LAB 6: LDPC Decoding

Student ID: 202101491

Name: Nancy Patel

This code was developed by Shaan Patel (202101259) and myself, Nancy Patel (202101491).

Honor Code:

I, Nancy Patel (202101491) (along with help of Shaan Patel (202101259)) declare that

• The work I am presenting is my work.

• I have not copied the work that someone else has done.

• Concepts understandings and insights I will be describing are my own

• Whenever I have relied on an existing work that is not my own, I have provided a proper reference citation.

• I make this pledge truthfully. I know that the violation of the pledge can carry grave consequences.

Q 1. Implement the above algorithm to decode the rate 1/4 (n = 12, k = 3, u = 9) LDPC code, whose parity check matrix H is listed in Eq. 1 on Page 2 of HW 2. (Hard decoding)

```
%% for hmatrix 9x12 use the following code
 h_mat=[ 1 0 0 0 0 1 0 1 0 1 0 0;
         1 0 0 1 1 0 0 0 0 0 1 0;
         0 1 0 0 1 0 1 0 1 0 0 0;
         0 0 1 0 0 1 0 0 0 0 1 1;
         0 0 1 0 0 0 1 1 0 0 0 1;
         0 1 0 0 1 0 0 0 1 0 1 0;
         1 0 0 1 0 0 1 0 0 1 0 0;
         0 1 0 0 0 1 0 1 0 1 0 0;
         0 0 1 1 0 0 0 0 1 0 0 1];

%reading from the Hmatrix9x12 text file:
%{
mat = fileread('Hmatrix9x12.txt');
mat = strtrim(mat);
rows = strsplit(mat, '\n');
h_mat = zeros(length(rows), length(strsplit(rows{1},' ')));

for i = 1:length(rows)
```

```
    values = strsplit(rows{i}, ' ');
    for j = 1:length(values)
        h_mat(i, j) = str2double(values{j});
    end
end
%}

%% for HMatrix.mat and HMatrix2.mat use this code:
%data=load('Hmatrix2.mat');
%h_mat=data.H;

COL = length(h_mat(1,:));
ROW = length(h_mat(:,1));

deg_CN = 0; % Degree of Cn
deg_VN = 0; % Degree of Vn

%Calculating Degree of CNs(no. of 1's in a row) and VNs(no. of 1's in a column)

for i = 1:COL
    if h_mat(1,i) == 1
        deg_CN = deg_CN + 1;
    end
end

for j = 1:ROW
    if h_mat(j,1) == 1
        deg_VN = deg_VN + 1;
    end
end

% storing the indexes of the VNs connected to a particular CN
VNs_conn_to_CN = -1*ones(ROW, deg_CN);

%storing the indexes of the CNs connected to a particular VN
CNs_conn_to_VN = -1*ones(COL, deg_VN);

% filling VNs_conn_to_CN
for i = 1:ROW
    k = 1;
    for j = 1:COL %keeping the row constant(CN) and iterating over columns(VNs)
to check for the 1's in a single row
        if h_mat(i,j) == 1
            VNs_conn_to_CN(i,k) = j;
            k = k + 1;
        end
```

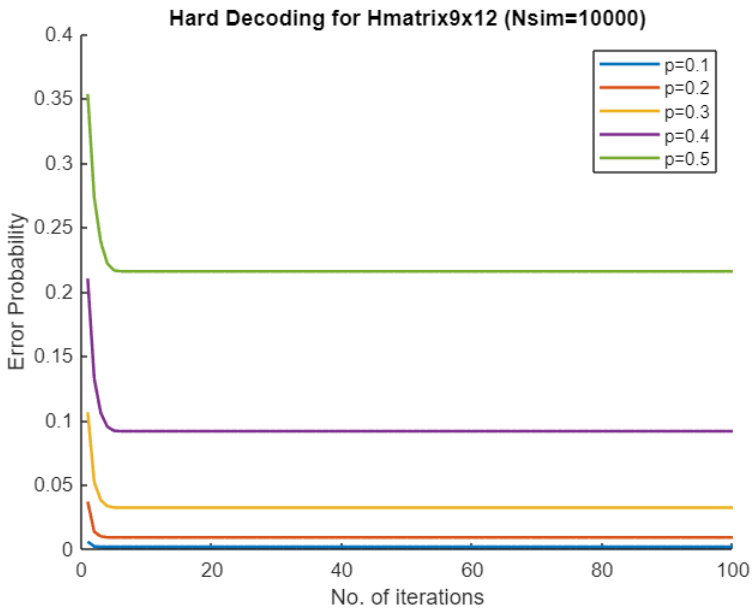
```
    end
end

%  filling CNs_conn_to_VN
for i = 1:COL
    k = 1;
    for j = 1:ROW % keeping the column constant(VN) and iterating over
rows(CNs) to check for the 1's in a single column
        if h_mat(j,i) == 1
            CNs_conn_to_VN(i,k) = j;
            k = k + 1;
        end
    end
end

p = 0;
Nsim = 10000;
P_success = [];
iter = 1:1:100;
num_iter = size(iter);
deltap=0.02;
Prob = 0:deltap:1;


fprintf('\nHMatrix with Rows: %d and Columns: %d', ROW, COL);
```

```
HMatrix with Rows: 9 and Columns: 12
```

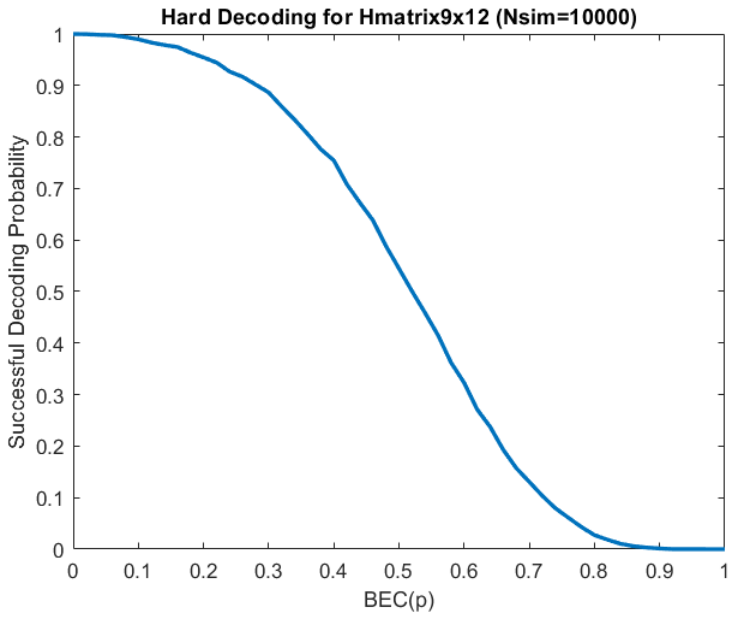
```
%fprintf('\nBEC(p)\tProbability of Success(LDPC)\n');

while p < 1.01
    No_of_err = 0;
    overall_err = zeros(num_iter);
    for ksim = 1:Nsim

        msg_from_bec = zeros(1,COL); %initially the msg is all 0's

        % For storing message that each CN will sent to VNs connected with it
        msg_from_CN_to_VNs = -1*ones(ROW,COL);

        % For storing message that each VN will sent to CNs connected with it
        msg_from_VN_to_CNs = -1*ones(ROW,COL);

        decoded_msg = zeros(1,COL); % final decoded msg will be stored here
```

```
        %noise introduced by the BEC channel with probability p
        for i = 1:length(msg_from_bec)
            % Generate a random number between 0 and 1
            r = rand();

            if r < p
                msg_from_bec(i) = -1; %erasure bit introduced
            end
        end

        decoded_msg = msg_from_bec;

        %Loading the VNs with the msg received from BSC channel that will be
sent to CNs
        for i = 1:ROW
            msg_from_VN_to_CNs(i,:) = msg_from_bec;
        end

        % iterations will stop if it_ind=100 or we receive the original msg of
all 0's which is checked by the function 'check'
        it_ind = 1;
        err_in_each_iter = zeros(num_iter);
        while it_ind <= num_iter(2) && ~check_all_zero(decoded_msg)

            % Sending the msg from CNs to connected VNs
            for i = 1:ROW
                for t = 1:deg_CN
                    msg_from_CN_to_VNs(i, VNs_conn_to_CN(i, t)) =
SPC_decoding(VNs_conn_to_CN(i, t), VNs_conn_to_CN, msg_from_VN_to_CNs, i,
deg_CN);
                end
            end

            % Sending the msg from VNs to connected CNs
            for j = 1:COL
                for t = 1:deg_VN
                    msg_from_VN_to_CNs(CNs_conn_to_VN(j, t), j) =
Majority_decoding(CNs_conn_to_VN(j, t), msg_from_bec(j), CNs_conn_to_VN,
msg_from_CN_to_VNs, j, deg_VN);
                end

                %fixing the value of the VNs after one complete msg passing
from CN to VN and VN to CN
                decoded_msg(j) = Majority_decoding(-1, msg_from_bec(j),
CNs_conn_to_VN, msg_from_CN_to_VNs, j, deg_VN);
            end
```

```
            err_in_each_iter(it_ind) = count_error(decoded_msg);
            it_ind = it_ind + 1;
        end
        overall_err = overall_err + err_in_each_iter;

        if ~isequal(decoded_msg, zeros(size(decoded_msg)))
            No_of_err = No_of_err + 1; %if the final msg is still not [0 0 ...
0]-->error present
        end

    end

    overall_err = overall_err./(COL*Nsim);
    hold on
    if (abs(p - 0.1) < eps || abs(p - 0.2) < eps || abs(p - 0.3) < eps || abs(p
- 0.4) < eps || abs(p - 0.5) < eps)
        plot(iter,overall_err,LineWidth=1.5);
    end

    %Probability of Success = 1 - Probability of Error
    P_success(end+1) = 1 - No_of_err / Nsim;

    %fprintf('\n%.2f\t\t%.4f', p, P_success(end));
    p = p + deltap;
end
ylabel('Error Probability');
xlabel('No. of iterations');
title('Hard Decoding for Hmatrix9x12 (Nsim=10000)')
legend('p=0.1','p=0.2','p=0.3','p=0.4','p=0.5');
hold off
```




```
%fprintf('\n\n\t P_success : ');
%for u = 1:numel(P_success)
%   fprintf(' %f', P_success(u));
%end
%fprintf('\n');


figure
plot(Prob,P_success,LineWidth=2);
ylabel('Successful Decoding Probability');
xlabel('BEC(p)');
title('Hard Decoding for Hmatrix9x12 (Nsim=10000)');
```




```
function cnt = count_error(msg)
cnt = 0;
for i = 1:length(msg)
    if msg(i) == -1
        cnt = cnt + 1;
    end
end
end

function result = check_all_zero(msg)
for i = 1:length(msg)
    if msg(i) ~= 0
        result = false;
        return
    end
end
result = true;
end

function result = SPC_decoding(vn, VNs_conn_to_CN, msg_from_VN_to_CNs, ind,
deg_CN)
s = 0;
for i = 1:deg_CN
    if VNs_conn_to_CN(ind,i) == vn %ignore the same vn
        continue;
    elseif msg_from_VN_to_CNs(ind,VNs_conn_to_CN(ind,i)) == -1
        result = -1;
        return;
```

```
    else
        s = s + msg_from_VN_to_CNs(ind,VNs_conn_to_CN(ind,i));
    end
end
result = mod(s, 2);
end

function result = Majority_decoding(cn, value, CNs_conn_to_VN,
msg_from_CN_to_VNs, ind, deg_VN)
Count = 0;
count_of_0s = 0;
count_of_1s = 0;
if (value == -1)
    Count = Count + 1;
end
for i = 1:deg_VN
    if CNs_conn_to_VN(ind,i) == cn %ignore the same cn
        continue;
    elseif msg_from_CN_to_VNs(CNs_conn_to_VN(ind,i),ind) == -1
        Count = Count + 1;
    elseif msg_from_CN_to_VNs(CNs_conn_to_VN(ind,i),ind) == 0
        count_of_0s = count_of_0s + 1;
    elseif msg_from_CN_to_VNs(CNs_conn_to_VN(ind,i),ind) == 1
        count_of_1s = count_of_1s + 1;
    end
end

if Count == deg_VN + 1 %all bits are erasure
    result = -1;
else
    if count_of_0s == count_of_1s
        result = value;
    else
        if count_of_0s > count_of_1s
            result = 0;
        else
            result = 1;
        end
    end
end
end
```