# Introduction

This report concerns the application of fundamental principles in the study of dynamical systems to computational neuroscience. More precisely, it demonstrates the implementation of the following concepts: state spaces (also known as phase spaces), nullclines, fixed points, divergence, convergence, stability, basic manipulations of ordinary differential equations and a method for solving them numerically (Euler's method). All of this in the context of simplified neural networks. We begin with the simplest models (autapse and mutual excitation), before moving to more complex models (Hopfield networks and ring attractors). Along the way, we also employ a series of visualization tools and techniques (bifurcation diagrams, PCA) to gain a better intuition for the trajectories of the dynamics for the systems in question.

# Part I

The first part of the report concerns the simplest possible network, an autapse, a system of a single node which is connected to itself by a single synapse.

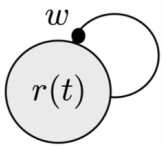



Figure 1: An autapse, first observed in 1972 in rabbit occipital cortex

## 1.1

We define the following activation function, $f(s) = 60 \cdot (tanh(s) + 1)$, and graph it for $s = [-5, 5]$:

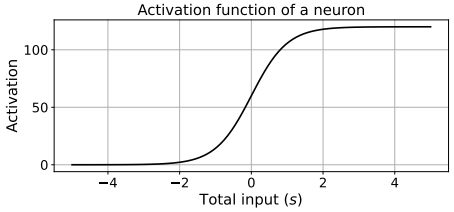



Figure 2: $f(s)$ is constrained to $(0, 120)$

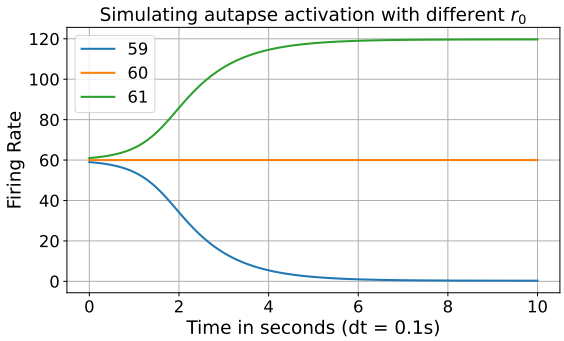
Here, $s$ represents the total input to the neuron. In a moment we will see how $f(s)$ interacts with the firing rate of the neuron $r(t)$.

## 1.2

Next, we simulate the system using Euler's method and the following diffential equation:

$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I) \tag{1}$$

Where the strength of the synapse is $w = 0.05$ and a constant external input to the neuron is described by $I = -3$. We plot the development of the system with $dt = 0.1$:




We observe that the system can converge to three stable points: $r(t) = 0$ when $r(0) < 60$, $r(t) = 120$ when $r(0) > 60$, and $r(t) = 60$ when $r(0) = 60$. To understand this behavior we can first solve for $dr(t)/dt$ when $r(t) = 60$:

$$\frac{dr(t)}{dt} = -60 + f(0.05 \cdot 60 - 3) = -60 + f(3 - 3) \tag{2}$$

$$= -60 + f(0) = -60 + 60 = 0 \tag{3}$$

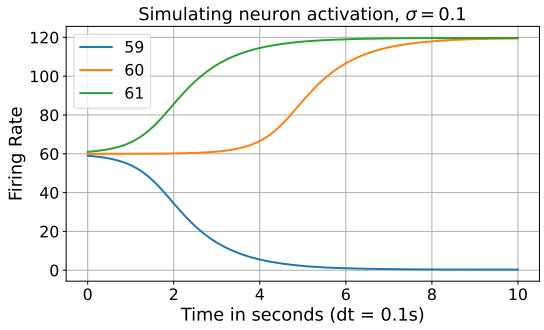
Where $\frac{dr(t)}{dt} = 0$, the system exhibits no change, hence why $r_0 = 60$ results in a straight line in our graph. Furthermore, because we know that $f(s)$ increases monotonically with $s$, any value under or over 60 will result in the system veering off to one of its other extremes, 0 or 120, given these initial conditions.

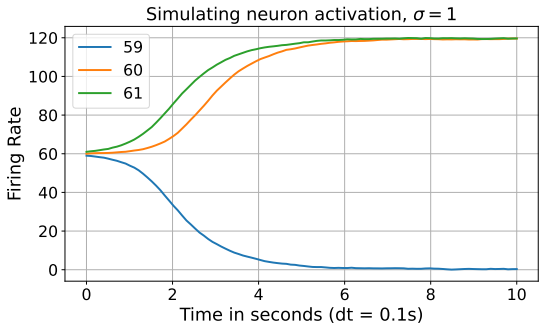
## 1.3

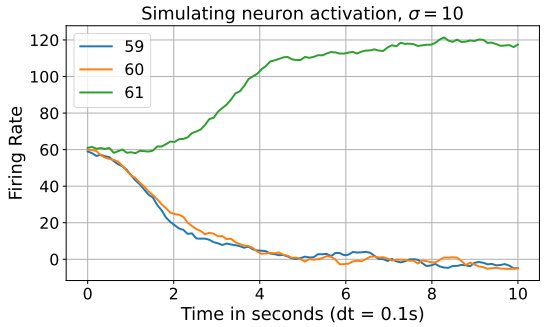
We modify our equation by adding a noise term $\sigma\eta(t)$ where $\sigma$ is a scalar and $\eta(t)$ samples from a gaussian distribution with a mean of 0 and a standard deviation of 1. This allows us to better approximate the messy biophysical reality of neuronal activation:

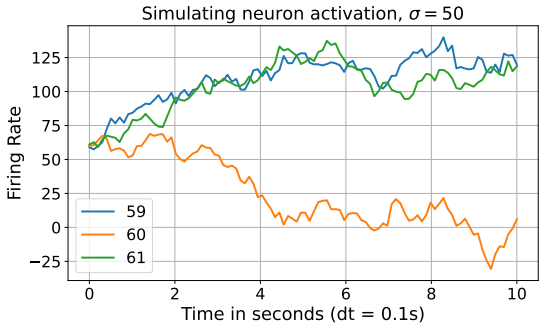
$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I) + \sigma\eta(t) \tag{4}$$

We can make a number of inferences as a result of these graphs. First and foremost, we see that simulations with the initial value $r_0 = 60$ no longer stay fixed at $r(t) = 60$, because slight perturbations in the firing rate cause the line to diverge from this fixed point. Furthermore, we see that every line that diverges from $r(t) = 60$ eventually converges on or around $r(t) = 0$ or $r(t) = 120$, even when there is significant noise added.



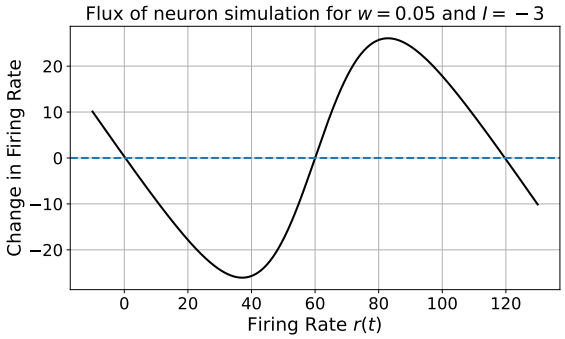



Thus, we have the intuition that $r(t) = 60$ is an unstable fixed point, while $r(t) = 0$ and $r(t) = 120$ are stable. At very high values of $\sigma$, the difference in initial condition is no longer enough to determine where the system will go, as shown in the fourth figure.

## 1.4

To gain a deeper understanding of the system dynamics, we can graph the "flux" of the system, that is, the values of $dr(t)/dt$ for relevant values of $r(t)$. The next figure is the flux of the previous simulation for $r(t) = [-10, 130]$, with the same initial conditions and no added noise:




Of particular interest to us are the points where the curve crosses the x-axis. These are values of $r(t)$ for which $dr(t)/dt = 0$, i.e. the fixed points of the system. If the system were to be initialized with any of these values, it's firing rate would not evolve over time.

As we alluded to before, fixed points can be either stable or unstable. This can be readily seen by observing the slope of the flux around the fixed points. For values close to $r(t) = 0$, $r(t) < 0$ gives us $dr(t)/dt > 0$, and $r(t) < 0$ gives up $dr(t)/dt > 0$. These values for $dr(t)/dt > 0$ push the system towards $r(t) < 0$, which is why we would call it a stable fixed point. Whereas, or values close to $r(t) = 60$, the opposite is true, making it an unstable fixed point.

## 1.5

Finally, we plot a bifurcation diagram to visualize how the number of solutions (zero-crossings) changes with different values of $w$ (synaptic strength) and $I$ (external input):

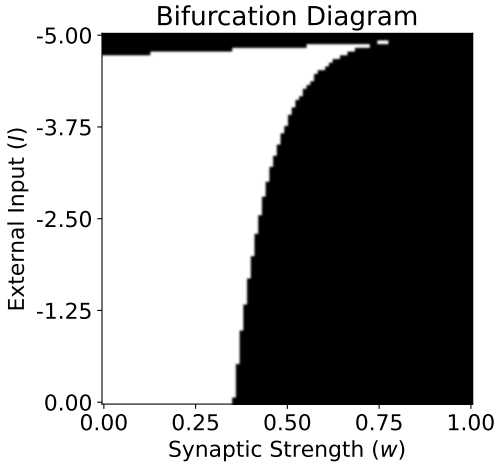



Figure 3: Black = 1 solution, White = 3 solutions

Some observations: beyond approximately $w = .75$, the system will always have only 1 solution, potentially because such a high synaptic weight always drives the neuron to its maximum firing rate eventually. Similarly, it sees that for $I$ close to $-5.00$, the system will always have only 1 solution, driven to a firing rate of 0 because of the high negative external input. In all other cases, the system has both the two extremes, and a single unstable solution somewhere in between them.

# Part II

The second part of the report deals with a neural network with two nodes which are connected to one another by two synapses.

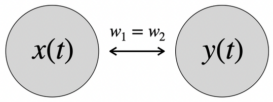



Figure 4: To simplify, we give the synapses equal weights

## 2.1

To begin, we plot the evolution of the firing rates of both neurons using the following equations, where $w_1 = w_2 = 0.4$, and $I = -10$:

$$\frac{dx(t)}{dt} = -x(t) + f(w_2 y(t) + I) \tag{5}$$

$$\frac{dy(t)}{dt} = -y(t) + f(w_1 x(t) + I) \tag{6}$$

$$f(s) = 50\frac{1}{1 + exp(-s)} \tag{7}$$

In the following figure, blue lines are $x(t)$ values and red lines are $y(t)$:

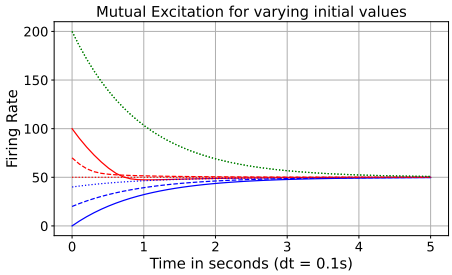
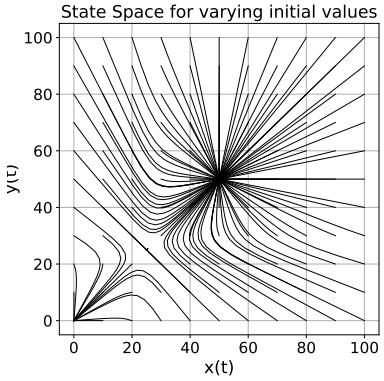



Figure 5: Neurons that were simulated together have the same linestyle

## 2.2

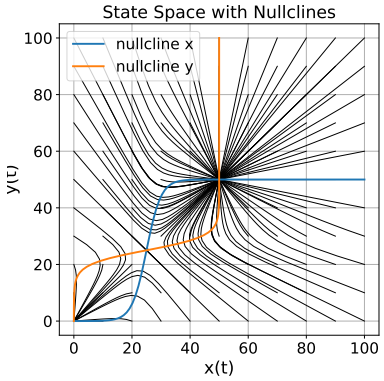
We can also plot the state space of the system for different initial values $x_0$ and $y_0$:




There are a number of things for us to note regarding this figure. First, notice that the equations $dx(t)/dt$ and $dy(t)/dt$ are symmetrical, which is evidenced by the symmetry of our state space. Second, we can identify three fixed points at $x(t) = y(t) = 0, 25, 50$.

## 2.3

Subsequently, we can plot the nullclines of the system, (i.e. the values of $x(t)$ for which $dx(t) = 0$, and likewise for $y(t)$) on top of our state space:




We can confirm the existence of the three fixed points at $x(t) = y(t) = 0, 25, 50$ because those are the same three points where the nullclines intersect, meaning they are the points where $dx(t) = dy(t) = 0$.

## 2.4

The crossings points of the nullclines indicate the fixed points of the system, where the rate of change for the firing rates of both neurons is 0. We can find these fixed points by solving the system of equations presented in section 2.1 numerically.

Specifically, we picked [0,0], [25,25], and [50,50] as our initial guesses and used the `fsolve` function from the `scipy.optimize` library to find the fixed points. As a result, we get the follwing fixed points: [0.00227196, 0.00227196], [25,25], and [49.99772804, 49.99772804]. For [25, 25], we can actually solve the equations analytically to find that it is a fixed point, as shown below:

$$\frac{dx(t)}{dt} = -25 + f(0.4 \cdot 25 - 10) = -25 + f(0) = -25 + \frac{50}{2} = 0 \tag{8}$$

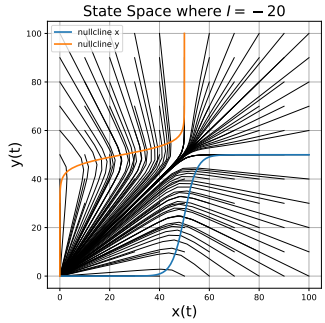
$$\frac{dy(t)}{dt} = -25 + f(0.4 \cdot 25 - 10) = -25 + f(0) = -25 + \frac{50}{2} = 0 \tag{9}$$

The graph in section 2.3, as well as the fixed points we found numerically, suggest that only one of our fixed points, [25, 25], is stable, while the other two are unstable. From the graph we can see that systems that fell along the diagonal line $y(t) = x(t) - 25$ always converged to [25, 25], while systems that were initialized closer to the other two fixed points diverged from them. It is no surprise then, that the corresponding solutions for the points we espect to be unstable are not whole numbers, but rather 0.00227196 and 49.99772804.

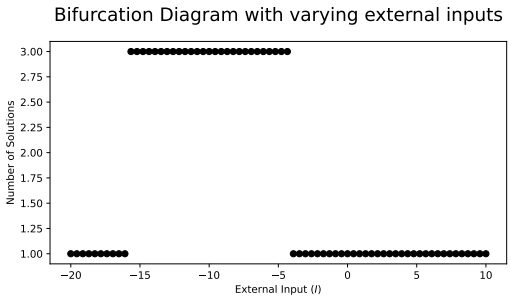
## 2.5

## 2.6

To conclude this section on mutual excitation, we will investigate how different values of external inputs $I$ affect the number of solutions (fixed points) of the system. First, we plot the nullclines and the state space for $I = -20$:



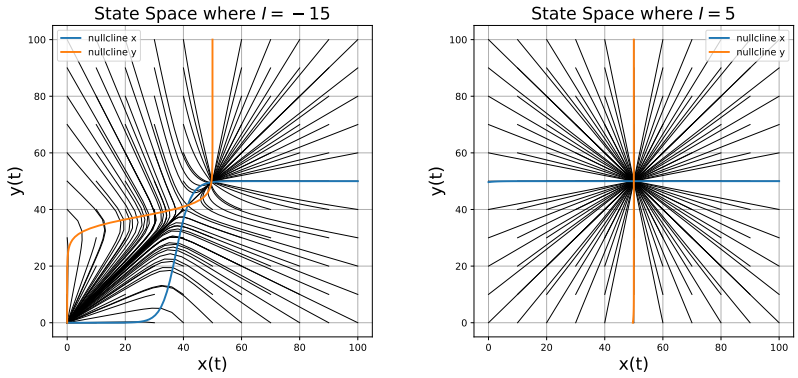

Then we plot a simple scatter plot bifurcation diagram for the system, varying $I$ from $-20$ to 10:



Our method for calculating the number of fixed points is perhaps a bit original, but it is effective. We simply count the number of times that the nullclines intersect in the state space. We could then go on to verify these results by solving the system of equations numerically as we did in the previous section, using the `fsolve` function.

Our bifurcation diagram tells us that two regimes exist, one with one solution, and another with three solutions. This suggests that our system experiences pitchfork bifurcation twice, once at around $I = -16$ and again at around $I = -4$.

To validate our results, we plot the nullclines and the state space for values of $I$ around these two points:




# Part III

In this next part of the report, we will learn about Hopfield networks, fully-connected recurrent neural networks which operate using a generalization of the Hebbian learning rule ("neurons that fire together, wire together"). They are typically used to model associative memory, wherein patterns that the network stores in the weights of its synapses can be recovered in the evolution of the system from random initial conditions.

## 3.1

In particular, we will study a Hopfield network with 64 neurons. To begin, we define a pattern $\boldsymbol{p}$ in this network as a vector of 64 elements, each of which is either -1 or 1. After generating such a pattern using a random process, we plot it as a $8x8$ image:

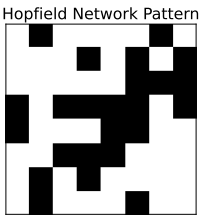



Figure 6: Randomly Generated Pattern

## 3.2

We use the following formula to calculate the weight matrix $W$, where $W_{ij}$ is the strength of the synapse from neuron $i$ to neuron $j$:

$$W = \frac{1}{N}\boldsymbol{p}\boldsymbol{p}^T \tag{10}$$

Then we simulate the evolution of the network over time using the following equation, where $dt = 0.1$, $\sigma\eta(t)$ is the same noise term as before, and $f(s)$ is the sign function:

$$\frac{dx(t)}{dt} = -x(t) + f(Wx(t)) + \sigma\eta(t) \tag{11}$$

Below, we plot plot 16 equidistant states in the evolution of the system which has a random pattern as its initial state:

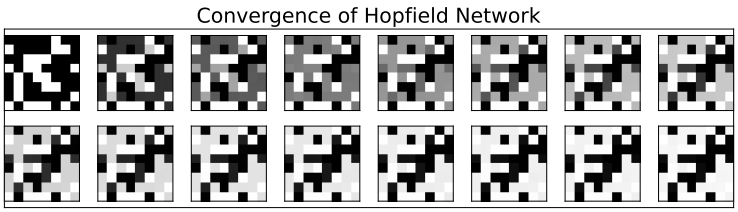
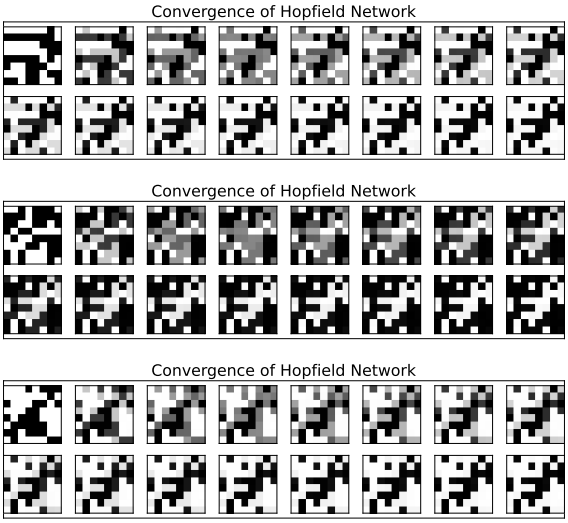



Figure 7: Note that the final state matches the pattern saved in the weights matrix

## 3.3

We repeat this simulation three more times, each time using a different starting point:




Interestingly, we find that we return to our original saved pattern $\boldsymbol{p}$ in only our second simulation. In the other two, we actually arrive at the "inverse" of this pattern, suggesting that the original pattern and its inverse are both fixed points of the system.

## 3.4

We now generate another random pattern $\boldsymbol{q}$ and recalculate the weight matrix $W = \frac{1}{N}(\boldsymbol{p}\boldsymbol{p}^T + \boldsymbol{q}\boldsymbol{q}^T)$:

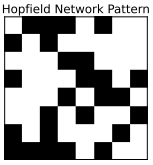
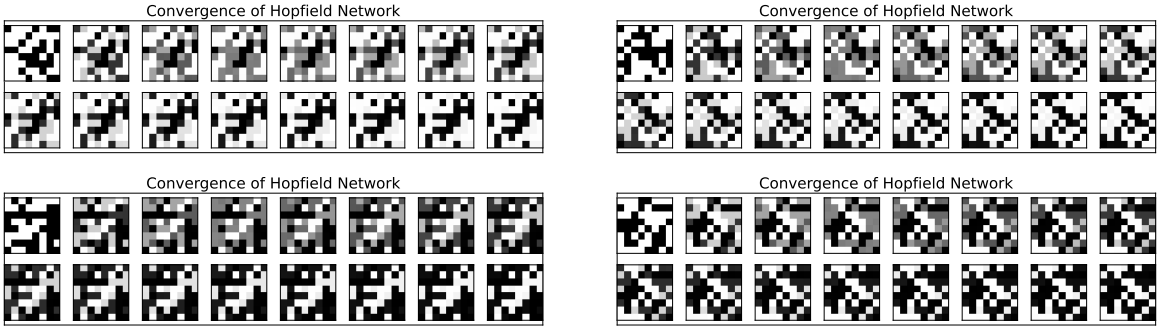


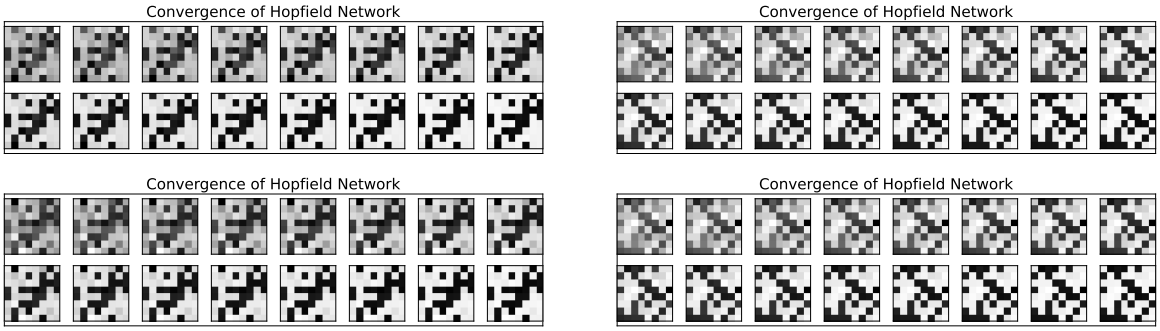

Figure 8: We denote this pattern as $\boldsymbol{q}$

Now we repeat the previous simulations four more times, using random initial states, to observe what fixed points now exist in the system. We find that our system can converge on both of our saved patterns $\boldsymbol{p}$ and $\boldsymbol{q}$ and their respective inverses:




## 3.5

Now we will use the same gaussian noise term as in the previous parts of this report to corrupt our patterns $\boldsymbol{p}$ and $\boldsymbol{q}$ and use them as initial states to see if we can recover them:




## 3.6

As a last exercise regarding Hopfield networks, will successively increase the number of saved patterns, using the the following formula $W = \frac{1}{N}\sum_i^M \boldsymbol{p}_i \boldsymbol{p}_i^T$, and then use a corrupted version of one of the patterns as our initial state to see if it can still be recovered:

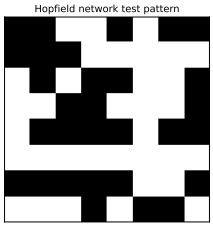
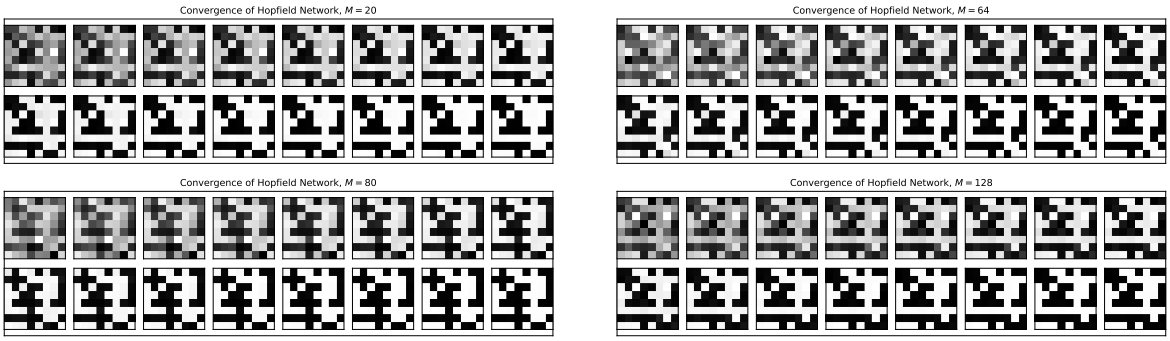



Figure 9: This is the pattern we will be attempting to recover




At lower values of $M$, we see that the system can still recover the pattern, but as we increase $M$, the system becomes less reliable at recovering the pattern. This is because the more patterns we save, the more likely it is that the system will converge on a fixed point that is not the one we are looking for. That being said, the system does seem to converge on patterns that are visibly similar to the one we are looking for, which is a testament to the robustness of the Hopfield network as a model for associative memory.

# Part IV

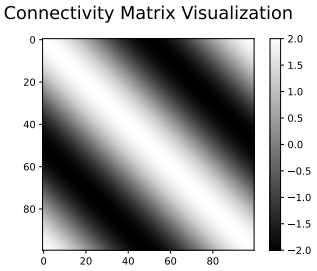
In this final part of the report, we will study a ring attractor, a network of neurons that are connected in a ring topology, meaning that each neuron is most strongly connected to its two "nearest" neighbors, and synaptic strength wanes as the "distance" increases. This network is used to model the representation of continuous variables in the brain, like the orientation of an object in space.

## 4.1

We begin by defining the connectivity matrix $J$, where $J_{ij}$ is the strength of the synapse from neuron $i$ to neuron $j$. We will use the following formula to calculate $J$:

$$J_{ij} = 2cos(\theta_i - \theta_j) \tag{12}$$

To calculate a particular instance of $J$, we set $N$ as the number of neurons in our network, and $2\pi/N$ as the angle "between" each neuron. Thus, $\theta_i = \pi i/50$ for $i = 0, 1, ..., N-1$. We will then plot the connectivity matrix $J$ as a $100x100$ image:



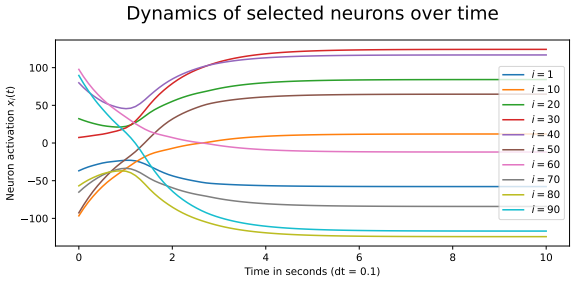

We observe that the matrix is symmetric, as expected, and that the diagonal is the strongest, which is also expected because each neuron is most strongly connected to itself. Furthermore, we see that the matrix is periodic, which is a result of the ring topology of the network.

## 4.2

Now we simulate the dynamics of the system, applying Euler's method to the following differential equation, where the activation function is $\phi(s) = tanh(s)$:

$$\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + J\phi(\boldsymbol{x}(t)) \tag{13}$$

We plot the evolution of the system over time, using a random initial state:




We see that each neuron in the network converges to a particular value, which is a result of the network's ring topology. This is because the network is designed to represent a continuous variable, and the neurons are arranged in such a way that they can represent this variable in a continuous manner.

## 4.3

Another way to visualize the relationships between the neurons is to use Principal Component Analysis (PCA) to reduce the dimensionality of the system to two dimensions. First, we have to compute the covariance matrix of the system by taking the output of our simulation $X$, a matrix with shape 100×1000 (100 neurons, 1000 time steps), and applying this formula:

$$C = \frac{1}{1000} XX^T \tag{14}$$

Matrix $C$ is an 100x100 matrix which denotes how much each neuron is correlated with every other neuron. We plot it below. We then calculate the eigenvalues and eigenvectors of $C$ using `NumPy`, find the eigenvector corresponding to the largest eigenvalue, and plot it to get the subsequent figure:

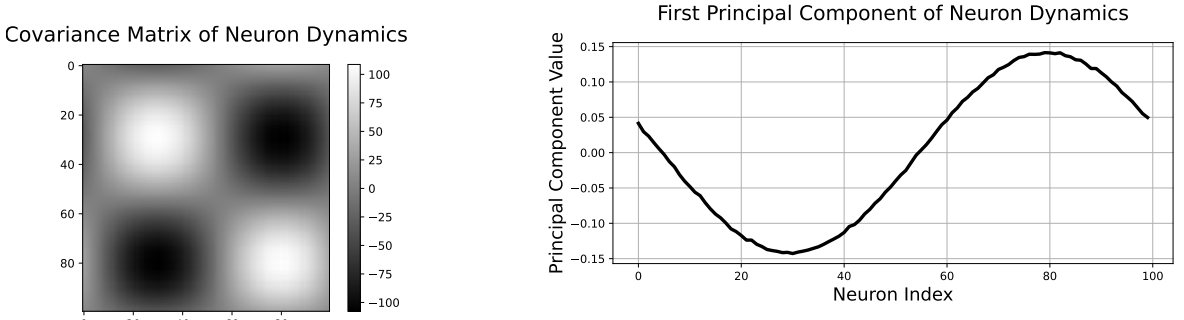



Figure 10: This plot has a cosine shape because of its activation function

Dividing the largest eigenvalue by the sum of all eigenvalues gives us the explained variance for the first principle component. Here, the first principle component captures over 99.9% of the variance of the system.

## 4.4

Every time that we simulate a new ring attractor with random initial values, every neuron in the attractor converges to a different value. Thus, if we repeatedly simulate the system dynamics (here, 500 times) and collect the final state of the 500 attractors into a 100x500 matrix $Q$, we can perform PCA on this matrix to visualize this "attractor" space:

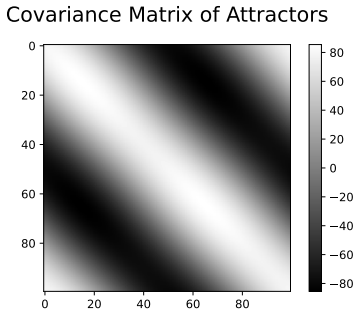



Figure 11: Note the similarity of this figure to our previous visualization of $C$ in 4.1

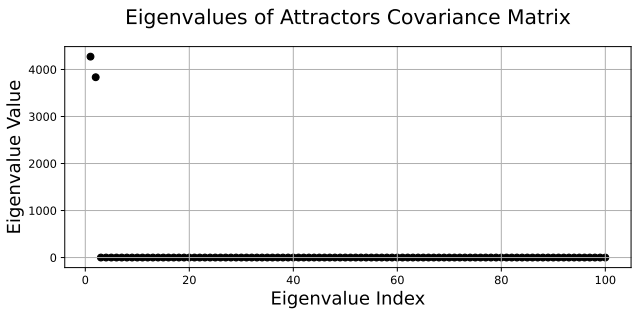
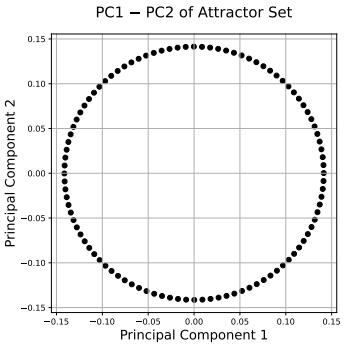



Figure 12: The first two principal components account for the vast majority of the variance

Seeing as the first two principal components account for over 99.9% of the variance (PC1 = 52.7%, PC2 = 47.3%), we can plot the attractor space in the plane defined by these two components:




The plot is a perfect circle, which is precisely what we would expect from a ring attractor, because it indicates that the main source of variability among the attractor states lies in their orientation along the ring.

## 4.5

Finally, we use the python library `scikit-learn` to perform PCA on the matrix $Q$ and plot the first two principal components of the attractor space as a sanity-check for our previous results:

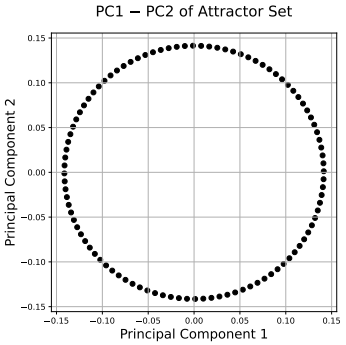



Figure 13: Explained variance with `scikit-learn`: PC1 = 52.7%, PC2 = 47.3%

## Conclusion

A great number of phenomena in nature can be described by dynamical systems. Among them, crucial for our purposes, are the dynamics of neural networks. In this report, we considered four fundamental systems, the autapse, mutual excitation in two-neuron systems, Hopfield networks, and ring attractors. Both Hopfield networks and ring attractors are instances of Recurrent Neural Networks (RNNs), and they are significant additions to our tool kit for understanding neural dynamics because they allow us to model how memory and representation may be implemented in the brain.

Dynamical systems are characterized by their differential equations and their initial conditions. With these starting points in hand, we can simulate the evolution of the system over time with a numerical process (like Euler's method), and sometimes we can even derive certain characteristics of the system analytically, like when we calculated the fixed points and the nullclines of our systems from the equations alone.

Perhaps the most essential insight offered by this report is the nature of the fixed points of our systems, which can be either stable or unstable. These points tells us where the states of our systems converge to or diverge from, and where they may reach an equilibrium that persists over time. Taking the work we did with Hopfield networks, for example, we can conceptualize how a certain pattern that the network has learned can be reached from various starting points, essential for the description of a cognitive function like memory which exhibits this kind of robustness.