# STOCHASTIC HOPFIELD NETWORK

## Table of Contents



# Initial setup

```
clear, clc
beta = 2;
N = 200;
T = 2e5;
number_of_updates = T/N;
number_of_tests = 100;
```

# For p = 7

```
p = 7;
m=zeros(1,number_of_tests);
for n_test = 1:number_of_tests
    patterns = generate_patterns(p,N);
    weight_matrix = generate_weight_matrix_zero_diag(patterns);
    initial_pattern = patterns(:,1);
    state = initial_pattern;
    sum_t = 0;
    for t = 1:number_of_updates
        for neuron_number = 1:N
            state = update_neuron_asynchronously(weight_matrix, state,
 neuron_number);
            sum_t = sum_t + 1/N*sum(state.*initial_pattern);
        end
    end
    m(n_test) = sum_t/T;
end
m_average = mean(m);
disp(strcat("The average m_1 for p = 7 is ", num2str(m_average)))
```

```
The average m_1 for p = 7 is 0.85115
```

# For p = 45

```
p = 45;
m=zeros(1,number_of_tests);
for n_test = 1:number_of_tests
    patterns = generate_patterns(p,N);
```

```
    weight_matrix = generate_weight_matrix_zero_diag(patterns);
    initial_pattern = patterns(:,1);
    state = initial_pattern;
    sum_t = 0;
    for t = 1:number_of_updates
        for neuron_number = 1:N
            state = update_neuron_asynchronously(weight_matrix, state,
 neuron_number);
            sum_t = sum_t + 1/N*sum(state.*initial_pattern);
        end
    end
    m(n_test) = sum_t/T;
end
m_average = mean(m);
disp(strcat("The average m_1 for p = 45 is ", num2str(m_average)))
```

```
The average m_1 for p = 45 is 0.11416
```

# Functions

```
disp('')
function g = g(b)
    beta = 2;
    g = 1./(1+exp(-2*beta*b));
end

function neuron_value = generate_neuron_value(b)
    x = rand;
    if x < g(b)
        neuron_value = 1;
    else
        neuron_value = -1;
    end
end

function weight_matrix = generate_weight_matrix_zero_diag(patterns)
    weight_matrix = patterns*patterns'/size(patterns,1);
    weight_matrix = weight_matrix - diag(diag(weight_matrix));
end

function patterns = generate_patterns(p, N)
    % Generates a p by N matrix with p N-bit patterns as columns
    patterns = 2.*randi([0 1],N, p) - 1;
end

function state = update_neuron_asynchronously(weight_matrix, state,
 neuron_number)
    b = weight_matrix(neuron_number,:)*state;
    state(neuron_number) = generate_neuron_value(b);
end
```

*Published with MATLAB® R2020a*