Table of Contents

Assignment 1-problem 1

```
clc
clear all

patterns = [12, 24, 48, 70, 100, 120]';  %pattern generation
N = 120; %No of bits
trails = 10^5; %trails
p_error_list= [];

for i = 1 : length(patterns) %loop for different pattern generation
    p_i = patterns(i);   % takaing individual pattern for differnt run
    error_count = 0;   % initialize error count to zero

    for j = 1 : trails  %loop for 10^5 different trails

        p = 2 * randi([0, 1], [N, p_i]) - 1;    % step to generate
 random pattern with +1 or -1

        random_p = randi(p_i);  % selecting random pattern from each
 actual pattern
        random_n = randi(N);   % selecting random neuron from the no
 of bits

        W_i = 1/N * p(random_n,:) * p';   % store a set of p random
 patterns
        %W_i(random_n) = 0;                % setting diagonal weights
 to zero

        S0 = sign(p(random_n,random_p));
        S1 = sign(W_i * p(:,random_p));  % Update single randomly
 chosen neuron

        if S1 == 0
            S1 = 1;          % keeping Signum(0) to Signum(1)
        end

        % Check dynamics and see if correct
        if S0 ~= S1
            error_count = error_count + 1;
        end
    end

    p_error = error_count/trails;
    p_error_list = [p_error_list p_error];
```

```
end
```

Assignment 1-problem2

```
%Recognising digits

clear all;close all; clc;

% loading the five patterns

x1=[ [ -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1,
 1, -1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, 1, 1, 1, -1,
 -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1,
 -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1,
 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1,
 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1,
 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1,
 -1],[ -1, -1, -1, -1, -1, -1, -1, -1, -1, -1] ];

x2=[ [ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1,
 -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1,
 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1,
 -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],
[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1,
 -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1,
 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1,
 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1,
 -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1] ];

x3=[ [ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1,
 -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1,
 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1,
 -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ 1,
 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1,
 1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1,
 -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1,
 -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, 1, 1,
 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1] ];

x4=[ [ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1,
 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1,
 -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1,
 -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1,
 -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],
[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1,
 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1,
 -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, 1,
 1, 1, 1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1] ];

x5=[ [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1,
 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1,
 -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1,
 -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1,
 1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1,
 -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1,
 -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1,
 -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1,
 -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1] ];

%loading pattern for respective questions

p_1 = [[-1, -1, 1, 1, 1, 1, 1, 1, -1, -1], [-1, -1, 1, 1, 1, 1, 1, 1,
 1, -1], [-1, -1, -1, 1, 1, 1, 1, 1, 1, -1], [-1, -1, -1, 1, -1, -1,
 1, 1, 1, -1], [-1, -1, -1, 1, -1, -1, 1, 1, 1, -1], [-1, -1, -1, 1,
 -1, -1, 1, 1, 1, -1], [-1, -1, -1, 1, -1, -1, 1, 1, 1, -1], [-1, -1,
 1, 1, 1, 1, 1, 1, -1, -1], [-1, -1, 1, 1, 1, 1, 1, 1, -1, -1], [-1,
 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
 [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1,
 -1], [-1, -1, 1, 1, 1, 1, 1, 1, 1, -1], [-1, -1, 1, 1, 1, 1, 1, 1, 1,
 -1], [-1, -1, 1, 1, 1, 1, 1, 1, -1, -1]];

p_2 = [[-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1,
 -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1,
 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1,
 -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1],
 [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1,
 -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1,
 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1,
 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1,
 -1, 1, 1, 1, 1, -1, -1, -1], [1, 1, 1, -1, -1, -1, -1, 1, 1, 1]];

p_3 = [[-1, -1, -1, -1, -1, -1, -1, -1, 1, 1], [-1, -1, -1, -1, -1,
 -1, -1, -1, 1, 1], [1, 1, 1, 1, 1, -1, -1, -1, 1, 1], [1, 1, 1, 1,
 1, -1, -1, -1, 1, 1], [1, 1, 1, 1, 1, -1, -1, -1, 1, 1], [1, 1, 1, 1,
 1, -1, -1, -1, 1, 1], [1, 1, 1, 1, 1, -1, -1, -1, 1, 1], [-1, -1, -1,
 -1, -1, -1, -1, -1, 1, 1], [-1, -1, -1, -1, -1, 1, 1, 1, -1, -1], [1,
 1, 1, -1, -1, -1, -1, -1, -1, -1], [1, 1, 1, -1, -1, -1, -1, -1, -1,
 -1], [1, 1, 1, -1, -1, -1, -1, -1, -1, -1], [1, 1, 1, -1, -1, -1, -1,
 -1, -1, -1], [1, 1, 1, -1, -1, -1, -1, -1, -1, -1], [1, 1, 1, 1, 1,
 1, 1, 1, -1, -1], [1, 1, 1, 1, 1, 1, 1, 1, -1, -1]];

p_stored = [x1',x2',x3',x4',x5'];  %stored pattern contains
 x1,x2,x3,x4,x5

p_feed = [p_1',p_2',p_3'];   %pattern to be feeded

N = length(x1);        % N-No of bits should be length of feed pattern

p_output = zeros(size(p_feed));   % output of the pattern

digit_output = zeros(1,size(p_feed,2));  % output of the digits

W = zeros(N);         % initializing weight matrix

for k = 1:size(p_stored,2)     % running for loop to add stored
 patterns in weight matrix
```

```
    W = W + p_stored(:,k)*p_stored(:,k)';
end

W = W.*(1/N);    % Normalize the weight matrix divided by N
W = W - diag(diag(W));    % step to diagonalize the weight matrix to 0

for i = 1:size((p_feed),2) % Loop to run inside the distorted pattern
    s = p_feed(:,i);
    s_old = zeros(size(s));

    while s ~= s_old       %loop inorder to check it reach the steady
 state or not
        s_old = s;
        for j = 1:length(s)   %loop for asyncronous update
            z = W(j,:)*s;
            S = sign(z);
            if S == 0
                S = 1;
            end
            s(j) = S;
        end
    end
    p_output(:,i) = s;


    for k = 1:size((p_stored),2)
        if s == p_stored(:,k)
            digit_output(i) = k;
            break
        elseif s == (-p_stored(:,k))
            digit_output(i) = -k;
            break
        elseif k == size((p_stored),2)
            digit_output(i) = k+1;
        end
    end
end

for i = 1:size(p_output,2)   %loop to check the output manually
    reshape(p_output(:,i),[10,16])';
    %disp([' classified digit of
 Q',num2str(i),num2str(digit_output(i))])
    %disp([' steady state pattern of Q',num2str(i)])
end
```

Assignment 1 - problem 3

```
% Stochastic Hopfield network

clear all;close all; clc;

%given parameters
```

```
T = 2*10^5;
N = 200; % neurons to use
p = 7; % random patterns to store

mu = zeros(1,100);

for i = 1:100     %in order to repeat the experiment for 100 times

    rand_patterns = 2 * randi([0, 1], [N, p]) - 1;   % generate random
 patterns of row N and column p

    W = zeros(N, N);  % creating weight matrix of N*N

    for j = 1 : p           %steps to store the patterns in the
 network
        W = W + rand_patterns(:, j) * rand_patterns(:, j)';  %hebbs
 rule
    end

    W = W / N; % Normalize the weight matrix
    W = W - diag(diag(W)); % diagonal elements to zero, comment to get
 ans for 3a

    S0 = rand_patterns(:,1);

    m1_T = zeros(1,T);  %initialise order parameter to zeros of 1*T

    for t = 1 : T   %for loop to get the order parameters
        S1 = S0;
        ni = randi(N);   %making random N
        bi = W(ni,:) * S0;
        probability = sigmf(bi, [4,0]);
        S1(ni) = randsrc(1, 1, [1,-1; probability, 1-probability]);
        m1_T(t) = 1/N * S1' * rand_patterns(:,1);
        S0 = S1;
    end
    mu(i) = 1 / T * sum(m1_T);    %order parameter or finite time
 average
end

m1_T_avg = 1/100 * sum(mu);  %resulting average order parameter
```

Published with MATLAB® R2020a