Methods in Computational Neuroscience

Lab Report 3: Firing Rate Models and Dynamical Systems

Alexander (Sasha) Lanine

2024-03-19

Contents

0 Introduction

A typical cortical neuron receives synaptic input from many thousands of neurons. Firing-rate based network models are often used to model the dynamics of these large neural circuits [1]. In this report, we will we use computational experiments to investigate these models through the lens of dynamical systems studies. In the first section, we will consider the simplest possible firing-rate based model: a network of a single neuron with a synaptic connection to itself. In the second section, we will then investigate the intertwining dynamics of a network of two neurons. As we will see, even these these one and two dimensional models are sufficiently complex to manifest fascinating dynamical properties.

In the third and forth sections, we will then investigate two popular high-dimensional firing rate models. The first will be the Hopfield network, which has been applied as a model of episodic memory. The second will be a *ring attractor network*: a high dimensional model whose dynamics can be captured on a low dimensional subspace. As such, we will use the ring attractor network to illustrate how to use a dimensionality reduction method called principal component analysis to analyze high dimensional network models.

1 Autaptic Neurons

In this first section, we will begin by investigating a single neuron with an *autapse*, which is a synapse from the neuron to itself. In our model, its dynamics are governed by the following equation

$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I), \tag{1.1}$$

where

$$f(s) = 60(1 + \tanh(s)), \tag{1.2}$$

which is a sigmoidal function. In the above, $r(t)$ represents the firing rate of the neuron at time t in Hz, I represents a background input current, and w represents the strength of the connection from the neuron to itself.

1.1 Dynamics

We begin our investigation by taking a closer look at the sigmoid function $f(s)$. In our model, it takes the total input from the neuron and background current $wr(t) + I$ as its argument. To better understand how this impacts the dynamics, we plot the sigmoid function for a range of possible inputs in Figure 1.1.

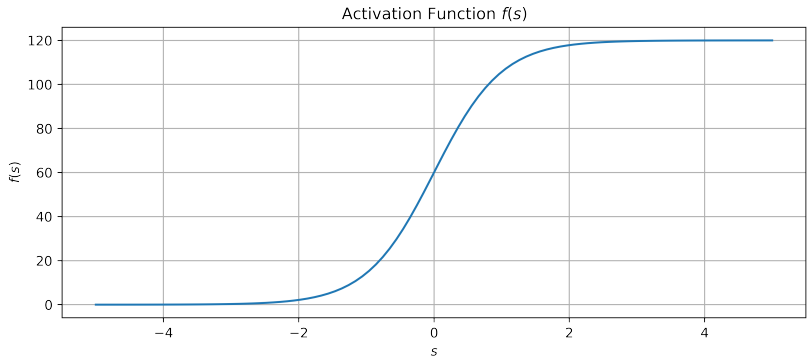


Figure 1.1: Activation function $f(s) = 60(1 + \tanh(s))$ plotted for different values of s.

What we see here that the output of the function scales roughly linearly in the domain where the input s is close to zero. As s gets large to the positive direction, the function converges to 120 Hz. Conversely, as s gets large in the negative direction, the function converges to 0 Hz.

This brings us to the start of our investigation of the dynamics of the autaptic neuron model. To begin, we will hold $I = -3$ and $w = 0.05$ constant, and numerically integrate Equation 1.1 for initial conditions $r(0) \in \{59, 60, 61\}$. The results are shown in Figure 1.2 below.

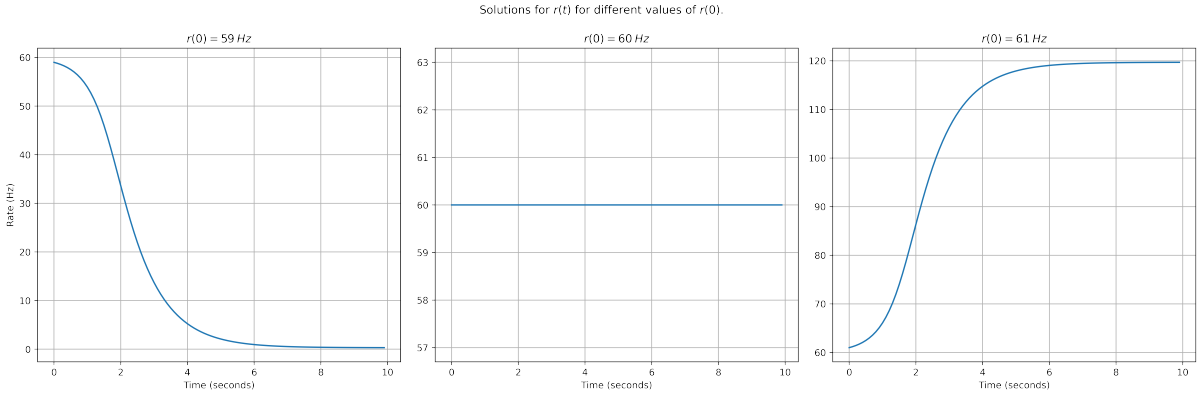


Figure 1.2: Time evolution of $r(t)$ for initial conditions $r(0) \in \{59, 60, 61\}$. Solution was numerically integrated using Euler's method with time steps $dt = 0.1$.

Note that we observe three different types of behavior for these three different initial conditions. First, when $r(0) = 59\,Hz$, the firing rate of the neuron decreases to zero; when $r(0) = 61\,Hz$, the firing rate increases to a rate of $120\,Hz$; and finally, when $r(0) = 60\,Hz$, the firing rate stays constant. We will demonstrate why this occurring in Section 1.3.

1.2 Adding Noise

In this section, we consider the effect of adding a noise term to our model:

$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I) + \sigma\eta(t), \tag{1.3}$$

where $\eta(t)$ is gaussian white noise and σ is a parameter that controls the noise magnitude.

Using Equation 1.3, we re-simulate the time evolution of the autaptic neuron for the initial conditions $r(0) \in \{59, 60, 61\}$ and values of σ ranging from 0.01 to 5. The results are shown in Figure 1.3 below.

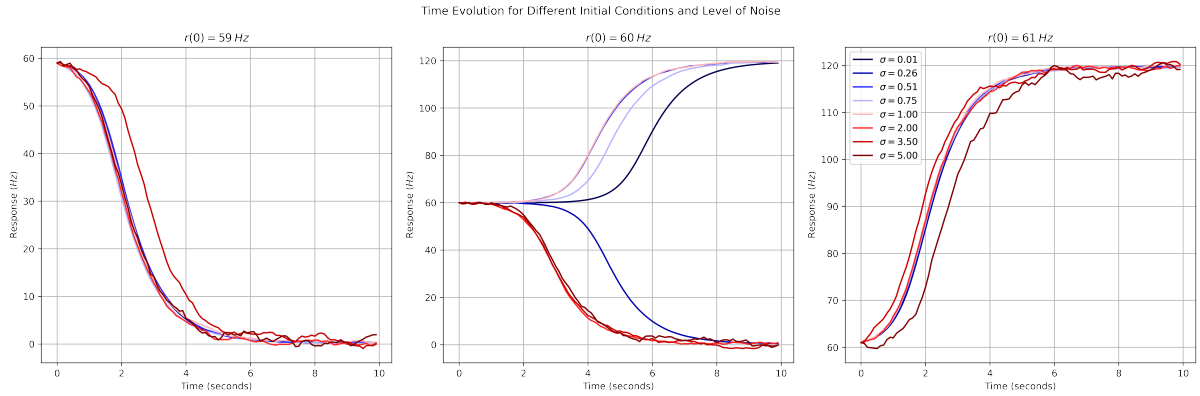


Figure 1.3: Time evolution of $r(t)$ for initial conditions $r(0) \in \{59, 60, 61\}$ and different values of σ. Solution were numerically integrated using Euler's method with time steps $dt = 0.1$.

We observe that both the $r(0) = 59\,Hz$ and $r(0) = 61\,Hz$ curves appear to be robust to small, noisy perturbations when compared to the noise-free simulations. By this we mean that, despite the introduction of noise, the neuron still evolves to roughly the same state. This suggests that $r = 0\,Hz$ and $r = 120\,Hz$ are stable fixed points of the system. The $r(0) = 60\,Hz$ curve, on the other hand, is not robust to even the smallest perturbations tested. Indeed, we see that even in the $\sigma = 0.01$, the neuron will eventually diverge from the $r(t) = 60\,Hz$ curve. This suggests that $r = 60\,Hz$ is an unstable fixed point of the system.

1.3 Flux and Bifurcations

The analysis in the previous two sections suggest that, in the absence of noise, the system will tend to stay at the states $r \in \{0, 60, 120\}$ (where these are only approximate values since we don't have analytic solutions for the fixed points). This raises the following question: Are these the only such points? To investigate this, we plot the flux dr/dt as a function of $r(0)$ in the case where $I = -3$ and $w = 0.05$ in Figure 1.4.

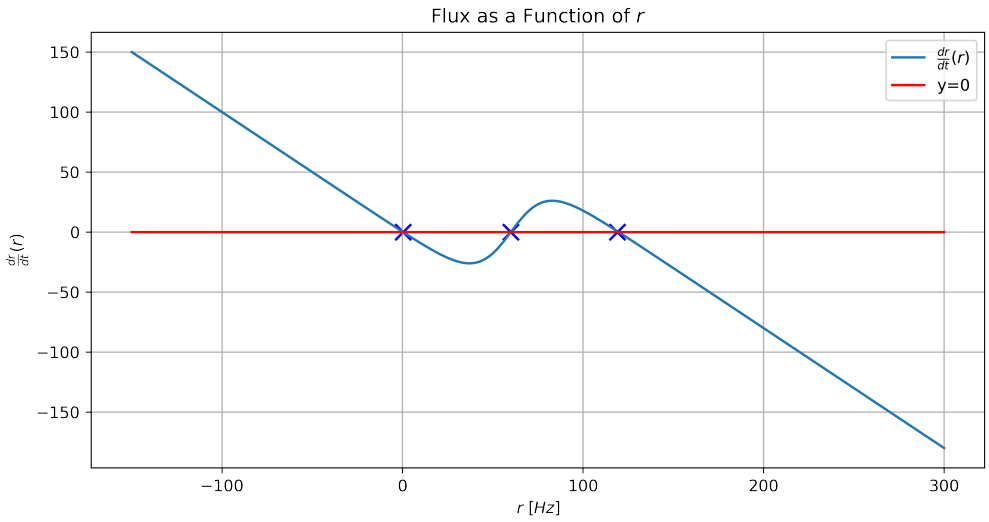


Figure 1.4: Flux $\frac{dr}{dt}$ as a function of the Neuron's firing rate $r(t)$ with $I = -3$ and $w = 0.05$.

We see that there are three points where $dr/dt = 0$, and that these are at roughly 0, 60, and 120, as we would expect from the previous analysis. These represent the points where the neuron's firing rate will stay constant in the absence of any perturbations. That is, they are the fixed points of the system. Moreover, observe that we can infer their stability from the graph. The fixed points $r = 0\,Hz$ and $r = 120\,Hz$ are stable because the slope of the flux is negative at those points, meaning that small perturbations to the left will return the flux to the right, and vice versa for perturbations to the right. Contrarily, the fixed point at $r = 60\,Hz$ is unstable since the slope of the flux is positive at that point. Small perturbations will be driven away to one of the other two fixed points.

A natural question to now ask is how the number of fixed points of the system depends on the parameters I and w. Figure 1.5 below is a *bifurcation diagram* that shows the number of zero crossings of the the flux as a function of w ranging from 0 to 1, and I ranging from -5 to 0.

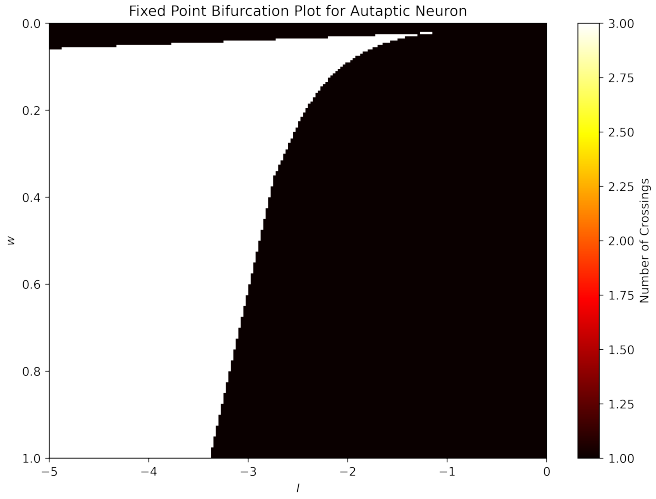


Figure 1.5: Bifurcation diagram of number of crossings as a function of w and I.

We see that there are two distinct regimes: one where the number of crossings is one (reflecting the existence of a unique fixed point), and one where the number of crossings is three.

2 Two Neuron Networks

In this section, we consider a model of a circuit of two neurons that excite each other. The dynamics of the system are governed by the equations below

$$\frac{dx(t)}{dt} = -x(t) + f(w_2 y(t) + I) \tag{2.1}$$

$$\frac{dy(t)}{dt} = -y(t) + f(w_1 x(t) + I), \tag{2.2}$$

where

$$f(s) = 50\sigma(s) = 50\left(\frac{1}{1+e^{-s}}\right). \tag{2.3}$$

Unless otherwise stated, all simulations in the following section will be run with parameters $w_1 = w_2 = 0.4$ and $I = -10$.

2.1 Dynamics

Let us start by numerically investigating the time evolution of the curves $x(t)$ and $y(t)$. We will begin with the simple case where $(x(0), y(0)) = (10, 20)$. The results of the simulation are shown in Figure 2.1 below.

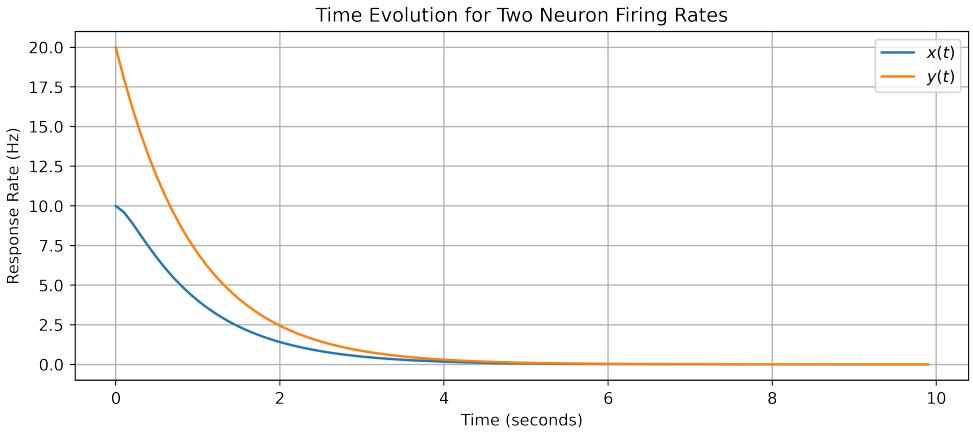


Figure 2.1: Time evolution of two neuron network with initial conditions $(x(0), y(0)) = (10, 20)$. The curves were integrated using Euler's method with time step size $dt = 0.1$.

We repeat this now for a number of simulations with $0\ Hz \leq x(0), y(0) \leq 60\ Hz$. The results are shown in Figure 2.2 below. Note that we only plot the trajectories of the x neuron for each trial. This is because the y curves would be redundant by the symmetry of Equations 2.1 and 2.2. For example, to see the evolution of y for initial condition $(10, 50)$, we can simply look at the evolution of x for initial condition $(50, 10)$.

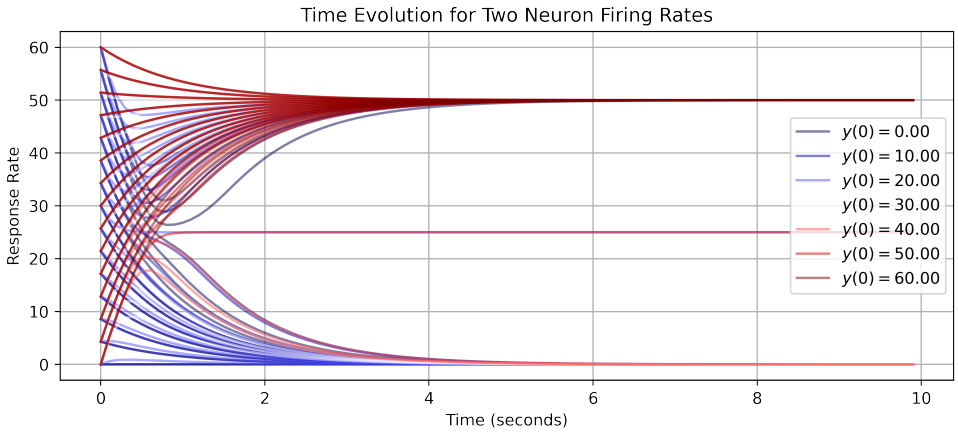


Figure 2.2: Time evolution of neuron x for a number of initial conditions in the range of $0\,Hz \leq x(0), y(0) \leq 60\,Hz$. The color indicates $y(0)$, the position on the vertical axis at $t = 0$ indicates $x(0)$.

We see that across all simulations, $x(t)$ converges to a firing rate of either $0\,Hz$, $25\,Hz$,or $50\,Hz$ (roughly) in time. Note that in all simulations $y(t)$ will converge to the same value as $x(t)$. This is because all fixed points of the system are symmetric (see Appendix A for a proof). At this point, we can judge qualitatively that $(x^*, y^*) = (0, 0)$ and $(x^*, y^*) = (50, 50)$ are likely stable fixed points, while $(x^*, y^*) = (25, 25)$ is likely an unstable fixed point or a saddle node. We will investigate this further in the following section.

We now plot a phase portrait for the system, wherein the trajectories are plotted in the x-y plane for many possible initial conditions (see Figure 2.3).

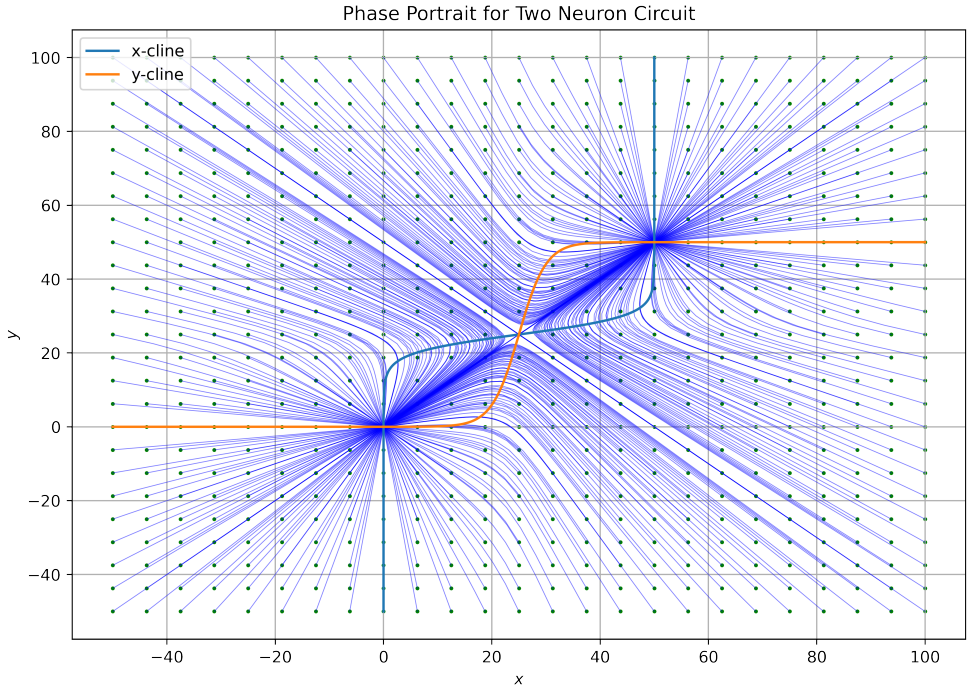


Figure 2.3: Phase diagram for two-neuron circuit with parameters $w_1 = w_2 = 0.4$ and $I = -10$. Green dots indicate initial states. The x- and y-nullclines are shown in colors blue and orange, respectively.

We again observe that the fixed points appear to be given by $(x^*, y^*) \in \{(0, 0), (25, 25), (50, 50)\}$. Moreover, this phase portrait is consistent with the conjecture that $(x^*, y^*) = (0, 0)$ and $(x^*, y^*) = (50, 50)$ are stable fixed points, and

$(x^*, y^*) = (25, 25)$ is a saddle note. In the next section, we will consider the nullclines and Jacobian of the system to better understand the location and stability of the fixed points.

2.2 Nullclines and Stability Analysis

The nullclines of a two-dimensional dynamical system $(F_1(x, y), F_2(x, y))$ are the curves/surfaces where $F_1 = 0$ or $F_2 = 0$. These correspond to to the x- and y-cline, respectively.

In the case of the system described by Equations 2.1 and 2.2, with parameters $w_1 = w_2 = 0.4$ and $I = -10$, the x-cline of the system is given by

$$0 = -x + f(0.4y - 10) \tag{2.4}$$
$$\Rightarrow x = f(0.4y - 10). \tag{2.5}$$

Likewise, the y-cline is given by

$$y = f(0.4x - 10). \tag{2.6}$$

Both are shown on the phase diagram for the circuit in Figure 2.3. The fixed points are given by the crossings of the nullclines. This is because, by definition, (x^*, y^*) is a fixed point just in case

$$\dot{x}(x^*, y^*) = \dot{y}(x^*, y^*) = 0. \tag{2.7}$$

By numerically comparing the distance between the two curves in the euclidean norm with a tolerance of 10^{-4}, we do indeed find that the crossings occur at roughly $(0, 0)$, $(25, 25)$, and $(50, 50)$, confirming our earlier observations.

We can classify the fixed points of the system by analyzing the Jacobian of the system at the fixed points, which is in general given by

$$J(x, y) := \begin{pmatrix} \partial_x F_1 & \partial_y F_1 \\ \partial_x F_2 & \partial_y F_2 \end{pmatrix} \tag{2.8}$$
$$= \begin{pmatrix} -1 & \partial_y 50\sigma(0.4y - 10) \\ \partial_x 50\sigma(0.4y - 10) & -1 \end{pmatrix} \tag{2.9}$$
$$= \begin{pmatrix} -1 & 20\sigma(0.4y - 10)(1 - \sigma(0.4y - 10)) \\ 20\sigma(0.4x - 10)(1 - \sigma(0.4x - 10)) & -1 \end{pmatrix} \tag{2.10}$$

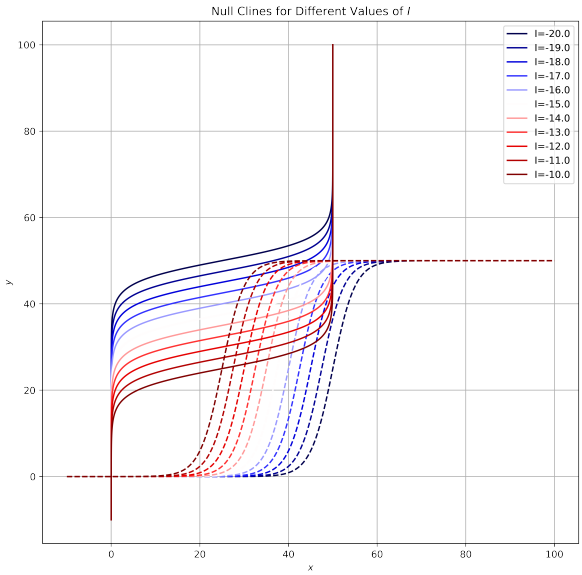
To analyze the stability of the fixed points, we follow the determinant- and trace-based classification procedure described in [2]. Since the trace is everywhere negative, all fixed points will either be stable or saddle nodes, where the former will occur if the determinant is positive, and the latter if the determinant is negative. Table 1 below shows the full classification, which confirms our earlier conjecture about the stability of the fixed points based on qualitative observations.

Fixed Point (x^*, y^*)	$detJ(x^*, y^*)$	$TrJ(x^*, y^*)$	Classification
$(0, 0)$	≈ 1	-2	Stable
$(25, 25)$	≈ -24	-2	Saddle
$(50, 50)$	≈ 1	-2	Stable

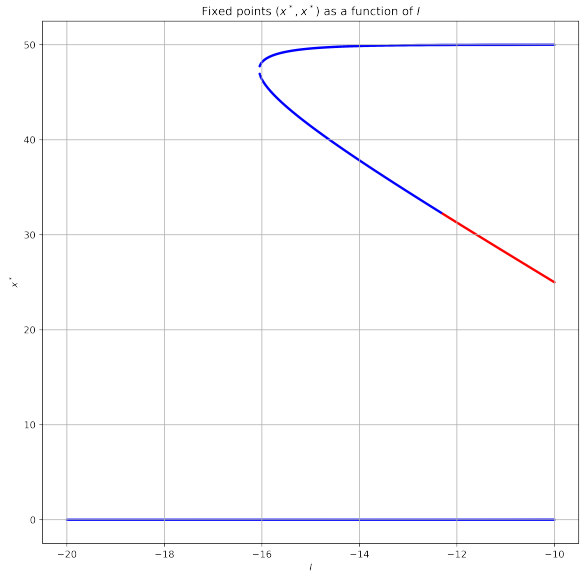
Table 1: Fixed point stability classification.

2.3 Bifurcation Plot

We now consider what will occur to the fixed point locations as we vary I from -20 to -10. In Figure 2.4a we plot the x- and y-clines for a number values in this range. We see that while I is small, there is only one fixed point at around $I \approx 0$. As we increase I to roughly -16, however, the x- and y-clines meet elsewhere for the first time, causing two new fixed points to come into existence.



(a) The x- and y-clines for the two neuron system plotted for different values of I. The value of I is denoted by color. The x-cline is plotted in a solid line, and the y-cline is dotted.



(b) Bifurcation plot of fixed point location for two neuron network. All fixed points are of the form (x^*, x^*), and thus x^* is plotted as a function of the parameter I. Blue points denote stable nodes, while red denote saddle nodes.

Figure 2.4

In Figure 2.4b, we show the location x^* of the fixed points (x^*, x^*) as a function of I. We also determined their stability numerically. We see that there are two regimes in the range of I we are considering. In the first, when I is small, there is one stable fixed node at $I \approx 0$. At around $I = -16$, we enter a new regime where there are three fixed points. We note however that while it seems that the middle fixed point appears to be stable for some time after the bifurcation before switching to saddle node, this is due to numerical error. Stability calculations are sensitive to small imprecisions in the neighborhood of a bifurcation, and since we have only approximate solutions for the fixed point locations, errors are likely to occur in this regime. The most likely explanation is that this is a *saddle node bifurcation* and that the middle solution is indeed a saddle node for all values of I after the bifurcation.

3 Hopfield Networks

We now turn our attention to more general N-dimensional neural networks. In particular, in this section we will explore a well-known neural network based model of episodic memory called a *Hopfield network*. Hopfield networks are recurrent neural networks with neurons who, in the absence of noise, can take values in $\{0, 1, -1\}$. Their dynamics are governed by the equation

$$\frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + f(W\mathbf{x}(t)) + \sigma\eta(t), \tag{3.1}$$

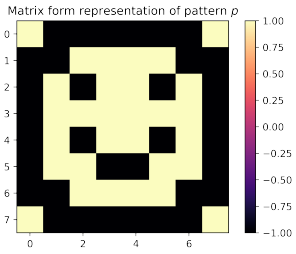
where

$$f(s) = \begin{cases} 1, & s > 0 \\ 0, & s = 0 \\ -1, & s < 0 \end{cases}. \tag{3.2}$$

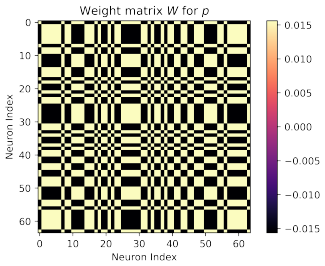
It has been popularized as a model of episodic memory since, for the right choice of weight matrix W, it is able to store specific patterns of activations. Moreover, the system will evolve to recover these patterns when starting from a similar state, akin to how a small stimulus, like a smell, can trigger the recovery of a whole memory. That is, these stored patterns are attractors of the network.

3.1 Storing One Pattern

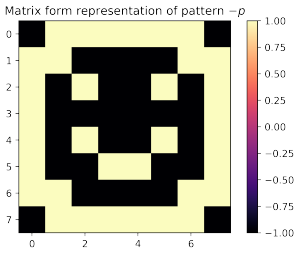
To begin, let's create a 64 neuron Hopfield network that stores a pattern $p \in \{-1, 1\}^{64}$. An 8×8 matrix form representation P of one such pattern is shown in Figure 3.1a, which was made via the rule $P(i, j) = p(8 * i + j)$.



(a) Matrix representation P of a smiley-face pattern p. Note that $P(i, j) = p(8 * i + j)$.



(b) Weight matrix for p created by the Hopfield memorization rule $W = \frac{1}{N} pp^T$.



(c) Matrix representation $-P$ of the inverse of the smiley-face pattern $-p$.

Figure 3.1

To create a Hopfield network that stores the pattern p, we compute its corresponding weight matrix W via the Hopfield memorization rule

$$W = \frac{1}{N} pp^T, \tag{3.3}$$

which we visualize in Figure 3.1b.

We now simulate the dynamics of the Hopfield network over a time frame of four seconds starting from random initial conditions. It can be seen that on this simulation the network was able to quickly recover the smiley-face pattern.

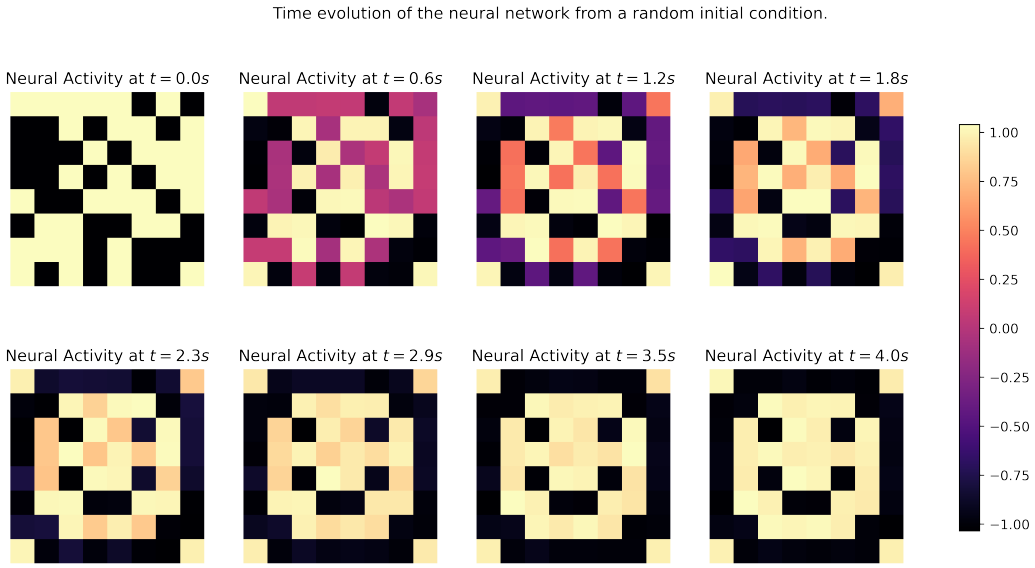


Figure 3.2: Time evolution of Hopfield network defined by W starting from random initial conditions. The simulation was performed via Euler integrations with time steps of $\Delta t = 0.1s$. We plot the matrix form visualization of the state of the network for eight linearly spaced time points.

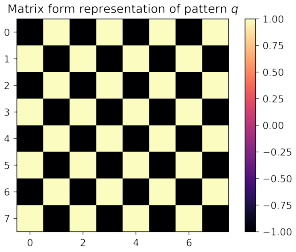
This is, however, not the only fixed point that the network will converge. It will also converge to the inverse of the pattern $-p$, as shown in Figure 3.1c. To see this, we repeat the simulation from random initial conditions for $N = 1000$ trials. We found that, within a tolerance of 0.2 in the euclidean norm, the network converged to the pattern p and $-p$ a total of 514 and 486 times respectively, suggesting that both final states may be more or less equally likely.

3.2 Two Pattern Networks

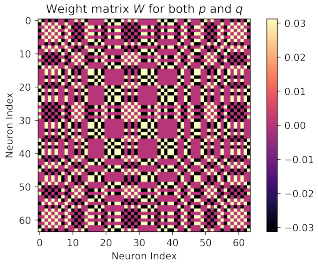
It is possible to store more than one pattern in a Hopfield network. The network defined by the weight matrix

$$W = \frac{1}{N}(pp^T + qq^T) \tag{3.4}$$

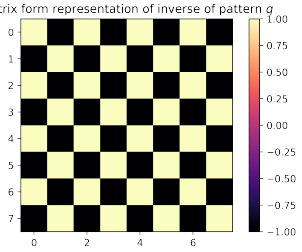
will be able to store both the patter p and q. We now repeat the experiments from the previous section with smiley-face pattern p (Figure 3.1a)and a checker-board pattern q (Figure 3.3a).



(a) Matrix representation Q of a checker-board pattern q.



(b) Weight matrix for p and q created by the Hopfield memorization rule $W = \frac{1}{N}(pp^T + qq^T)$.



(c) Matrix representation $-Q$ of the inverse of the checker-board pattern $-q$.

Figure 3.3

The weight matrix obtained via Equation 3.4 is show in Figure 3.3b. Figure 3.4 shows the time evolution of the Hopfield network defined by the weight matrix starting from a random initial condition. We see that it is able to quickly converge to the pattern q.

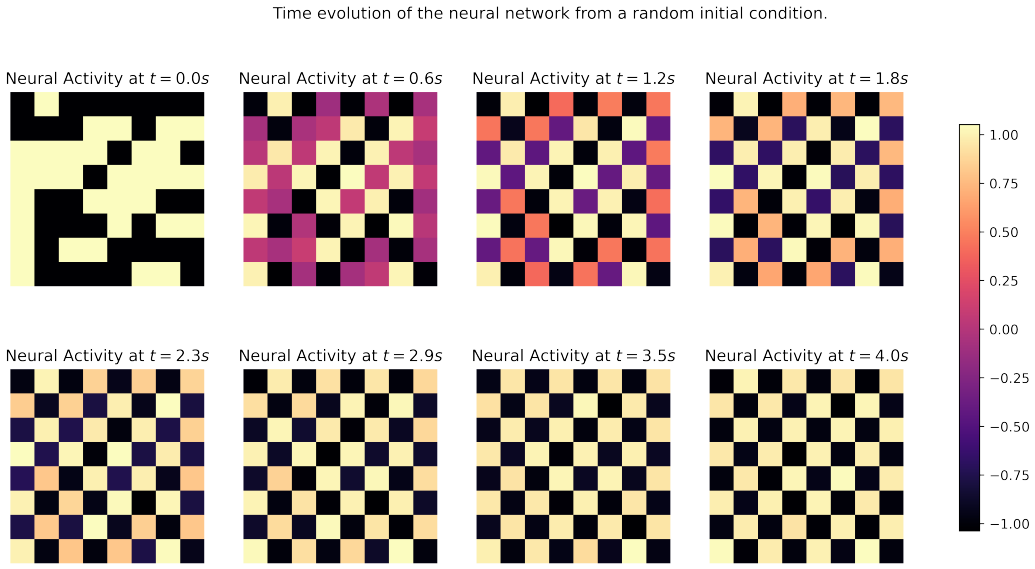


Figure 3.4: Time evolution of Hopfield network defined by $W = \frac{1}{N}(pp^T + qq^T)$ starting from random initial conditions. The simulation was performed via Euler integrations with time steps of $\Delta t = 0.1s$.

As before, the network also stores the negatives of the patterns p and q. After running the simulation 1000 times for 100 time steps from random initial conditions, we found that it converged to the patterns p, $-p$, q, and $-q$ a total of 270, 240, 255, and 235, respectively. As before, this suggests that the network has about an equal change to converge to any p, $-p$, q, or $-q$ when starting from random initial condition.

Moreover, if we give the network a "corrupt" pattern, it will relax back to the original pattern in time. Figure 3.5 shows the time evolution of the system starting from a copy of p that has has gaussian white noise added to each entry. We indeed

find that when starting from this initial condition, it is able to recover (or "remember") the correct pattern p.

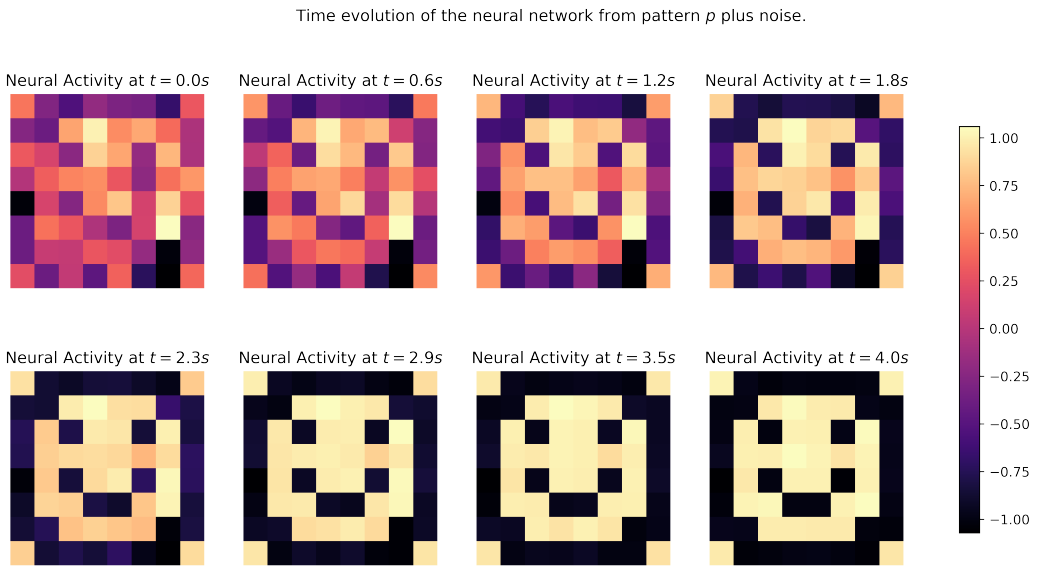


Figure 3.5: Time evolution of Hopfield network defined by $W = \frac{1}{N}(pp^T + qq^T)$ starting from random initial conditions. The simulation was performed via Euler integrations with time steps of $\Delta t = 0.1s$.

3.3 Performance:

More generally, a Hopfield network can store M separate patterns via the rule

$$W = \frac{1}{N}\sum_{i=1}^{M} p_i p^T. \tag{3.5}$$

There is, however, a limit to how many patterns can be stored before the network's capacity to perfectly recall patterns is corrupted. We investigate this phenomena in the case of an $N = 64$ neuron network. To start, for M ranging from 1 to 40, we create weight matrix that stores M randomly chosen patterns. We then feed each pattern back into the corresponding Hopfield network along with a noisy perturbation. If the network is able to recover the pattern, we count it as a success. Its recall rate is the proportion of the M patterns it was able to recall successfully. Figure 3.6 shows the recall rate for one such numerical experiment as a function of M.

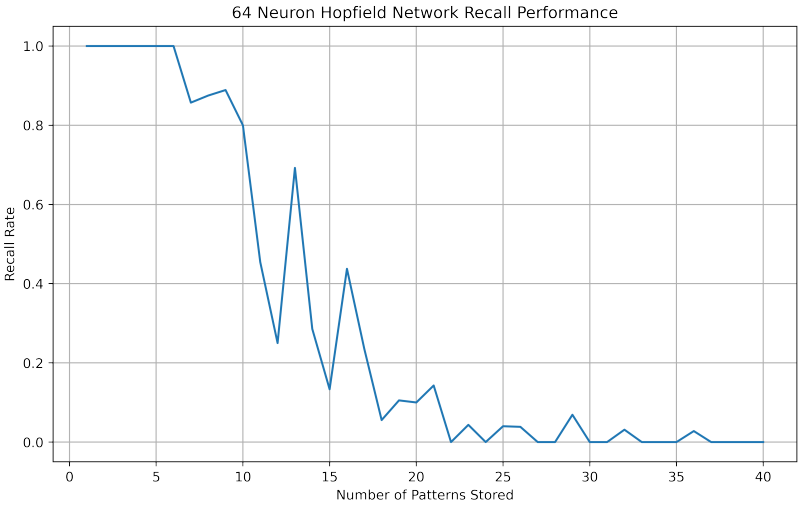


Figure 3.6: Recall rate of a 64 neuron Hopfield network storing M random patterns.

We see that performance begins to steeply decline at around $M = 6$ until around $M = 22$, at which point the recall rate forever stays below 10%.

4 Ring Networks and Dimensionality Reduction

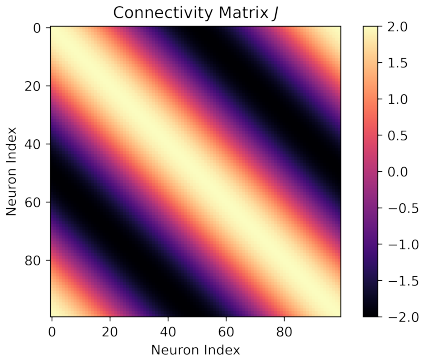
In this section, we will simulate an N-dimensional network called a ring attractor. We will then perform a dimensionality reduction method called *principal component analysis* on the neural data. The dynamics of the network are governed by the following equation

$$\frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + J\tanh(\mathbf{x}(t)), \tag{4.1}$$

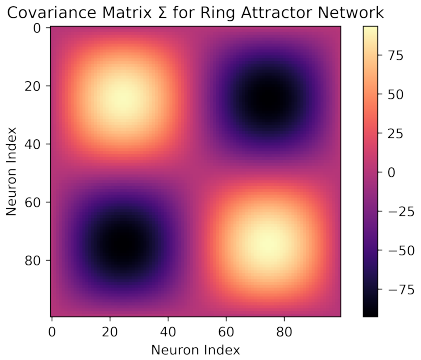
where $\mathbf{x}(t) \in \mathbb{R}^n$ is the current state of the system, and $J \in \mathbb{R}^{n\times n}$ is the connectivity matrix whose entries are given by

$$J_{ij} = 2\cos(\theta_i - \theta_j) \tag{4.2}$$

and reflect the strength of the connection between neurons i and j. For $N = 100$ neurons and linearly spaced $\theta \in [0, 2\pi)$, the connectivity matrix appears as in the following Figure 4.1a. We observe that connections are strongest for neurons that are close in index and decrease with distance. [1]



(a) Connectivity matrix J for the ring attractor network.



(b) Covariance matrix Σ for the ring attractor network.

Figure 4.1

[1] By close in index, we assume that the index set is toroidal, so that 40 and 50 are just as close as 75 and 5.

4.1 Dynamics

We now simulate the dynamics of the system for $N = 100$ neurons and for $t_f = 100$ seconds using Eulers method with time steps of $dt = 0.1$ and random initial conditions. The time evolution of all 100 neurons is shown below in Figure 4.2, which illustrates that the neurons quickly converge to a stable, attracting state.

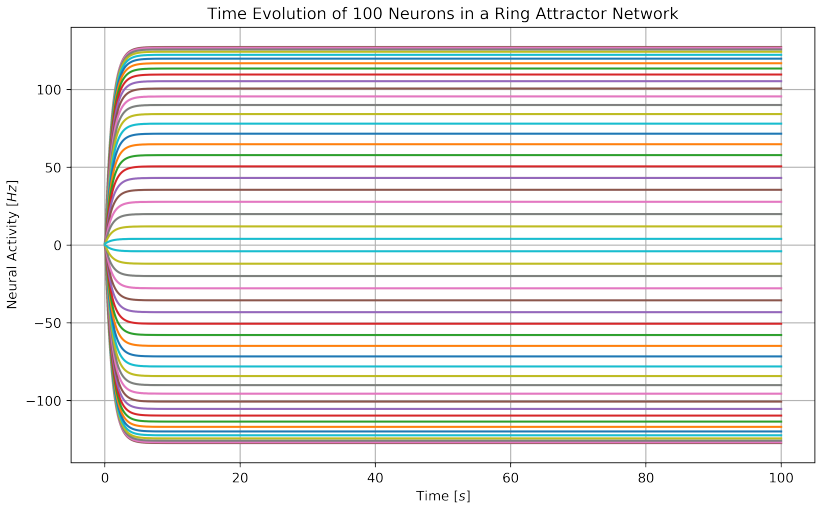


Figure 4.2: Time evolution of ring attractor network for each individual neuron starting from initial conditions $x_i(0) = 0$ for $1 \leq i \leq 50$ and $x_i(0) = 1$ for $51 \leq i \leq 100$.

We then collected the data from the simulation in a $N \times T$ matrix X, where $T = t_f/dt = 1000$. To better understand the intrinsic dimensionality of the data, we perform principal component analysis (PCA), which is a method of finding a new coordinate system for the data where each new ordered basis vector explains the maximum possible variance in the data not explained by the preceding basis vectors. We often find that PCA reveals that only a few components are needed to explain the vast majority of the variance of the data. By then projecting our data on to the few components that explain the most variability, we are able to reduce the dimensionality of the data without losing too much explanatory power.

To begin PCA, we must first center the data matrix by computing and subtracting the mean activity across each row to obtain our centered matrix X'. We then compute the covariance matrix via the formula

$$\Sigma = \frac{1}{T} X' X'^t, \tag{4.3}$$

where X'^t is the transpose of X. The resulting covariance matrix is depicted below in Figure 4.1b. Since Σ is a positive-definite matrix, all of its eigenvalues are positive and can be ordered by $\lambda_1 \geq \lambda_1 \geq \cdots \geq \lambda_N$. The principal components of the matrix are given by the corresponding eigenvectors who inherit this ordering. Furthermore, the variance explained by each principal component can be computed by

$$EV_i = \frac{\lambda_i}{\sum_{j=1}^{N} \lambda_j.} \tag{4.4}$$

Using Equation 4.4, we plot the variance explained by the first 10 principal components in Figure 4.3 below

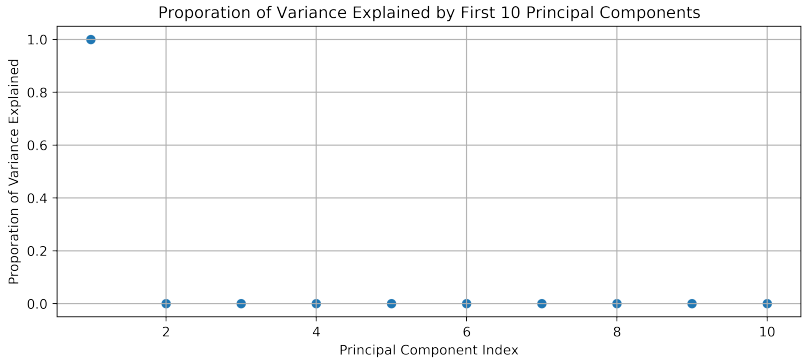


Figure 4.3: Variance in the data gathered from the ring attractor network simulation explained by the first 10 principal components.

In this case, we observe that just the first principal component can explain 99.99% of the variance in the data. In a sense, this is what we would expect from Figure 4.2, since each neuron quickly converges to a constant firing rate and it only takes one dimension to describe a constant $v \in \mathbb{R}^{100}$. A plot of the first principal component is shown below in Figure 4.4

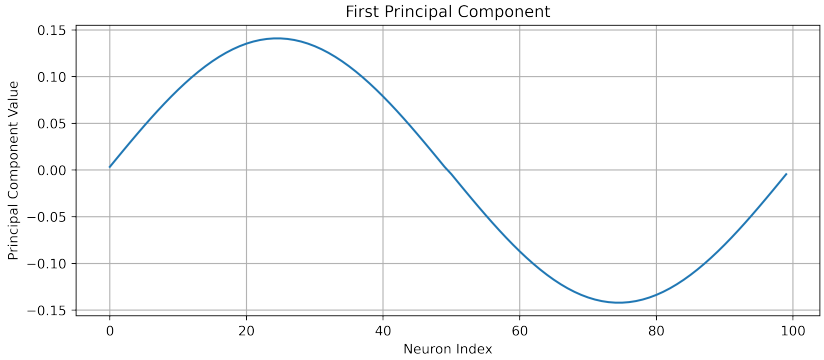


Figure 4.4: The first principal component in our simulation of the ring attractor network.

Unsurprisingly, we find the weight distribution from the first principal component across the neurons appears sinusoidal.

4.2 PCA on the Set of Attractors

We repeat the simulation process outlined above starting from random conditions for 500 trials. The resulting final state of each trial, which correspond to different attractors of the network, were gathered in a 100×500 matrix Q. To better understand the attractor structure of the network, we perform PCA on Q, following the procedure outlined in the previous section. To begin, we visualize the covariance matrix of the centered Q. It is shown in Figure 4.5.

We repeat the manual calculation of the eigenvalues and eigenvectors of the covariance matrix from the last section. We plot the first 10 eigenvalues in Figure 4.6 below. Using Equation 4.3 we can calculate the explained variance from each component, which shows that most of the variance in the attractor is explained by the first and second principal component (51.01% and 48.98%, respectively).

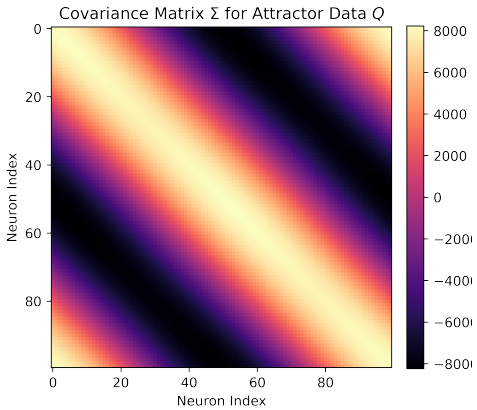


Figure 4.5: Covariance matrix Σ of Q.

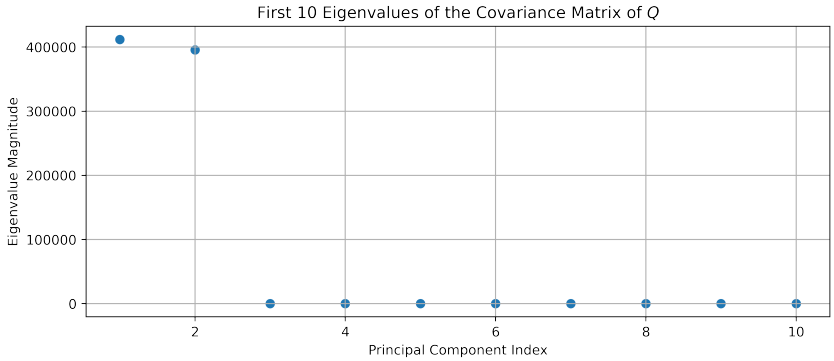
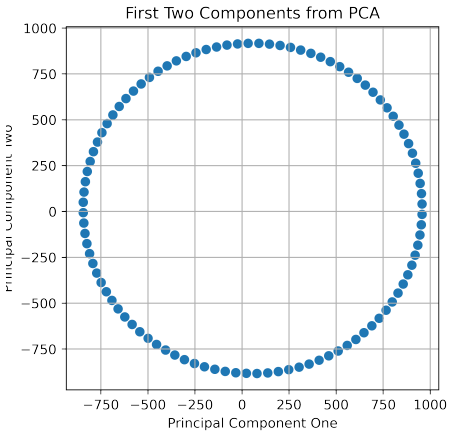


Figure 4.6: First ten eigenvalues of covariance matrix of Q.

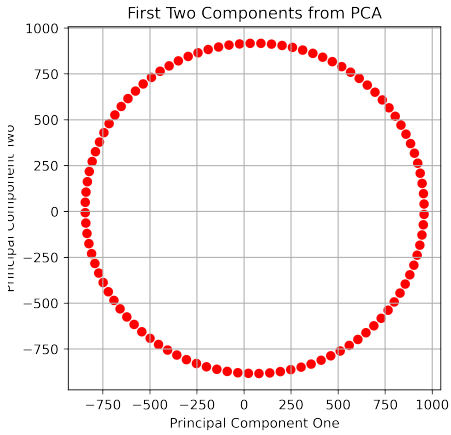
This suggests that it will be sufficient to consider only the first two principal components. We project the centered Q (which we denote Q') onto the components via

$$Q_{pc} = Q'^{t} V_d, \tag{4.5}$$

where V_d is 100×2 matrix whose columns are given by the the first two principal components. Figure 4.7a shows the projected data. This reveals where the ring attractor network gets its name: the attractors exist on a ring that embeds in a two dimensional space defined by the first two principal components.



(a) Attractor data projected onto first two principal components via manual calculations.



(b) Attractor data projected onto first two principal components using `decomposition.PCA`.

Figure 4.7

To verify the results of the analysis above, which was done by manually performing (with the aid of python) each step of the PCA procedure, we repeat the analysis by using the `decomposition.PCA` function from the library `sklearn`. We then call this function to recover the first two principal components via the command `PCA(n_components=2).fit_transform(Q)`. The results of projecting are shown below in Figure 4.7b. We find that the projected data is numerically equivalent.

5 Conclusion

In this report, we saw how firing-rate based network models could be used to model neural dynamics, and how the tools from dynamical systems and dimensionality reduction methods could be used to investigate their properties. In the initial part of our study we illustrated how even low-dimensional firing-rate based models could exhibit rich dynamical properties. Furthermore, we saw how the tools of dynamical system studies, such as bifurcation and stability analysis, could be fruitfully applied to study these models. We then saw how the Hopfield network could be used to store and recover "memories" of neural activity patterns, gesturing towards its applicability to fields such as cognitive science. Finally, we considered the ring attractor network, and demonstrated how its attractor dynamics could be described by only two principal components, despite its high extrinsic dimensions.

References

[1] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2001.

[2] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Studies in nonlinearity. Westview, 2000.

A Proof that Fixed Points are Symmetric

Claim: If (x^*, y^*) is a fixed point of the system described Equations 2.1 and 2.2 with $w_1 = w_2$, then $x^* = y^*$.

Proof. If (x^*, y^*) is a fixed point, then

$$\dot{x}(x^*, y^*) = \dot{y}(x^*, y^*) = 0. \tag{A.1}$$

From this, it follows that

$$-x^* + f(wy^* + I) = -y^* + f(wx^* + I) \tag{A.2}$$

$$\Rightarrow x^* + f(wx + I) = y^* + f(wy^* + I). \tag{A.3}$$

Since the sigmoid function $\sigma(s)$ is injective, however, so too is the function

$$g(z) = z + f(wz + I).$$

This follows because the composition of injective functions preserves injectivity. Thus Equation A.3 becomes

$$g(x^*) = g(y^*)$$

which entails that $x^* = y^*$ by injectivity. □