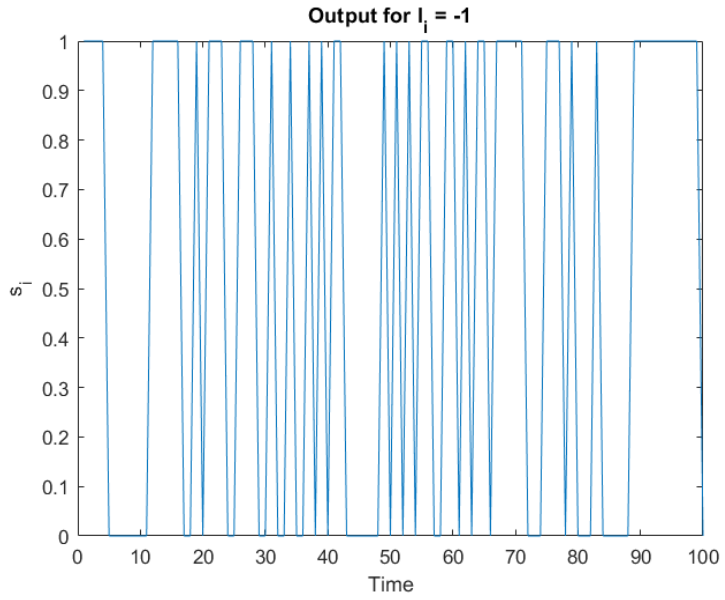
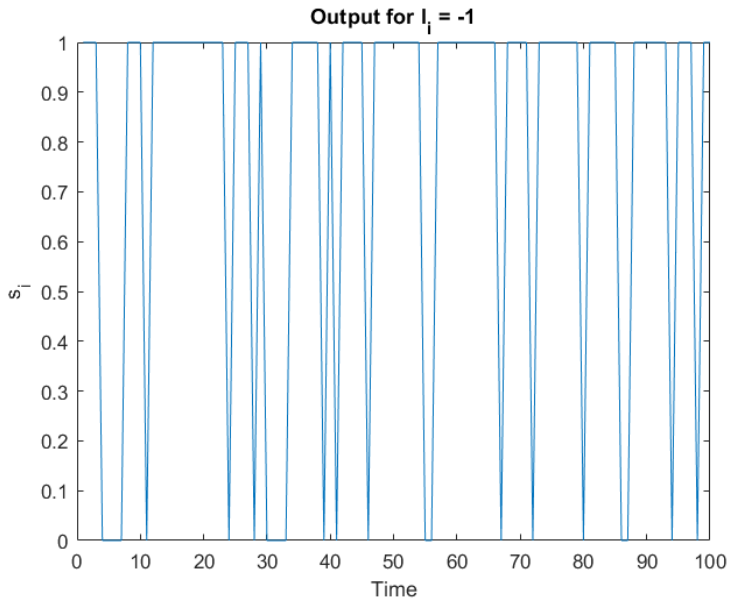
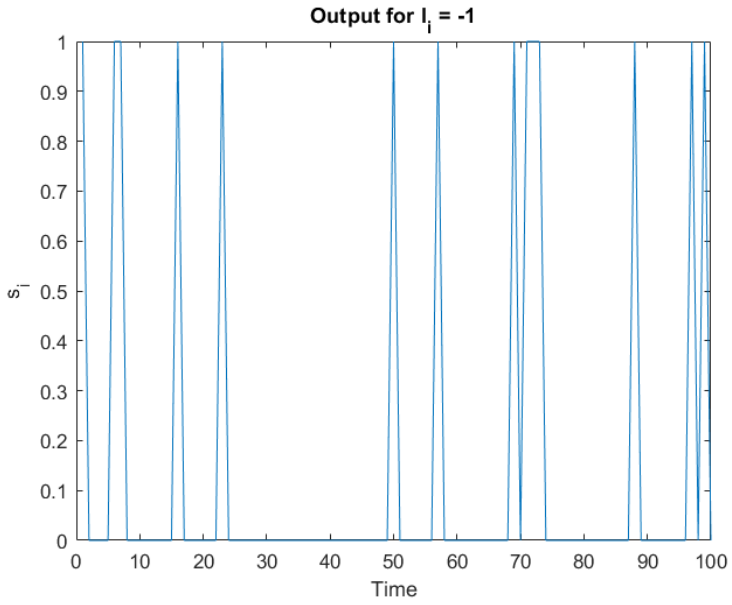
Homework-3
Umang Garg (Perm: 6787683)

Q1 part 1

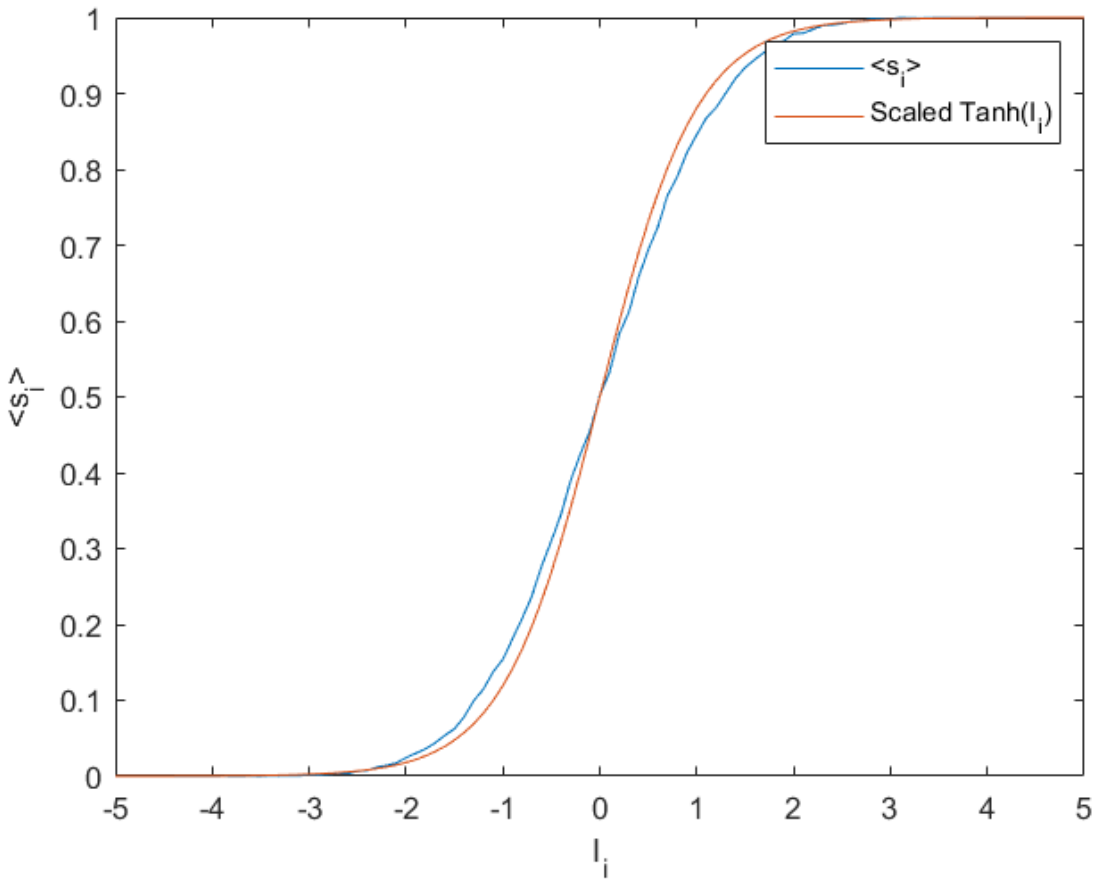
The three representative points as a function of time (assuming random numbers are generated in time) for Ii = −1, 0, +1 with at least 100 samples each are shown below:



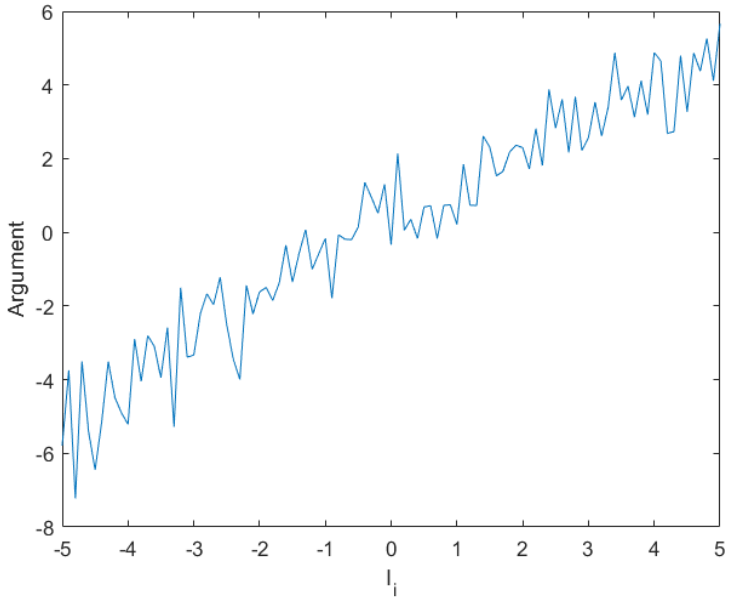




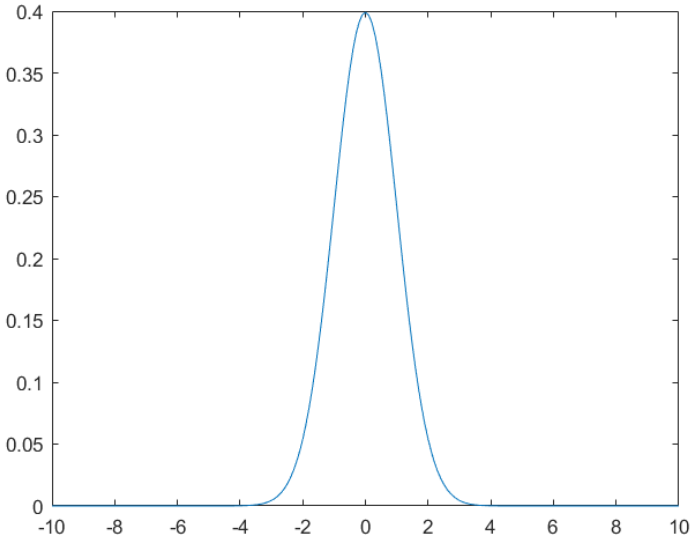
(b)



Yes, the 2 curves look approximately equal.



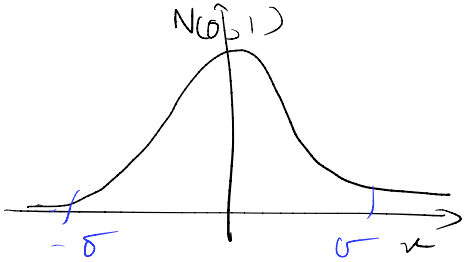
The argument of the Θ moving as a function of Ii for one iteration



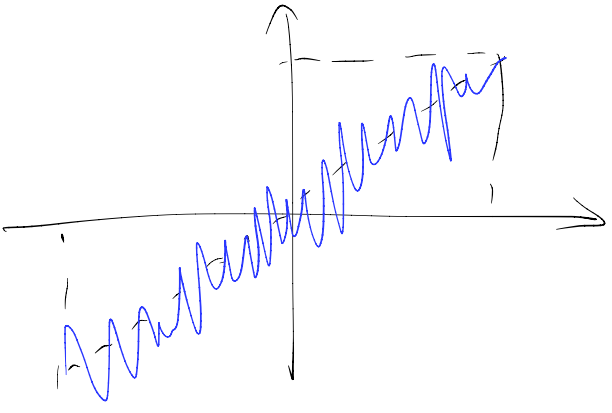
N is shown as graphically

Analytical calculations are shown on the next page.

(c)



Arg of $\theta = I_i + \gamma_N$



$$P(-\infty, 0) = 1 - P(0, \infty)$$

$$= 1 - \frac{1 + erf(\mu/\sqrt{2}\sigma)}{2}$$

$$= \frac{1 - erf(\mu/\sqrt{2}\sigma)}{2}$$

$$P(\text{drawing } +1 / I_i) = P(I_i + \gamma_n > 0)$$

$$= P(\gamma_n > -I_i)$$

$$= \int_{-I_i}^{\infty} N(\sigma, \mu)\, dx$$

$$P(\text{drawing } -1 / I_i) = P(I_i + \gamma_N < 0)$$

$$= P(\gamma_N < -I_i)$$

$$= \int_{-\infty}^{-I_i} N(\sigma, \mu)\, dx$$

$$\langle s_i \rangle = \left[P(\text{drawing } +1)\right](+1)$$

$$+ \left[P(\text{drawing } -1)\right](-1)$$

$$= -\int_{-\infty}^{-I_i} N(\sigma, \mu)\, dx + \int_{-I_i}^{\infty} N(\sigma, \mu)\, dx$$

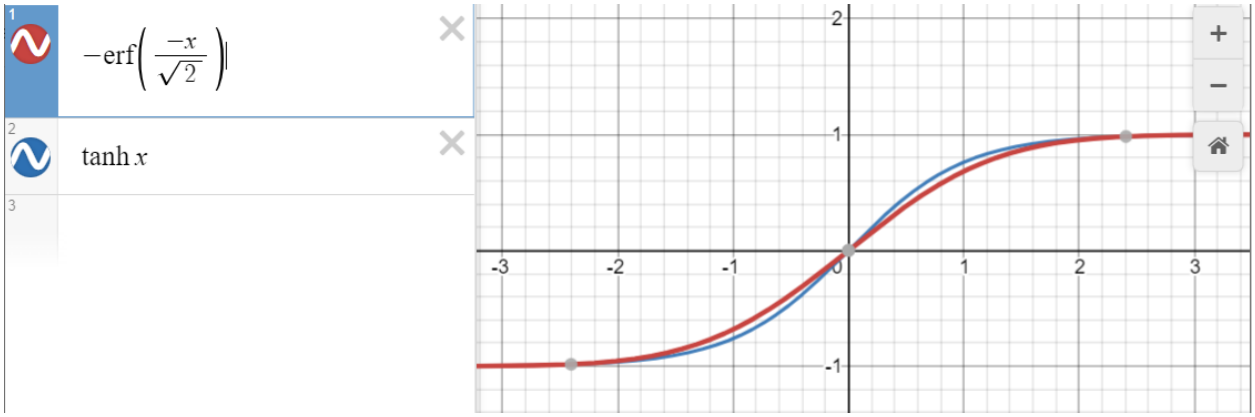
$$= -\left(1 - \int_{-I_i}^{\infty} N(\sigma, \mu)\, dx\right) + \int_{-I_i}^{\infty} N(\sigma, \mu)\, dx$$

$$= 2\int_{-I_i}^{\infty} N(\sigma, \mu)\, dx - 1$$

$$= 1 - 2\int_{-\infty}^{-I_i} N(\sigma, \mu)\, dx$$

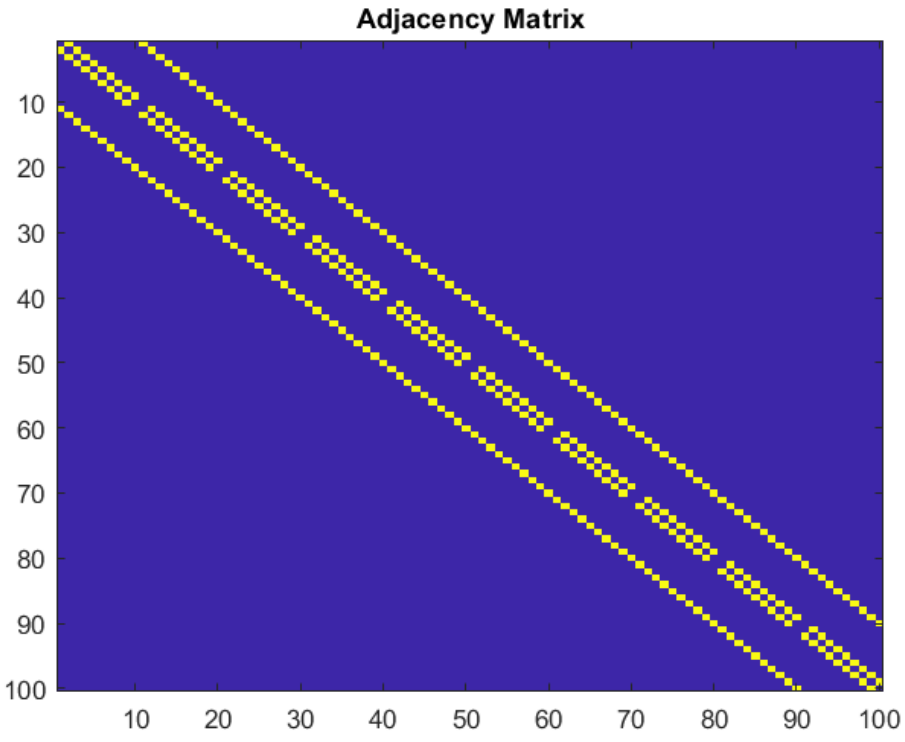
$$= 1 - 2\left(\frac{1}{2}\left(1 + \text{erf}\left(\frac{-I_i - \mu}{\sigma\sqrt{2}}\right)\right)\right) = -\text{erf}\left(\frac{-I_i}{\sqrt{2}}\right)$$

We conclude analytically that the results are similar, as shown in the plot below.



Q2 Part 1

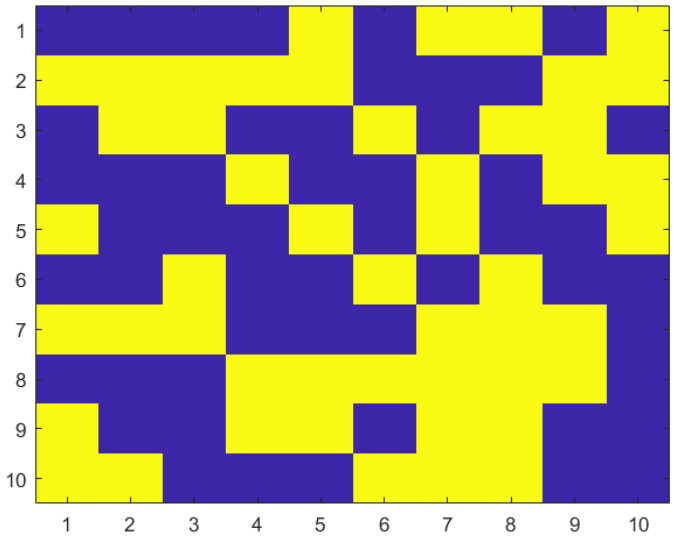
The adjacency matrix of the grid of 10x10 is plotted below:



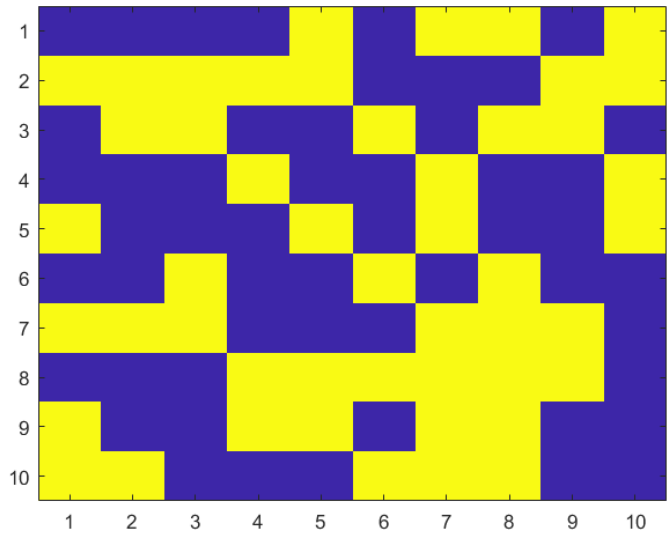
We observe that the adjacency matrix is mostly sparse, i.e. each spin in the lattice is connected to a limited number of structured neighbors. This also entails that for an effect to travel to the farther end of the lattice will take much longer as compared to an

all-to-all/ small-world network. If the connections were all-to-all, the adjacency matrix would have been a solid color block denoting, every spin is affected by all the other spins immediately. This assumes that the all-to-all connections also include recurrent connections.

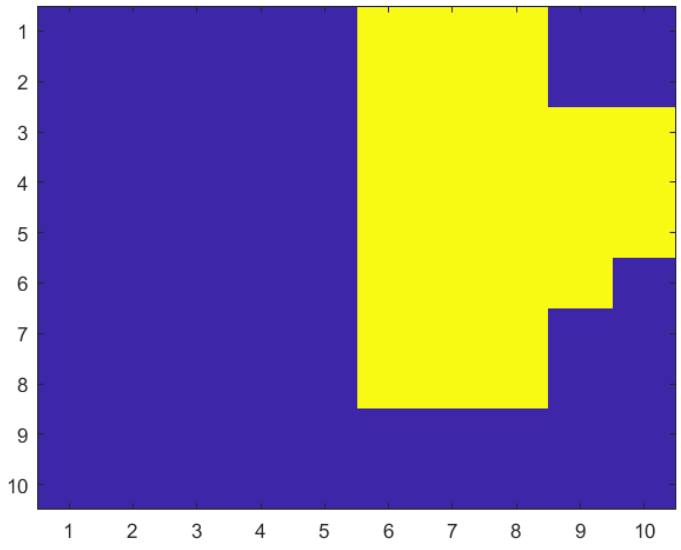
Q2 part 2



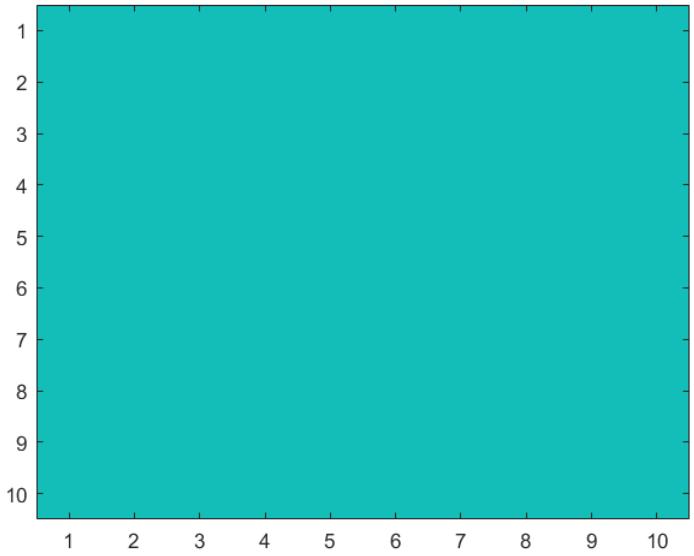
Time t = 0



Time t = 1



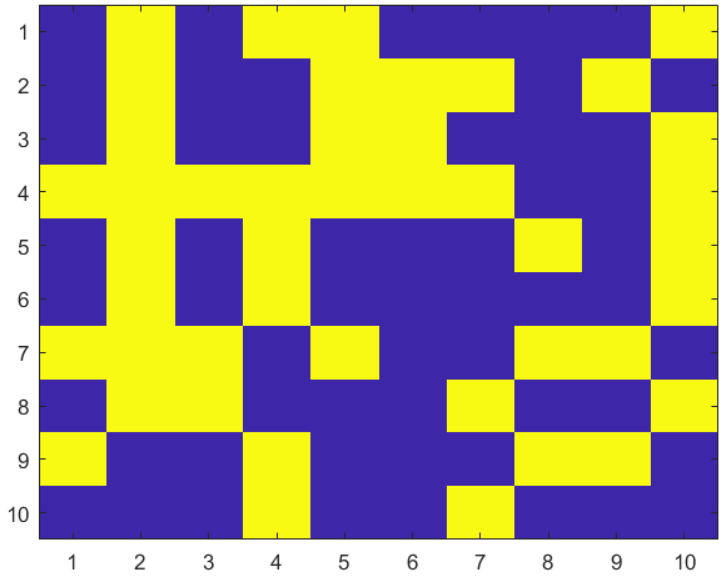
Time t = 1000



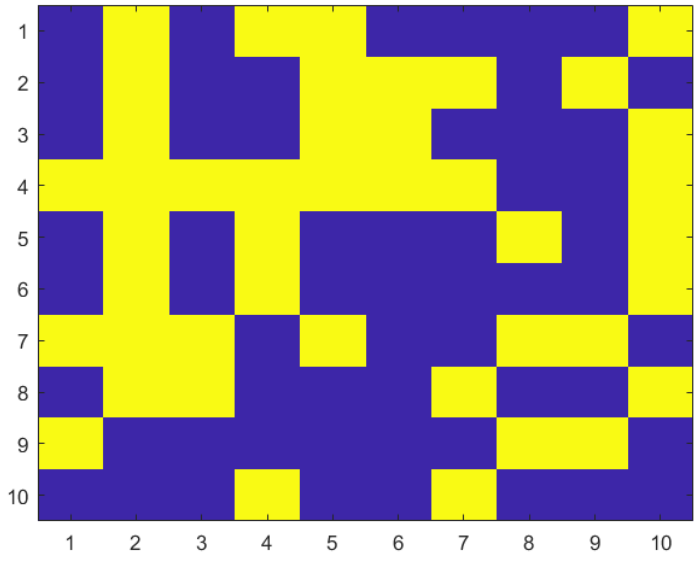
Time t = 1e5

Q2 PART 3

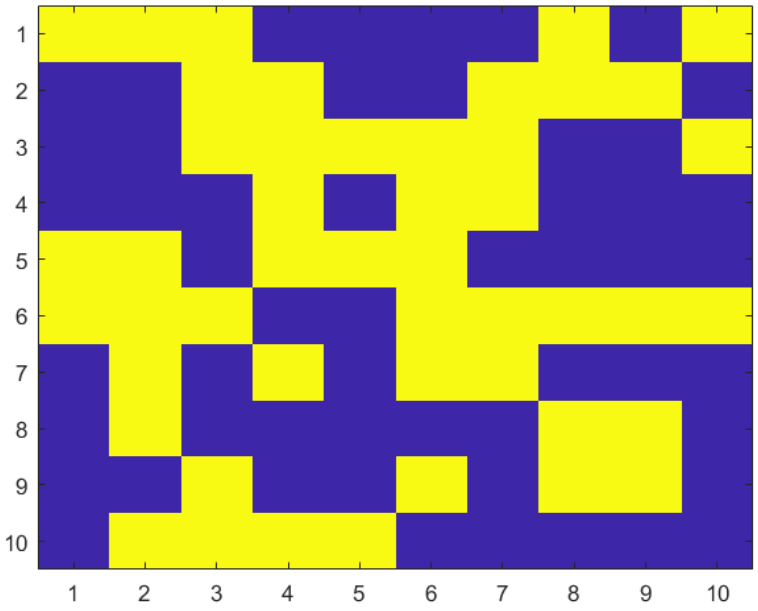
Sequentially reducing temperature (increasing beta) on each iteration.



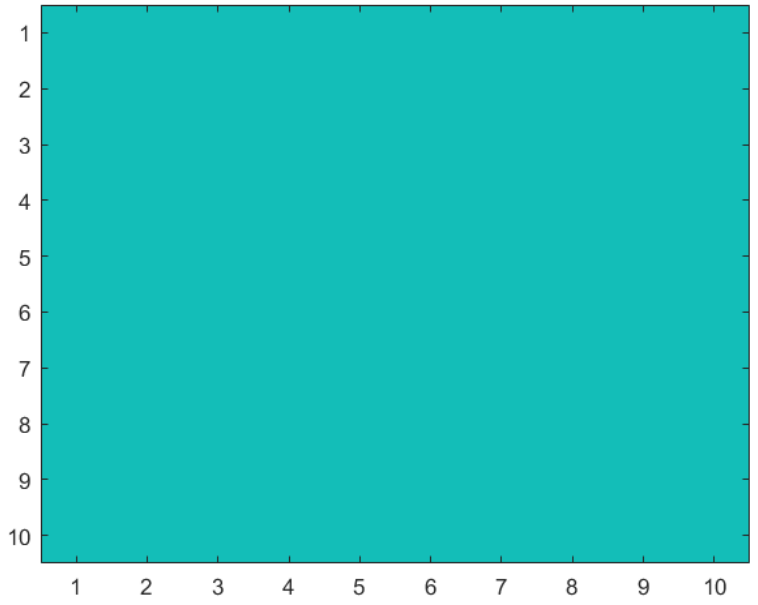
Time t = 0



Time t = 1



Time t = 1000



Time t = 1e5

(d) It was also noted that with increasing lattice size, the system was not converging to a single spin orientation with the same number of timesteps. It was rather a many-cluster (pockets) orientation of spins.

Implemented Matlab Code:

```
clear all;

%% Problem

mu = 0;
sigma = 1;
Ii = -5:0.1:5;
total_time = 10000;
time = 1:1:total_time ;

for i = 1:1:length(Ii)
   for t = 1:1:total_time

      r = normrnd(mu, sigma);
      temp = Ii(i) + r;

      if (temp<0)
         s(i,t) = 0;
      else
         s(i,t) = 1;
      end
   end
   avg_s(i) = sum(s(i,:))/total_time ;
end

% figure(1);
% p1 = plot(time,s);
% xlabel('Time');
% ylabel('s_{i}');
% title('Output for I_{i} = -1');

figure(2);
p2 = plot(Ii, avg_s);
xlabel('I_{i}');
ylabel('<s_{i}>');
hold on;
plot(Ii, (1+tanh(Ii))/2);
legend('<s_{i}>', 'Scaled Tanh(I_{i})');
```

```
%% Problem 1 part c

x = [-10:.1:10];
gaussian_distribution = normpdf(x,0,1);
figure();
plot(x,gaussian_distribution);

for i = 1:1:length(li)

   r = normrnd(mu, sigma);
   Arg(i) = r + li(i) ;
end

figure();
plot(li, Arg);
xlabel('l_{i}');
ylabel('Argument');



%% Problem 2

% L = 30;
% x = L;
% y = L;
% adj = zeros(x*y);
%
% for i=1:x
%     for j=1:y
%         k = sub2ind([x y],i,j);
%         if i>1
%             ii=i-1; jj=j;
%             adj(k,sub2ind([x y],ii,jj)) = 1;
%         end
%         if i<x
%             ii=i+1; jj=j;
%             adj(k,sub2ind([x y],ii,jj)) = 1;
%         end
%         if j>1
```

```
%            ii=i; jj=j-1;
%            adj(k,sub2ind([x y],ii,jj)) = 1;
%        end
%        if j<y
%            ii=i; jj=j+1;
%            adj(k,sub2ind([x y],ii,jj)) = 1;
%        end
%    end
% end
%
% figure(1); imagesc(adj)
% title('Adjacency Matrix');
%
% %% Problem 2 part b
%
% % allowed_spins = [-1 1];
% % spin_matrix = zeros(L);
% % beta = 1;
% % NT = 1;
% % J = 1;
% %
% % % Random spin matrix initialization
% % for i = 1:1:L
% %     for j =1:1:L
% %
% %         spin = randsample(allowed_spins,1,true);
% %         spin_matrix(i,j) = spin;
% %
% %     end
% % end
% %
% % disp('Initial spin matrix is: ')
% % disp(spin_matrix);
% %
% % figure();
% % imagesc(spin_matrix);
% %
% % % Calculate initial energy
% % E =0;
% %
```

```
% % for i = 1:1:L             %% First two loops select one spin in the matrix
% %     for j = 1:1:L         %%
% %
% %         index = sub2ind([x y],i,j);
% %         nbr_sum = 0;
% %
% %         for k = 1:1:L      % these 2 loops search for neighbors in adjacency
matrix
% %             for l = 1:1:L
% %                 if ( adj(index, sub2ind([x y],k,l)) ==1 )
% %                     nbr_sum = nbr_sum + spin_matrix(k,l);
% %                 end
% %             end
% %         end
% %
% %             E = E + (-1*J*spin_matrix(i,j)*nbr_sum);
% %
% %     end
% % end
% %
% % Net_energy = E/2;          % Initial energy
% % disp('Initial Total energy')
% % disp(Net_energy);
% %
% % NT =1e5;
% % N = 1:1:100 ;
% % for t= 1:1:NT
% %
% %     beta = 1*beta;
% %     % Select a random spin to flip
% %     %select_spin_number = 10;
% %     select_spin_number = randsample(N,1,true);
% %     [row,col] = ind2sub([x y],select_spin_number);
% %       h = 0;
% %       for k = 1:1:L       % these 2 loops search for neighbors in adjacency matrix
% %             for l = 1:1:L
% %                 if ( adj(sub2ind([x y],k,l),select_spin_number) ==1 )
% %                     h = h + spin_matrix(k,l);
% %                 end
% %             end
```

```
% %      end
% %
% %      del_E = 2*h*spin_matrix(row,col) ;       % Change in energy expected on
flip of selected spin
% %      gamma = exp(-1*beta*del_E);
% %
% %      random_number = rand;
% %      if (random_number < gamma)
% %         spin_matrix(row,col)= -1*spin_matrix(row,col) ;
% %         Net_energy = Net_energy + del_E ;
% %      end
% %
% %      if (t ==1)
% %         figure();
% %         imagesc(spin_matrix);
% %      end
% %
% %
% %      if (t ==1000)
% %         figure();
% %         imagesc(spin_matrix);
% %      end
% %
% %
% %      if (t ==1e5)
% %         figure();
% %         imagesc(spin_matrix);
% %      end
% %
% % end
% %
% % disp('Timesteps');
% % disp(NT);
% % disp('Updated Total energy')
% % disp(Net_energy);
% % disp('Updated spin matrix is: ')
% % disp(spin_matrix);
%
% %% Problem 2 part c
%
```

```
% allowed_spins = [-1 1];
% spin_matrix = zeros(L);
% beta = 0.01;
% J = 1;
%
% % Random spin matrix initialization
% for i = 1:1:L
%     for j =1:1:L
%
%         spin = randsample(allowed_spins,1,true);
%         spin_matrix(i,j) = spin;
%
%     end
% end
%
% disp('Initial spin matrix is: ')
% disp(spin_matrix);
%
% figure();
% imagesc(spin_matrix);
%
% % Calculate initial energy
% E =0;
%
% for i = 1:1:L             %% First two loops select one spin in the matrix
%     for j = 1:1:L         %%
%
%         index = sub2ind([x y],i,j);
%         nbr_sum = 0;
%
%         for k = 1:1:L      % these 2 loops search for neighbors in adjacency matrix
%             for l = 1:1:L
%                 if ( adj(index, sub2ind([x y],k,l)) ==1 )
%                     nbr_sum = nbr_sum + spin_matrix(k,l);
%                 end
%             end
%         end
%
%             E = E + (-1*J*spin_matrix(i,j)*nbr_sum);
%
```

```
%     end
% end
%
% Net_energy = E/2;           % Initial energy
% disp('Initial Total energy')
% disp(Net_energy);
%
% NT =1e5;
% N = 1:1:L*L ;
% for t= 1:1:NT
%
%     beta = 1.00005*beta;
%     % Select a random spin to flip
%     %select_spin_number = 10;
%     select_spin_number = randsample(N,1,true);
%     [row,col] = ind2sub([x y],select_spin_number);
%       h = 0;
%       for k = 1:1:L       % these 2 loops search for neighbors in adjacency matrix
%            for l = 1:1:L
%               if ( adj(sub2ind([x y],k,l),select_spin_number) ==1 )
%                  h = h + spin_matrix(k,l);
%               end
%            end
%       end
%
%       del_E = 2*h*spin_matrix(row,col) ;        % Change in energy expected on flip
of selected spin
%       gamma = exp(-1*beta*del_E);
%
%       random_number = rand;
%       if (random_number < gamma)
%          spin_matrix(row,col)= -1*spin_matrix(row,col) ;
%          Net_energy = Net_energy + del_E ;
%       end
%
%       if (t ==1)
%          figure();
%          imagesc(spin_matrix);
%       end
%
```

```
%
%       if (t ==1000)
%           figure();
%           imagesc(spin_matrix);
%       end
%
%
%       if (t ==1e5)
%           figure();
%           imagesc(spin_matrix);
%       end
%
% end
%
% disp('Timesteps');
% disp(NT);
% disp('Updated Total energy')
% disp(Net_energy);
% disp('Updated spin matrix is: ')
% disp(spin_matrix);
%
```