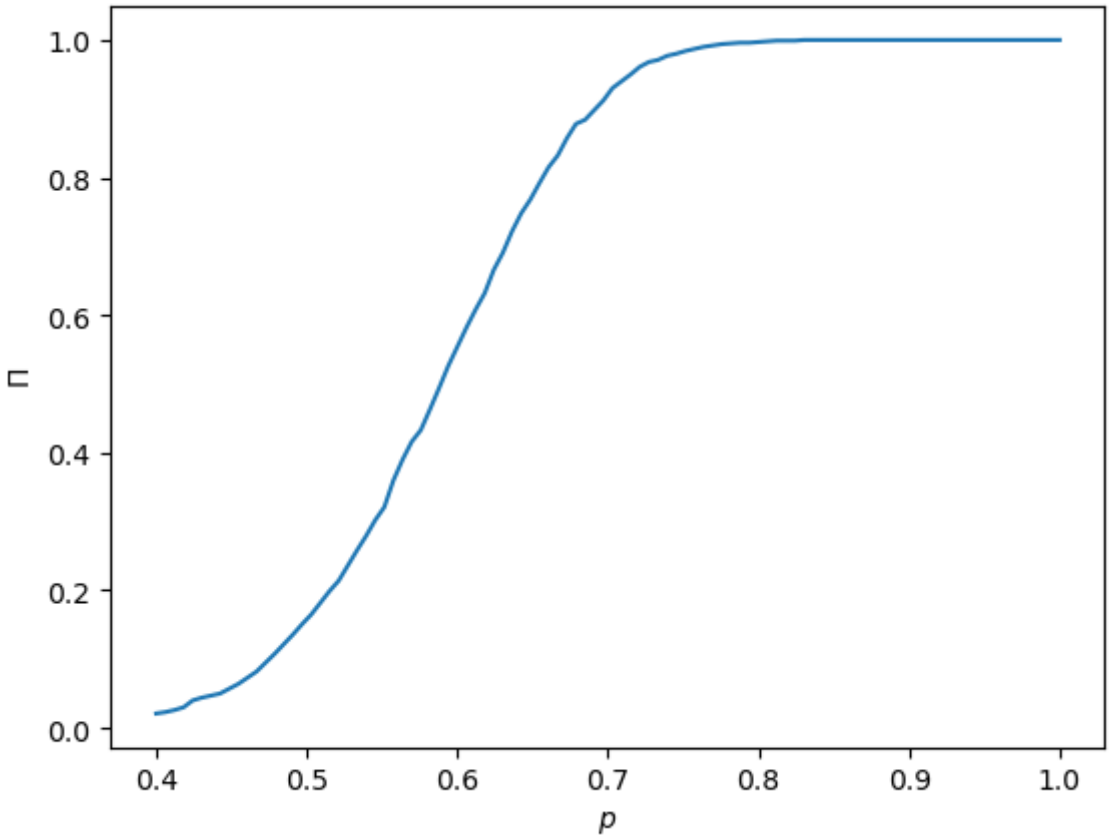
```
C:\Users\natha\AppData\Local\Temp\ipykernel_14924\97081088.py:9: DeprecationWarnin
g: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.measu
rements` namespace is deprecated.
  lw, num = measurements.label(m)
C:\Users\natha\AppData\Local\Temp\ipykernel_14924\97081088.py:10: DeprecationWarni
ng: Please use `sum` from the `scipy.ndimage` namespace, the `scipy.ndimage.measur
ements` namespace is deprecated.
  area = measurements.sum(m, lw, index=arange(lw.max() + 1))
```




In [44]:
```
from pylab import * #percolation probability for L= 10
from scipy.ndimage import measurements
p = linspace(0.4,1.0,100)
nx = len(p)
Ni = zeros(nx)
P = zeros(nx)
N = 1000
L = 10
for i in range(N):
    z = rand(L,L)
    for ip in range(nx):
        m = z<p[ip]
        lw, num = measurements.label(m)
        perc_x = intersect1d(lw[0,:],lw[-1,:])
        perc = perc_x[where(perc_x>0)]
        if (len(perc)>0):
            Ni[ip] = Ni[ip] + 1
Pi_2 = Ni/N
plot(p,Pi_2)
xlabel("$p$")
ylabel("$\Pi$")
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_18736\1872020846.py:13: DeprecationWar
ning: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.me
asurements` namespace is deprecated.
  lw, num = measurements.label(m)
```

Out[44]:
```
Text(0, 0.5, '$\\Pi$')
```

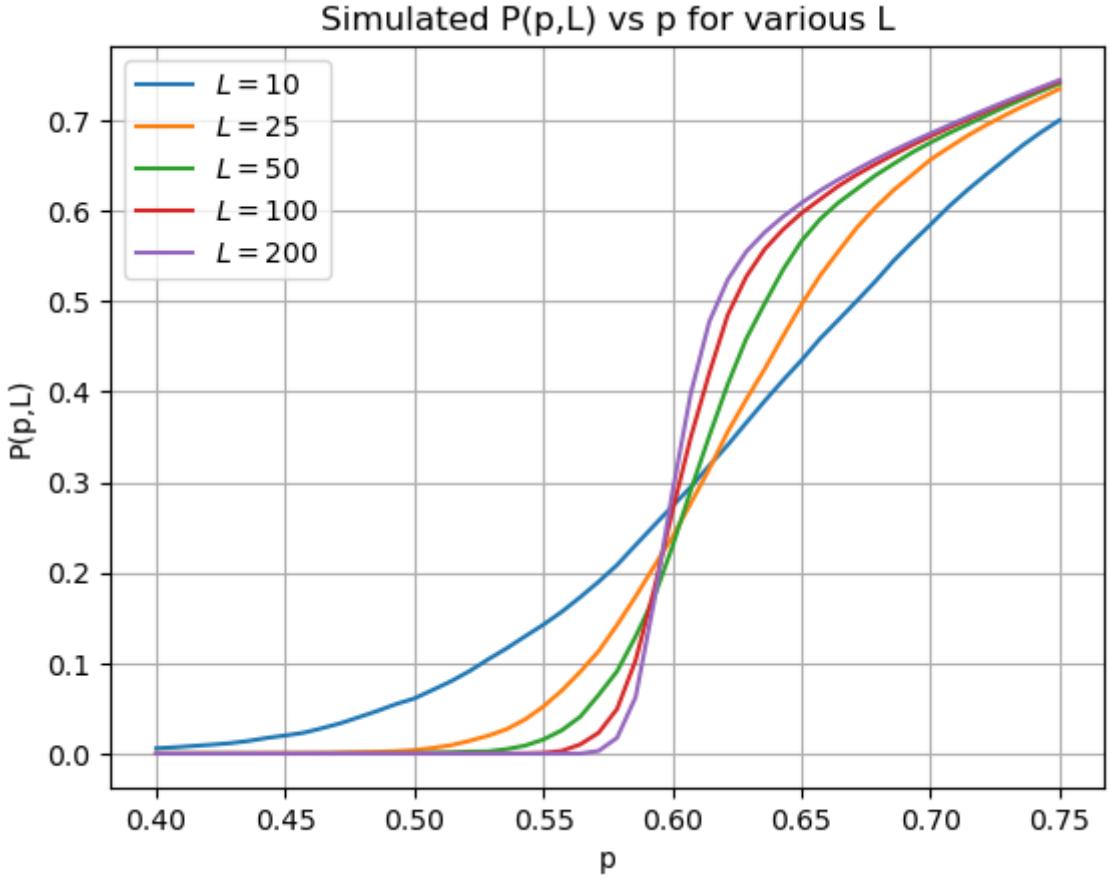



In [610…
```
perc_threshold = np.array([.56,.57,.59])
np.mean(perc_threshold)
```

Out[610]:
```
0.5733333333333333
```

In [94]:
```
from pylab import * #spanning cluster analysis
from scipy.ndimage import measurements
LL = [10,25,50,100,200]
p = linspace(0.4,0.75,50)
nL = len(LL)
nx = len(p)
P = zeros((nx,nL),float)
for iL in range(nL):
    L = LL[iL]
    N = int(2000*25/L)
    for i in range(N):
        z = rand(L,L)
        for ip in range(nx):
            m = z<p[ip]
            lw, num = measurements.label(m)
            perc_x = intersect1d(lw[0,:],lw[-1,:])
            perc = perc_x[where(perc_x>0)]
            if (len(perc)>0):
                area = measurements.sum(m, lw, perc[0])
                P[ip,iL] = P[ip,iL] + area
    P[:,iL] = P[:,iL]/((L*L)*N)
for iL in range(nL):
    L = LL[iL]
    lab = "$L="+str(L)+"$"
    plot(p,P[:,iL],label=lab)
    ylabel("P(p,L)")
    xlabel("p")
    legend()
    plt.grid(True)
    title("Simulated P(p,L) vs p for various L")
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_16528\3707868626.py:15: DeprecationWar
ning: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.me
asurements` namespace is deprecated.
  lw, num = measurements.label(m)
C:\Users\natha\AppData\Local\Temp\ipykernel_16528\3707868626.py:19: DeprecationWar
ning: Please use `sum` from the `scipy.ndimage` namespace, the `scipy.ndimage.meas
urements` namespace is deprecated.
  area = measurements.sum(m, lw, perc[0])
```

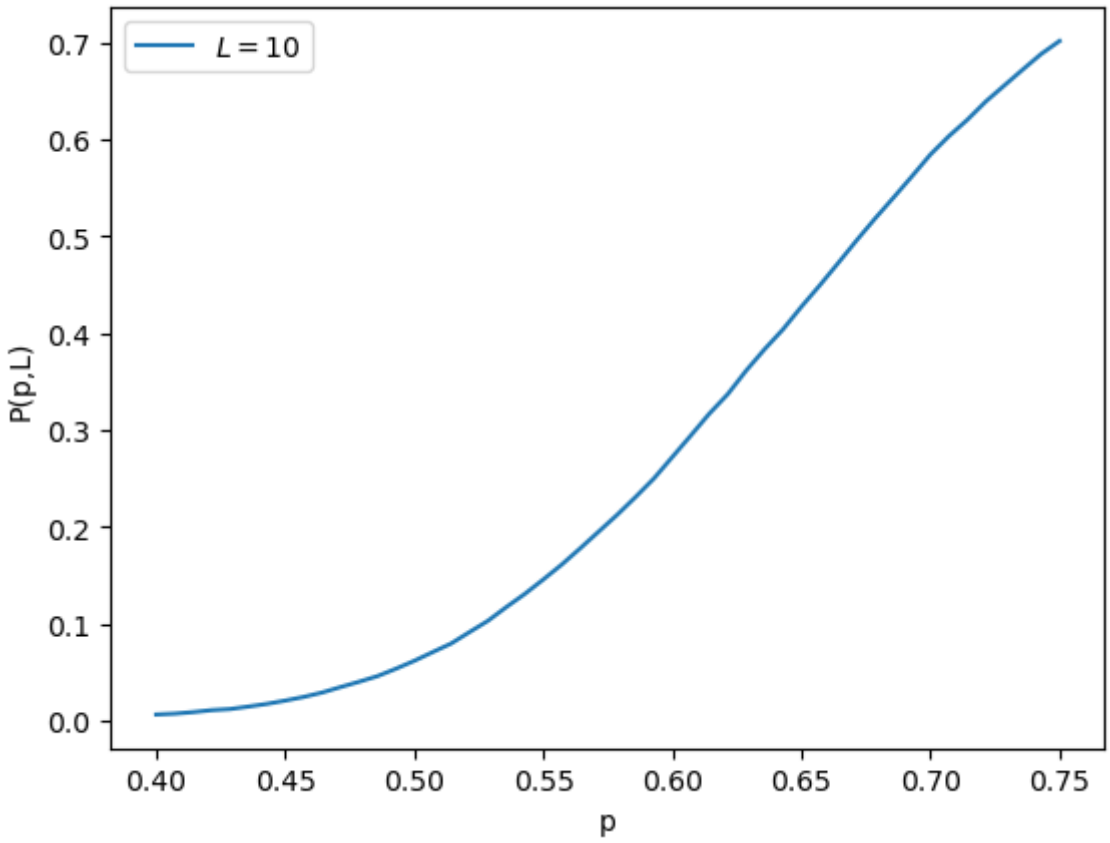



In [6]:

```
from pylab import * #spanning cluster analysis for L=10
from scipy.ndimage import measurements
LL = [10]
p = linspace(0.4,0.75,50)
nL = len(LL)
nx = len(p)
P = zeros((nx,nL),float)
for iL in range(nL):
    L = LL[iL]
    N = int(2000*25/L)
    for i in range(N):
        z = rand(L,L)
        for ip in range(nx):
            m = z<p[ip]
            lw, num = measurements.label(m)
            perc_x = intersect1d(lw[0,:],lw[-1,:])
            perc = perc_x[where(perc_x>0)]
            if (len(perc)>0):
                area = measurements.sum(m, lw, perc[0])
                P[ip,iL] = P[ip,iL] + area
    Pi3 = P[:,iL]/((L*L)*N)
for iL in range(nL):
    L = LL[iL]
    lab = "$L="+str(L)+"$"
    plot(p,Pi3,label=lab)
    ylabel("P(p,L)")
```

```
    xlabel("p")
    legend()
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_16528\491993534.py:15: DeprecationWarn
ing: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.mea
surements` namespace is deprecated.
  lw, num = measurements.label(m)
C:\Users\natha\AppData\Local\Temp\ipykernel_16528\491993534.py:19: DeprecationWarn
ing: Please use `sum` from the `scipy.ndimage` namespace, the `scipy.ndimage.measu
rements` namespace is deprecated.
  area = measurements.sum(m, lw, perc[0])
```

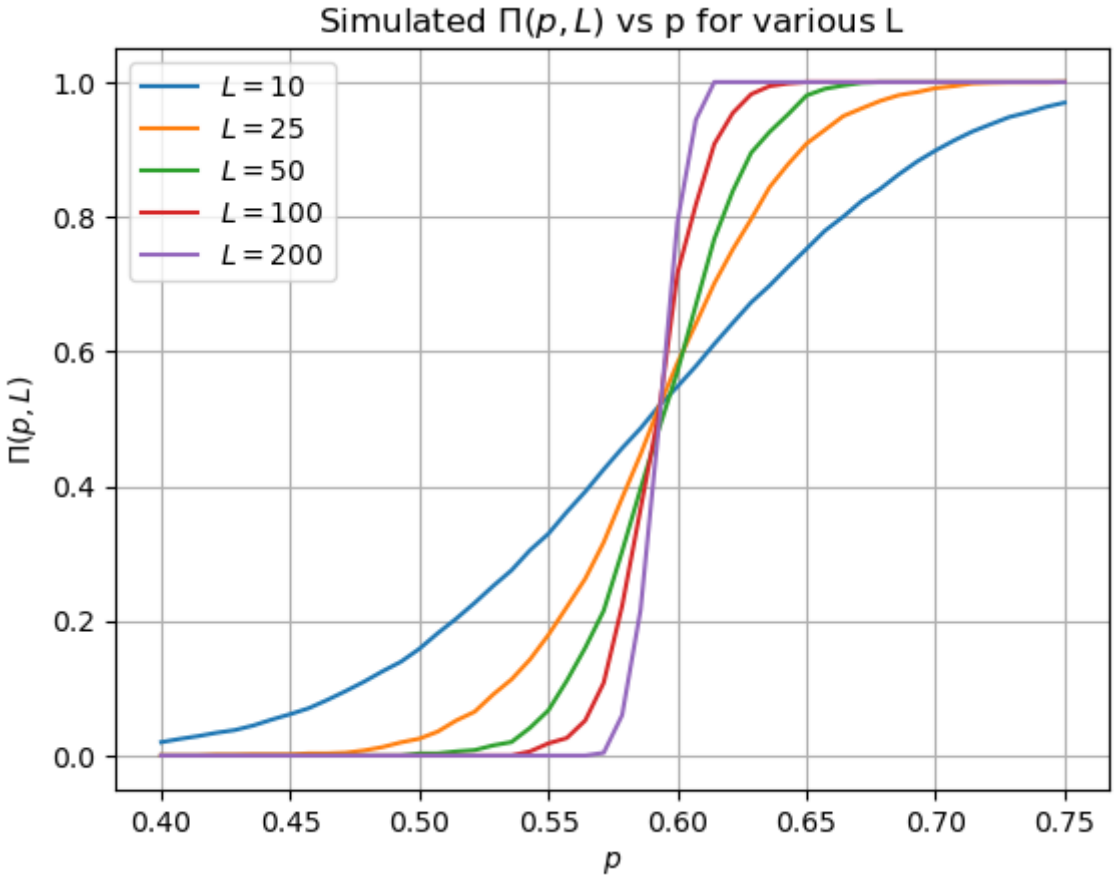



In [42]:

```
from pylab import * #percolation probabilty analysis
from scipy.ndimage import measurements
LL = [10,25,50,100,200]
p = linspace(0.4,0.75,50)
nL = len(LL)
nx = len(p)
Ni = zeros((nx,nL),float)
Pi = zeros((nx,nL),float)
for iL in range(nL):
    L = LL[iL]
    N = int(2000*25/L)
    for i in range(N):
        z = rand(L,L)
        for ip in range(nx):
            m = z<p[ip]
            lw, num = measurements.label(m)
            perc_x = intersect1d(lw[0,:],lw[-1,:])
            perc = perc_x[where(perc_x>0)]
            if (len(perc)>0):
                Ni[ip,iL] = Ni[ip,iL] + 1
    Pi[:,iL] = Ni[:,iL]/N
for iL in range(nL):
    L = LL[iL]
    lab = "$L="+str(L)+"$"
    plot(p,Pi[:,iL],label=lab)
```

```
    ylabel("$\Pi(p,L)$")
    xlabel("$p$")
    legend()
    grid(True)
    title("Simulated $\Pi(p,L)$ vs p for various L")
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_18736\245346442.py:16: DeprecationWarn
ing: Please use `label` from the `scipy.ndimage` namespace, the `scipy.ndimage.mea
surements` namespace is deprecated.
  lw, num = measurements.label(m)
```




In [102…
```
prob1 = np.array([0.52,0.51,0.54,0.57,0.716,0.721,0.726,0.734,0.8769,0.8777,0.9253
prob2 = np.array([0.616,0.733,0.783,0.85,0.95,0.966,0.966,1,1,1,1,1])
prob3 = np.array([0.55,0.65,0.75,0.74,0.725,0.8538,.95,1,1,1,1])
avg_prob = np.array([0.562,0.641,0.717666667,0.768666667,0.798666667,0.881933333,0
#data sets for percolation proability
```

In [104…
```
def funcB(p,A, B): #power law function
    return A*np.power(p-.57,B)
popt, pcov = curve_fit(funcB,np.linspace(0.57,.87,30),prob1)
popt,pcov #1.411+-0.044
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_18736\806346792.py:2: RuntimeWarning:
divide by zero encountered in power
  return A*np.power(p-.57,B)
```

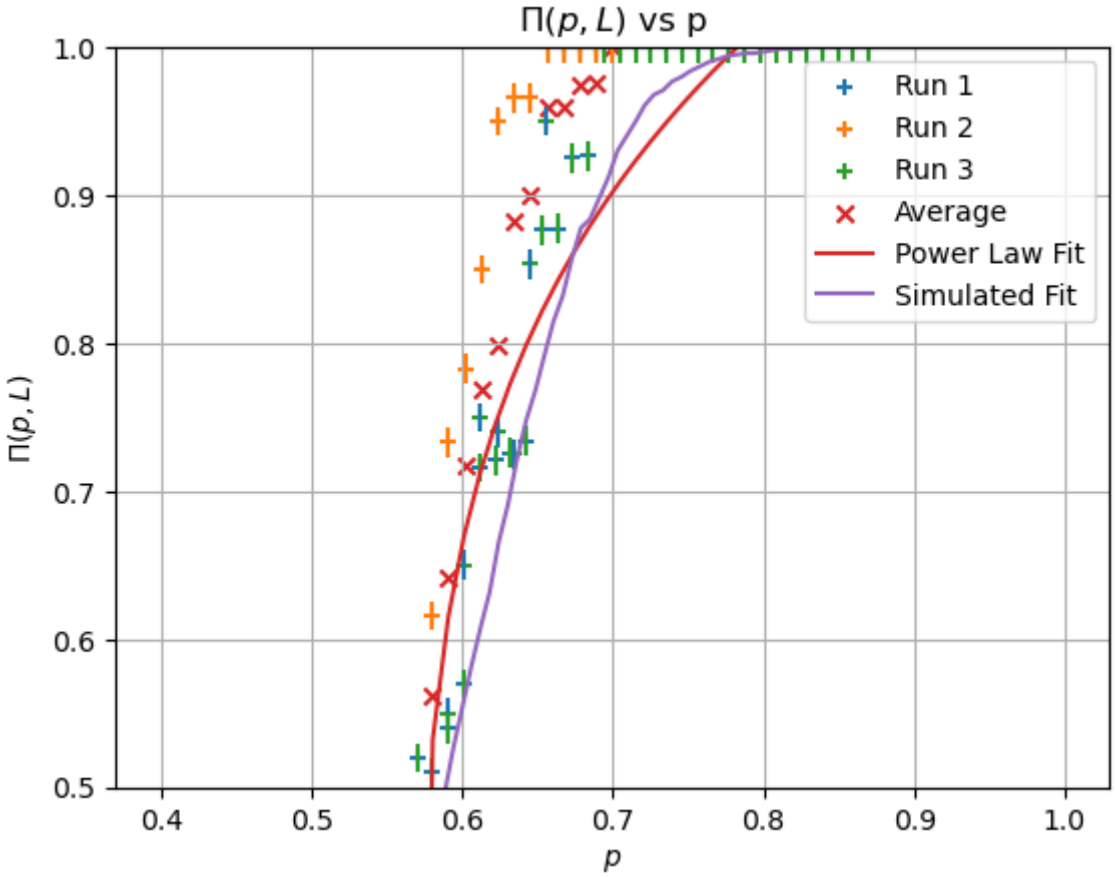
Out[104]:
```
(array([1.38177067, 0.2087422 ]),
 array([[0.00907136, 0.00310202],
        [0.00310202, 0.0011941 ]]))
```

In [49]:
```
def funcv(p,A, B): #power law function
    return A*np.power(p-.58,B)
popt, pcov = curve_fit(funcv,np.linspace(0.58,.7,12),prob2)
popt,pcov
```

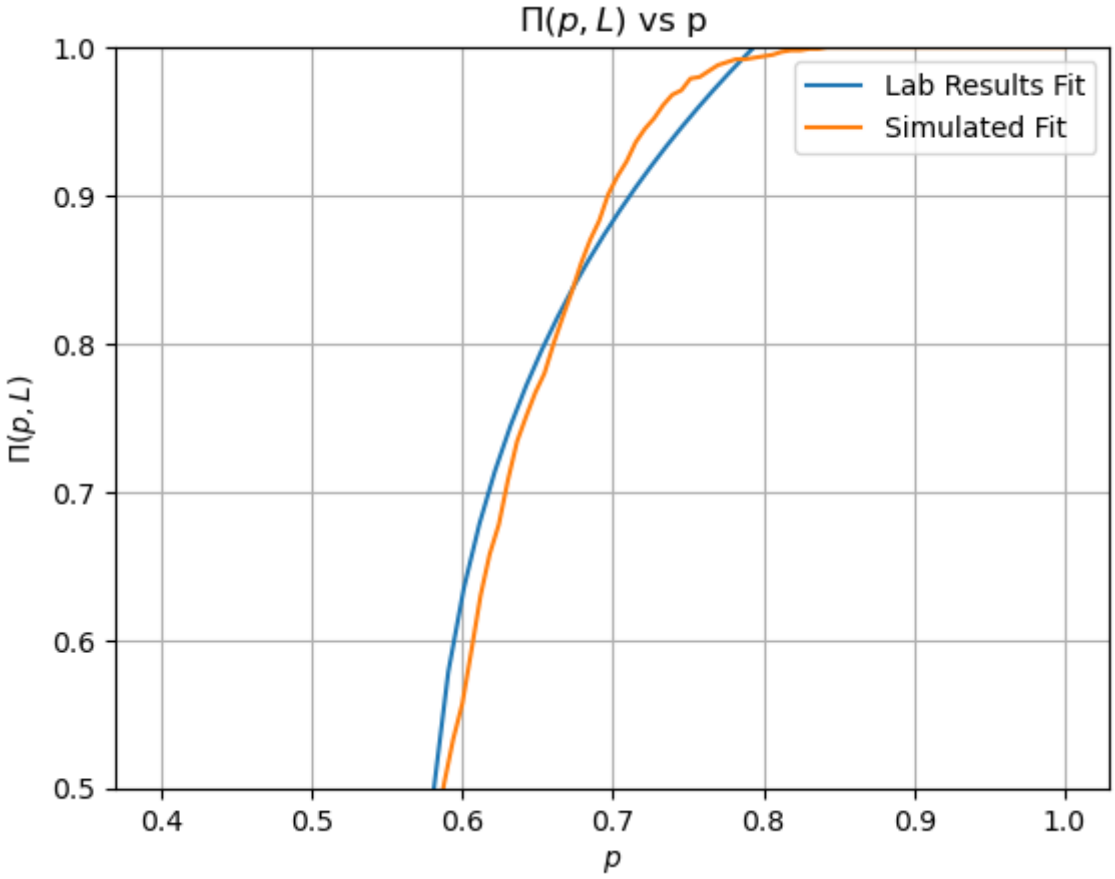
```
C:\Users\natha\AppData\Local\Temp\ipykernel_18736\2293379705.py:2: RuntimeWarning:
divide by zero encountered in power
  return A*np.power(p-.58,B)
```

Out[49]:
```
(array([1.39076811, 0.13846035]),
 array([[0.15485108, 0.03776871],
        [0.03776871, 0.00969949]]))
```

In [114…
```
plt.scatter(np.linspace(0.57,.87,30),prob1,marker="+")
plt.scatter(np.linspace(0.58,.7,12),prob2,marker="+")
plt.scatter(np.linspace(0.59,.7,11),prob3,marker="+")
plt.errorbar(np.linspace(0.59,.7,11),prob3,yerr=0.01,ls="none")
plt.errorbar(np.linspace(0.58,.7,12),prob2,yerr=0.01,ls="none")
plt.errorbar(np.linspace(0.57,.87,30),prob1,yerr=0.01,ls="none")
plt.scatter(np.linspace(0.58,.7,12),avg_prob,marker="x")
plt.plot(np.linspace(0.57,.87,30),funcB(np.linspace(0.57,.87,30),1.38177067, 0.208
plt.plot(p,Pi_2)
plt.ylim(0.5,1)
plt.grid(True)
plt.ylabel("$\Pi(p,L)$")
plt.xlabel("$p$")
plt.title("$\Pi(p,L)$ vs p")
plt.legend(["Run 1","Run 2", "Run 3","Average","Power Law Fit", "Simulated Fit"])
plt.grid(True)
```



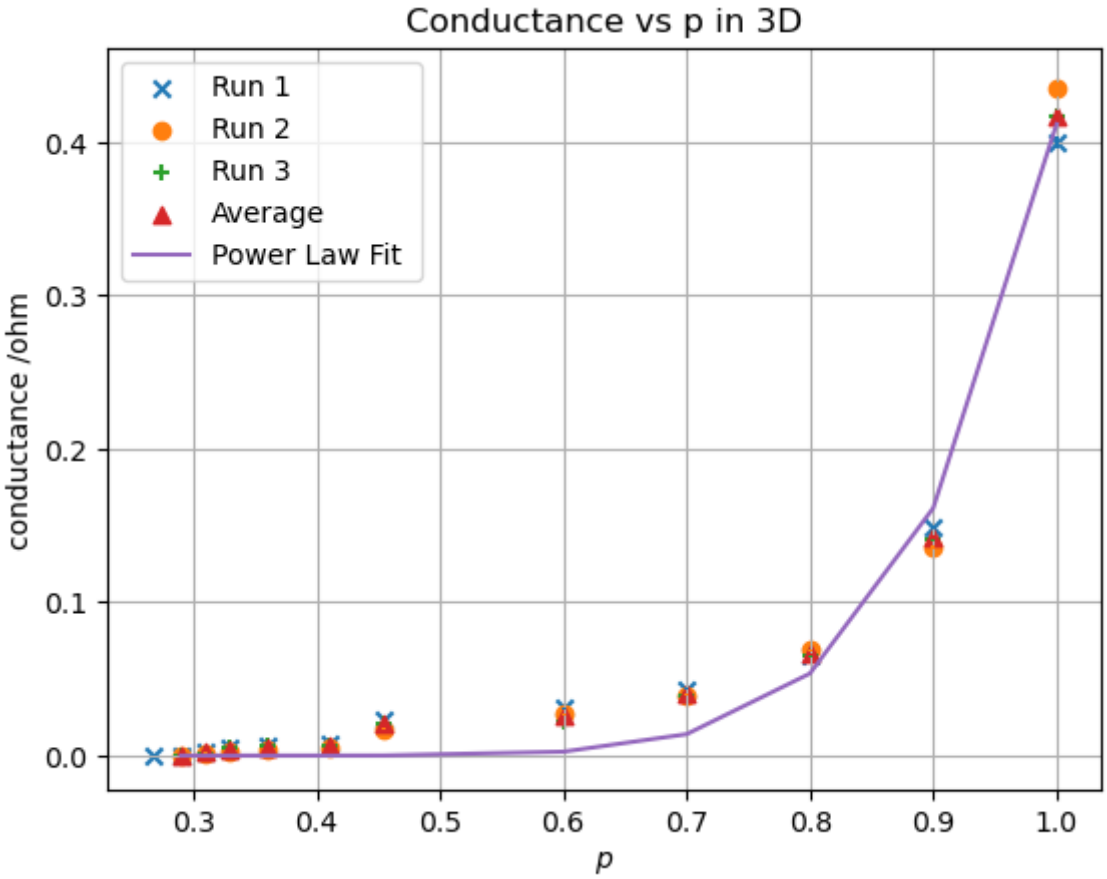

In [87]:
```
#plt.scatter(np.linspace(0.57,.87,30)-.57,prob1,marker="+")
plt.plot(np.linspace(0.57,.87,30),funcB(np.linspace(0.57,.87,30),1.41147855, 0.229
plot(p,Pi_2)
plt.ylim(0.5,1)
plt.grid(True)
plt.ylabel("$\Pi(p,L)$")
plt.xlabel("$p$")
plt.title("$\Pi(p,L)$ vs p")
plt.legend(["Lab Results Fit","Simulated Fit"])
plt.grid(True)
```




# 3D percolation

In [30]:
```
p_3d = np.array([1,.9,.8,.7,.6,.455,.41,.36,.33,.31,.29,]) #3D data sets
p_3d2 = np.array([1,.9,.8,.7,.6,.455,.41,.36,.33,.31,.29,.2675,])
run1_3d = np.array([2.5,6.7,15.475,23,32.4,43,129.1,154.3,200,475,34.4e06])
run1_3d2 = np.array([2.5,6.7,15.475,23,32.4,43,129.1,154.3,200,475,34.4e06,83.2e06
run2_3d = np.array([2.3,7.34,14.5,25.4,37.2,57.45,186.3,245,455,540,52.3e06])
run3_3d = np.array([2.4,7.1,15.4,25.5,46.1,48.23,152.3,170,200, 550, 31.24e06])
avg = (run1_3d + run2_3d + run3_3d)/3
```

In [35]:
```
plt.scatter(p_3d2,1/run1_3d2,marker="x")
plt.scatter(p_3d,1/run2_3d,marker="o")
plt.scatter(p_3d,1/run3_3d,marker="+")
plt.scatter(p_3d,1/avg,marker="^")
plt.errorbar(p_3d2,1/run1_3d2,yerr = 1/run1_3d2/100, ls = "none")
plt.errorbar(p_3d,1/run2_3d,yerr = 1/run2_3d/100, ls = "none")
plt.errorbar(p_3d,1/run3_3d,yerr = 1/run3_3d/100, ls = "none")
plt.errorbar(p_3d,1/avg,yerr = 1/avg/100, ls = "none")
plt.plot(p_3d,funcV(p_3d,3.40574211, 6.16916401))
plt.ylabel("conductance /ohm")
plt.xlabel("$p$")
plt.title("Conductance vs p in 3D")
plt.legend(["Run 1","Run 2", "Run 3","Average","Power Law Fit "])
plt.grid(True)
```




In [34]:
```
def funcV(P,A,V):
    return A*(P-.29)**V
popt, pcov = curve_fit(funcV,p_3d,1/avg)
popt,pcov
```

Out[34]:
```
(array([3.40574211, 6.16916401]),
 array([[0.49002355, 0.37929013],
        [0.37929013, 0.30286519]]))
```

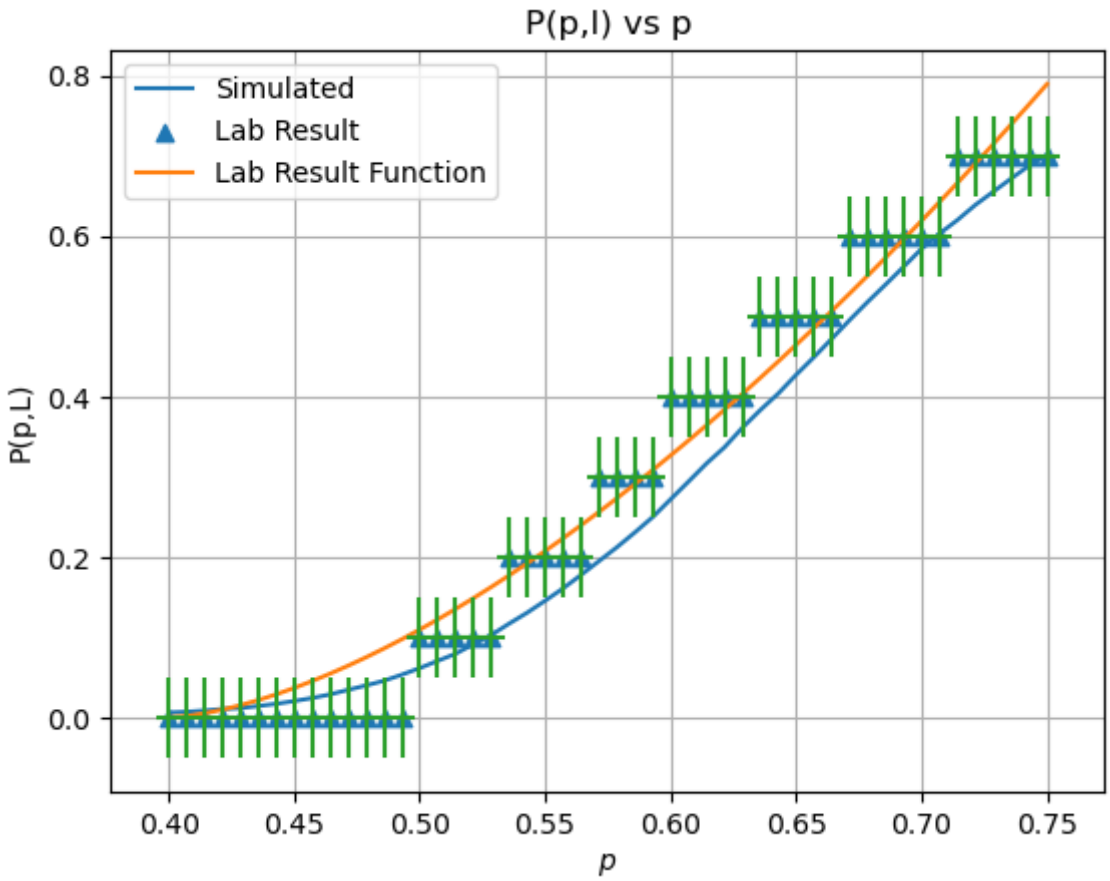
# Spanning Cluster Density for 2D

In [27]:
```
def funcP(P,A,V):
    return A*(P-.4)**V
popt, pcov = curve_fit(funcP,p,pi3_/10)
popt,pcov
```

Out[27]:
```
(array([4.13437894, 1.57718945]),
 array([[0.12848028, 0.02344185],
        [0.02344185, 0.00442813]]))
```

In [60]:
```
plt.plot(p,Pi3)
plt.scatter(p,pi3_/10,marker="^")
plt.plot(p,funcP(p,4.13437894, 1.57718945))
plt.errorbar(p,pi3_/10,yerr = 0.05,xerr=0.005 ,ls = "none")
plt.grid(True)
plt.legend(["Simulated","Lab Result", "Lab Result Function"])
plt.ylabel("P(p,L)")
plt.xlabel("$p$")
plt.title("P(p,l) vs p")
```

Out[60]:
```
Text(0.5, 1.0, 'P(p,l) vs p')
```




In [17]:
```
pi3_= np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3,3,3,3,4,4,4,4,4,
len(pi3_)
```

Out[17]:
```
50
```