Braden Bagby Computational Physics Assignment 6

Foundations of the Problem

A DLA Model is a model showing how particles undergo a random walk and form clusters. We can simulate this in physics using matrices. We will simulate two types of models.

One type, a walker is randomly placed and walks around randomly until it collides with the cluster, at that point becoming part of the cluster.

The other model, an Eden model, the walker is randomly placed and if and only if it happens to land on right next to the cluster it will become part of it.

Finally, we can use matrices to find the percolation threshold in a system of size n x n.

Text From the Program

I begin with the Eden model as it is the simplest.

We setup our initial values and loop a set amount of times. Each loop will be an attempted addition to the cluster.

Within the loop, we place a walker randomly

```
    x=randi(SIZE);
    y=randi(SIZE);
    W(y,x) = 1; %walker initial plop
```

We then check if the walker is on top of the cluster, and if it is, we go to the next iteration of the loop

```
    if sum(sum(W&H)~=0) %walker on top of the cluster
        continue
    end
```

Otherwise, we check and see if the walker has landed next to the cluster

```
    w2=zeros(SIZE);
    w2 = circshift(W,[1 0]) + circshift(W,[-1 0]) +circshift(W,[0 1])+circshift(W,[0
-1]);
    if sum(sum(w2&H)) ~= 0
```

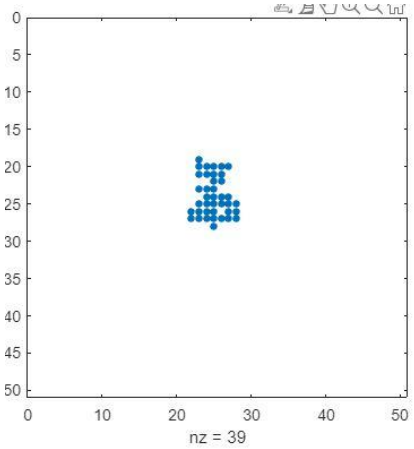
and if it has, we add it to the cluster and draw it to the screen

```
       H=or(W,H);

        spy(H);
        pause(0.01)
```

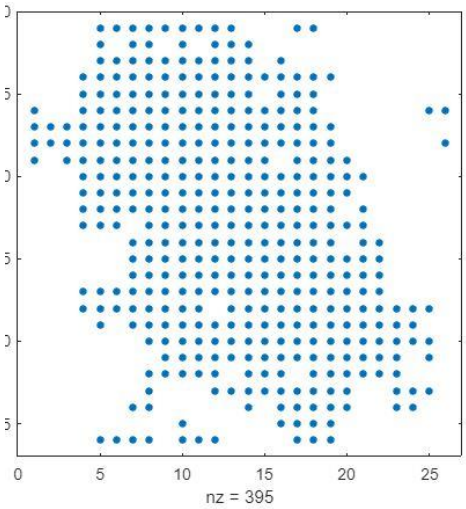
Braden Bagby Computational Physics Assignment 6

Here is the end result of the Eden cluster after 5000 iterations with a window size of 50x50

A small cluster size is expected because [Chance of landing next to original cluster] * [Number of iterations] or 4/(50x50) * 5000 = 8. This formula does not supply the actual expected number of clusters because as the cluster grows the chance of landing next to the cluster grows as well.



Here is another run with 5000 iterations and a window size of 25*25. The estimated expected outcome using the same formula as before would be 29.5, making this cluster larger. Notice there are a lot more than 29 pieces in the cluster, this is because as the cluster size increase the chance of landing next to it increases faster and faster.



Next we will take a look at another type of DLA model. This model adds a walker that randomly walks around until it collides with the cluster at which point it will become part of it.

Braden Bagby Computational Physics Assignment 6

We first initialize the variables and start a loop.

Inside the loop, we must place a walker randomly within the window size.

```
    x=randi(SIZE);
    y=randi(SIZE);
    W(y,x) = 1; %walker initial plop
```

If it lands on top of the current cluster we just continue to the next iteration

```
    if sum(sum(W&H)~=0) %walker on top of the cluster
        continue
    end
```

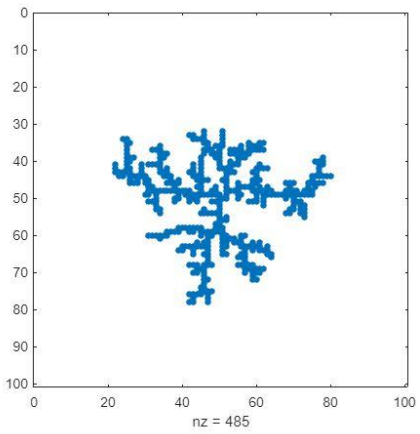
Otherwise, we move it randomly in any direction until it is next to the current cluster

```
    w2=zeros(SIZE);
    w2 = circshift(W,[1 0]) + circshift(W,[-1 0]) +circshift(W,[0 1])+circshift(W,[0
-1]);
    while sum(sum(w2&H)) == 0
        x = rand(1);
        if x < 0.25
            w2 = circshift(w2,[1 0]);
            W = circshift(W,[1 0]);
        else if x < 0.5
            w2 = circshift(w2,[-1 0]);
            W = circshift(W,[-1 0]);
        else if x < 0.75
            w2 = circshift(w2,[0 1]);
            W = circshift(W,[0 1]);
        else
            w2 = circshift(w2,[0 -1]);
            W = circshift(W,[0 -1]);
         end
        end
        end



    end
```

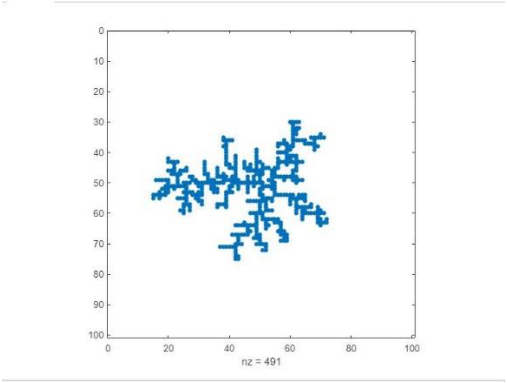
Once it is next to the current cluster, we add it to the cluster, draw to the screen, and continue

```
    H = or(W,H);
        spy(H);
        pause(0.01)
```

Here is a result of 500 iterations with a window size of 100. Notice this type of DLA grows a lot quicker than the Eden cluster as each walker is used to add to the cluster unless it lands on top.



There are 485 pieces to the cluster meaning only 15 pieces landed on top of the current cluster.



Here's another run where only 8 pieces landed on top of the cluster.

Percolation Threshold

To get one percolation threshold we need to define a nxn matrix of 0s and randomly set values in it to 1 based on a probability for each index. We then need to do this across from %0 probability to %100 to find at what the threshold is that the makes the system always percolate.

We start by looping through probabilities %1 to 100%

```
for prob = 1:100
    probability = prob; %we just want to do every 5%
    probabilities(prob) = probability;
    numberPerculated = 0
```

Next we need to perform a number of trials for each probability. We will see how many out of these trials percolated to calculate the probability of percolation for that probability.

for trial = 1:trialsEach

Within each trial we must setup a SIZExSIZE matrix and randomly set values in it to 1 based on the current probability

```
H=zeros(SIZE); %setup matrix
    for x=1:SIZE
        for y=1:SIZE
            if rand(1) < (probability/100)
                H(x,y) = 1;
            end
        end
    end
```

Then I use BW label to create a matrix of all the clusters in this matrix

bw = bwlabel(H,8);

Next, I figure out the max value in bw which will be the number of clusters

```
%get max from bw

    max = 0;
    for x=1:SIZE
        for y=1:SIZE
            if bw(x,y) > max
                max = bw(x,y);
            end
        end
    end
```

I then check every cluster and see if it reaches across from the top of the matrix to the bottom by seeing if the index of that cluster exists in every row

```
perculates = 0;
    for i=1:max
    if perculates == 1
       break;
    end

        gotInY = 1;
        for y=1:SIZE

            found = 0;
            for x = 1:SIZE
                if(bw(x,y) == 1)
                    found = 1;
                    break;
```

```
                end
            end

            if found == 0
                gotInY = 0;
                break; %not found in this y so this group doesnt perculate
            end

        end
        if gotInY
            perculates = 1;
            break;
        end
    end
```

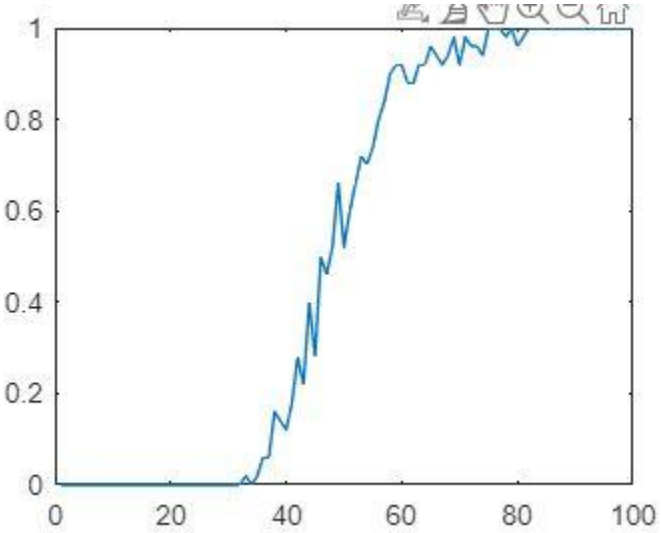
If it does cross from top to bottom, this system is percolating and we add it to the total count

```
numberPerculated = numberPerculated + perculates;
```

After doing N trials, we calculate the probability of percolation and add that to our matrix at the correct index for the probability we were testing

```
numberPerculated
probOfPerculation = numberPerculated / trialsEach
perculatedProbability(prob) = probOfPerculation;
```

Here is an example run of a size 20x20. Its percolation threshold is around 65



But we want to try this for a lot of different sizes, and plot the results.

So we put all of the already shown code in a loop and try many different sizes. If the probability of percolation passes 0.9 we count that as the threshold and save it in an array

```
if probOfPerculation > 0.9
          perculationThreshold = prob;
          break;

       end
```

At the end we plot all the sizes and percolation thresholds. Here is the result and it shows that the percolation threshold is a set number that doesn't depend on the size of the system. Its between 55 and 65 and probably around 60.