ONE STEP ERROR PROBABILITY

Note: This is the code for one step error probability with diagonal weights equal to 0.

```
% One-step error probability ( Weight matrix = 0)
clc;
clear all;


% Variable Initialization

nums_of_patterns = [12, 24, 48, 70, 100, 120];
N = 120;
n_Trials = 10^5;
error_prob_vec = [];

for i = 1 : length(nums_of_patterns)
    num_patterns = nums_of_patterns(i);
    %disp(num_patterns)
    no_of_errors = 0;

    for j = 1 : n_Trials
        patterns_vec = 2 * randi([0, 1], num_patterns, N)-
1;

% calculate the weights (Hebb's rule)

        weight_matrix = zeros(N,N);
        for k = 1:num_patterns
            weights = patterns_vec(k, :)'*patterns_vec(k, :
);

            weight_matrix = weight_matrix + weights;
        end
        weight_matrix = weight_matrix/N;
        weight_matrix = weight_matrix -
diag(diag(weight_matrix));
        %disp(weight_matrix)

% selecting a random stored pattern
```

```
        pattern_rindex = randi(num_patterns);
        input_pattern = patterns_vec(pattern_rindex, : );
% Selecting a random bit
        random_bit = randi(N);

        LocalField = weight_matrix * input_pattern()';
        if LocalField(random_bit) == 0
            LocalField(random_bit) = 1;
        end
% updating the bits
        updated_bit = sign(LocalField(random_bit));

% Check dynamics
        tran_input_pattern = input_pattern';
        if updated_bit ~= tran_input_pattern(random_bit)
            no_of_errors = no_of_errors + 1;
        end
        %disp(updated_bit)
        %disp(tran_input_pattern(random_bit))
    end
% Error Calculation (Check the bit)
    error_prob = no_of_errors/n_Trials;
    error_prob_vec = [error_prob_vec ,error_prob]
end
```

Note: This is the code for one step error probability with diagonal weights not equal to 0.

```
% One-step error probability ( Weight matrix = 0)
clc;
clear all;


% Variable Initialization

nums_of_patterns = [12, 24, 48, 70, 100, 120];
N = 120;
n_Trials = 10^5;
error_prob_vec = [];

for i = 1 : length(nums_of_patterns)
    num_patterns = nums_of_patterns(i);
    %disp(num_patterns)
    no_of_errors = 0;

    for j = 1 : n_Trials
        patterns_vec = 2 * randi([0, 1], num_patterns, N)-
1;

% calculate the weights (Hebb's rule)

        weight_matrix = zeros(N,N);
        for k = 1:num_patterns
            weights = patterns_vec(k, :)'*patterns_vec(k, :
);

            weight_matrix = weight_matrix + weights;
        end
        weight_matrix = weight_matrix/N;
        %disp(weight_matrix)

% selecting a random stored pattern

        pattern_rindex = randi(num_patterns);
        input_pattern = patterns_vec(pattern_rindex, : );
% Selecting a random bit
        random_bit = randi(N);

        LocalField = weight_matrix * input_pattern()';
        if LocalField(random_bit) == 0
```

```
            LocalField(random_bit) = 1;
        end
% updating the bits
        updated_bit = sign(LocalField(random_bit));

% Check dynamics
        tran_input_pattern = input_pattern';
        if updated_bit ~= tran_input_pattern(random_bit)
            no_of_errors = no_of_errors + 1;
        end
        %disp(updated_bit)
        %disp(tran_input_pattern(random_bit))
    end
% Error Calculation (Check the bit)
    error_prob = no_of_errors/n_Trials;
    error_prob_vec = [error_prob_vec ,error_prob]
end
```