```
nbrOfBits = 200;
nbrOfTrials = 100;
nbrOfUpdates = 10^5;
beta = 2;

nbrOfStoredPatterns = 40; %5;
m_1 = zeros(nbrOfTrials, 1);

for trial = 1:nbrOfTrials
    % Assign random patterns
    randomPatternsVec = randi(2, nbrOfBits, nbrOfStoredPatterns);
    randomPatternsVec(randomPatternsVec == 2) = -1;
    feedPatternInd = 1;
    feedPattern = randomPatternsVec(:, feedPatternInd);

    % Calculate the weight matrix
    W = AssignWeights(nbrOfBits, randomPatternsVec);

    % Update first pattern asynchronously <nbrOfUpdates> times
    s_new = feedPattern;
    for t = 1:nbrOfUpdates
        i_rndm = randi(nbrOfBits);
        b = W(i_rndm, :) * s_new;
        g_b = 1 / (1 + exp(-2 * beta * b));
        if rand(1) <= g_b
            s_new(i_rndm) = 1;
        else
            s_new(i_rndm) = -1;
        end
        m_1(trial) = m_1(trial) + s_new' * feedPattern;
    end

    m_1(trial) = m_1(trial) / (nbrOfBits * nbrOfUpdates);
    disp(trial)
end
mEnsembleAvg = sum(m_1) / nbrOfTrials;

function W = AssignWeights(nbrOfBits, randomPatternsVec)
    W = randomPatternsVec * randomPatternsVec' ./ nbrOfBits;
    W = W.*~eye(nbrOfBits);
end
```