1. **Hopfield Network**

Implement an associative neural network based on \( N \) discrete McCulloch-Pitts neurons, i.e. each neuron has only two states \( v_i = \pm 1 \) and its temporal evolution occurs in discrete steps based on the total input to the neuron

\[
v_i \rightarrow \text{sign} \left( \sum_j M_{ij} v_j \right).
\]

Use asynchronous update, i.e. update one, randomly selected neuron at a time.

The weights \( M_{ij} \) are chosen to store \( k \) patterns \( \hat{v}^{(\mu)}_i \)

\[
M_{ij} = \frac{1}{N} \sum_{\mu=1}^{k} \hat{v}^{(\mu)}_i \hat{v}^{(\mu)}_j.
\]

Plot the state every \( N \) iterations. To keep track of the convergence plot also the difference between these states. Plot also the stored patterns and the deviation of the final state \( v_i^\infty \) from all the stored patterns.

**Note:** for visualization purposes it is good to arrange the neurons in a square, i.e. as a matrix (using `reshape`).

(a) Generate \( k \) random patterns in which \( v_i = +1 \) and \( v_i = -1 \) have equal probability. Investigate the basin of attraction of these stored patterns. Use as initial condition a perturbation of one of these pattern \( \hat{v}^{(\mu)}_i \) by flipping the sign of the neuronal activity \( v_i \) with probability \( p \). Measure the difference between the final state that is reached with this initial condition and the pattern that was used to generate the initial condition. Plot, as a function of \( p \) with \( 0 \leq p \leq 0.8 \), the quantity \( \bar{v}_{\text{diff}}(p) = \frac{1}{N} \sum_i |v_i^\infty - \hat{v}_i^{(\mu)}| \).

Since the patterns are generated randomly and the updating contains also a random element you will need to average \( \bar{v}_{\text{diff}} \) over multiple runs (10 or more). For these production runs you better turn off the graphics. Generate the graphs \( \bar{v}_{\text{diff}}(p) \) for \( N = 16, 64, 225 \) and for \( k = 3, 6, 10 \). Comment on the ability of the network to retrieve the pattern as a function of the parameters.

(b) In general the patterns to be stored are not random. Correlations may deteriorate the performance. Investigate this aspect by using the patterns provided on the class web site. They are tif-files that you can read in using

```matlab
PLEVEL=160 \% YOUR CHOICE
PATT2D=DOUBLE(IMREAD(FILE));
PATT2D=SIGN(PATT2D-PLEVEL);
```
FOR IX=1:N %THE TIF FILES ARE OF VARYING SIZE, MAKE THEM ALL THE SAME SIZE
FOR IY=1:N
PTEMP(IX,IY)=PATT2D(IX,IY);
END
END
PATTERNS(1,:)=RESHAPE(PTEMP,N*N);
For $p = 0$ and for $p = 0.3$ investigate the recovery of a pattern when 3 patterns are stored and when 4 patterns are stored. How does the performance change?