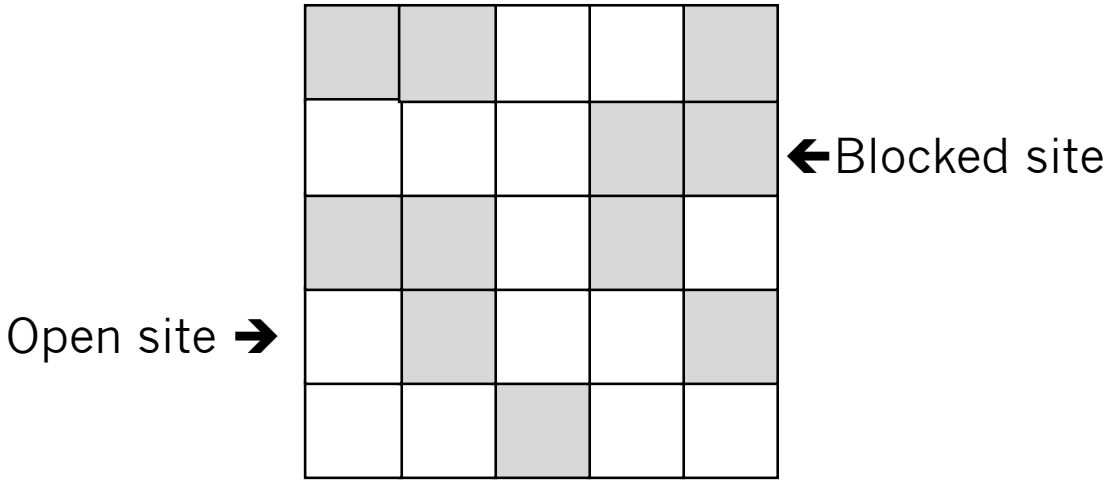
ENGI E1006

Percolation Handout

NOTE: This is not your assignment. These are notes from lecture about your assignment. *Be sure to actually read the assignment as posted on Courseworks and follow the instructions provided there.*

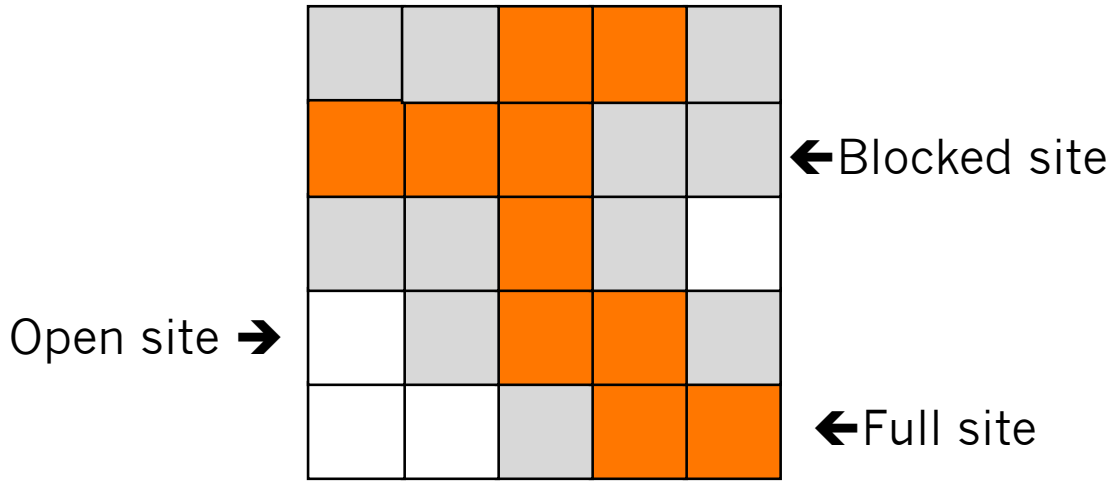
Percolation is the process of a fluid slowly traveling through a porous material. Example: If a liquid is poured on top of a layer of soil, will it manage to filter down through the soil and into what lies below? If it does we say it percolates.

We can model this phenomenon using a a grid consisting of blocked and open (vacant) sites to represent the soil.



We imagine pouring a liquid onto the top of the grid and consider how it would flow through the open and blocked

sites.



If the liquid can find a path through the grid, then it percolates in this instance. There are many other applications of the percolation model beyond liquid moving through soil. The model may be used to describe gas moving through a gas mask filter, electricity through a network of resistors, natural gas through semi porous rock, and many other important phenomena.

Question: *Okay, so if we're supposed to be modeling some porous material with this grid, how do we determine which sites are blocked and which sites are open?*

Answer: We'll use randomization. Let each site have a probability of being vacant (open) or blocked. Very porous materials will have a high *site vacancy probability* and less porous materials will have a low site vacancy probability. Let's call this site vacancy probability *p*. Using a pseudorandom number generator we can generate *n x n*

grids for any *p* we're interested in.

Question: *So if some material has a site vacancy probability p, will a corresponding n x n grid allow percolation or not?*

Answer: That's the million dollar question! First of all, notice that the answer to this question will be a probability. Since we are randomly generating the grid, we could get really lucky even with a low *p* and end up with a grid that allows percolation. Similarly even with a very high *p* we could end up with a system that does not percolate. So our answer will not be yes or no but rather a *percolation probability.* Fine, so how do we figure that out? The only way we know how to do that is via simulation. That is, we'll generate a whole bunch of these grids and then check each one to see if they permit percolation. The fraction of grids that get percolation is our estimate for the percolation probability. In a nutshell, that's what you'll be doing in the current project.

Question: How are we going to do this using Python?

Answer: We will use two-dimensional numpy arrays to model a grid. We'll learn about numpy arrays and how exactly one does this in the next lecture. For now think of them as very similar to lists in Python. In fact, we could do this just using lists but it would be very inefficient and take way too much time. That's where numpy will come in. So let's break it down a bit more:

Ultimate Goal: The question we want to answer is "Given an *n x n* grid with site vacancy probability *p*, whats the percolation probability of the system? We'll ultimately answer this by generating a graph of percolation probability versus site vacancy probability. Suppose we're considering a 10 by 10 grid and consider a single point on the graph: We're given a site vacancy and we want to determine the corresponding percolation probability. Here's what you need to do:

1) Randomly generate a 2-dimensional numpy array made up of zeros and ones. Each entry in the array will be 1 with probability *p*.
2) Determine what the system would look like if we poured liquid on the top. (Create an array representing the one with the orange in it above.)
3) Determine if the array you made in (2) is percolating or not. (Just check to see if any of the bottom sites are filled.)
4) Repeat the process a whole bunch of times keeping track of how many times it percolates. The fraction of times it percolates is the estimate for the percolation probability.
5) Graph your results.

That's it. That's all you have to do. That doesn't seem too bad, does it? Well, how do you do (2)? That is a good question. That part can be tricky. To help with all of this we will break the assignment into two parts. For Part 1

we're going to simplify step 2 by only modeling *vertical percolation*. That means that the liquid can only move vertically. So when you generate the flow matrix in step two, the only way a site can get filled is if it is vacant ***and*** the site above it is filled. That should make step 2 quite a bit easier. Next week will figure out how to remove this restriction and consider the more general percolation model. Here's Part 1 of your assignment below:

Percolation Part 1

Write a Percolation module in Python to solve the *vertical percolation* problem we saw in class. Your module should make use of the functions we designed in class. Specifically:

1. Functions to read and write *N x N* arrays (use numpy) of binary numbers representing a grid of blocked/ open sites (from or to text files). Please use the format below for text files.
2. A function that takes one of the arrays from (1) as input and outputs an array of vacant/full sites.
3. A function that takes as input the output from (2) and outputs a boolean indicating whether the system percolates or not.
4. A function that takes as input a number between 0-1 *p*, an integer *N,* and generates a random *NxN* array of blocked/open sites where each site is open with probability *p*. This function will be useful for testing your code.
5. I have provided function definitions in the attached

percolation.py file. I have provided a main function to test your code in a separate file hw5_1.py. Your code must work with the main function I have provided.

1. For section (1) above please use the following text file format: The first row of the text file should contain the integer *N*. That is the number of rows and columns in the system. The next *N* rows should contain *N 1*s and *0*s each separated by a space. The *0* indicates a blocked site and the *1* indicates an open site. So for example:

3

0 1 0

1 1 0

0 0 1

would represent a 3x3 system where the first row consists of a blocked site, an open site, and a blocked site.

NOTE: Please make sure your code works with the main function provided on Courseworks.