Problem Set #4: Networks

Hsieh Yu-Guan

June 1, 2017

Introduction

Besides studying the structure and function of a single neuron, it's also important to understand what may happen when neurons communicate between them. In this report we'll thus look at some simple models of neural networks. What will be their dynamics and expressive power? (P.S. we'll ignore all physical units for the whole report.)

1 Some simple networks

1.1 Neuron with autapse

Let's start by working on the simplest model that one can ever imagine: there's only one neuron in the network, and its output feeds back onto itself via a synapse (such a synapse is called an "autapse"). We note x the neuron's firing rate, and it obeys the equation

$$\dot{x}(t) = -x(t) + f(wx(t) + I)$$

where $w = 0.04$ is the strength of the synaptic connection and $I = -2$ is the external (and inhibitory) background input which is constant. Finally, f is the input-output (or activation) function of the neuron having a sigmoidal form and is given by

$$f(s) = 50(1 + \tanh(s))$$

where s is the total input of the neuron.

To see that f is indeed a sigmoidal function, we plot it for the range $s \in [-10, 10]$ as shown in Figure 1. Next, we plot the derivative of the firing rate $\dot{x}$ as a function of x (Figure 2). The form should be easily predictable. The function f is first stretched out and then shifted to the right, before we finally add the linear function $x \mapsto -x$ to it.

We observe three zero-crossings in this graph. Let's call them respectively x_1, x_2 and x_3 with $x_1 < x_2 < x_3$. In fact we get $x_1 \sim 2$, $x_2 = 50$ and $x_3 \sim 98$. They are the fixed points of the dynamics. However, x_1 and x_3 are stable while x_2 is unstable. We can see that if x lies between x_1 and x_2, $\dot{x}$ is negative so x will be "attracted" to x_1, and if x is smaller than x_1, $\dot{x}$ is positive and x will converge to x_1. The same analysis works for x_3.

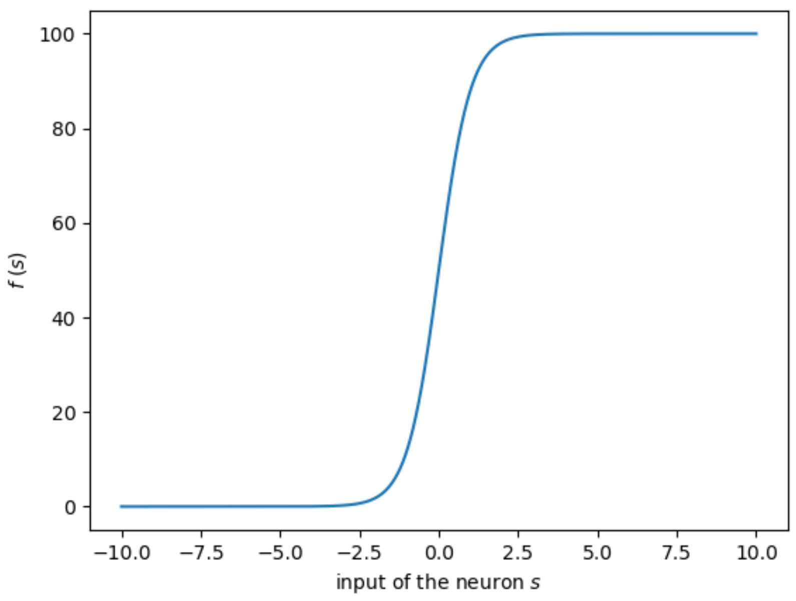

FIGURE 1: The activtion function f of the neuron

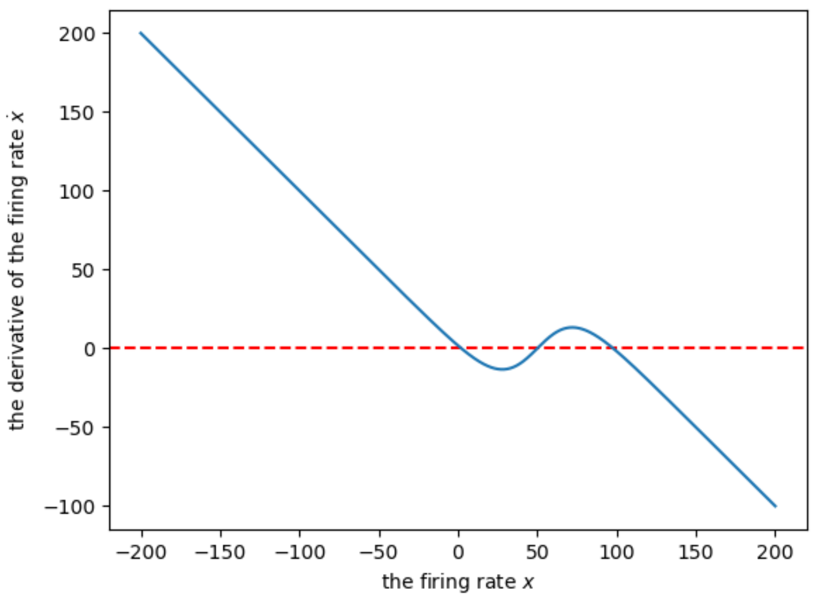

FIGURE 2: $\dot{(x)}$ as a function of x

Now we'll simulate the dynamics of the system by taking a time step $\Delta t = 0.1$ and a total time period $T = 100$. First consider $x(0) = 49$.

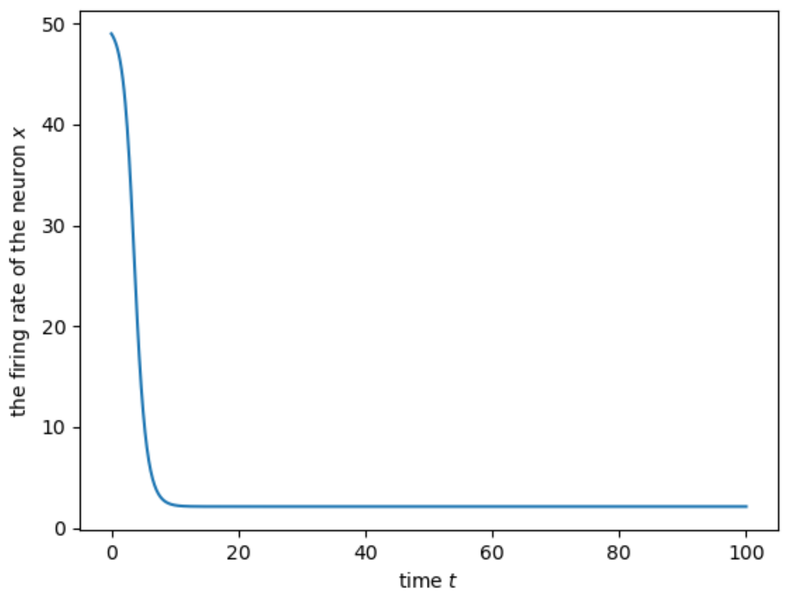

FIGURE 3: The evolution of x for $x(0) = 49$

As predicted before x is attracted to the dynamics attractor x_1. We redo the simulation for $x(0) = 50$.

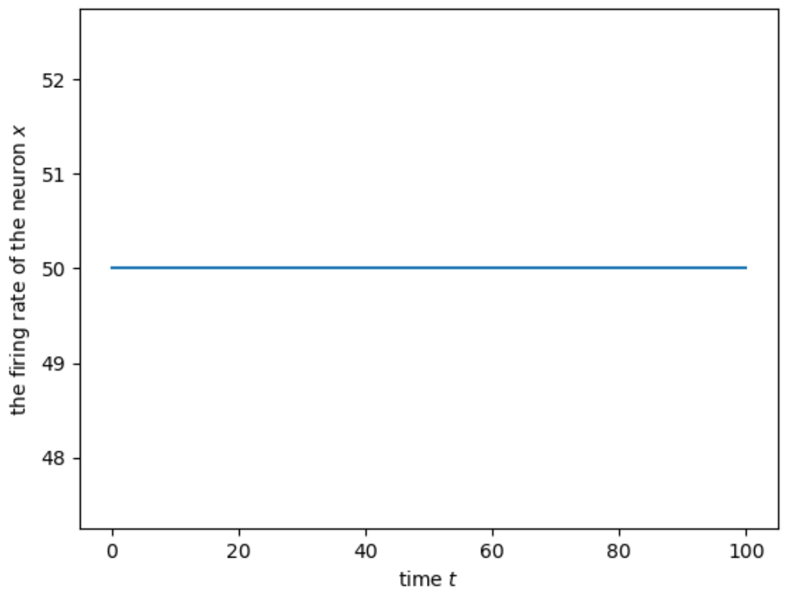

FIGURE 4: The evolution of x for $x(0) = 50$

This time since 50 is itself a fixed point of the dynamics, the system is at equilibrium and the solution doesn't change with time (though 50 is a repeller). Finally let $x(0) = 51$.

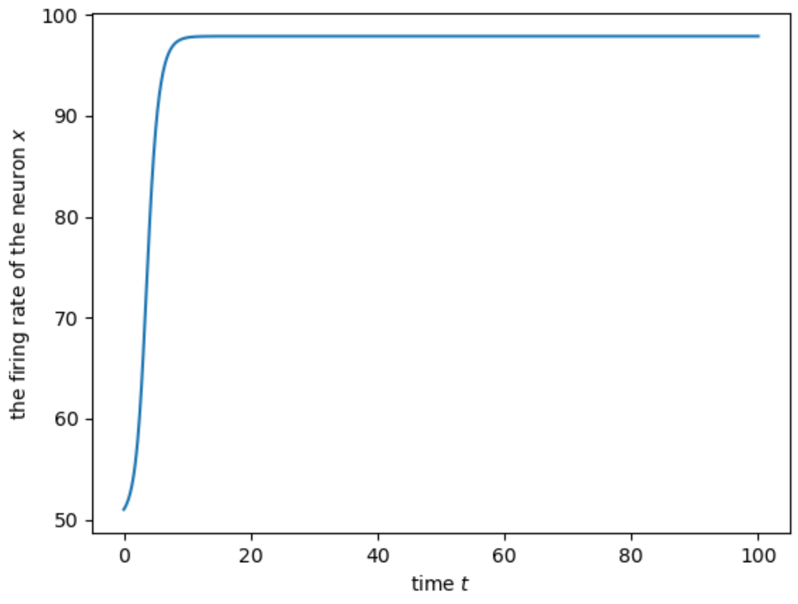

FIGURE 5: The evolution of x for $x(0) = 51$

It's the symmetry of the case $x(0) = 49$. Between the repeller x_2 and the attractor x_3, x evolves towards x_3.

We then add noise to the system, so the differential equation becomes

$$\dot{x}(t) = -x(t) + f(wx(t) + I) + \sigma\eta(t)$$

where $\eta(t)$ is Gaussian white noise with varaiance 1. First we suppose $\sigma = 5$ and we simulate for $x(0) = 49$.

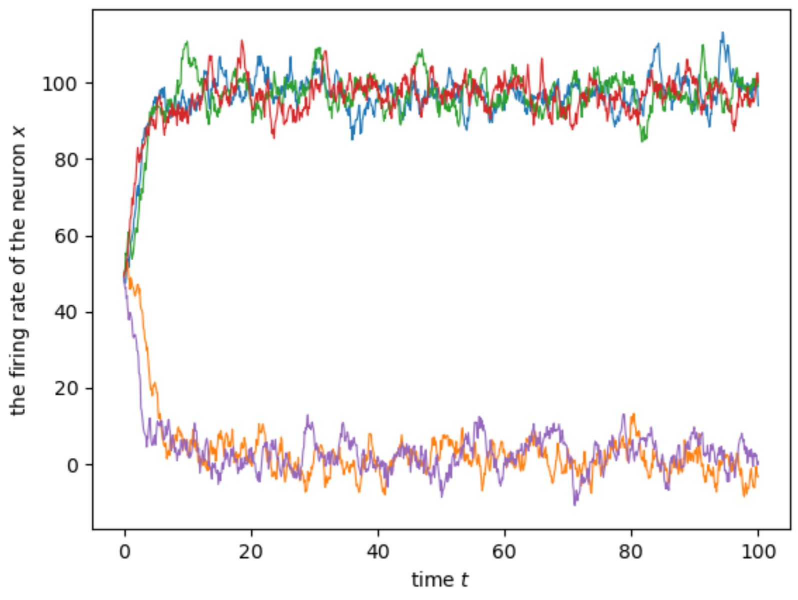

FIGURE 6: The evolution of x with noise $\sigma = 5$ for $x(0) = 49$

We see that there are two different scenarios. With noise we can no longer ensure that the system will converge towards x_1. Since the evolutions of the system are very different for

$x < 50$ and $x > 50$, and 49 is close to 50, slight noise in the model may lead to totaly distinct results. This can be again shown for $x(0) = 50$ and $x(0) = 51$.

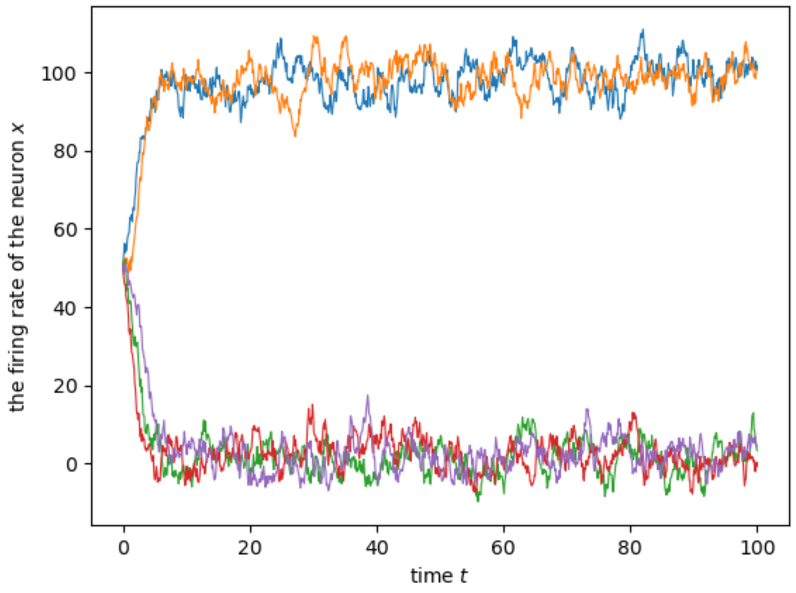

FIGURE 7: The evolution of x with noise $\sigma = 5$ for $x(0) = 50$

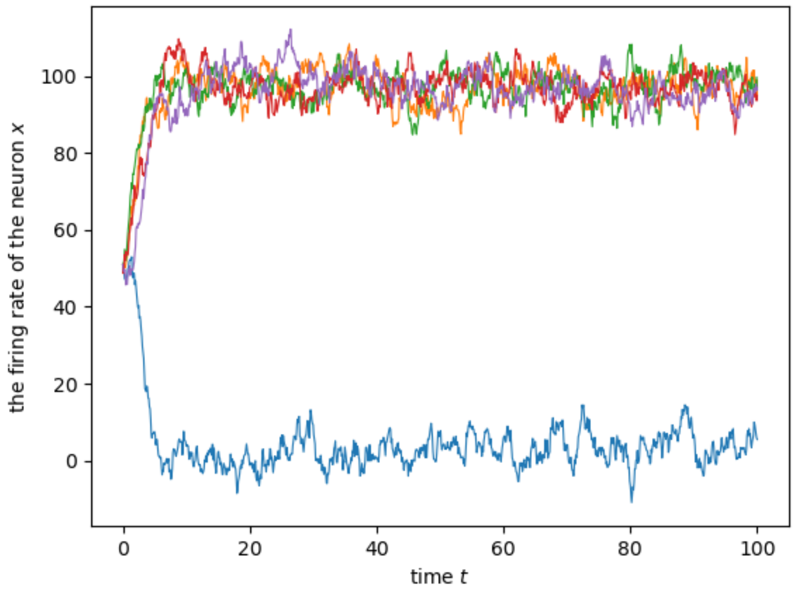

FIGURE 8: The evolution of x with noise $\sigma = 5$ for $x(0) = 51$

Also notice that for $x(0) = 50$, x will not stay anymore at the value 50 because as mentioned before, $x_2 = 50$ is a repeller. In a model with noise, the probability that x is always 50 becomes null.

In all the above examples, though noise can have great influence on the evolution of the system, once x gets far enough from x_2, the evolution is still mainly dominated by the drift term $-x(t) + f(wx(t) + I)$. Nonetheless, this is not the case for a greater noise level, for example, when $\sigma = 80$, as shown in the figure at the top of the next page.

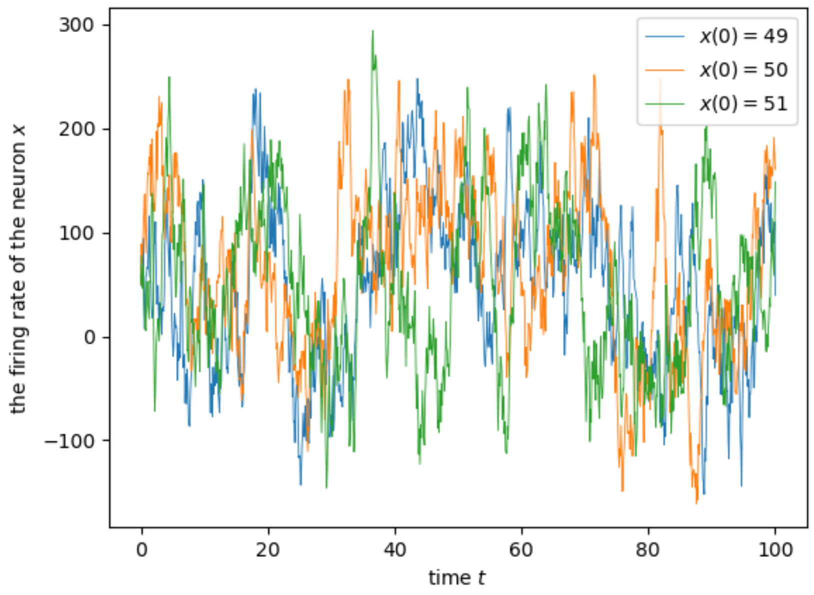

FIGURE 9: The evolution of x with noise $\sigma = 80$ for $x(0) = 49, 50, 51$

As the resulting curves follow almost the same pattern for different initial values of x, I plot all of them on the same graph. It's not worth it to plot several simulations for a same $x(0)$ because we would not be able to see great differences. In short, the evolution of x is donimated by the noise term and becomes just noisy.

1.2 Circuit with mutual inhibition

The second model we'll discuss here is made up of two neurons that are coupled by mutual inhibition. We note the firing rate of the two neurons respectively x_1 and x_2, then the whole system is governed by the differential equations

$$\dot{x}_1 = -x_1(t) + f(wx_2(t) + I)$$
$$\dot{x}_2 = -x_2(t) + f(wx_1(t) + I)$$

where f is defined as before and the inhibitory synaptic weights are given by $w = 0.1$. The external inputs are now excitatory, $I = 5$.

We may also want to use the vector notation that would turn out to be quite useful when the population of neurons gets larger, then the system of differential equations can be put in the form:

$$\dot{\mathbf{x}} = -\mathbf{x}(t) + f(\mathbf{W}\mathbf{x}(t) + \mathbf{I})$$

where $\mathbf{x}$ is the vector of firing rates, $\mathbf{W}$ is the synaptic weight matrix and $\mathbf{I}$ is the vector of input currents. In this particular case, we have

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \qquad \mathbf{W} = \begin{pmatrix} 0 & w \\ w & 0 \end{pmatrix} \qquad \mathbf{I} = \begin{pmatrix} I \\ I \end{pmatrix}.$$

To study the dynamics of this system, we first plot its nullclines, that is, the line for which

$\dot{x}_1(t) = 0$ and the line for which $\dot{x}_2(t) = 0$.

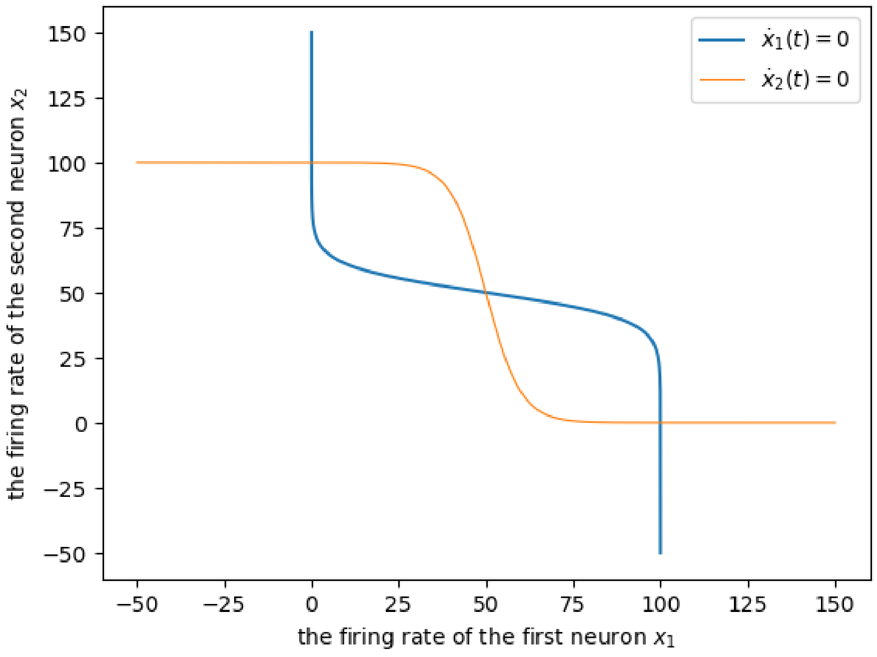

FIGURE 10: The nullclines of the system of two mutual inhibitory neurons

We observe three crossing points of the the two lines. From left to right, we name them respectively z_1, z_2 and z_3 (so $z_2 = (50, 50)$). These are the points such that $\dot{x}_1 = \dot{x}_2 = 0$. In other words, they're the fixed points of the dynamics. The plane is then divided into six zones and in each zone the system evolves in a specific direction. To put it simply, on the left of the blue line we have $\dot{x}_1 > 0$ while on the right $\dot{x}_1 < 0$. Similarly, below the orange line we get $\dot{x}_2 > 0$ whereas above it $\dot{x}_2 < 0$.

To better undestand what this implies, we simulate the system and plot the evolution of the firing rates for different initial conditions in the figure below.

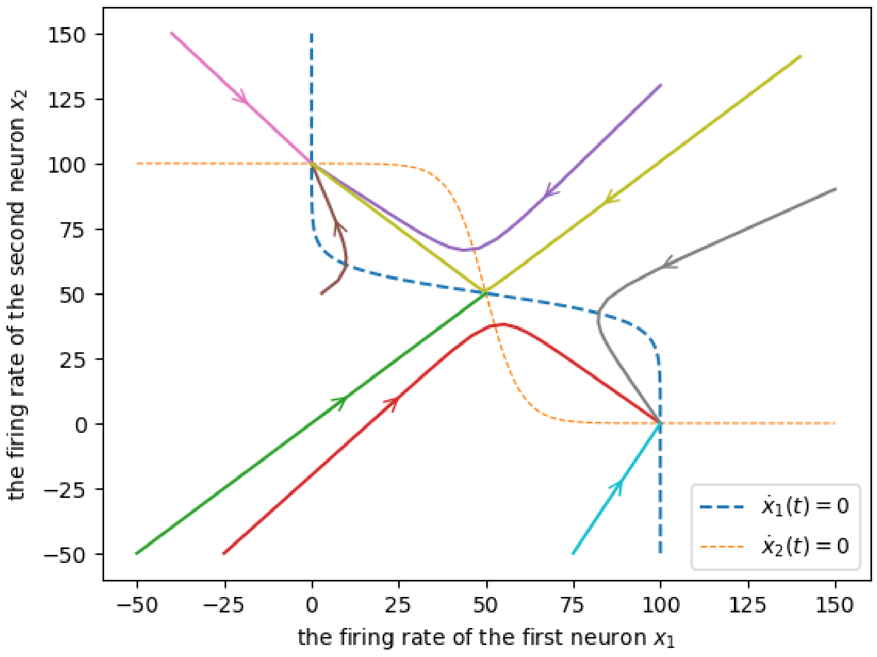

FIGURE 11: The evolution of the firing rates for different initial conditions

The initial conditions that are considered here are $(-50, -50)$, $(-40, 150)$, $(-25, 50)$, $(3, 50)$, $(75, -50)$, $(100, 130)$, $(140, 141)$ and $(150, 90)$. First we notice that all of the simulations end up in some fixed point and the directions of the evolutions follow roughly what is described above. Further, it seems that (but without rigorous mathematical proof here) given the initial condition $(x(0), y(0))$, if $x(0) < y(0)$ the system evolves to z_1; if $x(0) = y(0)$ the system converges to z_2; finally if $x(0) > y(0)$ the system moves to z_3.

The yellow arrow serves as quite a good example: the initial condition is $(150, 151)$, and the system gets once very clear to z_2 but then it again leaves away from this point and converges to z_1 at the end. As a result, being fixed point, z_1 and z_3 are stable while z_2 is unstable. Finally, a plot of the vector field of derivatives can better explain all of this.

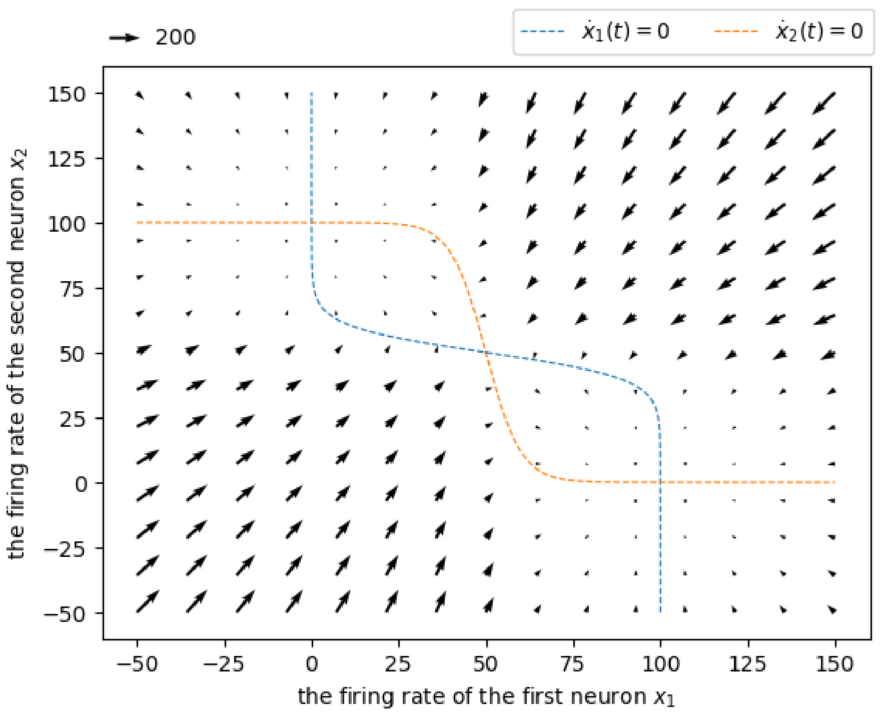

FIGURE 12: Derivatives at different points of the system

1.3 Conclusion

In this section, we have investigated two relatively simple networks and spent time studying their dynamics. Several simulations have also been carried out. We saw that there were diffrent kinds of fixed points of dynamics and if the external currents were constant, the system would evolve towards some final state (which was often an attractor). Finally, the presence of noise may more or less affect the evolution of the system.

2 Hopfield Network

2.1 Description

In the rest of the report we'll be interested in the continuous Hopfield model. The dynamics of the network is given the differential equation

$$\dot{\mathbf{x}} = -\mathbf{x}(t) + f(\mathbf{W}\mathbf{x}(t)) + \sigma\eta(t).$$

The vector notation is adopted and a noise is also introduced. Throughout the whole section, we set the dimension of $\mathbf{x}$ to be $N = 64$ and thus $\mathbf{W}$ is a 64×64 matrix. For the sake of simplicity, we will take $f(x) = \text{sgn}(x)$ for the moment being, and we use the convention that $\text{sgn}(0) = 0$. As a result, $\mathbf{x}$ is typically a vector of reels between -1 and 1.

A such network can act as an associative memory system. For example, to store one pattern $\mathbf{p}$ we set the weight matrix to

$$\mathbf{W} = \frac{1}{N}\mathbf{p}\mathbf{p}^\top.$$

Then it's quite easy to see that $\mathbf{p}$ is a fixed point of the dynamics. For illustration purpose, we'll plot the activity of the neurons as a figure containing $[8 \times 8]$ cases.

2.2 Simulations

To start we say that the network is meant to store just one pattern (Figure 13 - A) and the noise level is $\sigma = 0.1$. As shown in Figure 13 - B, the system may converge to $\mathbf{p}$. However, chances are that the it evolves towards $-\mathbf{p}$ (Figure 13 - C). We can easily show that $-\mathbf{p}$ is indeed also a fixed point of the dynamics, but since it's not a pattern that we aim to store, it's called a spurious state. Looking at the differential equation, one may notice that there is still another fixed point of the system: when $\mathbf{x} = \mathbf{0}$. Nonetheless, unlike the two previous fixed points, this one is not stable and noise can bring the system away from it (Figure 13 - D).

To store more patterns, say, M patterns from $\mathbf{p}_1$ to $\mathbf{p}_M$, the general rule is to construct the weight matrix as

$$\mathbf{W} = \frac{1}{N}\sum_{i=1}^{M}\mathbf{p}_i\mathbf{p}_i^\top.$$

For example in Figure 14 we store two patterns $\mathbf{p}$ and $\mathbf{q}$ in the network. We start from initial conditions that are respectively "close" to $\mathbf{p}$, $-\mathbf{p}$, $\mathbf{q}$ and $-\mathbf{q}$. By close we mean that we change the activity of a relative small number of neurons to its opposite value (from 1 to -1 and from -1 to 1). We see that the network is always able to evolve to the right fixed point.

However, surely we cannot store an unlimited number of patterns in the network. The network capcity depends on the explicit patterns that we want to store. In Figure 15 we try to store six different patterns in the Hopfield network, but it doen't work perfectly. In Figure 15 - B we start from a state that is close to the first pattern we'd like to store and we manage to retrieve it at the end. On the contrary, sometimes even though the initial condition is very similar to one of the stored patterns, we may fail to converge to this pattern and ends up in a state that is neither stored nor opposite of one that is stored in our network (Figure 15 - C, the initial condition is close to the fifth pattern we aim to store).

If we start from a random initial state, the network may converge to a stored pattern (Figure 15 - D), but as described above, it can also evolve towards an arbitrary spurious state that cannot be directly predicted from the stored patterns (Figure 15 - E, F). Moreover, the convergence might take a longer time and sometimes is not as direct as before (Figure 15 - D, the network finally converges to the second stored pattern though by showing only 50 simulation steps it may not yet be totally clear).

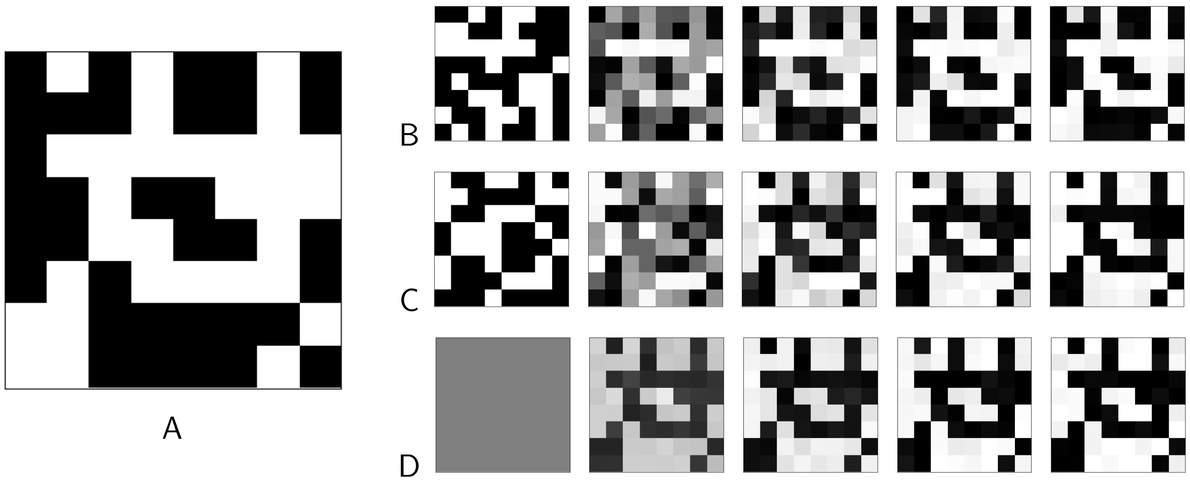

FIGURE 13: Store one pattern in a Hopfield network. A. The stored pattern. B.C.D. Simulations with different initial conditions (between two successive picutres we carry out 10 steps of simulation).

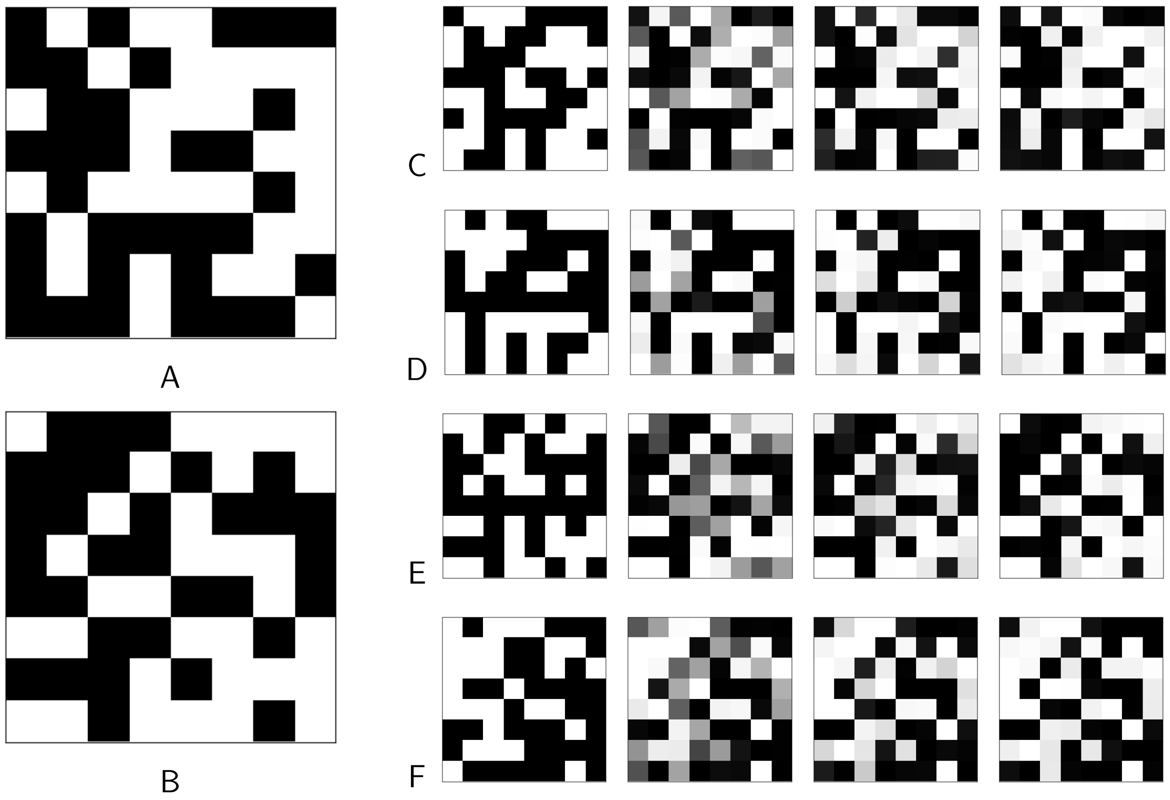

FIGURE 14: Store two patterns in a Hopfield network. A.B. Stored patterns. C.D.E.F. Simulations with different initial conditions that are close to stored patterns or their opposites.

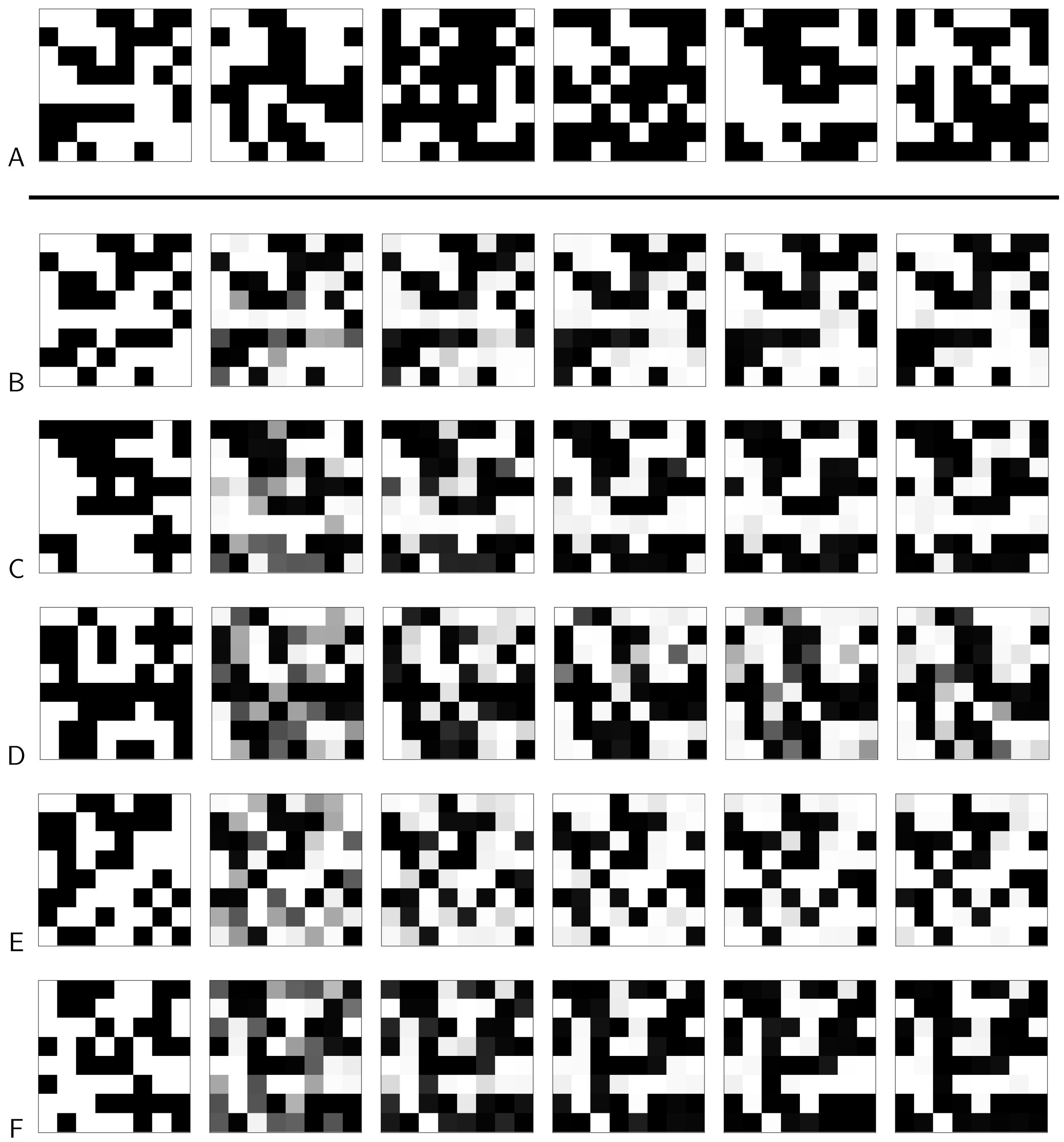

Figure 15: Try to store six patterns in a Hopfield network. A. Stored patterns. B.C.D.E.F. Simulations with different initial conditions.

Now how about replacing sgn by tanh for the activation function? To simplify the task, we'll first remove noise from the network and we attempt to store only one pattern in the network. As shown in Figure 16, the values of final states are proportional to those of **p**, but neuron activities are in general of much smaller magnitudes. The convergence is also much slower.

It's easy to imagine that in this case, the presence of noise can degrade severely the performance of the network. Indeed when we put again $\sigma = 0.1$, the result is shown in Figure 17. Neuron activities seem to be pure noise and we can hardly retrieve any information from it (notice that until now for the same simulation in one figure two successive images are separated by 10 simulation steps while from now on they'll be separated by 50 simulation steps).

The problem becomes even more complex if we store several patterns in the network. Again we'll remove noise from the model and this time we want to store two patterns in the network. Several scenarios can occur: the network may just work as before except a final state with much lower activity level (Figure 18); however, chances are the two stored patterns are no longer fixed points of the dynamics and even starting from the stored patterns we fall into another state that doesnt't appear to be reltatated to what we want to store (Figure 19). We notice also that the convergence of the system takes much longer time.

2.3 Conclusion

With the differential equation that is given at the beginning of the section and the general rule used to generate the weight matrix, we have simulated the dynamics of the network under various situations. We saw that when $f = \text{sgn}$ and if we stored just a small number of patterns the Hopfield network could act correctly as an associative memory system despite the presence of noise. However the network capacity is without doubt limited and many spurious states could appear if we tried to store too many patterns in a network.

When $f = \tanh$, the dynamics of the system seems to be more complicated. The values of the final states are much closer to 0, noise can be fatal to the system, the convergence is much slower and unexpected attractors may appear when we want to store more than one patterns.

We didn't do any mathematical proofs in this section, but to study more in details the Hopfield network, new notions must be introduced (e.g. energy of the network) and this will be a topic much more complex and beyond the scope of this report.

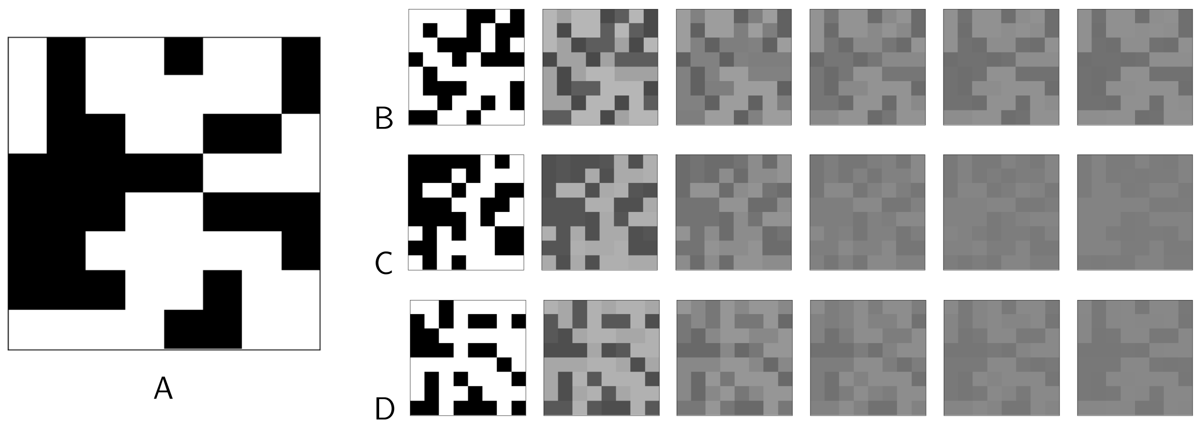

FIGURE 16: Store one pattern in a Hopfield network with $f = \tanh$ without noise. A. The stored pattern. B.C.D. Simulations with random initial conditions.

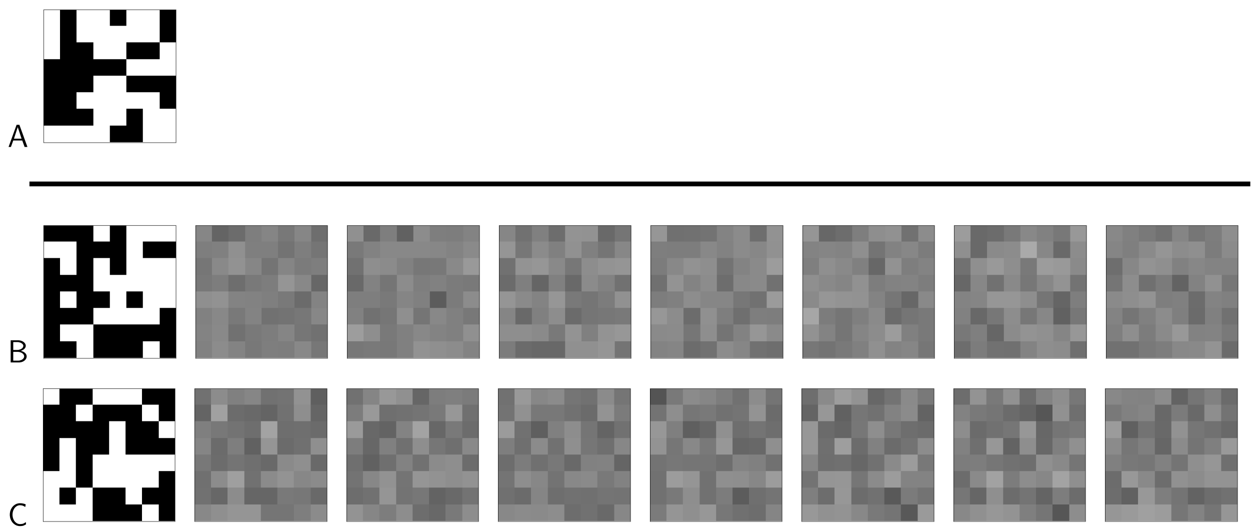

Figure 17: Store one pattern in a Hopfield network with $f = \tanh$ with noise. A. The stored pattern. B.C. Simulations with random initial conditions (between two successive picutres we carry out 50 steps of simulation).

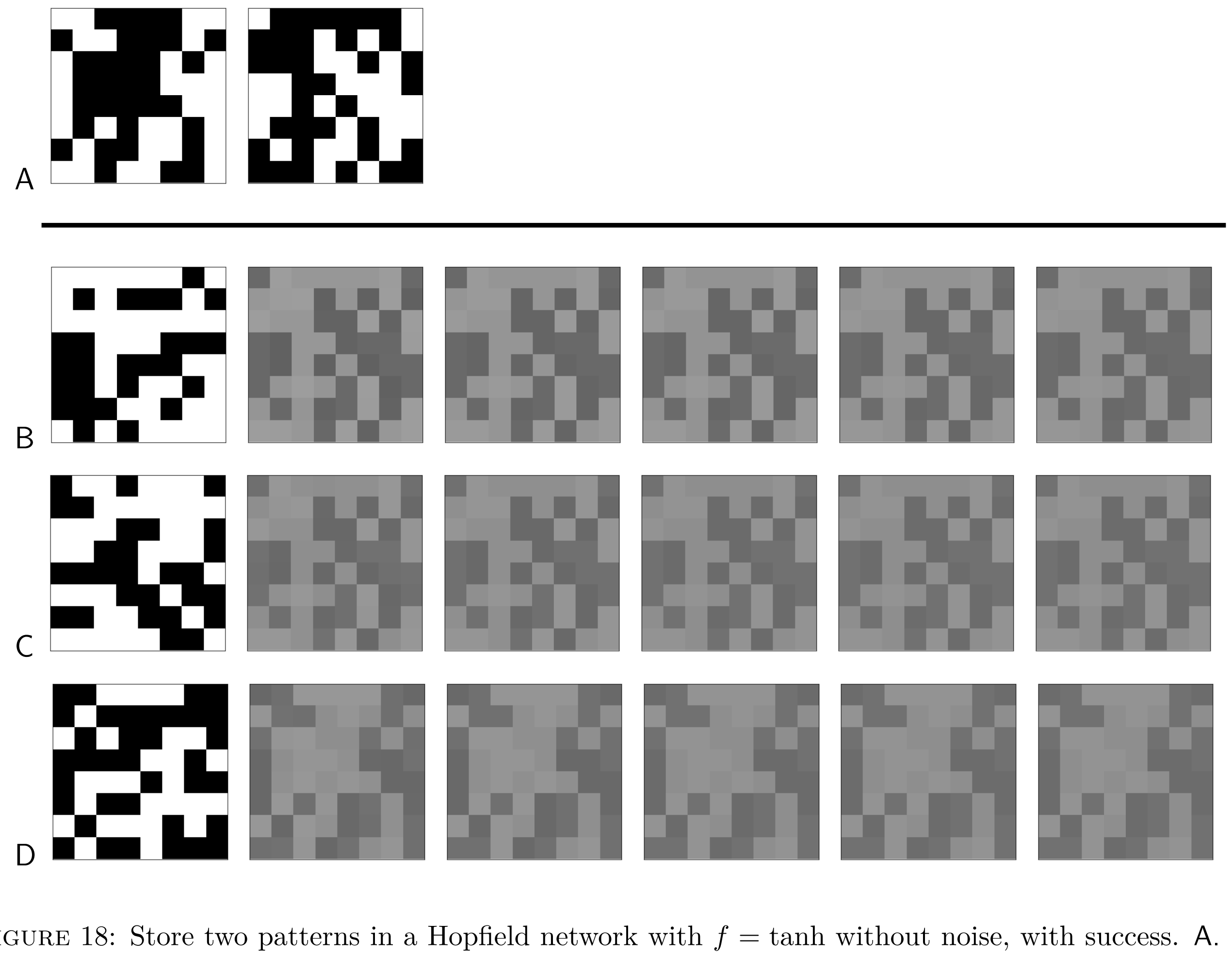

Figure 18: Store two patterns in a Hopfield network with $f = \tanh$ without noise, with success. A. Stored patterns. B.C.D. Simulations with random initial conditions.

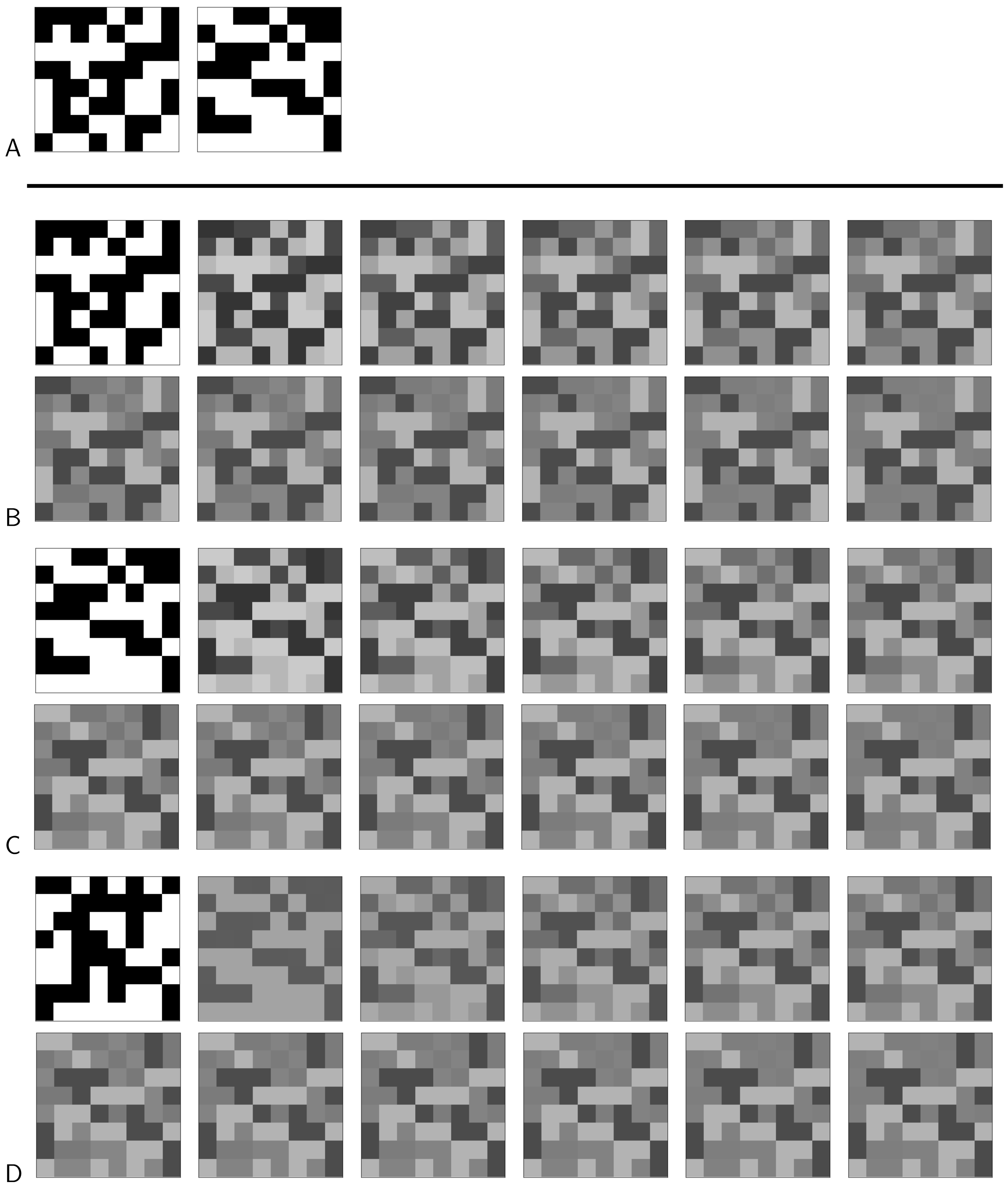

FIGURE 19: Store two patterns in a Hopfield network with $f = \tanh$ without noise, new fixed points appear instead of the stored ones. A. Stored patterns. B.C. Simulations starting from stored patterns. D. Simulations with random initial condition.