SHARIF UNIVERSITY OF TECHNOLOGY
PHYSICS DEPARTMENT

Problem Set 4

Student name: *Ali Fayyaz - 98100967*

Course: *Computational Physcis - (Spring 2023)*
Due date: *March 10, 2023*

Exercise 4.1

Percolation:

(*a*) Create an $L \times L$ 2D lattice for percolation.

(*b*) Turn the sites on with probability p.

(*c*) Write a program to check if a lattice is percolating; i.e. there is at least one spanning cluster connecting the left side of the lattice to the right side.

Answer. A random matrix with boolean elements was created. Two functions were defined as follows:

- **rand_lattice():**

  A basic function that takes as input the integer L for the side length of a lattice, and the float p. It returns a 2D numpy array of booleans, with shape (L,L), the elements of which will be *True* with probabilty p and *False* with probability $(1 - p)$.

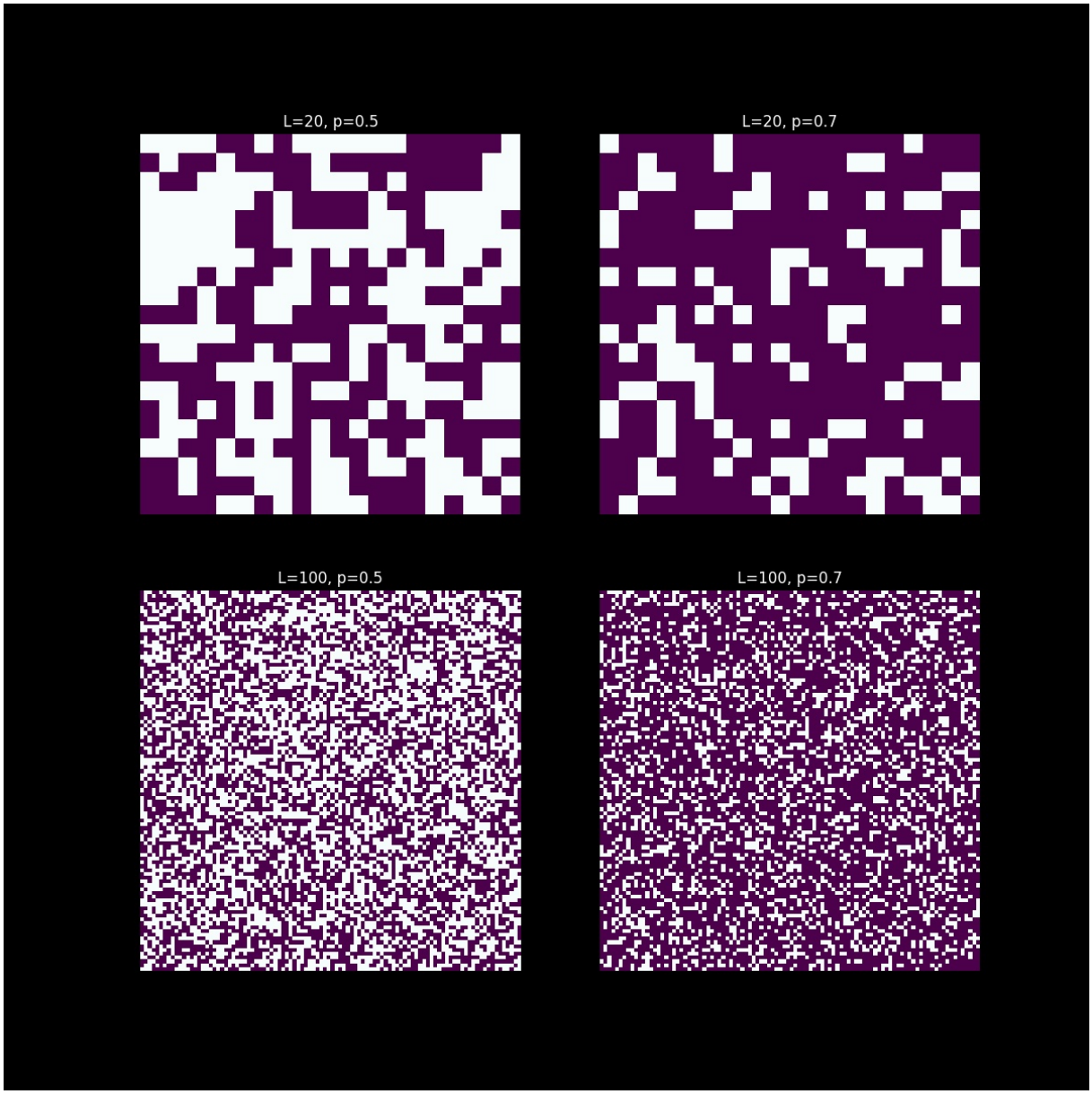
- **is_percolating():**

  Takes a numpy array of booleans and returns 1 if it is percolating, and 0 if it is not. It utilizes *scipy.ndimage.measurements.label()* to label the lattice; this *scipy* mehtod finds and distinguishes different clusters automatically and names them with integers: the *False* elements are denoted with 0, and the elements that belong to a cluster are assigned the same number, starting from 1. After that, *numpy.intersect1d()* is used to find the numbers in common between the left column of the matrix and the right column; if any exist, it means the lattice is percolating sideways.

Exercise 4.2

Demonstrate the lattice graphically.

Answer. The function *rand_lattice()* was used for two different lattice sizes and two different probabilities. *matplotlib.pyplot.imshow()* was used to represent the matrices graphically.



Exercise 4.3

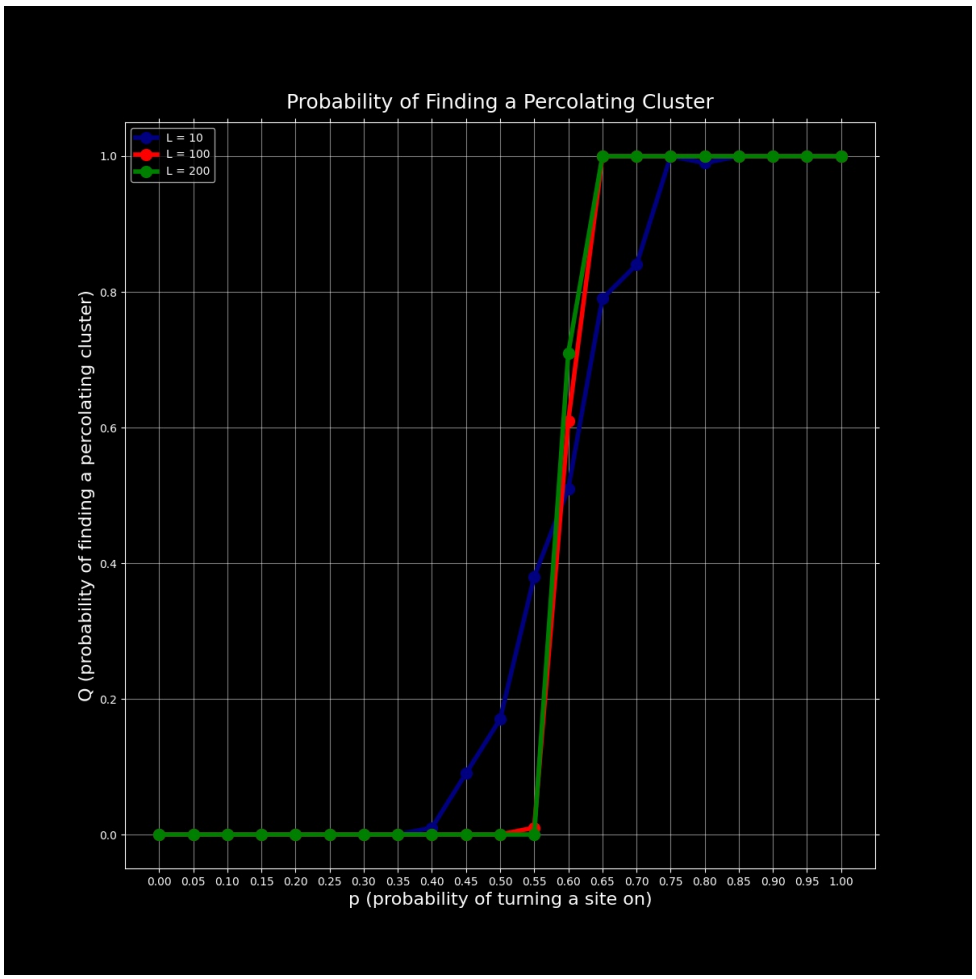
Probability Q of the existance of a spanning cluster:

(*a*) Take $L = 10$. For each value of p in the interval $0 \leq p \leq 1$ with steps $\Delta p = 0.05$, write a program to create 100 lattices and find the number of times it is percolating. Devide this number to 100 to find the probabilty of percolation, Q, for each value of p.

(*b*) Do the same for $L = 100$ and $L = 200$.

(*c*) Plot Q vs. p for each value of L.

Answer. The above mentioned functions were used to generate the data. The plot is as follows:



Exercise 4.4

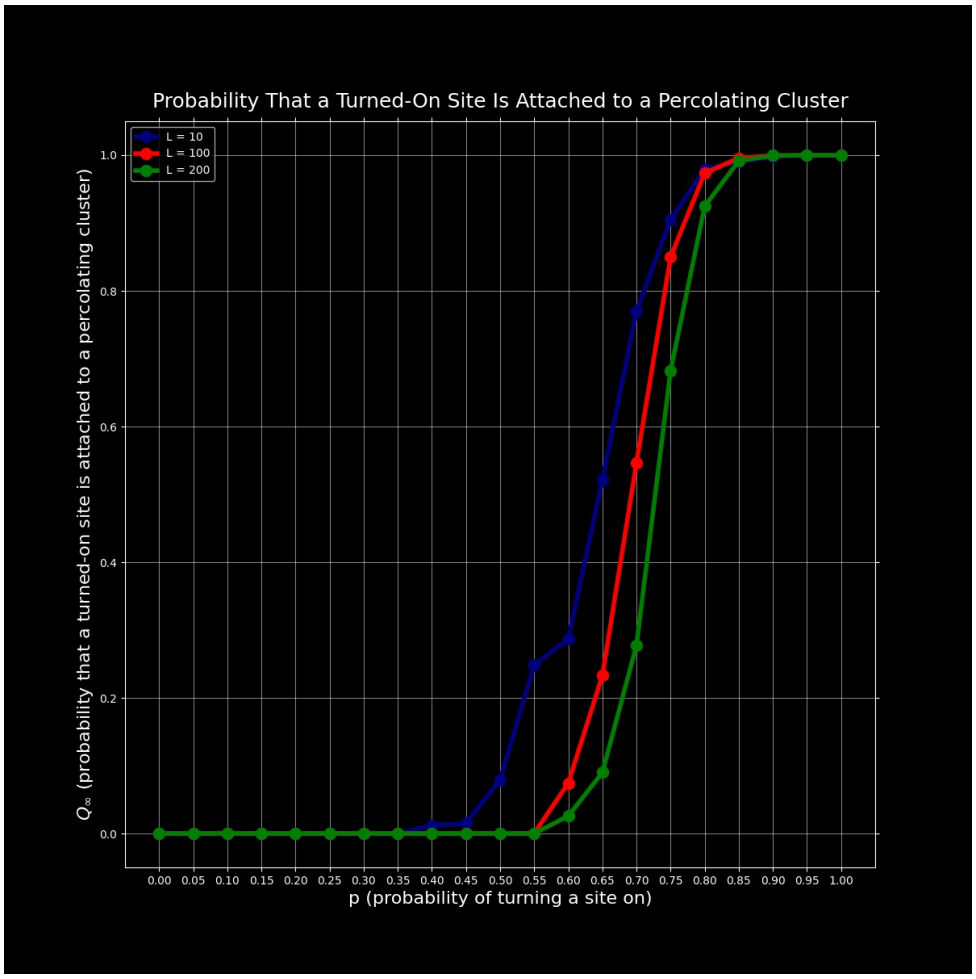
Probability Q_∞ of a Turned-On Site to Belong to a Spanning Cluster:

(*a*) Use your codes from the previous exercise, and adjust them to find Q_∞ in each run.

(*b*) Do this for $L = 10$, $L = 100$ and $L = 200$.

(*c*) Plot Q_∞ vs. p for each value of L.

Answer. The function *is_percolating()* was called 100 times on different values of L and for each sample, a counter recorded the number of times the lattice was percolating. The resulting plot is as follows:



Exercise 4.5

Correlation Length ξ:

(*a*) Use your codes from the previous exercises, and adjust them to find ξ in each run.

(*b*) Do this for $L = 10, 20, 40, 80, 160$.

(*c*) Plot ξ vs. p for each value of L.

(*d*) Run the program again around the critical point for smaller steps of Δp to increase the precision.

(*e*) The peak of ξ shows the critical probability p_c. Given that p_c is a function of L, can you find $p_c(\infty)$?

Answer. The radius of gyration was first calculated using the following function. For different values of $0 \leq p \leq 1$ and different values of L, 100 sample percolation lattices were created and the function was called to calculate the radius of gyration for each sample. Then the average of these 100 samples were taken as the R_g for a given L and given p.

The plots almost unanimously peaked at $p = 0.55$, indicating that $p_c(\infty)$ must be

near that point.

For $L \in \{10, 20, 100\}$ more verbose iterations were carried out around p_c and more precise plots were graphed.

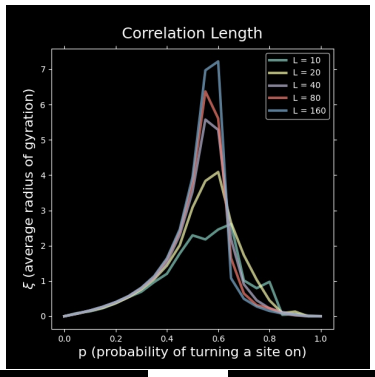
- **rgyration():**

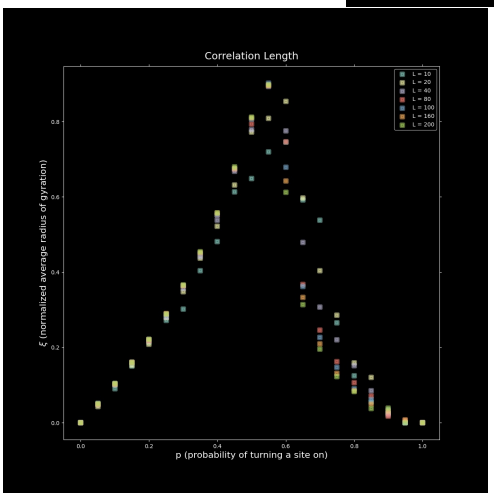
  This function takes a matrix of boolean elements. It uses *scipy.ndimage.measurements.center_of_mass()* to find the center of mass of all the clusters (including the spanning clusters if they exist) and then *scipy.ndimage.measurements.sum()* is used to find the area of each of them. Finally, *math.dist()* is called to find the distance of each site of each cluster with the CM of that particular cluster. The normalized radius of gyration is calculated using the following formula:

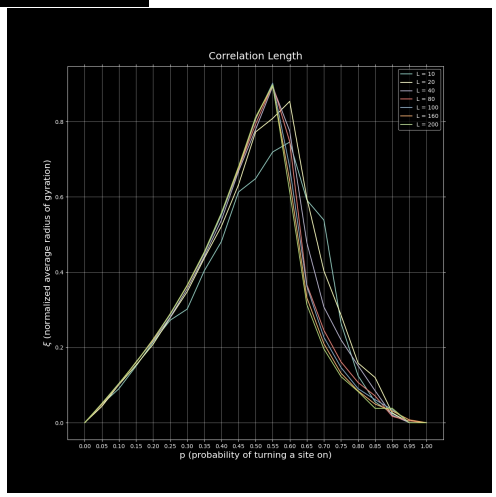
  $$R_g^2 = \frac{1}{s}\sum(\boldsymbol{r_i} - \boldsymbol{r_{cm}})^2$$

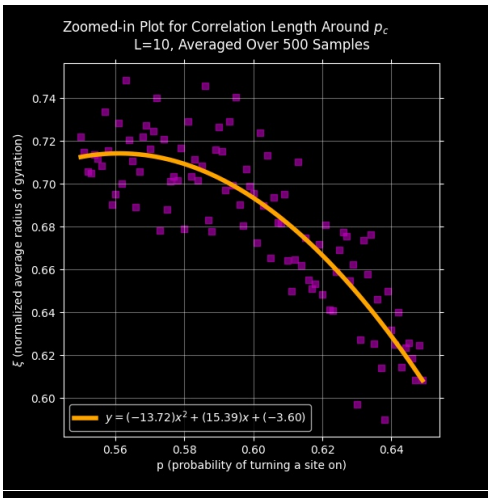
  where R_g is the radius of gyration of one particular cluster, s is its surface area, $\boldsymbol{r_i}$ is the position of a site on the cluster, and $\boldsymbol{r_{cm}}$ is the position of its center of mass. It is normalized because the sum is devided by the cluster area.

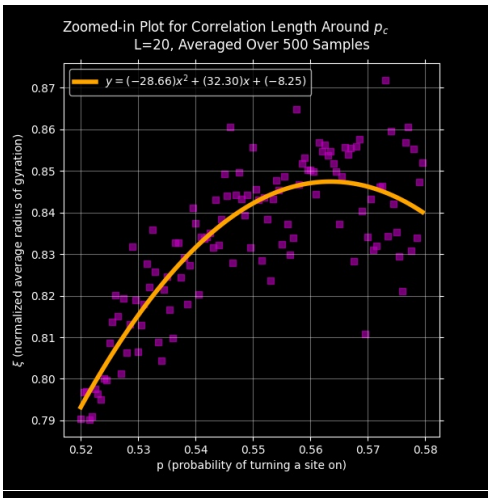
  This way, the radii of gyration of all the clusters are calculated and put into an array. As a last step, spanning clusters are identified and the corresponding radii of gyration in the array are replaced with *None*. The function returns the radii of gyration of all the non-spanning clusters.

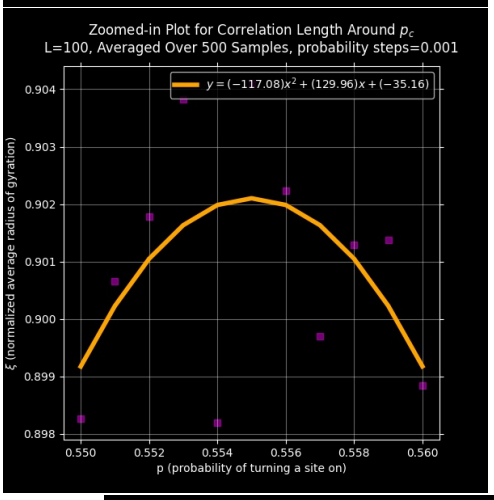


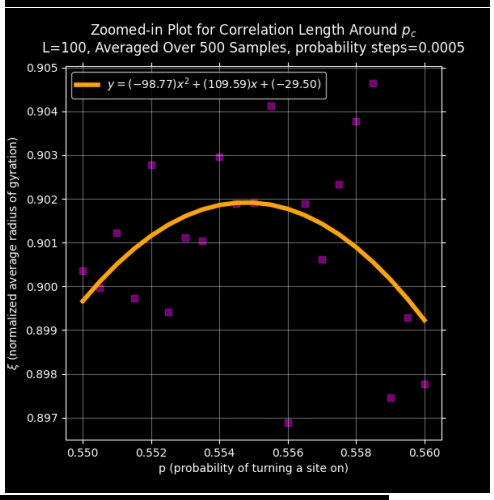


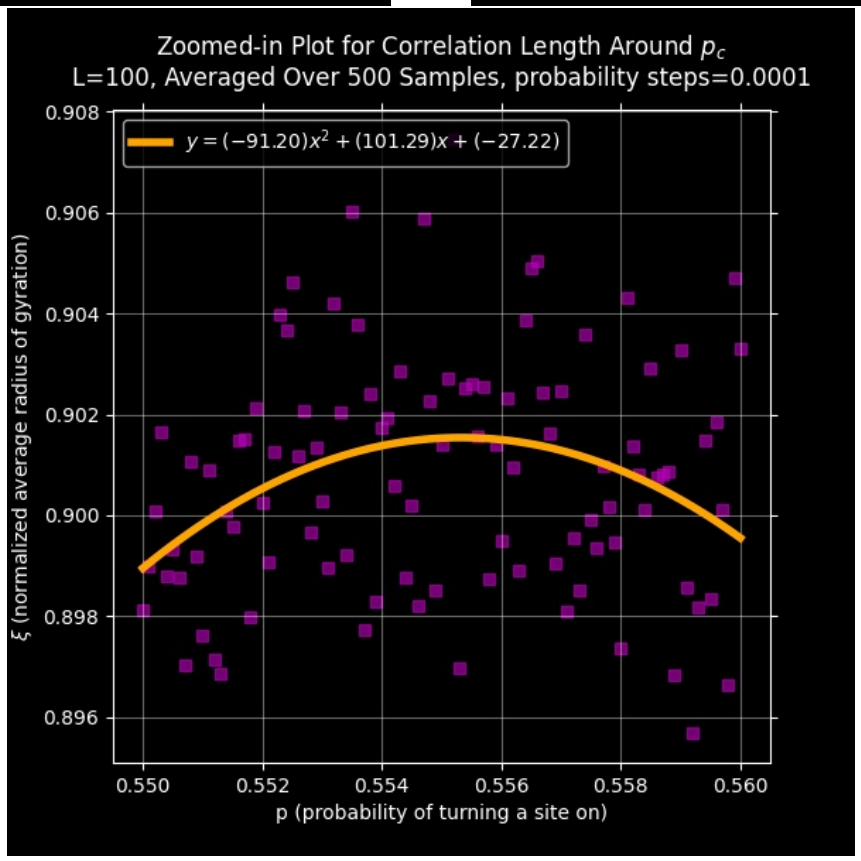



Zoomed-in Plot for Correlation Length Around p_c
L=10, Averaged Over 500 Samples
$y = (-13.72)x^2 + (15.39)x + (-3.60)$
ξ (normalized average radius of gyration)
p (probability of turning a site on)


Zoomed-in Plot for Correlation Length Around p_c
L=20, Averaged Over 500 Samples
$y = (-28.66)x^2 + (32.30)x + (-8.25)$
ξ (normalized average radius of gyration)
p (probability of turning a site on)


Zoomed-in Plot for Correlation Length Around p_c
L=100, Averaged Over 500 Samples, probability steps=0.001
$y = (-117.08)x^2 + (129.96)x + (-35.16)$
ξ (normalized average radius of gyration)
p (probability of turning a site on)

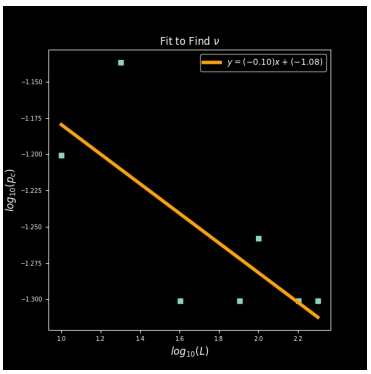

Zoomed-in Plot for Correlation Length Around p_c
L=100, Averaged Over 500 Samples, probability steps=0.0005
$y = (-98.77)x^2 + (109.59)x + (-29.50)$
ξ (normalized average radius of gyration)
p (probability of turning a site on)


Zoomed-in Plot for Correlation Length Around p_c
L=100, Averaged Over 500 Samples, probability steps=0.0001
$y = (-91.20)x^2 + (101.29)x + (-27.22)$
ξ (normalized average radius of gyration)
p (probability of turning a site on)

Exercise 4.6

According to the textbook, given $p_c(\infty) = \frac{1}{2}$, find ν.

Answer. Given that the data obtained from the previous section showed $p_c \simeq 0.55$ for almost all the values of L tested, a terrible fit was inevitable. So the data gathered from verbose analysis was used for $L \in \{10, 20, 100\}$. The fit was better but still very poor. From the data and the fit, ν was calculated to be 10; which is almost 10 times the correct value of $\frac{4}{3}$.



Exercise 4.7

Fractal Dimension of Percolation Clusters:

(*a*) For $p \in \{0.5, 0.55, 0.59\}$ generate lattices, find ξ and s, the surface area of the clusters.

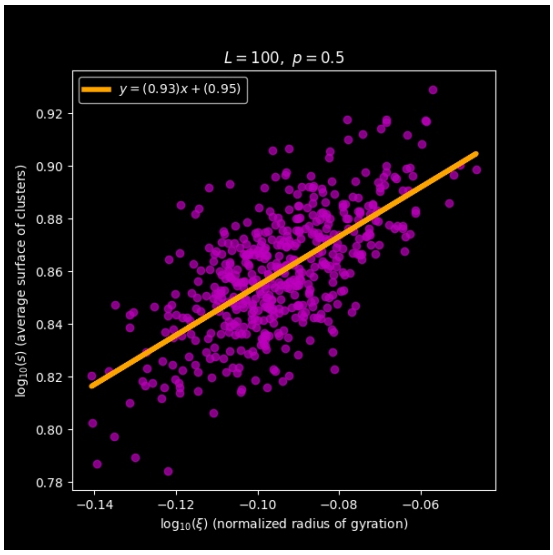
(*b*) Plot $\log(s)$ vs. $\log(\xi)$ for these lattices.

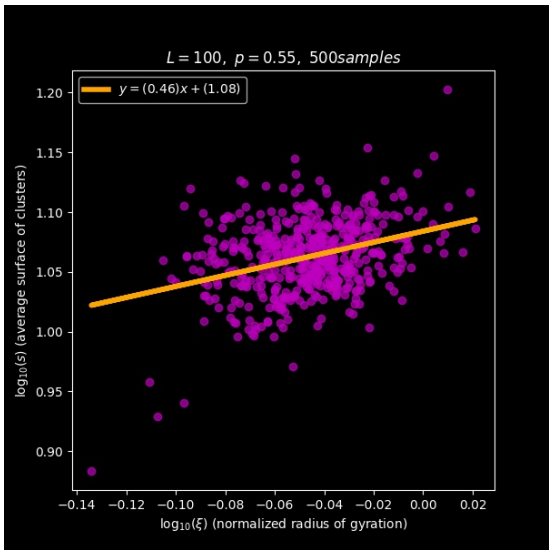
(*c*) Are the plots linear?

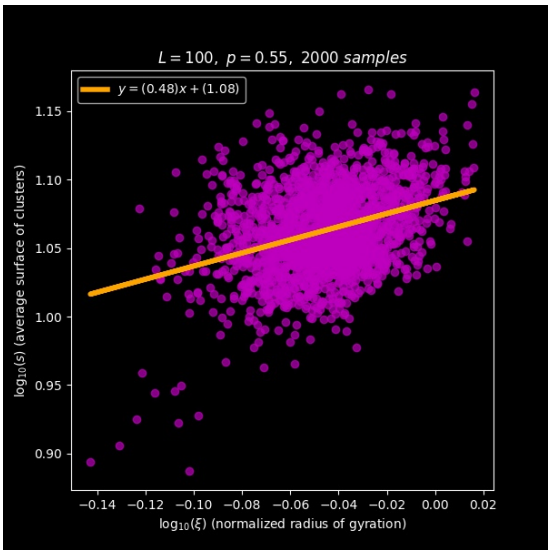
Answer. Via a small amendment to **rgyration()**, the function **rgyration_and_s()** was defined to return not only the radii of gyration for the non-spanning clusters on a lattice, but also the surface areas of the said clusters.

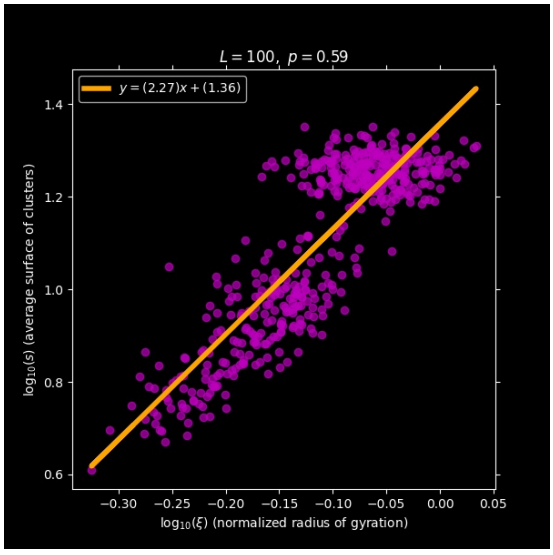
For each given probability, 500 sample lattices were generated, for each sample, the average radius of gyration, and the average cluster surface area was stored. The data for each value of p was then plotted separately.

As apparent from the plots below, near $p \simeq 0.55$, the linear behavior is weak and instead, the points seem to accumulate around a central point. This may not be irrelevent to $p_c \simeq 0.55$ found earlier in exercise 4.6 and may, other than be a validation for the p_c found in exercise 4.6, confirm that the behavior of data changes radically as one approaches the critical probability.









The book *Percolation theory using Python* by *Anders Malthe-Sørenssen, University of Oslo* was tremendously helpful in preparing this report.