Problem Set 7
EPS 528, Science of Complex Systems

Jonas Katona

November 16, 2022

Note: For this problem set, we define permeability as the existence of a continuous, uninterrupted path in *one* direction (e.g., in the z-direction but not considering the x- or y-directions), rather than *two* directions as is found in some texts which also deal with 3D cubic site percolation, e.g., in Tatsumi (1980).

Problem 1.

Solution. We count the number of internal arrangements in which a cube can be permeable given certain numbers of permeable elements. We list each case below:

- *8 permeable elements*: Clearly, descending down any of the four "columns" of pairs of elements, there will always be a path. There is only one arrangement of elements possible here.
- *7 permeable elements*: There are $\boxed{8}$ possible ways to arrange the elements here, and all of these result in at least one path, since we effectively only block one column each time. The reason why there are 8 possible ways is because there are 8 ways to place the one impermeable cubic element.
- *6 permeable elements*: There are $\binom{8}{2} = \boxed{28}$ possible ways to arrange the elements here, and all of these result in at least one path, because at *worst*, we can only block two columns at a time, leaving at least two other columns open for paths from top to bottom. There are $\binom{8}{2}$ possible ways because we are counting the number of ways to choose 2 impermeable cubic elements out of 8 total elements in the cube.
- *5 permeable elements*: There are $\binom{8}{3} = \boxed{56}$ possible ways to arrange the elements here, and all of these result in at least one path, because at *worst*, we can only block three columns at a time, leaving at least one column for a top-to-bottom path. There are $\binom{8}{3}$ possible ways because we are counting the number of ways to choose 3 impermeable cubic elements out of 8 total elements in the cube.
- *4 permeable elements*: In this case, we *can* block all four columns at once and cause the array to become impermeable, and hence, if we want to count as we did above, we have to subtract the cases where all four columns are blocked and there are no continuous paths from top to bottom. In total, there are similarly $\binom{8}{4}$ ways to choose

the 4 impermeable and 4 permeable cubic elements. However, all four columns will be blocked if we put one impermeable cubic element in each column; there are $2^4 = 16$ ways to do this, because there are two possible slots in each column for us to put one impermeable cubic element per column. In turn, this all gives us $\binom{8}{4} - 16 = \boxed{54}$ possible ways to arrange the elements in a way which will give us a permeable array.

- *3 permeable elements*: At first, we convince ourselves that at least one column must have both of its elements be permeable, because otherwise, there could not be a continuous path from top to bottom. There are 4 ways to choose a "permeable" column. Then, we must choose the slot for the other permeable cubic element to reside, of which there are $8 - 2 = 6$ possible slots in the other columns. Hence, we have $(4)(6) = \boxed{24}$ possible arrangements which will give us a permeable array in this case.

- *2 permeable elements*: As above, at least one column must have both of its elements be permeable, or else, there would not be a continuous path downwards. There are 4 ways to choose such a column, and then we have placed all of our permeable elements.

- *1 permeable element or no permeable elements*: There are no possible arrangements which result in a permeable array.

For a $2\times2\times2$ array with k permeable elements, the probability of a given arrangement follows a binomial distribution; i.e., $p^k (1-p)^{8-k}$. Hence, keeping in mind the number of possible arrangements for each case as deduced above, the renormalization group transformation $R_b(p)$ is

$$\begin{aligned} R_b(p) = p^8 + 8p^7(1-p) + 28p^6(1-p)^2 + 56p^5(1-p)^3 + 54p^4(1-p)^4 \\ +24p^3(1-p)^5 + 4p^2(1-p)^6 = 4p^2 - 6p^4 + 4p^6 - p^8 \\ \Rightarrow \boxed{R_b(p) = 4p^2 - 6p^4 + 4p^6 - p^8.} \end{aligned} \tag{1}$$

□

Problem 2.

(a) *Solution.* The code we used to do the first two parts of this problem is the following here below (`percolation3D.m`):

```
function [ data ] = percolation3D(N, dp)
tic
format long

pq = 0 : dp : 1;
Mq = zeros(size(pq));
for i = 1 : length(pq)
    % generate N 2x2x2 sample cubes
    samples = rand(2, 2, 2, N) > 1 - pq(i);
    % sum over top-to-bottom paths
    % (sum will be 2 if path exists in that column)
    paths = squeeze(sum(samples, 3));
    % check for paths
    checkpaths = reshape(paths, [4, N]) == 2;
```

```
    % if contains path, sum of column will be greater than
    % 0. use this to count the number of permeable cubes
    Mq(i) = sum(sum(checkpaths) > 0);
end
Mq = Mq / N;

R2 = @(p) 4 * p .^ 2 - 6 * p .^ 4 + 4 * p .^ 6 - p .^ 8;
R2zero = @(p) R2(p) - p;
% 0.3 is close to the p^{*}, so fzero should converge to
% p^{*} if we let 0.3 be our initial guess for fzero
pc = fzero(R2zero, 0.3);

% root-mean-square error between sampling
% and theoretical prediction
RMSE = sqrt((R2(pq) - Mq) * ((R2(pq) - Mq).') / length(pq));

figure
hold on
plot(pq, pq, ':', 'Color', 'k', 'HandleVisibility', 'off')
plot(pq, Mq, 'Color', '#77AC30')
plot(pq, R2(pq), 'Color', '#4DBEEE')
plot(0, 0, 'r*')
plot(pc, R2(pc), 'm*')
plot(1, 1, 'b*')
hold off
grid on
legend({['Numerical (RMSE: ', num2str(RMSE), ')'], ...
    'Theory', '$\left(0,0\right)$',...
    ['$\left(p^{*}, R_{2}\left(p^{*}\right)\right)\approx',...
    '\left(', num2str(pc), ',', num2str(pc), '\right)$'],...
    '$\left(1,1\right)$', }, 'Location', 'southeast',...
    'interpreter', 'latex', 'FontSize', 12)
title(['3-D site percolation simulation with $N=$',...
    num2str(N), ' and $\Delta p=$', num2str(dp)],...
    'interpreter', 'latex', 'FontSize', 16)
xlabel('$p$ (probability of permeability)',...
    'interpreter', 'latex', 'FontSize', 14)
ylabel(['$R_{2}(p)\approx M/N$ (renormalization',...
    ' group transformation)'], 'interpreter', 'latex',...
    'FontSize', 14)

dR2 = 8 * pc - 24 * pc .^ 3 + 24 * pc .^ 5 - 8 * pc .^ 7;
v = log(2) / log(dR2);

disp('The left value is the critical occupation probability.')
disp('The right value is the critical exponent.')
data = [pc v];

toc
```

The way that `percolation3D.m` works is non-trivial, and hence, even though I left ample comments to explain what many of the lines do, I will explain it in more detail below. Certain lines might seem opaque because I used some "tricks" to speed up the code. In fact, even running with parameters as precise as $N = 10^5$ and $\Delta p = 10^{-4}$ takes only 95 ± 5 seconds, which means that these "tricks" were all worthwhile. (I will explain what these two variables mean in a moment.)

`percolation3D.m` takes two inputs: `N` and `dp`. `N` (or N) gives the sample size for each probability that we are testing, i.e., the number of cubic arrays that we are percolating and checking for permeability. Meanwhile, `dp` (or Δp) gives the grid size for the probabilities that we are testing from $p = 0$ to $p = 1$, i.e., we are testing the following probabilities: $p = 0, \Delta p, 2\Delta p, \ldots, 1 - \Delta p, 1$. We start by creating an array of probabilities to test, which should be of size $\lfloor 1/\Delta p \rfloor - 1$, and alongside this we also preallocate an array of sample estimates for $R_b(p) \approx M/N$, where M will be counted as we test through each probability.

From here, we go through each probability in the $1 \times \lfloor 1/\Delta p \rfloor - 1$ array we made earlier. For each probability we test, we generate our $2 \times 2 \times 2$ sample cubes, which will be represented by 1s and 0s in a $2 \times 2 \times 2 \times N$ array. Of course, each cubic element is permeable (represented by a 1) with probability p and impermeable (represented by a 0) with probability $1 - p$, and `rand(2,2,2,N)>1-pq(i)` populates the $2 \times 2 \times 2 \times N$ array with elements that follow this probability distribution. More specifically, `rand(2,2,2,N)` populates a $2 \times 2 \times 2 \times N$ array with uniformly distributed real numbers in the interval $(0, 1)$, and hence, `rand(2,2,2,N)>1-pq(i)` returns 1s for elements above `1-pq(i)` and 0s otherwise, where `pq(i)` is the probability that any given element is permeable. It is not hard to see that this is equivalent to putting a 1 in each cubic array element with probability p (`pq(i)`) and a 0 otherwise, and is faster than checking and populating each element individually. The result of this we call `samples`, which is still a $2 \times 2 \times 2 \times N$ array.

Next, we sum over the third index in `samples`.[1] We can think of this as checking the four columns in each of the N sample cubic arrays. If there is a path in that column, then the sum of the two elements in that column will be 2. Else, it will be something less than 2. The result of this sum is stored in `paths`, and from there, we generate `checkpaths` by checking which columns generated a sum equal to 2, indicating that there does exist a top-to-bottom path in that sample cubic array. This is done via the line `reshape(paths,[4,N])==2`. Finally, if a cubic array contains a top-to-bottom path and is therefore permeable, we know that the sum of the column in `checkpaths` will be at least 1, i.e., greater than 0, because each column containing a top-to-bottom path is represented by a 1 in `checkpaths`; we check this by taking `sum(checkpaths)>0`. And then, by summing the result, we can compute M, the number of permeable cubes out of the N samples for that probability, which we store in `Mq(i)`. After the for loop concludes, we divide `Mq` by `N` to weight each element appropriately, and `Mq` then represents our numerical estimate for $R_b(p)$ as a grid function evaluated at each element in `pq`.

The rest of the code is relatively self-explanatory. We start by using `fzero` to find the critical probability numerically using an initial guess of $p_0 = 0.3$, since we know a priori by looking at the graph $y = R_2(p)$ ($b = 2$ in this case) and seeing where this

[1] Of course, nothing really changes if we sum over the first or second index instead, even given the way we wrote our code.

crosses the diagonal $y = p$ in the (p, y)-plane that the critical occupation probability satisfies the equation $R_2(p^*) = p^*$ at a non-trivial location (i.e., one in the interval $(0, 1)$) which visually lies just below 0.3.[2] The result is saved in `pc`. From there, we calculate the root-mean-square error between the samples and the theoretical result derived in Problem 1 at each probability p we tested; this result is saved in `RMSE`. Then, we produce a plot which I will explain more in detail through three sample simulations found in Figure 1. Finally, we compute the critical occupation probability using the approximate formula mentioned in lecture on 11-4-20 or in Professor Korenaga's notes on the real-space renormalization group, $\nu \approx \log b / \log\left(\left.\frac{dR_b}{dp}\right|_{p=p^*}\right)$, and save our results as `v`. The numerical results for `pc` and `v` are displayed alongside the time that it took to run the entire code (`tic` starts the timer at the start of the script and `toc` stops it).

In fact, given `pc` from earlier, we can compute `v` more precisely using our theoretical result from Problem 1. Note that

$$\frac{dR_b}{dp} = \frac{d}{dp}\left[4p^2 - 6p^4 + 4p^6 - p^8\right] = 8p - 24p^3 + 24p^5 - 8p^7 \tag{2}$$

$$\Rightarrow \nu \approx \frac{\log b}{\log\left(\left.\frac{dR_b}{dp}\right|_{p=p^*}\right)} = \frac{\log 2}{\log\left(8p^* - 24\left(p^*\right)^3 + 24\left(p^*\right)^5 - 8\left(p^*\right)^7\right)}, \tag{3}$$

and we can use (3) directly to calculate `v` in our code putting `pc` in place of p^* in (3).

We run `percolation3D` for $N = 10^3, 10^4, 10^5 \gg 10^2$ and at $\Delta p = 10^{-3}$ in each case; the graphical results can be found in Figure 1. In case you had issues reading the graphs, the RMSEs in each of the cases were 0.97%, 0.32%, and 0.10%, respectively, for the simulation results attached here, showing generally very robust agreement between our theoretical results from Problem 1 and the ones extracted via simulation. The entire script took 0.748872 seconds, 1.724430 seconds, and 13.792008 seconds to run in each case, respectively, which are certainly reasonable speeds. I also tested $N = 10^2$ and $\Delta p = 10^{-2}$, the results of which can be found in Figure 2; the script took 0.297700 seconds to run in this case. Despite the rather imprecise choices of N and Δp, the RMSE was only 2.61%, which is still quite respectable, given how small our sample size was relatively speaking for both the number of cubes and the probabilities tested.

Either way, we get an excellent match, which allows us to proceed to part (b). □

(b) *Solution.* We can use the same script `percolation3D` to return the results that we need for this part. Regardless of our choices for N and p, we should expect the same estimates for `pc` and `v`, since the way we computed `pc` (and therefore, `v` also as (3) shows) does not depend at all on what we chose for N and p. Indeed, in all our cases, we find that the critical occupation probability is $\boxed{p^* \approx 0.281837636555243}$ and that the critical exponent for correlation length is $\boxed{\nu \approx 1.227411558823035}$. Furthermore, the trivial fixed points for $R_b(p)$ are $\boxed{p = 0}$ and $\boxed{p = 1}$, which we can immediately see via brute force, i.e., just plug these into (1). We can also run `fzero` on the function `R2zero` with an initial guess of -1.5, from which we would find that $p = -1.502583842877299$ is another fixed point for $R_b(p)$, which is also essentially trivial or invalid because

[2] If we chose p_0 too close to 0 or 1, then our nonlinear solver would probably converge to the fixed points at 0 or 1, respectively.

it lies outside of $(0,1)$ and hence does not make any sense as a probability value. The same goes for the four complex solutions to $R_b(p) = p$. And finally, of course, $p^* \approx 0.281837636555243$ is the only non-trivial fixed point for $R_b(p)$.

Let us also check the stabilities for the three physical, sensible fixed points of $R_b(p)$: $p = 0$, $p \approx 0.281837636555243$, and $p = 1$. By substituing these into (2), we find that

$$\left.\frac{dR_b}{dp}\right|_{p=0} = \left.\frac{dR_b}{dp}\right|_{p=1} = 0, \quad \left.\frac{dR_b}{dp}\right|_{p=p^*} \approx 1.7589599713276754 > 0,$$

from which it is clear that $p = 0, 1$ are higher-order fixed points and $p = p^*$ is unstable. We would have been able to deduce this result from the 2D lattice percolation case, in which case the one non-trivial fixed point in $(0,1)$ was also unstable.

From here, to classify the stabilities for $p = 0$ and $p = 1$, we need to look at the higher-order derivatives of R_b. The second-derivative of R_b is

$$\frac{d^2R_b}{dp^2} = \frac{d}{dp}\left[8p - 24p^3 + 24p^5 - 8p^7\right] = -8\left(p^2-1\right)^2\left(7p^2-1\right) \tag{4}$$

$$\Rightarrow \left.\frac{d^2R_b}{dp^2}\right|_{p=0} = -8(0-1)^2(0-1) = 8 > 0, \quad \left.\frac{d^2R_b}{dp^2}\right|_{p=1} = -8(1-1)^2(7-1) = 0. \tag{5}$$

Hence, near $p = 0$, $R_b(p) \approx 8p^2/(2!) + \mathcal{O}(p^3) = 4p^2 + \mathcal{O}(p^3)$, from which we see that $p = 0$ is a stable fixed point. For $p = 1$, we still have failed to classify its stability, but I will spare you the details and assert that the third derivative of R_b is *still* zero when evaluated at $p = 1$. Thus, we must go to the fourth derivative of R_b, which is $d^4R_b/dp^4 = -48(3 - 30p^2 + 35p^4) \Rightarrow d^4R_b/dp^4|_{p=1} = -384 < 0 \Rightarrow R_b(p) \approx 1 - 384(p-1)^4/(4!) + \mathcal{O}\left((p-1)^5\right) = 1 - 16(p-1)^4 + \mathcal{O}\left((p-1)^5\right)$. Thus, $p = 1$ is also a stable fixed point for $R_b(p)$. □

(c) *Solution.* Recall that the critical occupation probabilities for the 1-D and 2-D cases are $p_c = 1$ and $p_c \approx 0.618033988749895$, respectively. Furthermore, in part (b), we just found that $p_c \approx p^* \approx 0.281837636555243$ in the 3-D case. To explain the decreasing trend as the number of dimensions increases, note that when $p < p_c$, no percolating infinite cluster emerges, but when $p > p_c$, there is one percolating infinite cluster. This illustrates how, as the number of dimensions increases, there are more possible paths for percolation to occur. Namely, if we fix $b = 2$, note that in 2D, we have two possible paths downwards, while in 3D, there are four possible paths downwards, and in 4D, there would be eight possible paths downwards, etc. Hence, p_c would be lower in higher-dimensions because p does not need to be as large for a percolating infinite cluster to emerge as the dimension increases, since the number of possible paths increases for say, some fluid particle to move through something from top to bottom as the dimension increases.

We can also think of the decreasing trend in p_c from a combinatorial perspective. In 1D, we only need to consider paths in one direction, and that is the *only* possible direction. For 2D, we need to consider paths in one direction, but exclude *one* path in the transverse direction. Meanwhile, in 3D, we again consider only the top-to-bottom direction, but by doing so, we need to exclude *two* paths along the "x"- and "y"-directions. Then, when we compute how $R_b(p)$ changes with p, we should expect $R_b(p)$ to slope up more slowly for large values of p — or equivalently, more quickly for

smaller values of p — because as we increase p, there are less possible top-to-bottom paths being activated if we make p really large since we already have many possible paths available, but as p increases from zero, the chance of some top-to-bottom path appearing is relatively higher for reasons described in the above paragraph. In other words, if many paths already exist, i.e., if p is large enough, then the renormalization group transformation value cannot change much since the chance of a percolating infinite cluster is already close to 1. Then, $R_b(p)$ we would expect to slope up quicker towards 1 in higher dimensions than in lower dimensions, which means that the graph $y = R_b(p)$ will cross the diagonal $y = p$ in the (p, y)-plane at a smaller value of p, or equivalently, the nontrivial fixed point $p^* \approx p_c$ in the interval $(0, 1)$ will be at a smaller value of p. □

Problem 3.

Solution. I have been working on a project about phase-field modeling of lava solidification. Namely, we have been studying the dynamics of lava emplacement (secondary, tertiary, etc. emplacements of lava lobes on top of one another), and trying to understand qualitatively how the solidification and cooling of each lobe is affected by the placement of lobes on top of one another. Our current results indicate roughly three regimes, where depending on the size of the lobes and the emplacement time intervals, the lobes will either recombine and form one massive lobe as the solid layer between them remelts; the lobes will solidify in parallel with another, with the lower one usually solidifying before the top one does; or the lower one will solidify completely before the second one is even emplaced.

Already, our model uses phase-field methods. These involve "smoothing" out the liquid-solid interface with a phase variable ϕ that ranges between $\phi = 0$ (liquid) and $\phi = 1$ (solid) as well as considering the bulk energy and entropy of the lava as an integral. Considering that we make the interface "sharp" enough, the introduction of ϕ should give us accurate results, and the latter is already extremely accurate given the sizes of the lobes we are simulating which are at least 0.1m in height. The result is a coupled system of nonlinear reaction-diffusion equations for temperature and ϕ.

Of course, we only really studied secondary emplacements of the same size. However, in actual field data, we often deal with dozens, possibly hundreds of smaller lobes of varying sizes which are emplaced on top of each other. We can sometimes track the temperature data based off of magnetic polarization, mineral composition, trapped gas bubbles, and other properties of the igneous rock layers that leave record of such flows. But how can we use this reconstruct things such as emplacement history? What can we say about the thermal activity in these multiply-emplaced flows on a much larger timescale than the emplacement intervals? Our model only suffices to provide qualitative and *sometimes* quantitative answers (the PDEs we have must be dealt with numerically, as they cannot be solved analytically), and to deal with secondary emplacements of varying sizes already introduces a massive number of degrees of freedom, since we can vary each lobe size separately as well as their emplacement times. And in actual field data, we are dealing with $N \gg 1$ lobes emplaced on top of each other with varying sizes *and* emplacement time intervals.

For larger flows of sizes 1 meter or larger, we generally are only dealing with up to around 10 emplacements, but usually only a few (around 2-4), so coarse-graining is not really appropriate. But to more satisfactorily answer the above questions regarding smaller, quick emplacements, we could certainly coarse-grain our model. We would probably want to introduce an average emplacement time interval τ and an average lobe size h, alongside perhaps standard deviations for each of these variables. We would also have $N \gg 1$ lobes in total. Unfortunately, I am afraid that we would only be able to deal with this problem

numerically, since we cannot even solve for the temperature in our phase-field PDE model explicitly. However, dealing with this problem numerically is not the worst option, but it could be computationally extremely tedious and time-consuming. Furthermore, it would not give us robust analytical results that are readily usable and generalizable.

On the other hand, there is an easier problem which *does* have an explicit, albeit somewhat cumbersome solutions: The Stefan problem. In this case, the solid-liquid interface is discontinuous, and we are effectively solving two different PDEs for the thermal evolution of the solid and liquid phases separately; the boundary condition at the interface expresses a conservation of energy between heat conduction through each phase and latent heat production. If we were somehow able to average the temperature dynamics over all of the lobes, we might be able to come up with some macroscopic thermal properties across the entire domain and determine things such as the maximum temperature across the domain and where this is located, the density or probability of certain spatial regions being liquid or solid, the *expected* solidification time at a given point in the domain, etc. And as already predicted with our original results for secondary emplacements, we would expect there to be some critical combination of τ and h below which all of the lobes would combine together and form one large lobe, perhaps some range where some lobes would combine and others would not, and after a certain point, *no* lobes would recombine.

I truly think that this might be a very worthwhile and useful continuation of my current research project, but of course, I should probably complete and finalize what I currently have before tackling more bold and unexplored questions. As of now, we will be presenting our work at the AGU Fall Meeting in several weeks in the form of a poster and are currently drafting our paper for submission. If you have any recommendations for pursuing such an avenue of research (e.g., research papers which have dealt with something similar or related), I would love to hear your suggestions. □

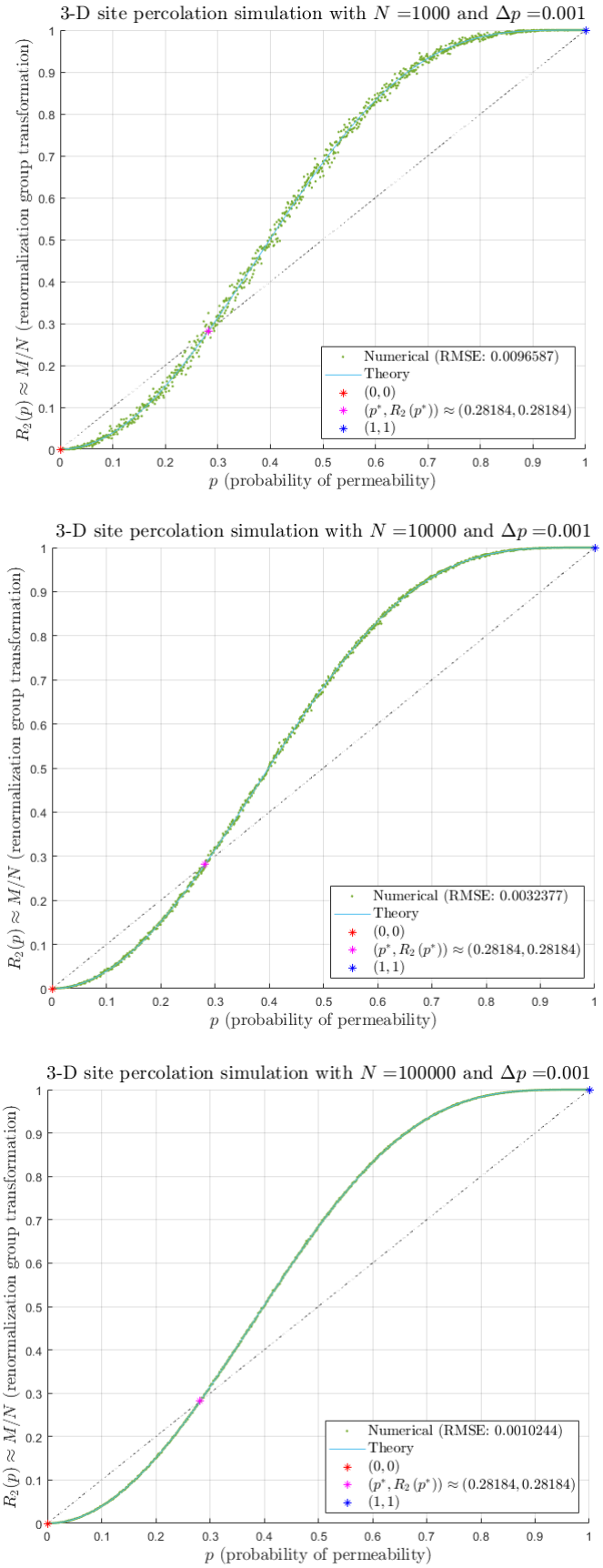

Figure 1: Graphical results for $N = 10^3, 10^4, 10^5$ and with $\Delta p = 10^{-3}$ in each case, respectively, in top-down order. Note how the points converge to the theoretical solution curve as N increases, indicating general agreement with our results from Problem 1.

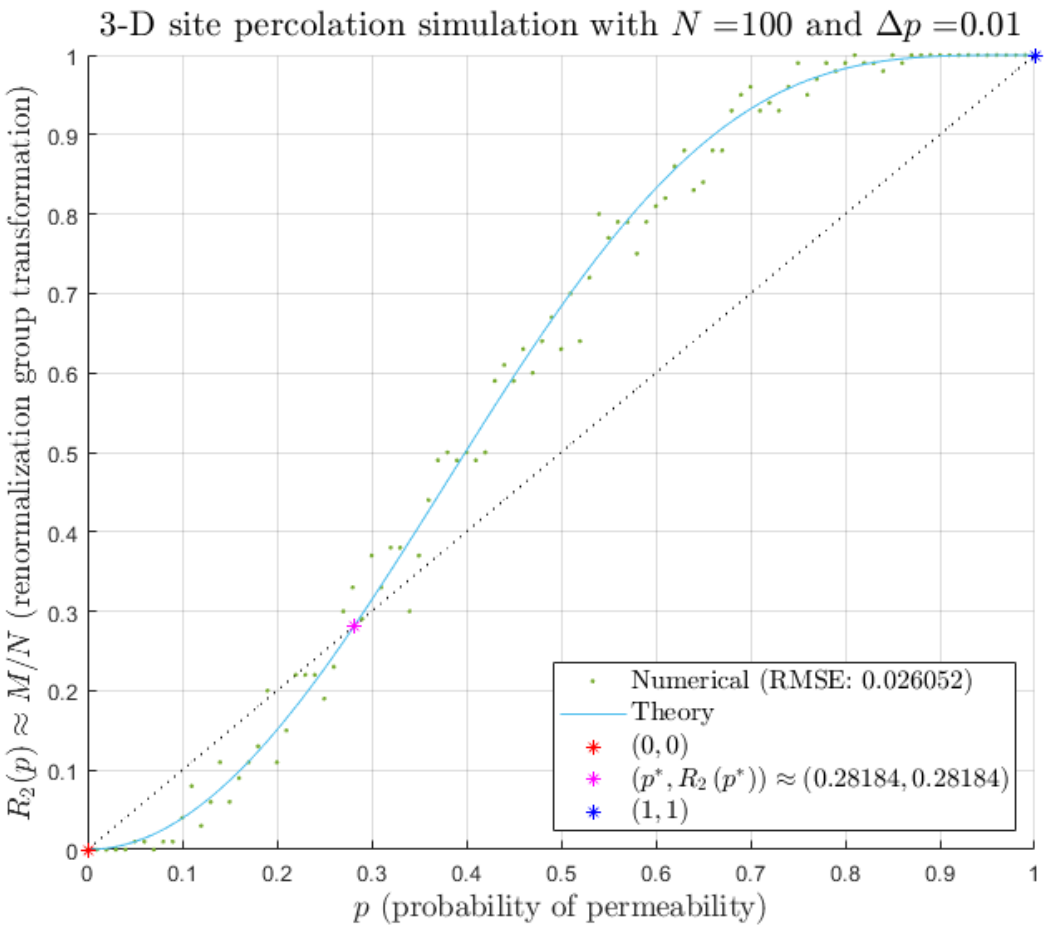

Figure 2: Graphical results for $N = 10^2$ and $\Delta p = 10^{-2}$, which are not too bad considering how N was not too large and Δp was not too small.