Equivalence of neural field dynamics with different embedding dimensionality

EPFL

Nicole Vadot

(Supervisor) Valentin Schmutz

(Supervisor, Director of LCN) Wulfram Gerstner

2023-06-22

Table of contents

Abstract

Early recordings of cats' somatosensory and visual cortices suggested that the cortical sheet is organized in vertical columns of functionally similar neurons. The columnar organization of the cortex motivated the design of spatially structured models of neural population dynamics, namely neural field models, where the spatial dimensions corresponded to the two dimensions of the cortical sheet. Later it has been shown that not only the physical position of the neurons can be used to parametrize their function, but also their location in more abstract "embedding spaces", for example a space of "neuron celltypes".

Decoupling the embedding space from the physical position of neurons on the cortical sheet begs the question of uniqueness of the embedding space. Given an embedding space and neural dynamics thereon, are there other embedding spaces which can express the same behavior? Is there a "natural" choice for the embedding space?

In this work, we show that the answer to the second question is highly nontrivial because given some neural dynamics, we can find multiple equivalent embedding spaces in different dimensions. We illustrate this using a toy example of a 2 (or 3)-dimensional embedding that is mapped to an equivalent 1-dimensional embedding. This is done by applying measurable bijective mappings $S : [0,1]^2 \mapsto [0,1]$, which lead, as we show, to well-behaved numerical simulation schemes. In this, we exploit the key concept of "locality", which expresses the property of (possibly fractal) neural fields, in which neighbouring neurons in the embedding space *tend* to have similar functions.

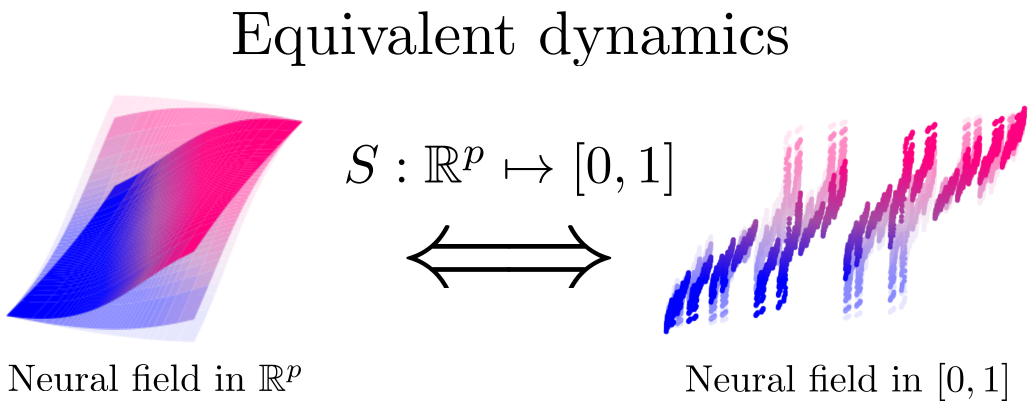


Figure 1: Smooth neural fields in p dimensions can be mapped to equivalent neural fields in one dimension, and the dynamics are equivalent.

1 Introduction

1.1 Neural field models in neuroscience

Early recordings of cats' somatosensory [1] and visual [2] cortices suggested that the cortical sheet is organized in vertical columns of functionally similar neurons.

The columnar organization of the cortex motivated the design of spatially structured models of neural population dynamics, namely neural field models [3]–[5], where the spatial dimensions corresponded to the two dimensions of the cortical sheet.

As models of spatiotemporal neural population dynamics on the cortical sheet, neural field models can be used to explain experimentally observed patterns of cortical activity, such as travelling waves in visual [6], [7], somatosensory [8], [9], motor [10], [11], and hippocampal [12]–[14] cortices (see [15] for a review), and are used to model large-scale brain signals, such as electroencephalography (EEG) recordings [16] (see also [17] for an example of neural field modelling of calcium imaging recording of the visual cortex).

More recent recording methods, especially in rodents, have provided evidence of low-dimensional organization of neuronal activity that does not depend on the physical locations of neurons on the cortical sheet. For example, contrary to cats' visual cortex, the orientation selectivity of pyramidal neurons in a rat's visual cortex does not depend on their locations, that is, orientation selectivity is heterogeneous at any given location [18]. Moreover, the activity of interneurons in any small volume of the visual cortex is also heterogeneous and seems to be structured by a low-dimensional manifold of fine cell subtypes [19]. The local functional heterogeneity of neuronal activity in the cortex challenges the old concept of functionally homogeneous cortical columns and, with it, classical neural field models.

While, historically, the spatial dimensions in neural field models have been the two dimensions of the cortical sheet (with the notable exception of the ring models for orientation selectivity in the visual cortex [20] and the head-direction system [21]), the "space" in neural field models does not need to represent physical space but can represent any suitable abstract embedding space. Neural field models with abstract embedding spaces could constitute a generalization of classical neural field theory for cortical networks that are not solely structured by distances of neurons on the cortical sheet.

If we consider neural fields in abstract embedding spaces, which embedding space should we choose? More fundamentally, given some spatiotemporal dynamics, is there a “natural” choice for the embedding space?

The goal of this Master Thesis is to show that the answer to the second question is highly nontrivial because even the dimensionality of the embedding space, for some given spatiotemporal dynamics, is not clearly defined. This means that it is possible to find two embeddings (each associated with a connectivity kernel) of different dimensions that give rise to identical dynamics.

In particular, we show that for any two-dimensional neural field equation (that is, with two spatial dimensions), there is a one-dimensional neural field equation, where the embedding space is simply the interval $[0, 1]$, from which the solution to the two-dimensional equation can be fully reconstructed. The mapping from the two-dimensional equation to the one-dimensional equation is done via well-known measurable bijections between the square $[0, 1]^2$ and the interval $[0, 1]$. Importantly, these bijections are not (and can't be) diffeomorphisms. Also, we show that although the connectivity kernel of the one-dimensional equation seems to have a fractal structure, it is sufficiently regular for its solution to be numerically approximated using standard grid-based simulations.

Our results suggest that the notion of (spatial) dimensionality in neural field dynamics is not well-defined if not associated with a constraint on the regularity of the neural fields (and the connectivity kernel). Thereby, by studying a simple toy model, this work illustrates the importance of the analytic notion of regularity when neural population dynamics over abstract continuous spaces are considered.

1.2 Neural fields as the limit of spatially-structured networks of neurons

Let us now give a short introduction to the mathematical theory of neural fields. Neural field equations emerge from the spatial structure of networks of rate neurons (or Poisson spiking neurons [22], [23]) when the number of neurons N becomes large. Positions $\boldsymbol{z}_i \in \mathbb{R}^p$ are attributed to each neuron, where $\mathbb{R}^p$ is the *embedding space*. Each neuron has a potential $h_i(t)$ that evolves in time, and the weight with which the neurons interact with each other is given by the entries of the connectivity matrix $J_{ij} \in \mathbb{R}^{N\times N}$. Additionally, the activation function $\phi(h)$ describes the firing rate of a given neuron with potential h. Then, the product $J_{ij}\phi(h_j)$ gives the *recurrent current* of neuron j on neuron i.

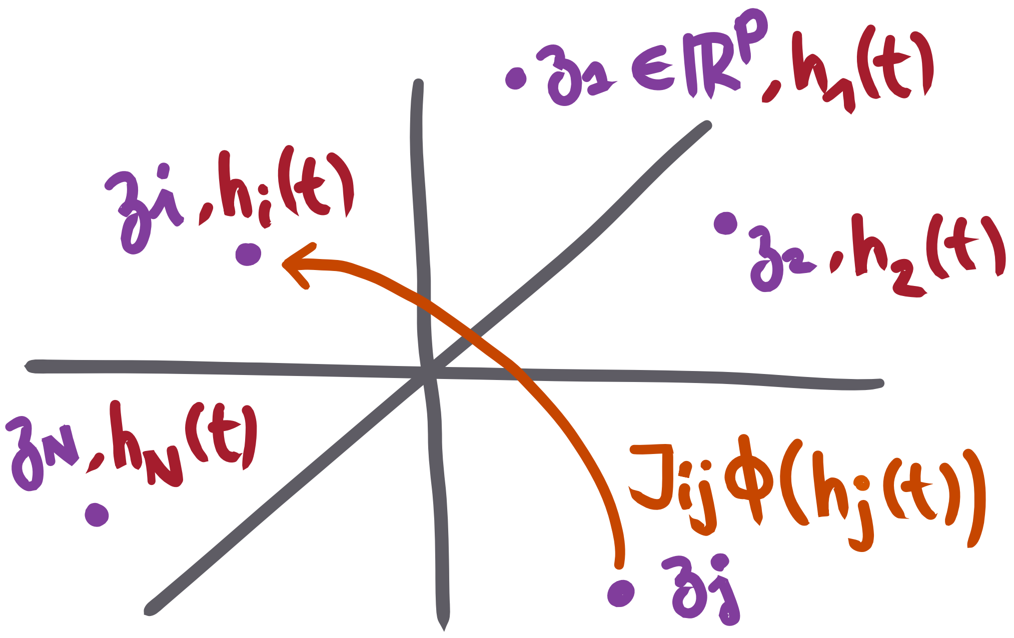


Figure 1.1: Every neuron in the neural network has a position in the p-dimensional embedding space.

If the embedding is well-chosen,[1] the connectivity matrix J_{ij} approaches a connectivity kernel $w(\boldsymbol{y}, \boldsymbol{z})$ as $N \to \infty$, and neurons close in the embedding space tend to have similar potentials, such that we can write the continuous *neural field* $h(\boldsymbol{z}, t)$ of neuron potentials. Before we write down the neural field equation which describes how the field evolves in time, we need to introduce the neuron density $\rho(\boldsymbol{z})$, which measures the number of neurons per unit volume of the embedding space. The neural field equation writes:

$$\partial_t h(\boldsymbol{z}, t) = -h(\boldsymbol{z}, t) + \int_{\mathbb{R}^p} w(\boldsymbol{z}, \boldsymbol{y})\phi(h(\boldsymbol{y}, t))\rho(\mathrm{d}\boldsymbol{y}). \tag{1.1}$$

1.3 Motivation for a one-dimensional neural field

This work was inspired by recent mathematical development in mean-field theories involving the theory of graphons [25]. Informally, these recent results suggest that, as long as weights scale as $1/N$, the dynamics of large networks can be described, as $N \to \infty$, by a neural field equation where the embedding is simply the interval $[0, 1]$, which motivates the search for a formulation of a one-dimensional neural embedding.

[1]Rigorous proofs of convergence are still an active area of study [23], [24] and are not the subject of this thesis.

The main idea is to define a measurable bijective mapping $S : \mathbb{R}^p \mapsto [0,1]$ and a measure $\lambda : \mathbb{R}^p \mapsto \mathbb{R}^+$, such that we can write a one-dimensional neural field $\tilde{h}(\alpha, t)$ with a connectivity kernel loosely defined as $\tilde{w} = w \circ S^{-1}$.

$$\partial_t \tilde{h}(\alpha, t) = -\tilde{h}(\alpha, t) + \int_{[0,1]} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right](\mathrm{d}\beta)$$

This change of variable in the kernel seems counter-intuitive: how is it possible to map dynamics in the embedding space $\mathbb{R}^p$ to dynamics in the lower dimensional embedding space $[0,1]$ without losing information? And even if mathematically such a change of variables is possible, is it even feasible to numerically simulate the dynamics in the lower dimensional embedding space?

1.4 Numerical simulations of neural fields

The goal of this thesis is to not only give an intuition on the mappings between the $[0,1]$ and $\mathbb{R}^p$ embedding spaces, but also show that the mathematical construction works numerically.

We distinguish two types of numerical simulations for the neural field in Equation 1.1:

1. A neuron-based simulation of N neurons. The location of each neuron is an independent sample from the distribution $\rho(z_1, \cdots, z_p)$. This case then just reduces to simulating the network of rate neurons. In other words, we are simulating the full network of rate neurons, and the simulation converges to the true mathematical neural field equation as N becomes large.
2. A grid-based simulation. In the $\mathbb{R}^p$ space, we discretize each dimension, forming a grid of p-dimensional bins. Each bin effectively corresponds to an infinite population of neurons, which are approximated to be the same (this is a mean-field approximation). This works because by continuity of the field, neurons which are close in $\mathbb{R}^p$ also have similar potentials. As we increase the number of bins, the simulation becomes more precise and the dynamics approach that of the true mathematical neural field.

With this in mind, we can try to understand why simulating a neural field in $[0,1]$ might be difficult. Suppose we take bins in the $\mathbb{R}^p$ space, and map each bin to a random position in $[0,1]$. Intuitively, this won't work (in the sense that the dynamics in $[0,1]$ will not be the same as the dynamics in $\mathbb{R}^p$), because the hypothesis of continuity is not longer true: the mapping is random, so the nearby neurons in $[0,1]$ will no longer have similar potentials, and the error of the mean-field approximation will no longer vanish as we take increasingly many (and smaller) bins. This is visualized in Figure 1.2.

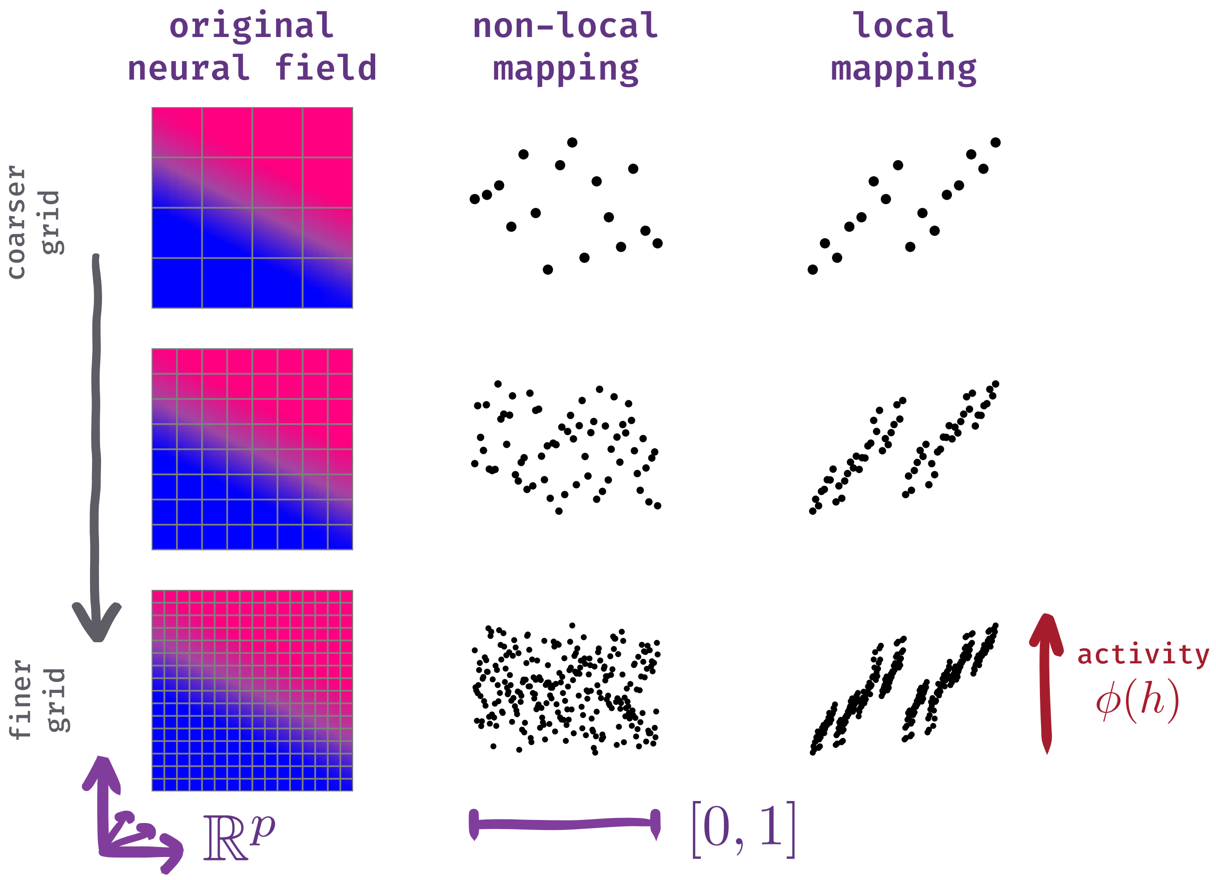


Figure 1.2: Good mappings from $\mathbb{R}^p$ to $[0, 1]$ should maintain some sense of “locality”.

In some sense, we would like to find a mapping that conserves “locality”: populations that are close in $[0,1]$ should represent nearby populations in $\mathbb{R}^p$, or at least there shouldn't be “too many large discontinuities”.

1.5 Summary of contents

Chapter 2 introduces a toy model for a neural field in p dimensions, and tools to study its dynamics. We also discuss interpretations of numerical simulation schemes.

Chapter 3 is dedicated to the discussion of mappings from 2 dimensions to 1 dimension. We give intuitions accompanied by simulations, and discuss in detail how “locality” emerges from these mappings.

Finally, Chapter 4 tries to formalize the intuitive results obtained in the previous chapters. We build a metric to measure the “discretization error”, and provide the key arguments for a proof of convergence of the numerical approximations.

2 Neural field toy model and simulations

In this chapter, we introduce a toy model so that we can later study the mappings of embedding spaces. We stress that this toy model serves only for illustration purposes, and our results apply to any neural field. For this reason, we only cite or give short derivations of results used to understand the model, and the influence of the mappings that are later applied to it.

2.1 Networks of neurons to neural fields

2.1.1 Low-rank networks of neurons

The toy model we introduce is an instance of a low-rank neural network. Such models hold their name from the form of their connectivity matrix, which has a rank of (at most) p. We write the connectivity matrix as follows:

$$J_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} F_{\mu i} G_{\mu j}. \tag{2.1}$$

The geometric view of these networks is that the recurrent currents lie in a p-dimensional subspace of $\mathbb{R}^N$ spanned by the vectors $\{\boldsymbol{F_1}, \cdots, \boldsymbol{F_\mu}, \cdots, \boldsymbol{F_p}\}$ (where each $\boldsymbol{F_\mu} = (F_{\mu 1}, \cdots F_{\mu N})$), which therefore define a "natural" linear embedding (in the sense of [26]) of the neural population activity.

Recalling the notation from the introduction, the $h_i(t)$ are neuron potentials (with initial condition $h_i(0)$), $\phi : \mathbb{R} \mapsto \mathbb{R}^+$ is the monotonic increasing activation function, and we write the evolution equation for this network of neurons:

$$\dot{h}_i(t) = \underbrace{-h_i(t)}_{\text{exponential decay}} + \underbrace{\frac{1}{N} \sum_{\mu=1}^{p} \sum_{j=1}^{N} F_{\mu i} G_{\mu j} \phi(h_j(t))}_{\text{recurrent current } I_i^{\text{rec}}(t)}. \tag{2.2}$$

The exponential decay term describes how, in isolation, a neuron's potential tends to zero, biologically corresponding to the decay of the membrane potential to its value at rest. The recurrent current term $I_i^{\text{rec}}(t)$ describes the "currents" received by neuron i from all the other neurons in the network.

We would like to add a few remarks on the physical units of Equation 2.2. Strictly speaking, a prefactor τ with units of seconds^{-1} should be multiplied with $\dot{h}_i(t)$ of units volt $\cdot$ seconds^{-1}, such that the left-hand side of the equation is consistent with the unit of voltage. Similarly, the activation function has units seconds^{-1} and the connectivity matrix units of coulomb, and a multiplicative term R with units ohm should multiply the recurrent currents in order to yield a voltage. By rescaling time or connection weights, both τ and R can be set to one, and so we omit to write them.

One last remark is that in our model, there is no external current; we only study the autonomous dynamics of the system.

2.1.2 Gaussian low-rank network of neurons

Let us now specify the low-rank model introduced in the previous section, and use the model introduced in [27], which can be seen as a simple example of a "Gaussian mixture low-rank network" [28]. This paper defines the low-rank vectors $F_{\mu i}$ such that each component independently samples a standard Gaussian (zero mean, unit variance). In other words, every neuron i samples a vector $\boldsymbol{F_i} = (F_{1i}, \cdots, F_{pi})$ from the p-dimensional gaussian. We write

$$\boldsymbol{F_i} = \boldsymbol{z_i}, \ \ \boldsymbol{z_i} \overset{\text{iid}}{\sim} \rho(z_1, \cdots, z_p),$$

where

$$\begin{aligned} \rho(z_1, \cdots, z_p) &= \prod_{\mu=1}^{p} \mathcal{N}(z_\mu) \\ \mathcal{N}(z) &= \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}z^2}. \end{aligned}$$

The vectors $G_{\mu i}$ are defined as

$$G_{\mu i} = \tilde{\phi}(z_{\mu i}) \overset{\text{def}}{=} \frac{\phi(z_{\mu i}) - \langle \phi(z_{\mu i}) \rangle}{\text{Var}[\phi(z_{\mu i})]}.$$

The motivation for this choice will be presented in a few paragraphs. Finally, the activation function remains to be defined. For simplicity, we take it to be the logistic function, although the following results are not sensitive to this choice,

$$\phi(h) = \frac{1}{1+e^h}.$$

Putting everything together, we get the following expression for the equation of evolution of the network of neurons:

$$\dot{h}_i(t) = -h_i(t) + \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} z_{\mu i}\tilde{\phi}(z_{\mu j})\phi(h_j(t)) =: \mathrm{RHS}(h_1, \cdots, h_N). \tag{2.3}$$

We note that contrary to the model introduced in [27], we ignore the self-connections term, because it introduces a vanishing correction of order $\mathcal{O}(1/N)$.

To understand this model better, we can analyze its fixed points and their stability. Here we summarize the results from the derivation presented in Appendix A. We use the notation $h^\star$ to refer to fixed points of the network. The fixed points solve the roots of the evolution equation:

$$\mathrm{RHS}(h_1^\star, \cdots, h_N^\star) = -h_i^\star + \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} z_{\mu i}\tilde{\phi}(z_{\mu j})\phi(h_j^\star) = 0$$

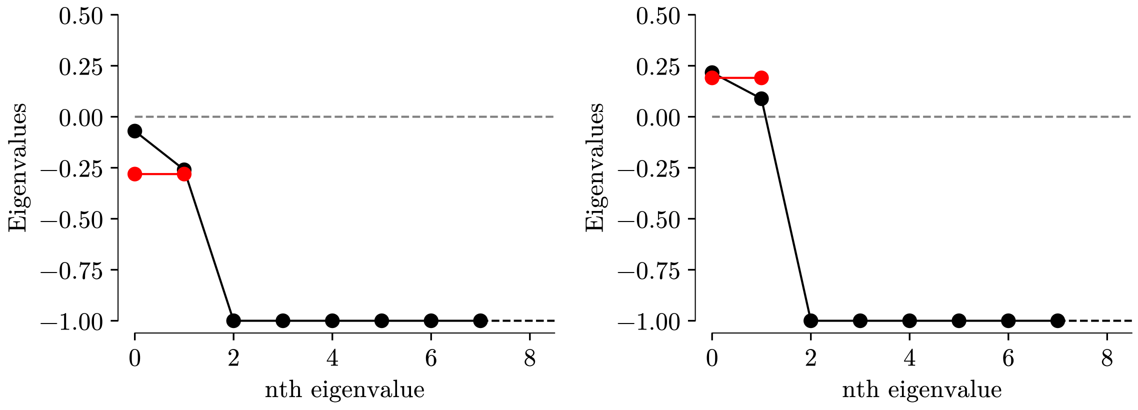
- $h_i^\star = z_{\mu i}$ is a stable fixed point for all $\mu \in \{1, \cdots, p\}$. Additionally, because the Gaussian distribution is symmetric around zero, $h_i^\star = -z_{\mu i}$ are also stable fixed points. We refer to these fixed points as the "pattern" fixed points, because the network of neurons "remembers" the vectors $\boldsymbol{F_1}, \cdots, \boldsymbol{F_p}$.
- $h_i^\star = 0$ is an unstable fixed point. It corresponds to all the neuron potentials being set to zero.

The analysis of stability from Section A.3 can be summarized by the study of the eigenvalues of the matrix $K_{ij} = J_{ij}\partial\phi(h_j^\star) - \mathrm{Id}_{ij}$.

1. Taking Taylor expansions of the activation functions reveals that – at least up to order 3 – uneven powers tend to increase stability of the pattern fixed points (and reciprocally, decrease stability of the zero fixed point), and vice-versa for the even powers (with the notable exception of the constant offset, which doesn't play a role in this analysis).
2. A steeper slope at the origin of the activation function also seems to improve stability of the pattern fixed points. This corresponds to a sharper difference between the "inactive" (low potential) and the "active" (high potential) neurons.
3. The spectrum of K is composed of p eigenvalues (labeled λ_μ for all $\mu \in \{1, \cdots, p\}$) that depend on the fixed point, and $N - p$ eigenvalues $\lambda_\perp = -1$ corresponding to the orthogonal complement of the pattern fixed points.

In our case, this analysis in Taylor expansions was sufficient to explain the observed stability resulting from a logistic activation function.

We should however note that the results obtained in Appendix A are valid in the $N \to \infty$ limit. A numerical analysis shows in Figure 2.1 that at $N = 250$, the numerical eigenvalues approximate the analytical eigenvalues, and this correspondence improves when we take larger N (typically we will for the rest of this thesis take $N > 10^3$).



(a) Pattern (stable) fixed point (b) Zero (unstable) fixed point

Figure 2.1: Spectrum of K for $N = 250$ and $p = 2$. The numerical estimation (black, eigenvalues λ_μ and $\lambda_\perp = -1$) of the eigenvalues is close to the analytical ($N \to \infty$) derivation (red, only eigenvalues λ_μ). Eigenvalues above the dotted gray line (positive values) correspond to unstable fixed points, and below (negative values) correspond to stable fixed points.

2.1.3 Emergence of structure in the networks of neurons

In the introduction, we introduced the convergence of a network of neurons to a smooth neural field when the embedding space is well-chosen. For our toy model, it seems natural to try to embed the neurons according to their positions on the p-dimensional Gaussian.

2.1.3.1 Numerical aspects of simulating a low-rank network of neurons

We first demonstrate what happens with numerical simulations in the simple case of $p = 1$. We generate a network of 50,000 neurons, where each neuron samples a one-dimensional Gaussian to get its position in the embedding. The function $\tilde{\phi}$ is computed by rescaling ϕ with the numerically computed mean and variance.[1]

Instead of numerically computing the full connectivity matrix J (which would require storing $N^2 = 50,000^2 \sim 10^{10}$ entries, taking about 40 GiB if we use 32-bit floating point numbers... yikes!), we take advantage of its low-rank structure to compute recurrent currents, therefore only storing $p \times N$ numbers instead of N^2.[2]

2.1.3.2 Naive and natural embeddings for $p = 1$

Initially, we set all the potentials to zero, then simulate the differential equation by using the Runge-Kutta of order 4 integration schema (see `scipy.integrate.solve_ivp`). In Figure 2.2, we show the resulting dynamics for a few thousand neurons. We order the neurons as they appear in the numerical array (a naive "random" one-dimensional embedding), and compare this to the ordering of neurons by their sampled position $z_1 \sim \mathcal{N}$.

The results of the simulation show that the random ordering of the neurons does not yield a smooth embedding, in contrast to the smooth surface that appears when we order the neurons by their position z_{1i}. The simulation serves as a demonstration that the neurons converge to the stable fixed point $h_i^\star = z_{1i}$, as well as confirming that the zero fixed point is unstable. Indeed, despite the zero initialization, the simulation did not stay there because small numerical fluctuations were amplified by the instability.

The observed smooth surface corresponds to the activation function $\phi(h^\star) = \phi(z_1)$. Therefore, the Gaussian patterns define a natural embedding of the network, and the neural field at the fixed point reads $h^\star(z_1, t) = z_1$. We can write the neural field equation for $p = 1$ in the Gaussian embedding as

$$\partial_t h(z_1, t) = -h(z_1, t) + \int_{\mathbb{R}} z_1 \tilde{\phi}(y_1) \phi(h(y_1, t)) \mathcal{N}(\mathrm{d}y_1). \tag{2.4}$$

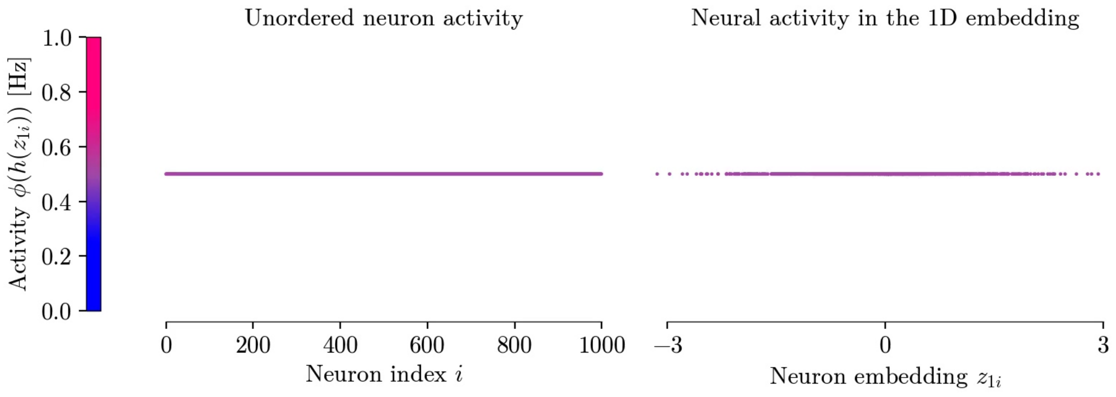
2.1.3.3 A natural embedding for $p = 2$

Because Equation 2.4 already involves a one-dimensional integral, the $[0, 1]$ embedding can easily be found by applying rescaling methods, such as the inverse CDF method described in Section 2.5 later in this chapter.

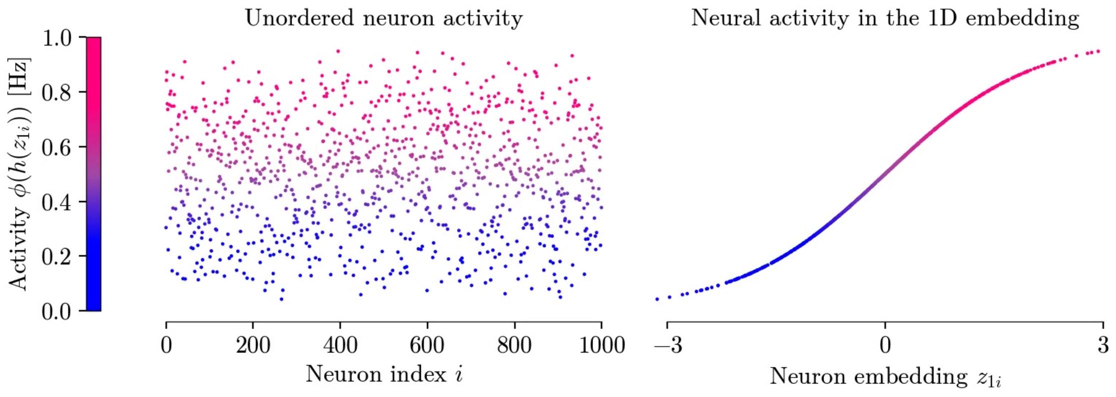
The question of finding a one-dimensional embedding becomes non-trivial when we consider $p > 1$, here we take $p = 2$. In Figure 2.3, we repeat the same simulations as

[1] In the source code, implementation for this can be found under `notebooks/neurodyn/_rnn.py`, in the method `LowRankRNNParams.new_sampled_valentin`.

[2] See the implementation in `notebooks/neurodyn/_rnn.py`, in the method `LowRankRNN.I_rec`.



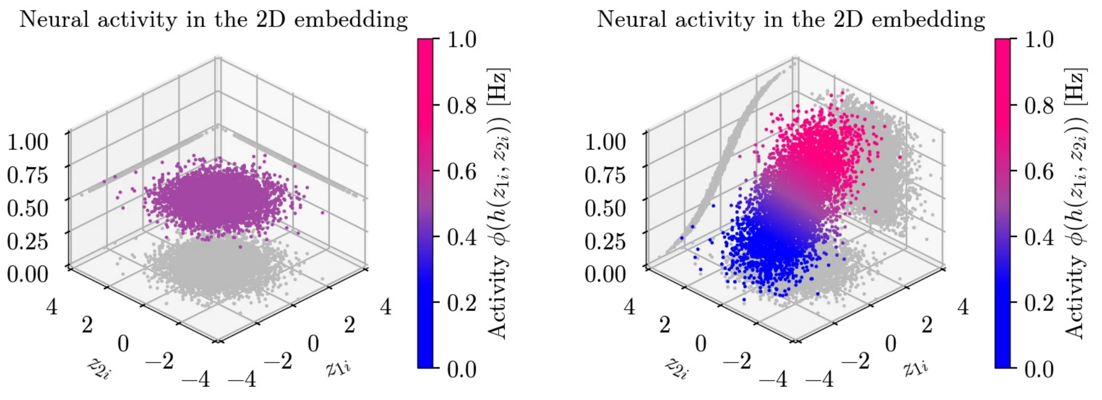
(a) First frame



(b) Last frame

Figure 2.2: Simulation of the Gaussian network of neurons with $p = 1$. Every neuron is represented by a dot, of which the height represents its activity level, and the colorbar doubles as a y-axis. The first and the last frame of the animation are shown, available here.

in the previous section. Again, the structure of the network emerges naturally as a smooth surface in $\mathbb{R}^2$.



(a) First frame

(b) Last frame

Figure 2.3: Simulation of the Gaussian network of neurons with $p = 2$. Every neuron is represented by a dot, of which the height represents its activity level. Projections are shown by gray points. The first and the last frame of the animation are shown, available here.

We however note that in this case, there are two pattern fixed points that the network can converge to. In the example given here, the network coincidentally converged again to the first pattern fixed point. We see this because the neural field is constant in the z_2 direction. We write the fixed point $h^\star(z_1, z_2, t) = z_1$. The neural field is now written as

$$\begin{aligned}\partial_t h(z_1, z_2, t) &= -h(z_1, z_2, t) \\ &\quad + \int_{\mathbb{R}} \left(z_1\tilde{\phi}(y_1) + z_2\tilde{\phi}(y_2)\right) \phi(h(y_1, y_2, t))\mathcal{N}(\mathrm{d}y_1)\mathcal{N}(\mathrm{d}y_2).\end{aligned} \tag{2.5}$$

More generally, we can write the p-dimensional neural field, where $\boldsymbol{z} \in \mathbb{R}^p$:

$$\begin{aligned}\partial_t h(\boldsymbol{z}, t) &= -h(\boldsymbol{z}, t) + \int_{\mathbb{R}^p} w(\boldsymbol{z}, \boldsymbol{y})\phi(h(\boldsymbol{y}, t))\mathcal{N}^p(\mathrm{d}\boldsymbol{y}) \\ w(\boldsymbol{z}, \boldsymbol{y}) &= \sum_{\mu=1}^{p} z_\mu \tilde{\phi}(y_\mu) \\ \mathcal{N}^p(\mathrm{d}\boldsymbol{y}) &= \frac{\exp(\frac{1}{2}\sum_{\mu=1}^p z_\mu^2)}{(2\pi)^{p/2}}\mathrm{d}y_1 \cdots \mathrm{d}y_p.\end{aligned} \tag{2.6}$$

2.2 Simulation schemes for neural fields

Now that we have introduced the analytical neural fields, let us discuss how we can simulate them numerically, more specifically how we can estimate the p-dimensional integral.

2.2.1 Sampling method (Monte Carlo integration)

Due to the formulation of the neural field weighed by a probability distribution $\rho(y_1, \cdots, y_p)$, a natural way to estimate the integral is by Monte Carlo integration. The essence of this method is to use the Central Limit Theorem, where we take N independent samples $\boldsymbol{y_i} = (y_{1i}, \cdots, y_{pi})$ from the distribution ρ, and use them to estimate the integrand $I(\boldsymbol{z}, \boldsymbol{y})$.

$$\begin{aligned} \mathcal{I}(\boldsymbol{z}) &= \int_{\mathbb{R}^p} \underbrace{w(\boldsymbol{z}, \boldsymbol{y})\phi(h(\boldsymbol{y}, t))}_{I(\boldsymbol{z},\boldsymbol{y})} \rho(\mathrm{d}\boldsymbol{y}) \\ \hat{\mathcal{I}}(\boldsymbol{z}) &= \frac{1}{N}\sum_{i=1}^{N} I(\boldsymbol{z}, \boldsymbol{y_i}),\ \boldsymbol{y_i} \overset{\text{iid}}{\sim} \rho \end{aligned} \tag{2.7}$$

Applying the Central Limit Theorem, we get that the estimation $\mathcal{I}(\boldsymbol{z})$ from Equation 2.7 has a convergence rate of $\mathcal{O}(1/\sqrt{N})$, since

$$\mathrm{Var}\left[\hat{\mathcal{I}}(\boldsymbol{z})\right] = \frac{1}{N}\mathrm{Var}_{\boldsymbol{y}\sim\rho}[I(\boldsymbol{z}, \boldsymbol{y})].$$

2.2.2 Grid method (Trapezoidal integration)

Another method to estimate an integral numerically is by discretizing the integrand on a regular grid. We simply state the well-known result that the convergence rate of the estimation is $\mathcal{O}(N^{-2/p})$. Interestingly, the convergence rate depends on the dimension p of the integral. For $p \geq 4$, Monte-Carlo integration shows better scaling behaviour (but depending on the use case, prefactors of the error can change which method is better).

In our case, since the distribution $\rho(z_1, \cdots, z_p)$ can be factorized into the product of p independent Gaussian distributions, we can decouple the p-dimensional integral into the product of p one-dimensional integrals. Then the error estimation of the integral scales as $\mathcal{O}(N^{-2})$. For this reason, we use the grid method for the remainder of this thesis.

2.3 Characterizing dynamics of the low-rank neural field

We now introduce some tools that will help us understand the toy model. We stress that in general, other neural field models might not have this luxury, and that we only make use of these tools to help explain the dynamics of the simulated neural fields. The overlaps and projections introduced here are only computed from the results of the simulation (see `overlap.py` in the source code).

2.3.1 Overlap variables

Overlap variables in continuous time were introduced by [29] to study spiking networks models of associative memory (Hopfield-type networks), which can be seen as a low-rank network with discrete spatial structure instead of our continuous $\boldsymbol{z}$. The overlap variables measure the correlation between the state of the network (the value of the potential $h(\boldsymbol{z}, t)$ at each point in space) and the pattern fixed point.

For the network for neurons, we write the overlap with the pattern μ as

$$\begin{aligned} m_\mu(t) &= \frac{1}{N} \sum_{i=1}^{N} \tilde{\phi}(z_{\mu i}) \phi(h_i(t)) \\ &\xrightarrow{N\to\infty} \int_{\mathbb{R}^p} \tilde{\phi}(z_\mu) \phi(h(\boldsymbol{z}, t)) \mathcal{N}^p(\mathrm{d}\boldsymbol{z}). \end{aligned} \tag{2.8}$$

We note that similarly to [29], we can write the dynamics of the neural field in terms of the overlap variables:

$$\partial_t h(\boldsymbol{z}, t) = -h(\boldsymbol{z}, t) + \sum_{\mu=1}^{p} z_\mu m_\mu(t). \tag{2.9}$$

This formulation "hides" the integral inside the overlap variables, but one should not forget that the overlaps depend on the neural field at any given time.

2.3.2 Low-rank neural fields as p-dimensional closed systems

Equation 2.9 hints towards the possibility of writing the dynamics of the low-rank system as a closed system of p variables. As explained in Section 2.1.1, the recurrent currents span a p-dimensional subspace, and the orthogonal component behaves independently. This motivates the formulation of a p-dimensional closed system.

We refer to [30], section 4.3 for the reduction of general low-rank networks to p-dimensional closed systems, and give the equations applied to our toy model. Defining the projections $\kappa_\mu(t)$ of the neural field onto the patterns

$$\kappa_\mu(t) = \int_{\mathbb{R}^p} y_\mu h(\boldsymbol{y}, t) \mathcal{N}^p(\mathrm{d}\boldsymbol{y}), \tag{2.10}$$

we can decompose the neural field onto the basis of patterns (mathematically, the patterns form an orthonormal basis of functions):

$$h(\boldsymbol{z}, t) = h^\perp(\boldsymbol{z}, t) + \sum_{\mu=1}^{p} \kappa_\mu(t) z_\mu.$$

The orthonormal component $h^\perp(\boldsymbol{z}, t)$ is independent of the rest of the system and decays exponentially. Then, the equations of evolution for the projections are given by the following closed system:

$$\begin{aligned}
\dot{\kappa}_\mu(t) &= -\kappa_\mu(t) + \int_{\mathbb{R}^p} \tilde{\phi}(y_\mu) \phi(h(\boldsymbol{y}, t)) \mathcal{N}^p(\mathrm{d}\boldsymbol{y}) = -\kappa_\mu(t) + m_\mu(t) \\
\kappa_\mu(0) &= \int_{\mathbb{R}^p} y_\mu h(\boldsymbol{y}, 0) \mathcal{N}^p(\mathrm{d}\boldsymbol{y}),
\end{aligned}$$

and the orthogonal component evolves according to

$$\begin{aligned}
h^\perp(\boldsymbol{z}, t) &= h^\perp(\boldsymbol{z}, 0) \mathrm{e}^{-t} \\
h^\perp(\boldsymbol{z}, 0) &= h(\boldsymbol{z}, 0) - \sum_{\mu=1}^{p} \kappa_\mu(0) z_\mu.
\end{aligned}$$

The set of $\kappa_\mu(t)$ defines a trajectory in a p-dimensional *latent space*. We note that similarly to Equation 2.9, the overlaps intervene in the equations of evolution, and carry the information on the neural field.

Additionally, it can be seen from the expression of the overlaps in Equation 2.8, that in the case of a linear activation function $\phi(h) = c_0 + c_1 h$, we have the equality $m_\mu(t) = \kappa_\mu(t)$.

2.4 A cycling neural field

In Section 2.1 we introduced a simple toy model of a neural field with a natural embedding on the p-dimensional Gaussian. Analytical derivations accompanied by numerical simulations showed the dynamics of this model can be summarized by the convergence to pattern fixed points. In this section, we modify the toy model minimally, such that we can observe more interesting behaviour in the form of cycling.

We modify Equation 2.6 in two ways:

1. We define a time delay δ with which a neuron will "wait" before reacting to a change in its potential.
2. We define "rolling" as the response of a neuron to shift towards the "next" pattern given its state matching a current pattern. This is done by adding a shift $\mu + 1$ in the connectivity kernel, with the convention $p + 1 = 1$ (see [31], Chapter 5).

The resulting cycling neural field is written as such:

$$\partial_t h(\boldsymbol{z}, t) = -h(\boldsymbol{z}, t) + \sum_{\mu=1}^{p} \int_{\mathbb{R}^p} z_{\mu+1} \tilde{\phi}(y_\mu) \phi(h(\boldsymbol{y}, t - \delta)) \mathcal{N}^p(\mathrm{d}\boldsymbol{y}) \tag{2.11}$$

The geometric intuition behind this formulation is that now the recurrent currents are "rotated by half a turn" (around the axis normal to the plane $(z_\mu, z_{\mu+1})$) in the embedding. When the network is at a pattern fixed point μ, the recurrent drive will then push it towards the pattern $\mu+1$. Figure 2.4 shows the difference in the behaviour of the original and the cycling neural field.

We note however that Equation 2.11 no longer is a partial differential equation, but rather a delayed differential equation, which are in general much more complicated to solve. A simplifying assumption that we make is we extend the initial condition back in time, such that $h(\boldsymbol{z}, t < 0) = h(\boldsymbol{z}, 0)$ is a constant function. Numerically, this also means that we have to store a history of the neural fields instead of only the current one. Since the `solve_ivp` method dynamically adapts the timestep of integration, we use linear interpolation between two known points in the stored neural field states to estimate $h(\boldsymbol{z}, t - \delta)$ (see `lagging.py` and its use).

With these considerations, we simulate a $p = 2$ cycling neural field with $\delta = 6$ (since the membrane time constant has been fixed to one, this means that $\delta = 6\tau = 6$) and initial condition $h(z_1, z_2, t < 0) = z_1$ in Figure 2.5. The latent trajectory is estimated from the simulation results, and animated simultaneously with the neural field. We clearly see the oscillations between the patterns $\mu = 1$ and $\mu = 2$.

Relating to Section 2.3.2, the dynamics of the projection now read:

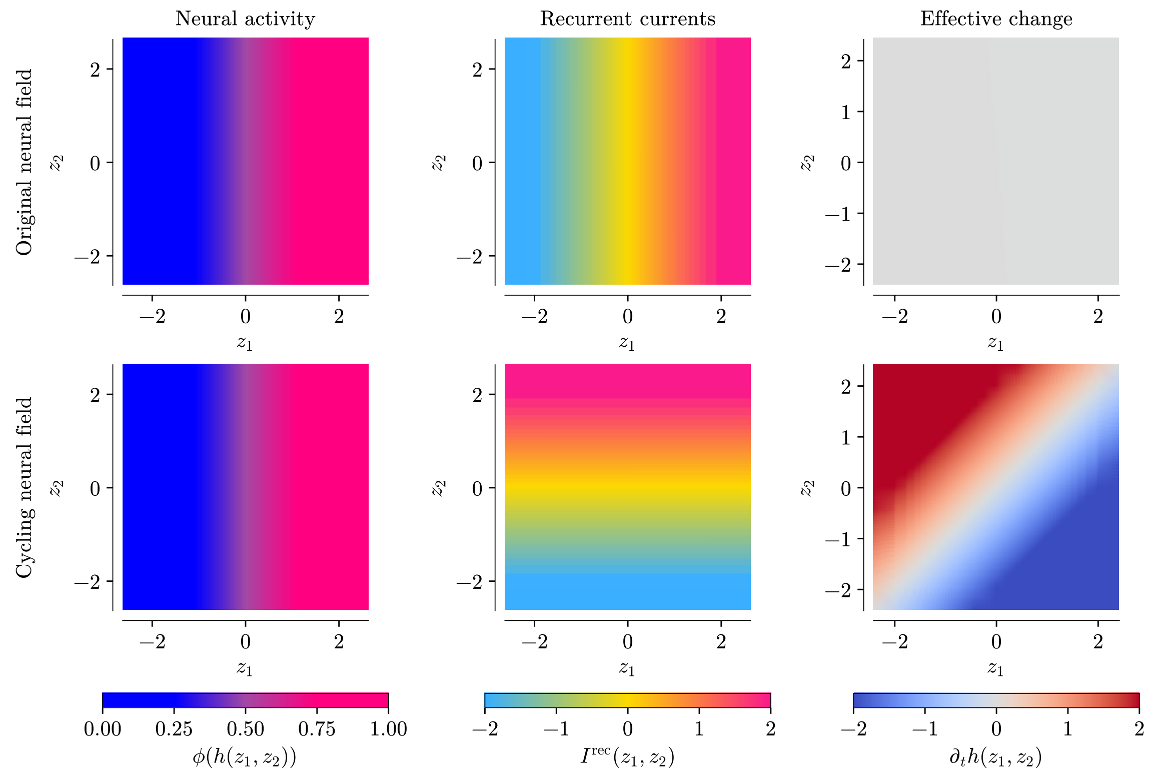
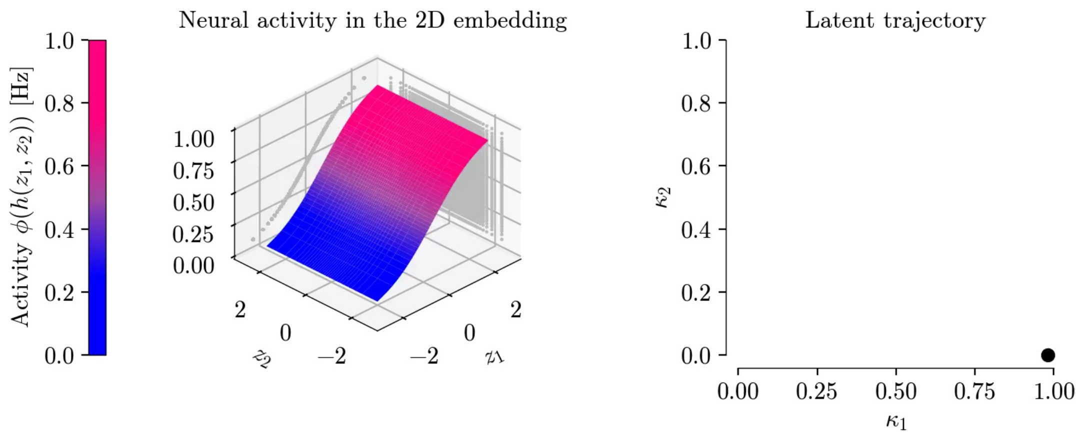
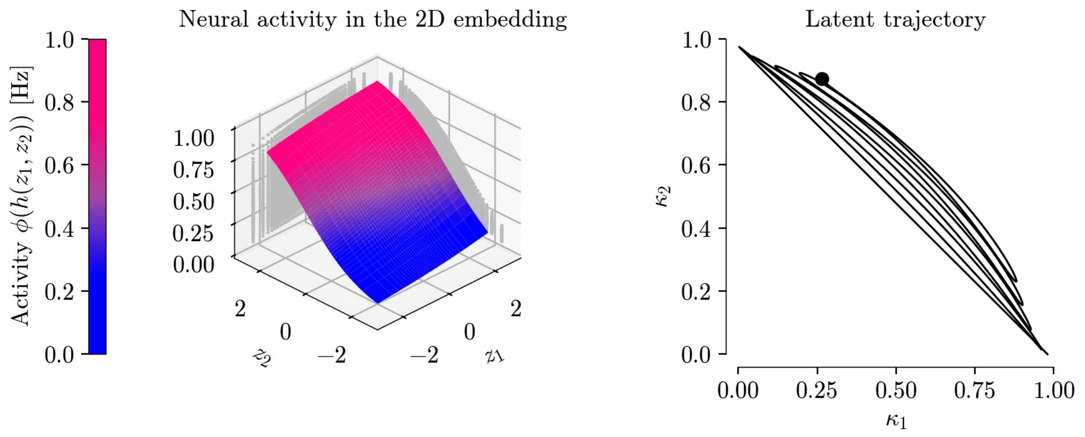


Figure 2.4: Intuition for the cycling behavior. We consider the network in the state $h(z_1, z_2) = z_1$, and look at the recurrent currents. The recurrent currents align with the activity for the original neural field, and the effective change in potential is zero everywhere. For the cycling neural field, the recurrent currents are orthogonal to the activity, such that with the decay term, the effective change leads to a rotation in the counter-clockwise direction, towards the state $h(z_1, z_2) = z_2$.



(a) First frame



(b) Last frame

Figure 2.5: A cycling neural field with $\delta = 6$ and initial condition $h(z_1, z_2, t < 0) = z_1$. The latent trajectory is also computed, and demonstrates the oscillatory behaviour. The first and the last frame of the animation are shown, available here.

$$\dot{\kappa}_\mu(t) = -\kappa_\mu(t) + m_{\mu+1}(t-\delta).$$

2.5 Mapping $\mathbb{R}^2$ to $[0,1]^2$ using the CDF

In preparation to the next chapter, we first map the 2-dimensional neural field equation in $\mathbb{R}^2$ to $[0,1]^2$, and, in general $\mathbb{R}^p$ to $[0,1]^p$. There are many ways to do this by using functions $\mathbb{R} \mapsto [0,1]$, but a practical choice is the Gaussian CDF (Cumulative Density Function), because it has the benefit of absorbing the density into the kernel, in the sense that the integral $[0,1]^p$ becomes weighted by the uniform distribution. The one-dimensional Gaussian CDF is defined as

$$\mathrm{CDF}(z) = \int_{-\infty}^{z} \mathcal{N}(\mathrm{d}y)$$

and maps $\mathbb{R}$ to $[0,1]$,[3] as pictured in Figure 2.6.

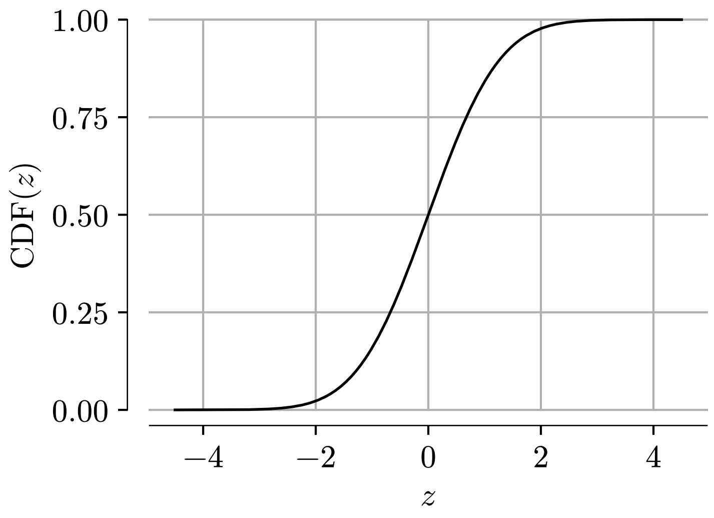


Figure 2.6: CDF of the Gaussian distribution

Defining the change of variables $v_\mu = \mathrm{CDF}(z_\mu), u_\mu = \mathrm{CDF}(y_\mu)$, we can define the neural field h_U and the connectivity kernel w_U for the uniform distribution as

[3]Strictly speaking the image of $\mathbb{R}$ is in $(0,1)$, but for simplicity we consider $\mathbb{R} \cup \{-\infty, +\infty\}$ which maps to $[0,1]$. This is not fundamentally important, because numerically the infinities are never touched, and analytically the recurrent currents given by the integration vanish at infinity due to the Gaussian weighting.

$$
\begin{aligned}
h_U(\boldsymbol{v}, t) &= h(\mathrm{CDF}^{-1}(\boldsymbol{v}), t) = h(\mathrm{CDF}^{-1}(v_1), \cdots, \mathrm{CDF}^{-1}(v_p)) \\
w_U(\boldsymbol{v}, \boldsymbol{u}) &= w(\mathrm{CDF}^{-1}(\boldsymbol{v}), \mathrm{CDF}^{-1}(\boldsymbol{u})).
\end{aligned}
\tag{2.12}
$$

After the change of variables, the neural field in $[0,1]^p$ is equivalent to the neural field in $\mathbb{R}^p$, and its equation reads

$$
\partial_t h_U(t, \boldsymbol{v}) = -h_U(t, \boldsymbol{v}) + \int_{[0,1]^p} w_U(\boldsymbol{v}, \boldsymbol{u}) \phi(h_U(t, \boldsymbol{u})) \mathrm{d}\boldsymbol{u}.
$$

Defining the neural field in $[0,1]^p$ also addresses the problem that we tried to hide in Section 2.2.2 relating to the difficulties of defining a grid on $\mathbb{R}^p$. With the integral on $[0,1]^p$, we can simply use a uniform grid, which maps back to samples in $\mathbb{R}^p$ by using the inverse CDF, as shown in Figure 2.7.

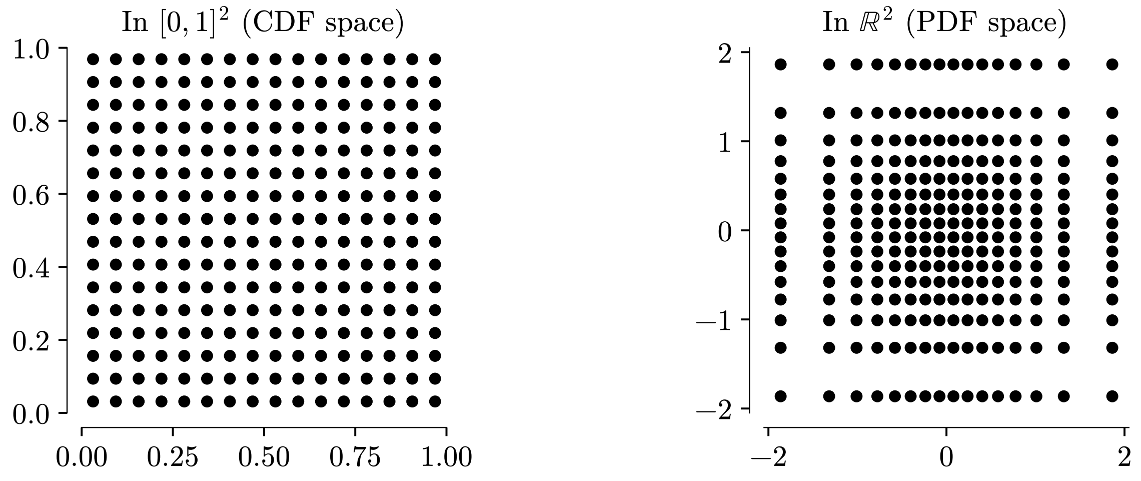


Figure 2.7: A grid in $[0,1]^2$ and the corresponding samples in $\mathbb{R}^2$. The numeric density of points in $\mathbb{R}^2$ approaches the normal distribution as the grid becomes finer (here, the mesh size is 2^{-4} along each dimension).

We make the final remark that this method of mapping $\mathbb{R}^p$ to $[0,1]^p$ by using the CDF does not work for any neural field. In our case, we use the fact that the distribution factorizes into $\rho(z_1, \cdots, z_p) = \rho_1(z_1) \cdots \rho_p(z_p)$, such that the components of $\boldsymbol{v}$ are independent. We can then define the "componentwise CDF" that relates to the total CDF via

$$\begin{aligned}
\mathrm{CDF}(z_1,\cdots,z_p) &= \int_{-\infty}^{z_1}\cdots\int_{-\infty}^{z_p}\rho(\mathrm{d}y_1,\cdots,\mathrm{d}y_p) \\
&= \int_{-\infty}^{z_1}\rho_1(\mathrm{d}y_1)\cdots\int_{-\infty}^{z_p}\rho_p(\mathrm{d}y_p) \\
&= \mathrm{CDF}(z_1)\cdots\mathrm{CDF}(z_p) \\
&= v_1\cdots v_p = \prod_{\mu=1}^{p} v_\mu .
\end{aligned}$$

3 Mapping neural fields to lower-dimensional embeddings

In the previous chapter, we have established a simple toy model for a neural field in p dimensions, and have shown how it can be embedded in $[0,1]^p$. In this chapter, we discuss in-depth what types of mappings can be used to map the p-dimensional space to $[0,1]$, and in particular we study the case $p=2$. Before getting into too much detail, let us give some insight as to why this problem is not trivial.

3.1 Population dynamics: an introductory example of a 1D embedding

3.1.1 Embedding a finite number of populations in $[0,1]$

Let us for a moment consider another type of model for networks of neurons, that of multiple homogeneous population dynamics, describing how N_{pop} different populations interact with each other. These types of models emerge, for example, when networks of neurons exhibit a finite number of "neuron types". We then group neurons of the same type, and make the approximation that they have identical potentials and the same interaction weights with other neurons. As a consequence, every population represents a homogeneous population of identical neurons.

This results in writing population dynamics with the same equation as a finite network of neurons. However, the equation *already represents* the $N \to \infty$ limit. We write the potential of the populations $H_a(t)$ and the population connectivity $\tilde{J}_{ab}$, where $a, b \in \{1, \cdots, N_{\text{pop}}\}^2$ in Equation 3.1.

$$\dot{H}_a(t) = -H_a(t) + \sum_{b=1}^{N_{\text{pop}}} \tilde{J}_{ab}\phi(H_b(t)) \tag{3.1}$$

We don't show the derivation of population dynamics here (see [32], chapter 12.3 for a detailed analysis), but we point out the implicit scaling behavior of $\tilde{J}_{ab}$. Letting i be a neuron of population a and j be a neuron of population b, the connectivity between the two neurons is J_{ij}, and the population connectivity scales with the number of neurons in the presynaptic population $|b|$. Furthermore, we make the reasonable assumption that J_{ij} scales as J^0_{ij}/N, where $J^0_{ij} \sim \mathcal{O}(1)$. We then have

$$\tilde{J}_{ab} = \frac{|b|}{N} J^0_{ij}.$$

As $N \to \infty$, the ratio $\frac{|b|}{N}$ converges to a constant p_b, the probability of the population, normalized such that $\sum_{a=1}^{N_{\text{pop}}} p_a = 1$. It expresses the probability of uniformly sampling a neuron belonging to population b. In effect, p_b can be interpreted as the "weight" of the population.

So far, we have not considered that the populations are embedded in some abstract space; but the notion of population weights gives rise to the obvious embedding in $[0, 1]$ given by the cumulative sums of the population weights, illustrated in Figure 3.1. We define (somewhat informally) the connectivity kernel $w(\alpha, \beta) = J^0_{ab}$ for α, β in the regions corresponding to populations a and b respectively.[1]

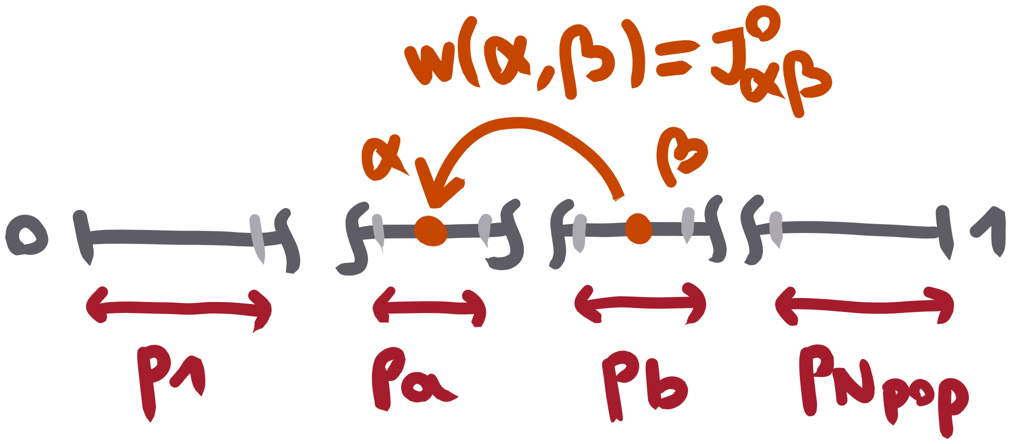


Figure 3.1: Embedding of the population dynamics in $[0, 1]$.

Defining the neural field $H(\alpha, t) = H_a(t)$, we can write a "neural field equation" on the $[0, 1]$ space.

[1]Formally, we would write the cumulative sum $K_a = \sum_{b=1}^{a} p_b$. Then, we have $\alpha \in [K_{a-1}, K_a)$ and $\beta \in [K_{b-1}, K_b)$. We abusively wrote J^0_{ab} since all neurons in each population are identical, therefore we can index into the connectivity J^0_{ij} with the population indices a, b instead of neuron indices i, j.

$$\begin{aligned}
\partial_t H(\alpha,t) &= -H(\alpha,t) + \int_0^1 w(\alpha,\beta)\phi(H(\beta,t))\mathrm{d}\beta \\
&= -H(\alpha,t) + \sum_{b=1}^{N_{\mathrm{pop}}} p_b w(\alpha,b)\phi(H_b(t)) \\
\iff \partial_t H_a(t) &= -H_a(t) + \sum_{b=1}^{N_{\mathrm{pop}}} p_b w(a,b)\phi(H_b(t)) \\
&= -H_a(t) + \sum_{b=1}^{N_{\mathrm{pop}}} p_b J^0_{ab}\phi(H_b(t)) \\
&= -H_a(t) + \sum_{b=1}^{N_{\mathrm{pop}}} \tilde{J}_{ab}\phi(H_b(t))
\end{aligned}$$

We hereby showed (having taken many writing liberties, but this is just to give an idea) how the neural field reduces back to the original population dynamics, which was in itself already the $N \to \infty$ limit of the network. The case of multi-population dynamics therefore shows a trivial example of an embedding in $[0,1]$.

3.1.2 Population dynamics with an infinite number of populations

When taken in isolation, the previous developments might sound a little bit silly, though they serve to give a feeling for the problem at hand, and a motivation for the rest of this work. For a finite number of populations N_{pop}, the population dynamics is well-defined; however, what happens as $N_{\mathrm{pop}} \to \infty$? The core of the question lies in the fact that we can easily *enumerate* a finite number of populations, but the sum $\sum_{b=1}^{N_{\mathrm{pop}}}$ must have a clearly defined interpretation when this number tends to infinity.

In light of the toy model presented in Chapter 2, we can reinterpret the density ρ of the neural field equation. Every point $\boldsymbol{z} \in \mathbb{R}^p$ (let us consider $p = 2$ in the following) describes an infinite population of homogeneous neurons, associated with a probability (density) $\rho(\boldsymbol{z})$. Therefore, in order to write an equivalent neural field in $[0,1]$, we need a way to *enumerate* the populations. This is done using a mapping $S : [0,1]^2 \mapsto [0,1]$, where we cover the $[0,1]^2$ space by following the order assigned to the populations in the image $[0,1]$.

Additionally, the notion of population dynamics helps give another interpretation of the grid integration method given in Section 2.2.2. When we discretize the $[0,1]^2$ space, we are effectively performing a coarse-graining approximation and simulating a finite number of populations. For every square bin localized by (v_1, v_2), we are sampling the connectivity kernel $w_U((v_1,v_2),\cdot)$ and the 2D neural field $h_U((v_1,v_2),t)$ (see Figure 3.2). In the rest of the chapter, we refer to these 2D bins as “square populations”, and to the corresponding 1D bins obtained by the mapping S as “segment populations”.

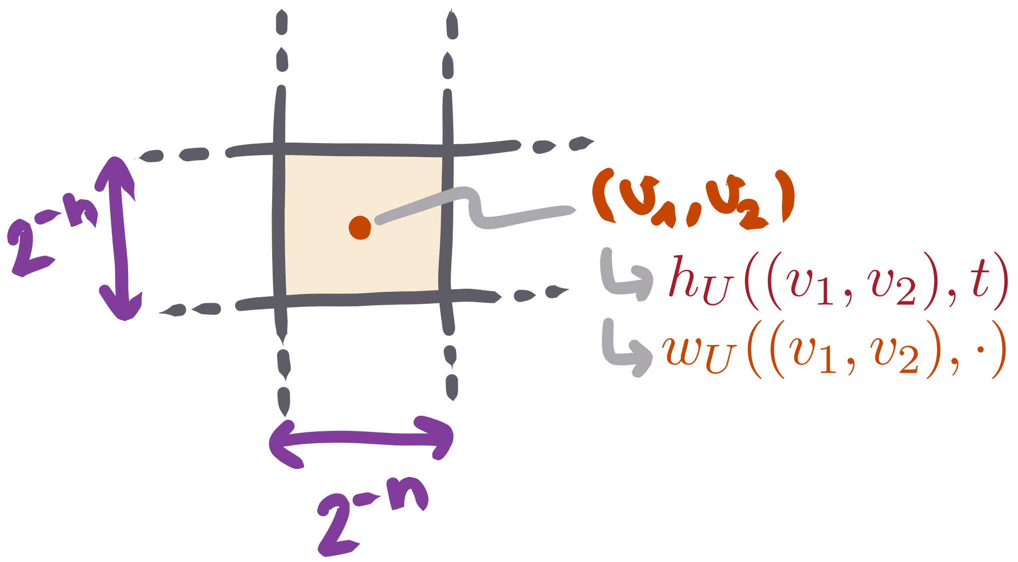


Figure 3.2: Every point of the grid effectively performs coarse-graining, and defines a point where we sample the connectivity kernel w_U and the 2D neural field h_U.

For a finite number of populations N_{pop}, applying a bijective mapping S will evidently give an equivalent neural field, because the connectivity matrix is permutation-invariant, and the resulting connectivity matrix $J_{S(a),S(b)}$ will therefore give equivalent dynamics. As we increase N_{pop} by taking finer and finer grids, the image of the grids defines a sequence of mappings S^n : grid in $[0,1]^2 \mapsto$ grid in $[0,1]$, which we aim to study in the rest of this chapter. The main question is:

> How do we define a sequence of mappings S^n such that the limiting $S : [0,1]^2 \mapsto [0,1]$ can express a 1D neural field with identical dynamics ?

3.2 Defining the neural field in $[0,1]$

We start by writing the neural field equation in the $[0,1]$ space. For $\lambda : [0,1]^2 \to [0,\infty]$ a measurable function (e.g. the Lebesgue measure, which is nothing but the uniform probability distribution), we let the composition $\lambda \circ S^{-1}$ define a measure on $[0,1]$, which keeps track of how 2D surfaces are transformed to (possibly disjoint) 1D segments through the mapping S.

The connectivity kernel $\tilde{w}$ defines the interactions between the populations in the $[0,1]$ embedding, which we define relating to the kernel w_U, defined in Section 2.5 as:

$$\tilde{w}(\alpha, \beta) = w_U(S^{-1}(\alpha), S^{-1}(\beta)) \tag{3.2}$$

Defining the initial condition for the 1D neural field as $\tilde{h}(\alpha, t=0) = h_U(S^{-1}(\alpha), t=0)$, we have fully defined the equation of evolution in the one-dimensional embedding in Equation 3.3.

$$\partial_t \tilde{h}(\alpha, t) = -\tilde{h}(\alpha, t) + \int_{[0,1]} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right](\mathrm{d}\beta) \tag{3.3}$$

The definition of the connectivity kernel in Equation 3.2 might seem innocent, but the key element here is the inverse S^{-1}. We recall that in the toy model, the original connectivity kernel is written as $w(\boldsymbol{z}, \boldsymbol{y}) = \sum_{\mu=1}^{p} z_\mu \tilde{\phi}(y_\mu)$. This kernel is highly regular, by which we mean that, by its continuity in each of its arguments, populations nearby in the $\mathbb{R}^p$ will have similar input weights. Since it is also differentiable, we can write this as a Taylor expansion in the first argument:

$$\begin{aligned} w(\boldsymbol{z} + \boldsymbol{\epsilon}, \boldsymbol{y}) &= w(\boldsymbol{z}, \boldsymbol{y}) + \sum_{\mu=1}^{p} \epsilon_\mu \frac{\partial w}{\partial z_\mu} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2) \\ &= w(\boldsymbol{z}, \boldsymbol{y}) + \sum_{\mu=1}^{p} \epsilon_\mu \phi(y_\mu) + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2) \\ &\approx w(\boldsymbol{z}, \boldsymbol{y}). \end{aligned}$$

When we do the change of variables with the CDF, we don't encounter problems because the Gaussian (inverse) CDF is differentiable and its derivative is continuous;[2] therefore $w_U(\boldsymbol{v}, \boldsymbol{u}) = w(\mathrm{CDF}^{-1}(\boldsymbol{v}), \mathrm{CDF}^{-1}(\boldsymbol{u}))$ is still highly regular.

For these reasons, we can see why the composition of w_U with a possibly non-differentiable mapping S might be problematic. Since the only thing we know about S is that it is measurable and bijective, *there is no guarantee that the regularity (continuity and the differentiability) of the kernel is conserved through composition with the mapping S.*

3.3 Mappings of $[0,1]^2$ to $[0,1]$

3.3.1 The search for differentiable mappings

We start our search of mappings by attempting to find a mapping S that is differentiable. The easiest example of such a mapping would be a linear projection, for instance, $P(v_1, v_2) = v_1$. This mapping is obviously continuous and differentiable in both

[2]Strictly speaking, we have the problem that $\mathrm{CDF}^{-1}(v) \xrightarrow{v \to 1} \infty$, and so the derivative diverges when $v \to 1$ (and similarly, when $v \to 0$). However, this is not a problem, since these limits correspond to $z \to \pm\infty$, where the density vanishes.

components, and surjectively maps the unit square to the unit segment. However, the obvious problem with this approach is that linear projections are not invertible. Hence, should we nevertheless define an inverse P^{-1} that computes the preimage of $\alpha \in [0,1]$, we have that:

$$P^{-1}(\alpha) = \{(\alpha, v_2) \mid v_2 \in [0,1]\}.$$

In other words, $P^{-1}(\alpha)$ defines a vertical line in the $[0,1]^2$ embedding. When we take the composition $w_U \circ P^{-1}$, the "locality" is destroyed, in the sense that the same population α in the 1D embedding will "sense" all the connectivity kernels on the vertical line.

A more sophisticated attempt, in the spirit of Section 2.5, might be to consider the joint cumulative function

$$\alpha = \mathrm{CDF}(z_1, z_2) = \int_{-\infty}^{z_1} \int_{-\infty}^{z_2} \rho(y_1, y_2) \mathrm{d}y_1 \mathrm{d}y_2.$$

However, this again fails because the joint cumulative function is not bijective, and isolines of the CDF define continuous curves in the uniform embedding space $[0,1]^2$.[3]

Perhaps these examples are a bit too naive, and we need to look for more complex mappings. We might ask the question of whether a diffeomorphism (a differentiable bijection with a differentiable inverse) even exists between the unit square and the unit segment. Unfortunately, results from topology give a negative answer, and even worse, it can be shown that there is no homeomorphism (a continuous bijection with continuous inverse) between the unit square and the unit segment. This is stated by Netto's theorem, which imposes continuous bijections to conserve the dimensionality between the domain and the image.

Figure 3.3 shows a visual representation of Netto's theorem. The 3-way intersection of the Venn diagram represents that functions which are both bijective (surjective and injective) and continuous do not exist. For the three remaining 2-way intersections, we annotate, without going into detail, the types of functions which they represent (see [33] for Cantor and Peano functions, [34] for Peano and Jordan functions).

[3]We note to avoid any confusion, that in Section 2.5 the CDF is taken component-wise, instead of jointly. The univariate Gaussian CDF defines a bijection between $\mathbb{R}$ and $(0,1)$, but the joint CDF does not.

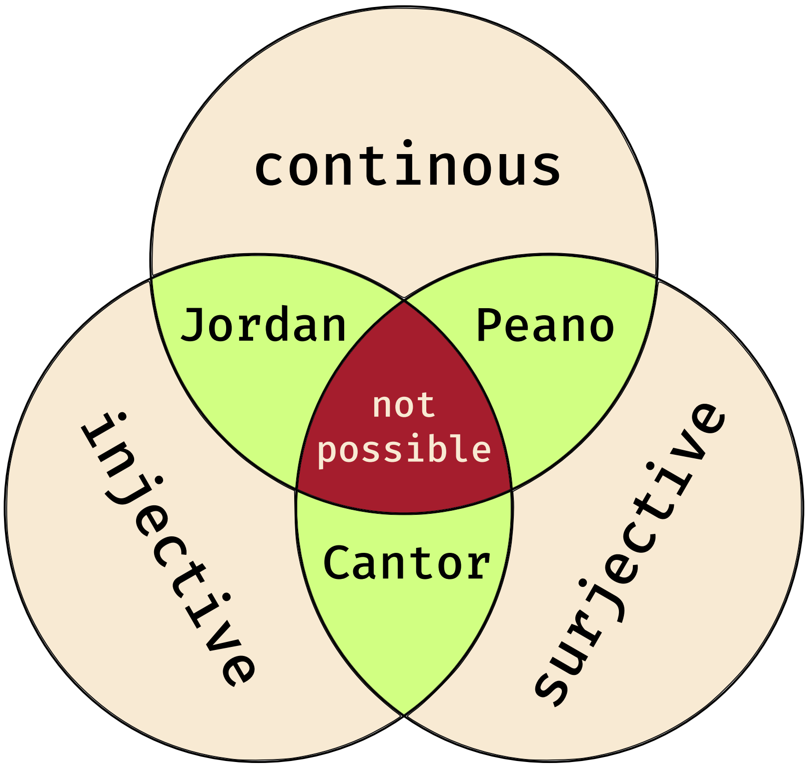


Figure 3.3: Venn diagram reprensenting the properties of functions $[0, 1] \mapsto [0, 1]^2$ which are allowed by Netto's theorem.

3.3.2 Sequences of bijective mappings

3.3.2.1 Mappings as binary expansions

Previously, we motivated S as the limit of functions S^n acting on finer and finer grids. Let us consider $(v_1, v_2) \in [0,1]^2$. Given $n \in \mathbb{N}$, we write a (truncated) binary expansion of v_1 and v_2.

$$
\begin{aligned}
v_1^{(n)} &= \sum_{l=1}^{n} b_l^1 2^{-l} = 0.b_1^1 b_2^1 \cdots b_n^1 \iff b_l^1 = \mathrm{Ind}\left\{2^{l-1}v_1 - \lfloor 2^{l-1}v_1 \rfloor \geq \frac{1}{2}\right\} \\
v_2^{(n)} &= \sum_{l=1}^{n} b_l^2 2^{-l} = 0.b_1^2 b_2^2 \cdots b_n^2 \iff b_l^2 = \mathrm{Ind}\left\{2^{l-1}v_2 - \lfloor 2^{l-1}v_2 \rfloor \geq \frac{1}{2}\right\}
\end{aligned}
\tag{3.4}
$$

Figure 3.4 illustrates this expansion: we locate each component bit by recursively splitting the segment in two sub-segments of equal length. At every step, taking the left or right segments defines the sequence of binary digits.

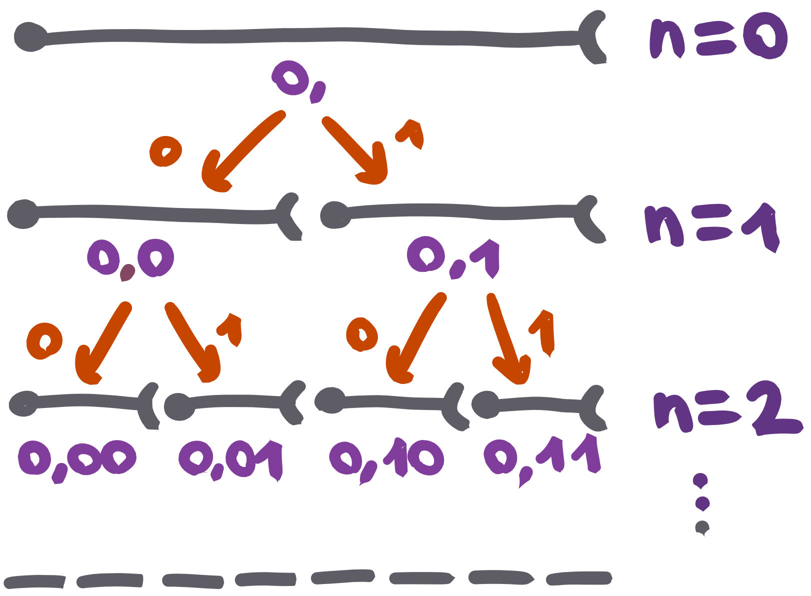


Figure 3.4: Splitting $[0,1]$ recursively with the indicator function gives the binary bit expansion of v_1

We define the mapping S^n that takes the n-bit truncations $v_1^{(n)}$ and $v_2^{(n)}$ as input, and outputs a $2n$ bit truncation $\alpha^{(n)}$ of $\alpha = S(v_1, v_2)$, where S is the $n \to \infty$ pointwise limit of the sequence S^n (when the sequence S^n converges).

$$
\begin{aligned}
S^n : &\{0, \tfrac{1}{2^n}, \cdots, 1 - \tfrac{1}{2^n}\}^2 \mapsto \{0, \tfrac{1}{4^n}, \cdots, 1 - \tfrac{1}{4^n}\} \\
&(v_1^{(n)}, v_2^{(n)}) = (0.b_1^1 b_2^1 \cdots b_n^1, 0.b_1^2 b_2^2 \cdots b_n^2) \mapsto \alpha^{(n)} = 0.b_1 b_2 \cdots b_{2n}
\end{aligned}
\tag{3.5}
$$

In general, each bit b_k of the image would be a binary function of the n bits of v_1 and the n bits of v_2, i.e. $b_k = b_k(b_1^1, b_2^1, \cdots, b_n^1, b_1^2, b_2^2, \cdots, b_n^2)$. With this, we can formulate any mapping between sets of 4^n distinct elements.

The intuition behind the use of $2n$ bits for the image is that we take $2^n \times 2^n = 4^n$ squares of size $2^{-n} \times 2^{-n}$ in $[0,1]^2$, and map those to 4^n segments of size 4^{-n} in $[0,1]$. Informally, we map $n+n$ bits onto $2n$ bits.

In the following, we will restrict ourselves to mappings where the functions b_k are defined in such a way that each bit of the inputs v_1 and v_2 is used only once. Therefore, for each bit k of the output, there is a function $b_k(b_{l_k}^{\mu_k})$ acting on only one bit of the input, where $\mu_k \in \{1,2\}$ is the input component and $l_k \in \{1, \cdots, n\}$ the bit number corresponding to k. Since this function is binary, the only allowed definitions are

- $b_k(b_{l_k}^{\mu_k}) = b_{l_k}^{\mu_k}$ (identity),
- $b_k(b_{l_k}^{\mu_k}) = 1 - b_{l_k}^{\mu_k}$ (inversion),
- $b_k(b_{l_k}^{\mu_k}) = 0$ (constant zero),
- $b_k(b_{l_k}^{\mu_k}) = 1$ (constant one).

The constant functions are ignored, since they effectively just restrict the output domain to a subset of $[0,1]$; and since only $2n$ bits are allowed for the output, they are a “waste” by ignoring information of the input. Because they don’t add information to the output (they just “swap” the values of the output bits at known locations), inversion functions are also not considered.

All this mathematical formalism expresses, in essence, that the mapping S^n is just a *reordering* of the input bits. S^n takes 4^n points embedded in $[0,1]^2$, and reorders them onto 4^n points in $[0,1]$.[4]

Therefore, there is a one-to-one mapping between the points on the square and the points on the segment, and the mapping S^n is bijective. In the following, we will, for ease of notation and when it is clear in the context, drop the index n when writing S^n. This is often justified (particularly with the Z-mapping and Column mapping introduced later), because the finite-n truncations $(v_1^{(n)}, v_2^{(n)})$ result in finite truncations of S, therefore $S(v_1^{(n)}, v_2^{(n)}) = S^n(v_1^{(n)}, v_2^{(n)})$.

We will show in the rest of this thesis that this simple formulation is flexible enough to express mappings that have good properties such as “locality”, while being simple enough to help build intuitions.

[4]We note that in our definition, S^n maps $(0., 0.)$ onto 0. Numerically, we encounter the problem that the inverse of the Gaussian CDF, used to compute the connectivity, goes to infinity when v_μ (one of the components of $(v_1, v_2) = S^{-1}(\alpha)$) goes to 0 (or 1). To avoid this, we offset the squares (by 2^{-n-1} along each dimension), so that geometrically their position is not indexed by their lower-left corner, but rather their center (this is represented in Figure 3.2). Analytically, this is not a problem, because the kernel defined in Equation 2.12 diverges only linearly, compared to the exponential cutoff of the Gaussian distribution, so that the integrand still vanishes at infinity.

3.3.2.2 A geometric view relating to curves in $[0,1]^2$

With this formalism in mind, we give a geometric interpretation to the mappings S^n. The core idea is that the populations in $[0,1]$ have a clear ordering to them, and we can enumerate them simply by their position in the embedding. Since we constructed S^n to be bijective, this enumeration traces a path in the corresponding populations of the $[0,1]^2$ embedding.

We illustrate this by introducing the "column" mapping.

$$\alpha^{(n)} = C(v_1^{(n)}, v_2^{(n)}) = 0.b_1^1 b_2^1 \cdots b_n^1 b_1^2 b_2^2 \cdots b_n^2 \tag{3.6}$$

The intuition behind this mapping is that we enumerate the populations on a grid column by column, letting the index of the rows vary the fastest. We visualize this mapping in Figure 3.5, by coloring the square populations by the position of the corresponding $[0,1]$ population, and additionally drawing lines between the points to help guide the eye.

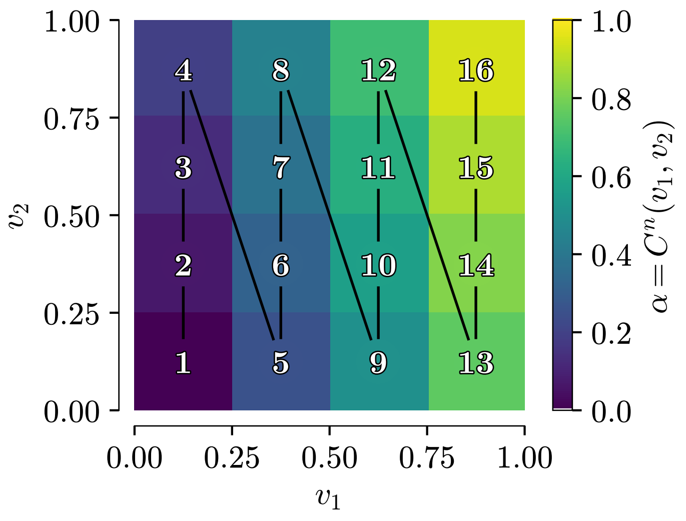


Figure 3.5: Column mapping for $n = 2$. Numbers represent the ordering of the populations in the $[0,1]$ embedding, neighbouring populations are connected by lines.

For finite n, we write the inverse of the column mapping:

$$\begin{aligned} v_1^{(n)} &= C_1^{-1}(\alpha^{(n)}) = 0.b_1 b_2 \cdots b_n = \sum_{k=1}^{n} b_k 2^{-k} \\ v_2^{(n)} &= C_2^{-1}(\alpha^{(n)}) = 0.b_{n+1} b_{n+2} \cdots b_{2n} = \sum_{k=1}^{n} b_{n+k} 2^{-k}. \end{aligned} \tag{3.7}$$

This way of enumerating the square populations might be familiar to people who have worked with numerical arrays. In such (contiguous) arrays, the values are laid out in memory in "row-major" order (C-style) or "column-major" order (Fortran-style); and we can go from the 1D memory representation to a 2D matrix representation by using "reshape" operations. Effectively, this is how this mapping is implemented in code, using `numpy.ravel_multi_index` (2D to 1D) and `numpy.unravel_index` (1D to 2D).

3.3.2.3 "Local" curves

Research in computer science and efficient data structures has given rise to other ways of storing 2-dimensional information in a 1-dimensional memory. A problem in GPU computing is that of texture locality: sampling textures often involves reading data corresponding to a small 2D region, and therefore to improve speed (minimize cache misses), the 2D data should be packed close together in the 1D memory. One way of ordering the 2D texture data in the 1D memory is by using the Z-order curve (also known as "Morton mapping", and similar to the "Lebesgue curve", see [34], Chapter 6). In this way, points close in 1D *tend* to be tightly packed in 2D. We define the Z-mapping in the following way:[5]

$$\alpha^{(n)} = Z(v_1^{(n)}, v_2^{(n)}) = 0.b_1^1 b_1^2 b_2^1 b_2^2 \cdots b_n^1 b_n^2 = \sum_{k=1}^{n} b_k^1 2^{1-2k} + b_k^2 2^{-2k} \tag{3.8}$$

And its inverse is:

$$\begin{aligned} v_1^{(n)} &= Z_1^{-1}(\alpha^{(n)}) = 0.b_1 b_3 \cdots b_{2n-1} = \sum_{k=1}^{n} b_{2k-1} 2^{-k} \\ v_2^{(n)} &= Z_2^{-1}(\alpha^{(n)}) = 0.b_2 b_4 \cdots b_{2n} = \sum_{k=1}^{n} b_{2k} 2^{-k}. \end{aligned} \tag{3.9}$$

Figure 3.6 is a visualization of the Z-mapping when $n = 2$, and shows how it is composed of the "Z" shapes giving it its name.

To illustrate this notion of locality, we take a few consecutive populations in 1D and show where the corresponding populations fall in the 2D embedding. This is done in Figure 3.7, in which we see that the Z-mapping seems to conserve locality from 1D to 2D: populations close in 1D seem to be close in 2D.

[5]The Z-order curve is often defined in a transpose manner, putting first the bits of v_2 (vertical component) then the bits of v_1 (horizontal component), instead of v_1 then v_2. By doing this, we get the "upright" Z shapes, as shown here.

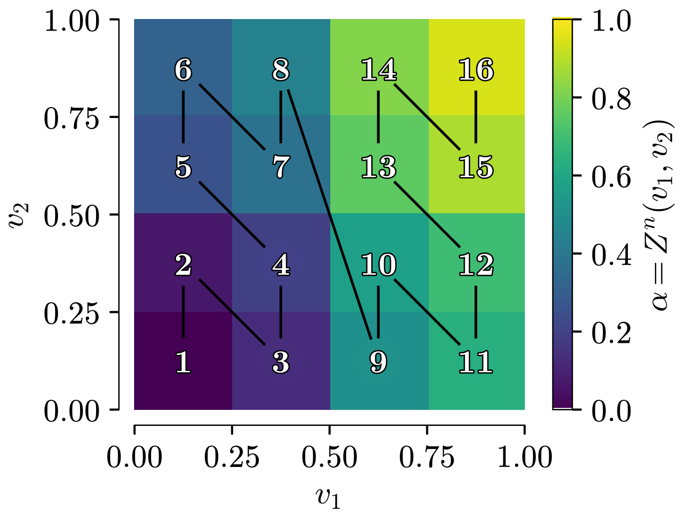


Figure 3.6: Z-mapping for $n = 2$. Numbers represent the ordering of the populations in the $[0, 1]$ embedding, neighbouring populations are connected by lines.

We also see that the Column mapping is not local in this sense, since there is a large variation along the vertical direction, which seems to always be "of order one". This can be explained by looking at the expression for $C_2^{-1}(\alpha^{(n)}) = 0.b_{n+1}b_{n+2}\cdots b_{2n}$: small variations of the order of 2^{-n} in the value of α always result in variations of order one in C_2^{-1}.

Note that, in some sense, the Column mapping is local, from 2D to 1D. This can be seen from the definition of $\alpha^{(n)} = C(v_1^{(n)}, v_2^{(n)}) = 0.b_1^1 b_2^1 \cdots b_n^1 b_1^2 b_2^2 \cdots b_n^2$: small variations of order 2^{-n} in $v_1^{(n)}$ result in small variations also of order 2^{-n} in $\alpha^{(n)}$. Small variations of order 2^{-n} in $v_1^{(n)}$ result in even smaller variations also of order 2^{-2n} in $\alpha^{(n)}$! We therefore have two notions of locality: from 1D to 2D, and from 2D to 1D; and the Column mapping shows that they are not equivalent.

We argue that in our case only the notion of locality from 1D to 2D matters. In essence, the dynamics in the 2D embedding are already known, and we ask the question of whether it is possible to write an equivalent neural field in 1D. Without repeating the intuition given in Section 1.4, if neighbouring populations in 1D have similar potentials, then this allows us to write a neural field equation in $[0, 1]$. Therefore, we would like to have the property that populations close in 1D are (inversely) mapped to populations close in 2D, which already have similar potentials.

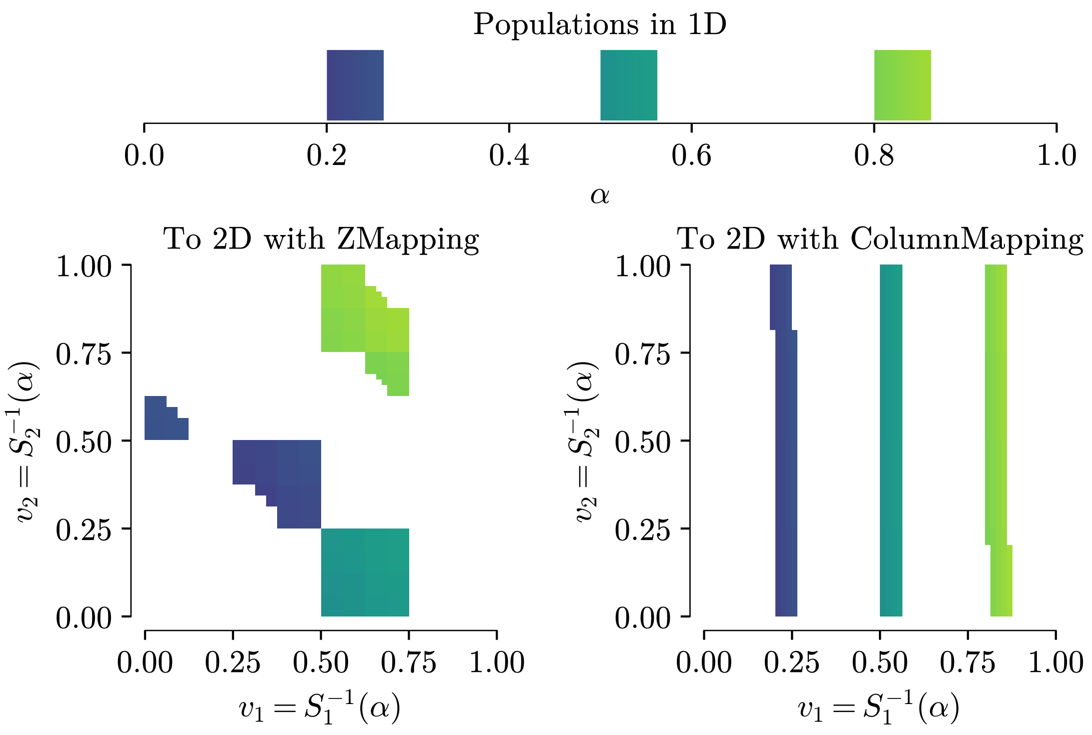


Figure 3.7: Locality of the mappings. The 2D populations corresponding to three small segments (each of length $\frac{1}{16}$) of 1D populations are shown. The figures are generated with $n = 6$.

3.3.3 Sequences of mappings and their limit

3.3.3.1 Pointwise convergence of mappings

We now discuss what happens to the bijective mappings when $n \to \infty$. For this, let us consider again the "random mapping" considered in the introduction Section 1.4. We discussed how this hypothetical mapping maps each position in $[0,1]^2$ into a "random position" in $[0,1]$. A finite-n formulation would be that the mapping corresponds to a *random permutation* of the bits, therefore, its limit is obviously not well-defined. For a given position of the input (v_1, v_2), we would like that the image is stable in the $n \to \infty$ limit.

$$S^n(v_1, v_2) \xrightarrow{n\to\infty} \alpha =: S(v_1, v_2)$$

This is the condition of "pointwise convergence" of the mappings. Let us introduce another mapping, which will help to give a better intuition of what pointwise convergence means in our context. We define, for finite n, the "anti-Z" mapping and its inverse:

$$\alpha^{(n)} = A(v_1^{(n)}, v_2^{(n)}) = 0.b_n^1 b_n^2 b_{n-1}^1 b_{n-1}^2 \cdots b_1^1 b_1^2 = \sum_{k=1}^{n} b_{n+1-k}^1 2^{1-2k} + b_{n+1-k}^2 2^{-2k}$$

$$\iff$$

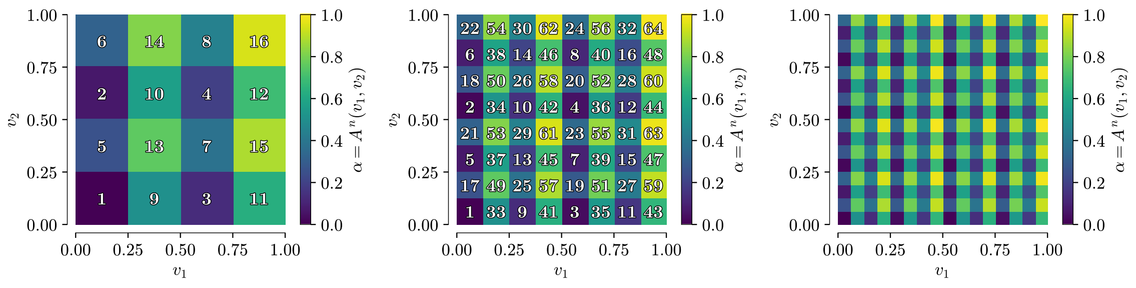
$$v_1^{(n)} = A_1^{-1}(\alpha^{(n)}) = 0.b_{2n-1} b_{2n-3} \cdots b_1 = \sum_{k=1}^{n} b_{2(n-k)-1} 2^{-k}$$

$$v_2^{(n)} = A_2^{-1}(\alpha^{(n)}) = 0.b_{2n} b_{2n-2} \cdots b_2 = \sum_{k=1}^{n} b_{2(n+1-k)} 2^{-k}.$$

Figure 3.8 gives some insight into the problem of this mapping. For a fixed input (v_1, v_2), we see that the output point position jumps around when n changes, and that these jumps seem to always be of order 1. The intuition behind pointwise convergence is that as n increases, the image of a point by S^n is continually refined.

Consider how $\alpha^{(n)}$ fluctuates as the precision of the input increases with $n \to \infty$. As we refine the input with fluctuations of order 2^{-n} (we change only the bits that come after b_n^1 and b_n^2), the output $\alpha^{(n)} = b_n^1 4^{-1} + b_n^2 4^{-2} + b_{n-1}^1 4^{-3} + b_{n-1}^2 4^{-4} \cdots$ fluctuates (at least) with amplitude $\sim 4^{-2}$, independantly of n. Therefore the output never converges, and the anti-Z mapping is not pointwise convergent.

In our setting, how can we guarantee that the mappings are pointwise convergent? We need to show that small fluctuations in the input result in small fluctuations in the output. The binary expansion allows us to argue that for the sequence of mappings



(a) $n = 2$ (b) $n = 3$ (c) $n = 4$

Figure 3.8: Anti-Z mapping for $n = 2, 3, 4$. Numbers represent the ordering of the populations in the $[0, 1]$ embedding. We don't plot the trace, because it has many crossovers and overlaps which makes the plot hard to read.

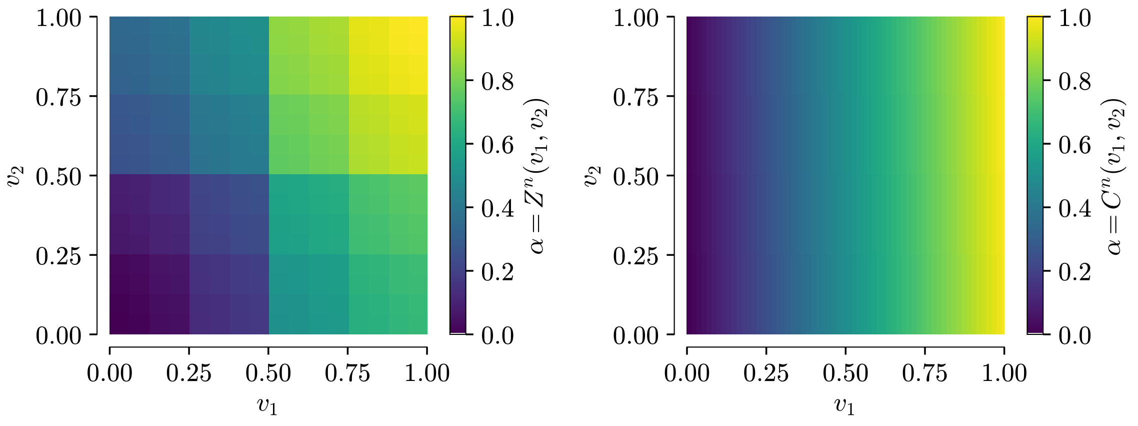
to be pointwise convergent, we need that the "least significant bits" (LSB, the bits to the right of the binary expansion) of the input are mapped to the LSB of the output. Conversely, the "most significant bits" (MSB, the bits to the left of the binary expansion) should be mapped to the MSB of the output.

We illustrate this with the Z-mapping, written in Equation 3.8. Corrections of order 2^{-n} in v_1 and v_2 induce corrections of order 2^{1-2n} and 2^{-2n} in α respectively. Therefore, it is easy to see that the Z-mapping is pointwise convergent as $n \to \infty$. Similarly, the Column mapping in Equation 3.6 is also pointwise convergent, since as discussed Section 3.3.2.3 it has a locality of 2D to 1D.

3.3.3.2 Bijectivity in the $n \to \infty$ limit

However, the behavior of the Column mapping and that of the Z-mapping are very different in the $n \to \infty$ limit. Up until now, we have avoided this issue by considering the finite-n approximation of the mappings, which by construction is numerically bijective (that is, bijective on the discretization of $[0, 1]^2$ and $[0, 1]$ corresponding to finite n). We illustrate the infinite-n (numerically, large n) limit of these two mappings in Figure 3.9.

We see that the Column mapping converges to a projection on v_1, in other words, $\alpha = C(v_1, v_2) = v_1$, showing that despite every C^n being numerically bijective, the limit of the sequence is not bijective. This becomes clear when looking at the binary expansion of C:



(a) Z-mapping (b) Column mapping

Figure 3.9: The $n \to \infty$ limit of mappings.

$$
\begin{aligned}
C(v_1, v_2) &= \lim_{n\to\infty} C^{(n)}(v_1, v_2) \\
&= \lim_{n\to\infty} 0.b_1^1 b_2^1 \cdots b_n^1 b_1^2 b_2^2 \cdots b_n^2 \\
&= \lim_{n\to\infty} 0.b_1^1 b_2^1 \cdots \\
&= v_1
\end{aligned}
$$

The limit is not bijective, because the bits of v_2 are "lost" in the $n \to \infty$ limit, and the inverse $C_2^{-1}(\alpha)$ is not well-defined, since it "takes bits between ∞ and 2∞".

The Z-mapping seems to converge to a fractal structure. Numerically, there seems to be an infinite number of discontinuities, but the sizes of these discontinuities are, for the most part, small, so that *on average* they vanish as $n \to \infty$. We won't go further into detail about this, but this gives some intuition for the arguments presented in Chapter 4.

We finish this discussion on the limit of the sequence of mappings by asking if there are other mappings similar to the Z-mapping, in the sense that they are pointwise convergent and the limit is bijective. In an informal way, we established two conditions for this in the context of the mappings described in Section 3.3.2.1:

1. condition of pointwise convergence: the LSB of the input must be mapped to the LSB of the output,
2. condition of "no information loss": the MSB of the input must be mapped to the MSB of the output.

These two conditions do not leave a lot of flexibility in the design of new mappings. We can evidently define mappings where we interleave the bits in a different order, such as $0.b_1^2 b_1^1 b_2^2 b_2^1 \cdots$; or mappings where we reorder the k bits of the input, as in $0.b_2^1 b_2^2 b_1^1 b_1^2 b_4^1 b_4^2 b_3^1 b_3^2 \cdots$, but the conditions restrict the reordering to be permutations of neighbouring bits. In essence, these mappings are not fundamentally different, and the Z-mapping can be seen as a "standard example" of the pointwise convergent mappings with bijective limits, which can be expressed as the permutation of binary digits.

3.4 Coarse-graining the neural field in $[0,1]$

3.4.1 Naive simulations of the neural field on $[0,1]$

In the continuation of the simulation methods established in the Section 2.2, we adopt a grid discretization scheme to simulate the neural field on $[0,1]$, as the grid naturally yields square populations, which can directly be used with the mappings introduced previously.[6]

We now address the elephant in the room: how do we numerically simulate a neural field defined on – what appears to be – a fractal curve? Naively, we could just proceed as follows:

1. Create a grid of $2^n \times 2^n = 4^n$ squares of size $2^{-n} \times 2^{-n}$ in the $[0,1] \times [0,1]$ embedding. Each square is located by a position $\boldsymbol{v_i} = (v_{1,i}, v_{2,i})$, $i \in \{1, \cdots 2^n\}$.
2. Compute a discretized version of the 1D connectivity kernel

$$\begin{aligned} \tilde{w}(\alpha_i, \beta_j) &= \tilde{J}_{\alpha_i,\beta_j} \\ &= \tilde{J}_{S(i),S(j)} \\ &\overset{\text{def}}{=} J_{ij} \\ &= w_U(\boldsymbol{v_i}, \boldsymbol{v_j}) \end{aligned}$$

   for every pair of square locations (where we denote for simplicity of notation $S(i) = S(\boldsymbol{v_i})$).
3. Simulate the neural field in $[0,1]$ on the corresponding discretized space, with a mesh size 4^{-n}.

The problem with this approach is that since the mappings are just *permutations* of the square populations, the resulting connectivity matrix $\tilde{J}_{\alpha_i,\beta_j}$ is just a permutation of the rows and columns of J_{ij}, and will therefore trivially lead to identical neural field dynamics.

[6] In order to simulate the mappings using Monte-Carlo schemes, binning methods can be applied. The intuition behind this is that sampled populations inside a small square of size $2^{-n} \times 2^{-n}$ give rise to the notion of an "average connectivity kernel", and in our case the notion of "mean patterns". More details are given in Appendix B.

3.4.2 Coarse-graining the one-dimensional neural field

In some sense, we would like to “take into account” the fact that a mapping of the square populations has been performed. Simulations of the neural field equation on $[0,1]^2$ are done by using 4^n square populations on a grid, but, as long as we have 4^n corresponding segment populations in $[0,1]$, the dynamics will trivially be the same, *independantly of the mapping* S^n. Therefore, we need a way to reduce the number of segment populations in $[0,1]$.

We do this using “coarse-graining”, in which we average 2^n consecutive segment populations, each of length 4^{-n}, in order to obtain 2^n coarse-grained segment populations, each of length 2^{-n}. This operation is illustrated in Figure 3.10.

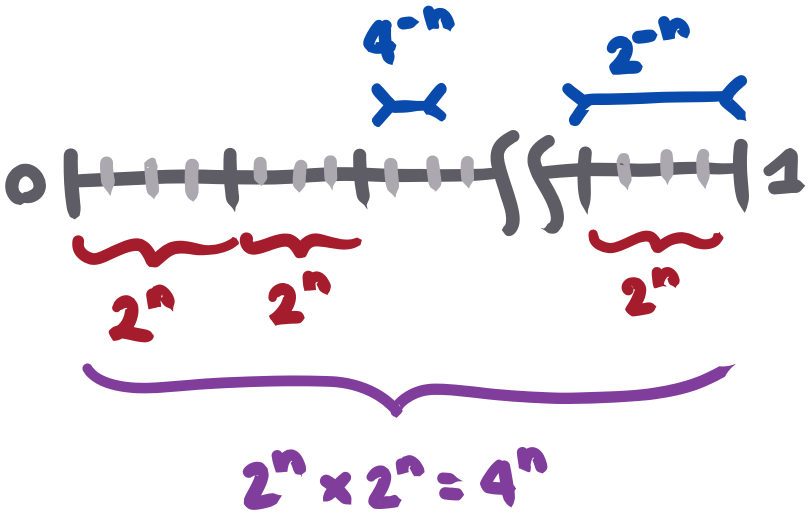
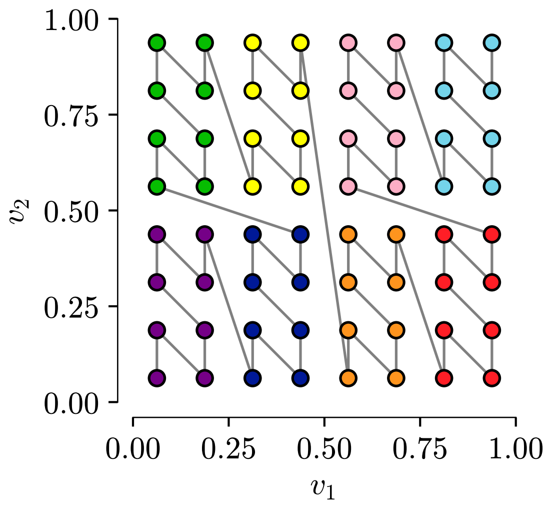


Figure 3.10: Averaging 2^n of the 4^n segments populations of S^n results in the 2^n segments populations

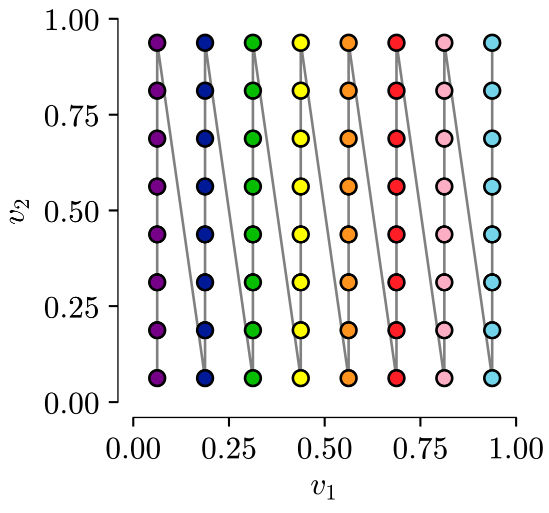
Without coarse-graining, the mesh size along each spatial dimension is 2^{-n} in 2D, but directly applying the mapping results instead in a mesh size of 4^{-n} in 1D, which is much finer than the 2^{-n} per dimension. However, after coarse-graining, the effective mesh size in 1D becomes 2^{-n}, which matches the 2D mesh size. Therefore, we are are not compensating the reduction of dimensionality by an increase in the numerical resolution.

A geometrical intuition we can give for the coarse-graining is how the averaging of the segment populations leads to sampling the connectivity kernel at “average” positions in the $[0,1]^2$ embedding. In Figure 3.11, we colour-code the square populations corresponding to the binned segment populations before coarse-graining.

In summary, coarse-graining allows us to compare the dynamics of a neural field in $[0,1]^2$ simulated using $2^n \times 2^n = 4^n$ square populations; with the dynamics of a neural field in $[0,1]$ simulated using 2^n segment populations.



(a) Z-mapping



(b) Column mapping

Figure 3.11: Square populations corresponding to the segment populations before coarse-graining are colored identically when they are the same bin of 1D length 2^{-n}. Here we show $n = 3$.

3.5 Simulations of neural fields in 3D, 2D and 1D

In this section, we show numerical simulations of the neural field in $[0, 1]$, obtained from applying mappings on higher-dimensional neural fields in $[0, 1]^p$. We see that the coarse-graining procedure works when applied from $[0, 1]^2$ to $[0, 1]$. Then, we demonstrate that it can be applied iteratively, to map from $[0, 1]^3$ to $[0, 1]^2$, and from $[0, 1]^2$ to $[0, 1]$.

3.5.1 Practical aspects of implementing the simulations of the one-dimensional neural field

We give a quick overview of the numerical details that go into implementing the methods presented above. Since the mappings (Z-mapping, Column mapping) can be written as binary expansions, they naturally lend themselves to numerical approximations (see `ZMapping`, `ColumnMapping`). Furthermore, this is made especially easy since the mappings we defined are simply permutations of bits (in the $n < \infty$ case), and so we can write a one-to-one correspondence between unsigned integers.

We have implemented the mappings as all inheriting from the same abstract class, and each mapping simply has to define a method `indices_to_indices2d(indices: array[int, 1]) -> array[int, 2]` to go from 1D to 2D, and a method `indices2d_to_indices(indices2d: array[int, 2]) -> array[int, 1]` to go from 2D to 1D.[7] The abstract class simply defines additional quality-of-life functionality, which makes use of these two methods.

Finally, the coarse-graining is easily implemented as the combination of a reshape operation (to bring all consecutive sequences of 2^n segment populations together), followed by averaging (see the implementation here)

3.5.2 Application of S: from $[0,1]^2$ to $[0,1]$

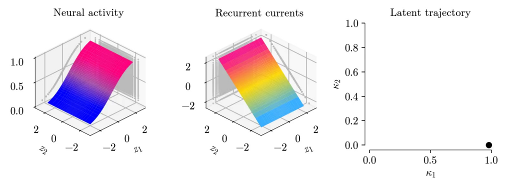
We start with the cycling neural field presented in Section 2.4. The kernel $\tilde{w}$ of the one-dimensional neural field is numerically approximated using the coarse-graining procedure of different mappings. Simulations are presented in Figure 3.12 (we again set $\delta = 6$ and the initial condition $h(z_1, z_2, t < 0) = z_1$), showing the activity levels, recurrent currents, and the latent trajectory $(\kappa_1(t), \kappa_2(t))$. In order, we discuss:

- The random mapping can be numerically evaluated as a trivial case of a mapping that does not have locality. The coarse-graining averages out all the structure of the activity, and the recurrent currents are zero, because the contributions from excitatory and inhibitory populations cancel out. This results in the erasure of the cycling behavior in the latent trajectory.
- The Column mapping shows the projection of the initial condition onto the z_1 axis, yielding the sigmoid in the $[0,1]$ embedding. Again, the current currents cancel out and yield zero, and the latent trajectory quickly decays towards $(\kappa_1, \kappa_2) = (0,0)$.
- The Z-mapping shows that despite the fractal structure of the activity, there is still regularity in the $[0,1]$ embedding, resulting in non-zero recurrent currents. In consequence, the latent trajectory shows a clear cycling behavior, that closely matches that of the 2D neural field.

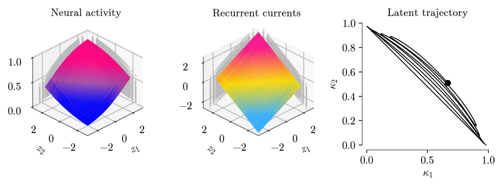
3.5.3 Iterative application of S: from $[0,1]^3$ to $[0,1]^2$, then to $[0,1]$

In the previous section, we showed that the coarse-graining procedure works from $[0,1]^2$ to $[0,1]$. However, one might argue that the reason why it worked, is not the mapping, but rather the continuity of the neural field in $[0,1]^2$. To show the generality of our results, we show that coarse-graining works, even when the original neural field is fractal.

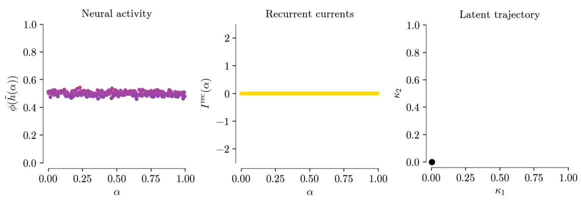
[7] In the source code, only the method `indices(v12: array[float, 2]) -> array[int, 1]` is defined. This is because some mappings are implemented (for instance, the `RecursiveLocalMapping`, that we don't discuss here), that don't lend themselves to the simple permutation of bits formulation.



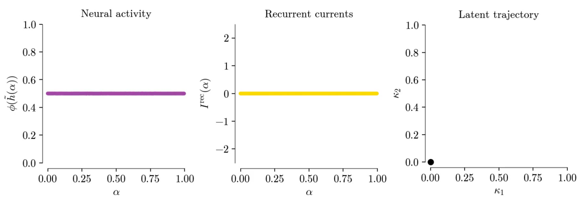
(a) Neural field in the 2D embedding. First frame, animation available here.



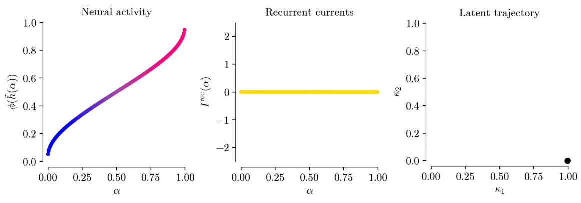
(b) Neural field in the 2D embedding. Last frame, animation available here.



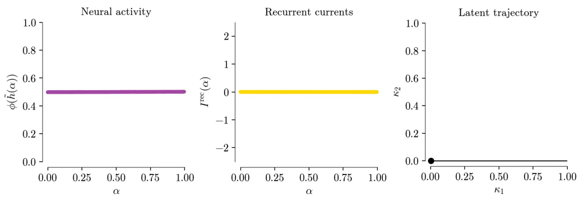
(c) Random mapping followed by coarse-graining. First frame, animation available here.



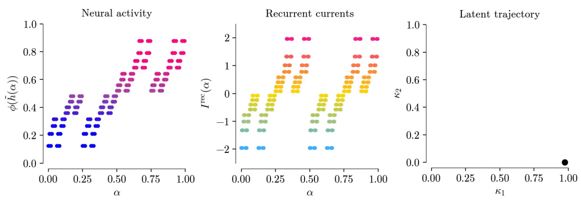
(d) Random mapping followed by coarse-graining. Last frame, animation available here.



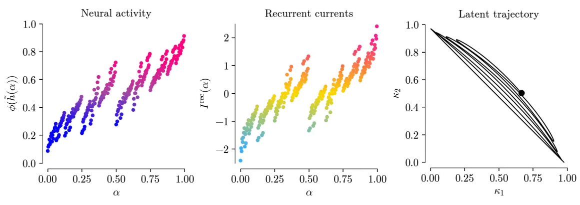
(e) Column mapping followed by coarse-graining. First frame, animation available here.



(f) Column mapping followed by coarse-graining. Last frame, animation available here.



(g) Z-mapping followed by coarse-graining First frame, animation available here.

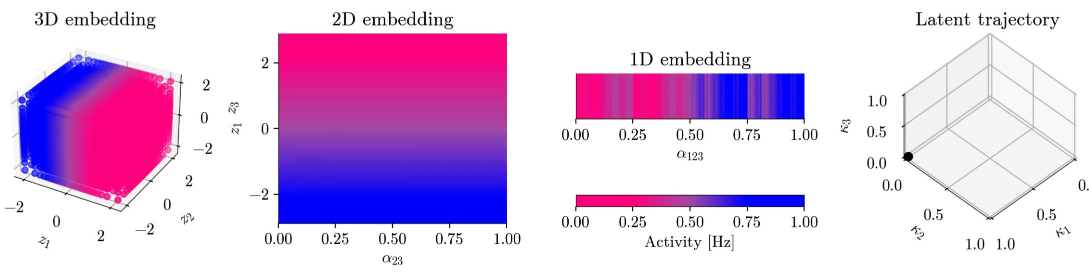


(h) Z-mapping followed by coarse-graining. Last frame, animation available here.

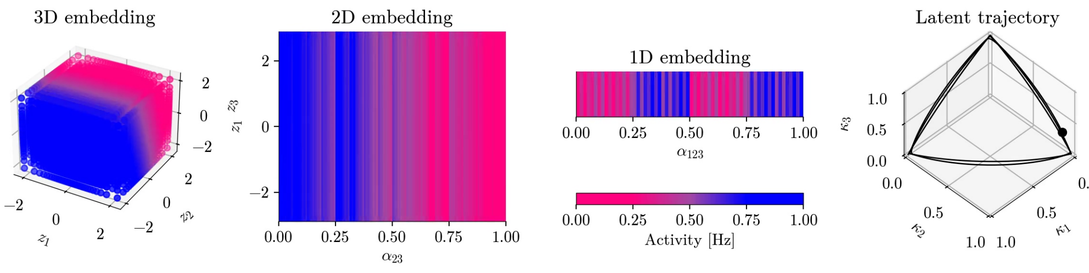
Figure 3.12: The cycling 2D neural field (4^8 square populations) is mapped to $[0, 1]$ using different mappings, then coarse-grained (resulting in 2^8 segment populations).

For this, we again consider the same cycling field, but with $p = 3$, such that the natural embedding space is $\mathbb{R}^3$, with the equivalent formulation on $[0,1]^3$. We again consider a grid with a mesh size of 2^{-n} along each spatial dimension, resulting in $2^n \times 2^n \times 2^n = 8^n$ "cube populations", at positions $(v_1, v_2, v_3) \in [0,1]^3$.

To go from $[0,1]^3$ to $[0,1]^2$, we consider a slice (v_2, v_3) at constant v_1, map (v_2, v_3) to α_{23}, then perform coarse-graining, which brings down the number of square populations in $[0,1]^2$ from 8^n to 4^n. The result of this is an embedding of the neural field on $(v_1, \alpha_{23}) \in [0,1]^2$. The corresponding neural field is continuous in its v_1 component, but fractal in α_{23}. This is visualized in Figure 3.13, where we see that in the state $h(z_1, z_2, z_3) = z_1$, the corresponding 2D neural field is constant along the α_{23} component, since the slice (v_2, v_3) has a constant activity level. However, as soon as h evolves to a new state, non-orthogonal to z_2 and z_3, the fractal structure becomes clear, because the slice (v_2, v_3) is no longer constant.



(a) First frame



(b) Last frame

Figure 3.13: Equivalence of dynamics, from 3D to 2D to 1D. The first and the last frame of the animation are shown, available here.

We close this chapter by repeating this process to go from $[0,1]^2$ to $[0,1]$, mapping (v_1, α_{23}) to α_{123}, and coarse-graining. In total, we have coarse-grained two times: once from $[0,1]^3$ (8^n cube populations) to $[0,1]^2$ (4^n square populations), and again from $[0,1]^2$ to $[0,1]$ (2^n segment populations). The corresponding 1D neural field is now fully fractal, and again exhibits the same dynamics. This shows that "local mappings conserve regularity": we started with a fully continous neural field in $[0,1]^3$, and obtained a partially fractal neural field in $[0,1]^2$. Despite the fractal structure, the

2D neural field is still regular enough that the dynamics are conserved, but more importantly that it can *again* be mapped to an equivalent 1D neural field.

4 Locality and numerical convergence

The previous chapter gave a detailed description of the mappings $S : [0,1]^2 \mapsto [0,1]$, as well as a few numerical considerations. We found that mappings that yield equivalent dynamics crucially seem to have in common the property of *locality* (from 1D to 2D), even if the image seems to have a fractal structure. Intuitively, we saw that the locality seems to emerge from the mapping having for the most part small discontinuities.

In this chapter, we will further explore these results, and attempt to formulate a measure of this "locality". As a preamble, we show that bijectivity and measurability are sufficient conditions for the 2D and 1D neural field dynamics to be equivalent. Then, we apply our measure of "locality" to the special cases of the Column and Z-mappings, and show how this notion can discriminate between the two mappings. Finally, we show that our measure of locality is sufficient to ensure that the numerical estimation of the integral converges to the analytical integral; therefore, such embeddings can be simulated numerically.

4.1 Analytical equivalence of the dynamics when S is measurable and bijective

Let us assume that the mapping S is bijective and measurable. This is not a strong assumption, since the limit of pointwise convergent measurable functions is also measurable. In our case, we have written the finite-n expansions S^n, which, by being sums of piecewise constant functions, are also measurable. The measurability of S allows us to write $\lambda \circ S^{-1}$, the pushforward measure (or image measure) of λ under the mapping S.

With these two assumptions, let us prove that by defining the initial condition and connectivity

$$\begin{aligned}
\tilde{h}(\alpha, t=0) &= h_U(S^{-1}(\alpha), t=0) \\
\tilde{w}(\alpha, \beta) &= w_U(S^{-1}(\alpha), S^{-1}(\beta)),
\end{aligned}$$

then the solution $h_U(\boldsymbol{v}, t)$ of the neural field equation in $[0,1]^2$ uniquely defines the solution of the neural field equation in $[0,1]$ by writing

$$\tilde{h}(\alpha, t) = h_U(\cdot, t) \circ S^{-1}(\alpha). \tag{4.1}$$

We start by recalling the expression of the one-dimensional neural field, then substitute Equation 4.1 and $\boldsymbol{v} = S^{-1}(\alpha)$, $\boldsymbol{u} = S^{-1}(\beta)$.

$$\begin{aligned}
\partial_t \tilde{h}(\alpha, t) &= -\tilde{h}(\alpha, t) + \int_{[0,1]} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \\
&\iff \\
\partial_t h_U(S^{-1}(\alpha), t) &= -h_U(S^{-1}(\alpha), t) \\
&\quad + \int_{[0,1]} w(S^{-1}(\alpha), S^{-1}(\beta))\phi(h_U(S^{-1}(\beta), t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \\
&= -h_U(\boldsymbol{v}, t) \\
&\quad + \int_{[0,1]} w(\boldsymbol{v}, S^{-1}(\beta))\phi(h_U(S^{-1}(\beta), t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \\
&= -h_U(\boldsymbol{v}, t) + \int_{[0,1]^2} w(\boldsymbol{v}, \boldsymbol{u})\phi(h_U(\boldsymbol{u}, t))\lambda(\mathrm{d}\boldsymbol{u})
\end{aligned}$$

In the last line, we used a basic property of the pushfoward measure (see [35], Theorem 3.6.1, p. 190). The Lebesgue measure $\lambda : [0,1]^2 \mapsto \mathbb{R}^+$ simply plays the role of a uniform distribution on $[0,1]^2$, and we simply write

$$\partial_t h_U(\boldsymbol{v}, t) = -h_U(\boldsymbol{v}, t) + \int_{[0,1]^2} w(\boldsymbol{v}, \boldsymbol{u})\phi(h_U(\boldsymbol{u}, t))\mathrm{d}\boldsymbol{u}.$$

Therefore, if $h_U(\boldsymbol{v}, t)$ solves the neural field equation in $[0,1]^2$, then $\tilde{h}(\alpha, t) = (h_U(\cdot, t) \circ S^{-1})(\alpha)$ solves the equivalent neural field equation in $[0,1]$. The same reasoning applies in reverse: if $\tilde{h}(\alpha, t)$ solves the neural field equation in $[0,1]$, then $h_U(\boldsymbol{v}, t) = (\tilde{h}(\cdot, t) \circ S)(\boldsymbol{v})$ solves the neural field in $[0,1]^2$.

We do not address the question of well-posedness of the neural field equations in this thesis; for proofs of the well-posedness of neural field equations, we refer the reader to [30], [36], [37].

4.2 Locality

4.2.1 Locality as vanishing average binned variation

The previous chapter gave the intuition that local mappings map populations close in 1D to populations close in 2D. To verify this, we might try to compare the distance between all segment populations and the population at $\alpha = 0$, to the distance of all square populations to the population at $S^{-1}(\alpha = 0)$. This is done in Figure 4.1, where the 2D distance is taken to be the ℓ^2 norm. If the distances matched perfectly, then we would see a line with slope $\sqrt{2}$ in the graph.

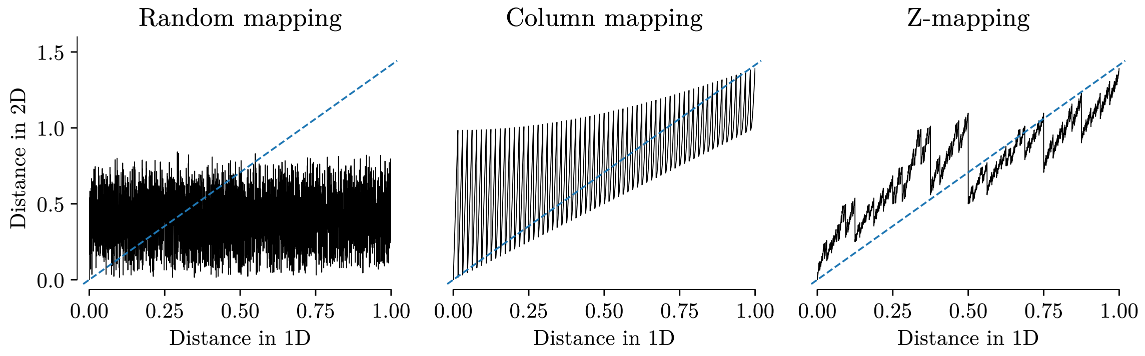


Figure 4.1: Distance between the first population and all other populations, in 1D and 2D. The dotted line has a slope $\sqrt{2}$. This figure is generated using $n = 6$.

For control, we also demonstrate that the random mapping has no locality: there is no correlation between the positions in 1D and the positions in 2D. Both Column and Z-mappings show a clear trend, the distances in 2D are correlated to the distances in 1D. The notable difference is that distances for the Column mapping seem to "oscillate" wildly, while the Z-mapping seems much more regular and matches the straight line more closely.

With this view, we understand that the coarse-graining performed in Section 3.4 makes use of that fact that locality helps by bounding the variation inside of each averaged bin. Therefore, we would like for mathematically formulate that on average, the variation inside of each bin before coarse-graining is small. Figure 4.2 gives an intuition of this. Even though the $[0, 1]$ embedding might be discontinuous in places, on average nearby neurons should exhibit the same activity levels.

Given n, let us define the average binned variation as the maximum difference of the 2D positions associated to segment populations in the same bin of size 2^{-n}:

$$V_n(S^{-1}) = \frac{1}{2^n} \sum_{i=1}^{2^n} \sup_{\alpha,\alpha' \in \left[\frac{i-1}{2^n}, \frac{i}{2^n}\right]^2} \|S^{-1}(\alpha') - S^{-1}(\alpha)\|_1 \tag{4.2}$$

The definition of locality can essentially be seen as a weakened version of continuity, or "continuity on average". A continuous function f evidently verifies $V_n(f) \xrightarrow{n\to\infty} 0$, since small neighbourhoods are mapped to small neighbourhoods. The reverse is not true, since for instance the metric V_n vanishes when applied to a step function, which is only piecewise continuous.

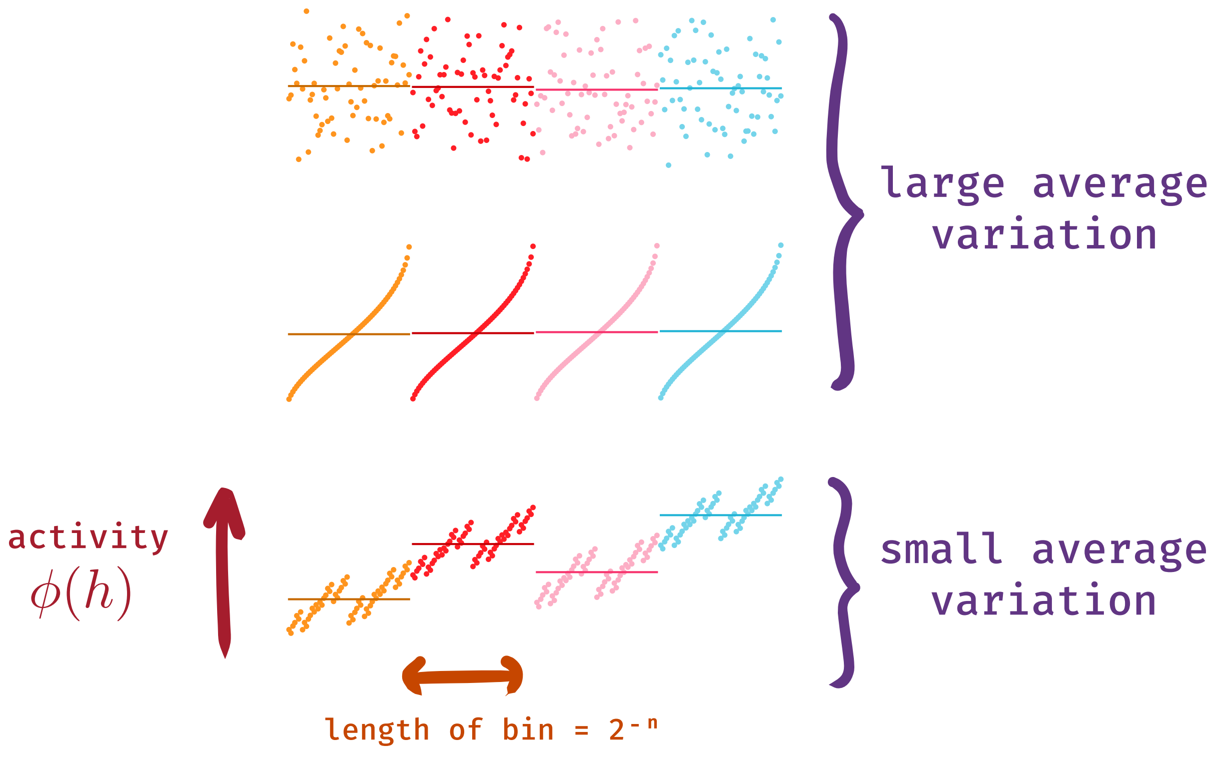


Figure 4.2: Intuition behind the average binned variation. To simplify the visualization, activity levels are used in lieu of the 2D positions. 4 bins are each shown with a different colour. The horizontal line represents the average activity in each bin. From top to bottom: Random, Column, Z-mapping.

Before we move to analytical derivations applying the average variation to mappings, we can first check numerically if it gives the expected results; that is V_n vanishes when $n \to \infty$ for local mappings, whereas V_n should be lower bounded by a constant for non-local mappings. Figure 4.3 shows this is indeed the case, rejoice!

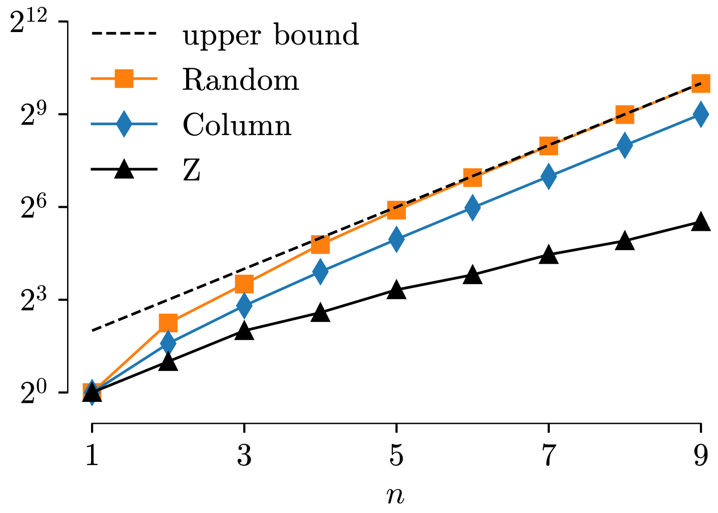


Figure 4.3: (Numerator of the) average binned variation as a function of n. The Z-mapping has a growth slower than the upper bound 2×2^n. The Column mapping grows at the same rate as the upper bound, and the Random mapping saturates it.

4.2.2 Computations of average variation

To compute (a bound on) V_n for some mappings, we consider the binary expansion of α and α' which appear in the supremum. For clarity and to represent actual numerical implementations, we furthermore truncate the binary expansion to $2m$ bits (on most machines, $2m = 64$ bits).

$$\alpha = 0.b_1 b_2 \cdots b_{2m-1} b_{2m}$$
$$\alpha' = 0.b'_1 b'_2 \cdots b'_{2m-1} b'_{2m}$$

Furthermore, since $\alpha \in \left[\frac{i-1}{2^n}, \frac{i}{2^n}\right]$, $i \in \{1, \cdots, 2^n\}$, we may express the first n (where $n < 2m$) bits of α with the bits of $i = \sum_{k=0}^{n-1} i_k 2^k$ (we ignore $i = 2^n$ for the sake of simplicity, which would introduce one more bit, $n+1$ in total).

$$\alpha = 0.i_{n-1} i_{n-2} \cdots i_1 i_0 b_n b_{n+1} \cdots b_{2m-1} b_{2m}$$
$$\alpha' = 0.i_{n-1} i_{n-2} \cdots i_1 i_0 b'_n b'_{n+1} \cdots b'_{2m-1} b'_{2m}$$

4.2.2.1 Z-mapping

Let us consider the Z-mapping, and the two terms compositing its inverse. In the case of α, α' above, we have if n is even:

$$
\begin{aligned}
Z_1^{-1}(\alpha) &= 0.\underbrace{i_{n-1}i_{n-3}\cdots i_1}_{n/2 \text{ bits}} b_n b_{n+2} \cdots b_{2m-1} \\
Z_2^{-1}(\alpha) &= 0.\underbrace{i_{n-2}i_{n-4}\cdots i_0}_{n/2 \text{ bits}} b_{n+1} b_{n+3} \cdots b_{2m}.
\end{aligned}
$$

If n is odd, we have:

$$
\begin{aligned}
Z_1^{-1}(\alpha) &= 0.\underbrace{i_{n-1}i_{n-3}\cdots i_0}_{(n+1)/2 \text{ bits}} b_n b_{n+2} \cdots b_{2m-1} \\
Z_2^{-1}(\alpha) &= 0.\underbrace{i_{n-2}i_{n-4}\cdots i_1}_{(n-1)/2 \text{ bits}} b_{n+1} b_{n+3} \cdots b_{2m}.
\end{aligned}
$$

We only consider even n in the following, as the proof for the latter remains very similar.

The core of the proof relies on the fact the first $n/2$ bits of the difference of the inverses cancel out when computing the average binned variation.

$$
\begin{aligned}
|Z_1^{-1}(\alpha) - Z_1^{-1}(\alpha')| &= \left|0.0\cdots 0(b_n b_{n+2}\cdots b_{2m-1} - b'_n b'_{n+2}\cdots b'_{2m-1})\right| \\
&= \left| 2^{-n/2} \sum_{k=1}^{\frac{2m-n+1}{2}} (b_{n+2k-1} - b'_{n+2k-1})2^{-k} \right| \\
&\le 2^{-n/2} \\
|Z_2^{-1}(\alpha) - Z_2^{-1}(\alpha')| &= \left| 2^{-n/2} \sum_{k=1}^{\frac{2m-n+1}{2}} (b_{n+2k} - b'_{n+2k})2^{-k} \right| \\
&\le 2^{-n/2}
\end{aligned}
$$

Putting everything together, we get a vanishing upper bound for the average binned variation.

$$
V_n(Z^{-1}) \le \frac{1}{2^n}\sum_{i=1}^{2^n} 2^{-n/2} + 2^{-n/2} = 2^{-n/2+1} \xrightarrow{n\to\infty} 0
$$

4.2.2.2 Column mapping

We now consider the Column mapping. Since here we consider finite-precision α and α', the invere is well-defined and we have

$$\begin{aligned}
C_1^{-1}(\alpha) &= 0.i_{n-1}i_{n-2}\cdots i_1 i_0 b_n b_{n+1}\cdots b_m \\
C_2^{-1}(\alpha) &= 0.b_{m+1}b_{m+2}\cdots b_{2m}.
\end{aligned}$$

The first component can be bounded using the same trick from the previous proof :

$$\begin{aligned}
|C_1^{-1}(\alpha) - C_1^{-1}(\alpha')| &= \left| 2^{-n}\sum_{k=1}^{m-n+1}(b_{n+k-1} - b'_{n+k-1})2^{-k}\right| \\
&\leq 2^{-n}.
\end{aligned}$$

However, now problems arise when we compute the averge binned variation, since the second component is not well-behaved. In particular, since we take the supremum on $\alpha, \alpha' \in \left[\frac{i-1}{2^n}, \frac{i}{2^n}\right]^2$, we can pick the specific values such that $b_k = 1$ and $b'_k = 0$ for all $k \in \{m+1, \cdots 2m\}$.

$$\begin{aligned}
|C_2^{-1}(\alpha) - C_2^{-1}(\alpha')| &= \left|\sum_{k=m+1}^{2m}(b_k - b'_k)2^{-(k-m)}\right| \\
&= \sum_{k=m+1}^{2m} 2^{-(k-m)} \\
&> \frac{1}{2}
\end{aligned}$$

Putting both together, we can write:

$$\begin{aligned}
V_n(C^{-1}) &= \frac{1}{2^n}\sum_{i=1}^{2^n}\sup_{\alpha,\alpha'\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]^2} |C_1^{-1}(\alpha') - C_1^{-1}(\alpha)| + |C_2^{-1}(\alpha') - C_2^{-1}(\alpha)| \\
&\geq \frac{1}{2^n}\sum_{i=1}^{2^n}\sup_{\alpha,\alpha'\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]^2} |C_1^{-1}(\alpha') - C_1^{-1}(\alpha)| \\
&\qquad + \sup_{\alpha,\alpha'\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]^2} |C_2^{-1}(\alpha') - C_2^{-1}(\alpha)| \\
&\geq \frac{1}{2} + \mathcal{O}(2^{-n}).
\end{aligned}$$

Therefore we have found a lower bound, which proves the average binned variation does not converge to zero (if it converges at all).

$$V_n(C^{-1}) > \frac{1}{2} \implies \lim_{n\to\infty} V_n(C^{-1}) > \frac{1}{2}$$

4.3 Numerical convergence of the finite-n estimations

In Section 4.2.1, we have introduced the average variation metric V_n. Assuming that $V_n(S^{-1}) \xrightarrow{n\to\infty} 0$ (which expresses the locality property), does this imply that the numerical estimation of the integral in the neural field equation in $[0,1]$ converges to its true value?

The intuition behind the proof of convergence is that the variation, coming from the coarse-graining, inside each numerical bin vanishes as we increase the number of iterations n, even though the neural field $\tilde{h}$ and the connectivity kernel $\tilde{w}$ become fractal.

We define the one-dimensional field on the grid:

$$\tilde{h}_i(t) = \tilde{h}(\tfrac{i-1}{2^n}, t) = \tilde{h}(\beta_i, t),\ \beta_i = \tfrac{i-1}{2^n},\ i \in \{1, \cdots, 2^n\}.$$

Let us first recall the numerical and analytical integral. For any $\alpha \in [0,1]$, let us define:

$$\begin{aligned} \mathrm{NI}_n &= \frac{1}{2^n}\sum_{i=1}^{2^n} \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) \\ \mathrm{AI} &= \int_{[0,1]} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right](\mathrm{d}\beta). \end{aligned}$$

We wish to prove that $|\mathrm{NI}_n - \mathrm{AI}| \xrightarrow{n\to\infty} 0$.

The first part of the proof involves splitting the integral on $[0,1]$ into 2^n integral over segments of length 2^{-n}, and using the triangle inequality.

$$
\begin{aligned}
|\mathrm{NI}_n - \mathrm{AI}| &= \left| \frac{1}{2^n} \sum_{i=1}^{2^n} \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) - \int_0^1 \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \right| \\
&= \left| \frac{1}{2^n} \sum_{i=1}^{2^n} \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) - \sum_{i=1}^{2^n} \int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \right| \\
&= \left| \sum_{i=1}^{2^n} \frac{1}{2^n} \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) - \int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \right| \\
&\leq \sum_{i=1}^{2^n} \left| \frac{1}{2^n} \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) - \int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}} \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \right|
\end{aligned}
$$

We now want to prove that evaluating the integrand at $\beta_i = \frac{i-1}{2^n}$ gives a good approximation of the integral. Let us assume that S is the Z-mapping. Then, S is measure-preserving (if λ is the Lebesgue measure on $[0,1]^2$, then $\lambda \circ S^{-1}$ is the Lebesgue measure on $[0,1]$). This implies that we have $\int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}} [\lambda \circ S^{-1}] (\mathrm{d}\beta) = \frac{1}{2^n}$, and we write

$$
\begin{aligned}
|\mathrm{NI}_n - \mathrm{AI}| &\leq \sum_{i=1}^{2^n} \left| \int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}} \Delta(\beta) \left[\lambda \circ S^{-1}\right] (\mathrm{d}\beta) \right| \\
\text{where } \Delta(\beta) &= \tilde{w}(\alpha, \beta_i)\phi(\tilde{h}_i(t)) - \tilde{w}(\alpha, \beta)\phi(\tilde{h}(\beta, t)) \\
&= w_U(\alpha, S^{-1}(\beta_i))\phi(h_U(S^{-1}(\beta_i), t)) - w_U(\alpha, S^{-1}(\beta))\phi(h_U(S^{-1}(\beta), t)).
\end{aligned}
$$

The main source of variance inside the integrand $\Delta(\beta)$ is the term S^{-1}, since w_U and h_U are "nice" functions. We express this regularity by the added assumption that $w_U(\alpha, \boldsymbol{u})\phi(h_U(t, \boldsymbol{u}))$ is Lipschitz in $\boldsymbol{u}$, with Lipschitz constant $L(\alpha)$, and $\|\alpha\|$ is the corresponding norm on $[0,1]^2$. We argue this is not a strong assumption, because this is just expressing the regularity of the integrand in the integral over $[0,1]^2$.

A more pragmatic justification of this is to notice that the "density of recurrent currents" is expressed as:

$$
I_U^{\mathrm{rec}}(\boldsymbol{v}, \boldsymbol{u}) = w_U(\boldsymbol{v}, \boldsymbol{u})\phi(h_U(t, \boldsymbol{u})),
$$

and that in Section 2.4, numerical simulations showed that $I_U^{\mathrm{rec}}(\boldsymbol{v}, \boldsymbol{u})$ (the integrand of the integral on $[0,1]^2$) is a well-behaved function.

Applying the Lipschitz assumption yields the following inequality on the integrand:

$$\begin{aligned}\Delta(\beta) &= w_U(\alpha, S^{-1}(\beta_i))\phi(h_U(S^{-1}(\beta_i), t)) - w_U(\alpha, S^{-1}(\beta))\phi(h_U(S^{-1}(\beta), t)) \\ &\leq L(\alpha)\|S^{-1}(\beta_i) - S^{-1}(\beta)\|.\end{aligned}$$

We now make use of our intuition of "locality": points close in $[0,1]$ should be (on average) mapped to points close in $[0,1]^2$. Therefore, taking the sup inside each 2^{-n} segment does not introduce a too large error.

$$\begin{aligned}|\mathrm{NI}_n - \mathrm{AI}| &\leq L(\alpha)\sum_{i=1}^{2^n}\left|\int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}}\|S^{-1}(\beta_i) - S^{-1}(\beta)\|\left[\lambda \circ S^{-1}\right](\mathrm{d}\beta)\right| \\ &\leq L(\alpha)\sum_{i=1}^{2^n}\left|\int_{\frac{i-1}{2^n}}^{\frac{i}{2^n}}\sup_{\beta\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]}\|S^{-1}(\beta_i) - S^{-1}(\beta)\|\left[\lambda \circ S^{-1}\right](\mathrm{d}\beta')\right| \\ &= L(\alpha)\frac{1}{2^n}\sum_{i=1}^{2^n}\sup_{\beta\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]}\|S^{-1}(\beta_i) - S^{-1}(\beta)\| \\ &\leq L(\alpha)\frac{1}{2^n}\sum_{i=1}^{2^n}\sup_{\beta,\beta'\in\left[\frac{i-1}{2^n},\frac{i}{2^n}\right]^2}\|S^{-1}(\beta') - S^{-1}(\beta)\| \\ &= L(\alpha)V_n(S^{-1}) \\ &\xrightarrow{n\to\infty} 0\end{aligned}$$

We see that it is the notion of "locality" of the mapping S that allowed us to write the convergence of the numerical integral to the analytical value. In doing so, we exploited the fact that the "variance" inside each small segment in $[0,1]$ is small on average, and that the errors vanish as we take finer and finer bins.

4.4 Generalizing to $[0,1]^p \mapsto [0,1]$

We finish this thesis by tying up the loose ends. In the abstract, we wrote that any p-dimensional neural field equation is equivalent to a neural field equation in $[0,1]$, but we only gave demonstrations for $[0,1]^2$ to $[0,1]$. Section 3.5.3 showed that it is possible to iteratively apply the mapping $Z : [0,1]^2 \mapsto [0,1]$ to reduce dimensionality of the neural field, but it might be more practical to have a direct mapping $Z_p : [0,1]^p \mapsto [0,1]$.

Let us give a generalization of the Z-mapping. Defining $\boldsymbol{v^{(n)}} = (v_1^{(n)}, \cdots, v_p^{(n)})$, and the finite binary expansions $v_\mu^{(n)} = \sum_{k=1}^n b_k^\mu 2^{-k}$, we can define Z_p^n as

$$\begin{aligned} Z_p^n(\boldsymbol{v^{(n)}}) &= \sum_{k=1}^{n}\sum_{\mu=1}^{p} b_k^{\mu} 2^{-(p(k-1)+\mu)} \\ &= 0.b_1^1 b_1^2 \cdots b_1^p b_2^1 b_2^2 \cdots b_{n-1}^p b_n^1 b_n^2 \cdots b_n^p. \end{aligned}$$

The same arguments apply to show that this generalized Z-mapping converges pointwise to Z_p, has the locality property, and that $V_n(Z_p^{-1}) \xrightarrow{n\to\infty} 0$. Then the proof of convergence, albeit more laborious, can be repeated using the same procedure.

5 Conclusion

In this thesis, we have constructed an example of neural field dynamics in $\mathbb{R}^2$, that can be mapped to a neural field equation in $[0,1]$, via well-known measurable bijections (such as the Z-mapping) between $[0,1]^2$ and $[0,1]$. To illustrate this, we have introduced a *coarse-graining* procedure in which numerical simulations of 2^n segment populations in $[0,1]$ converge to the same dynamics obtained from simulations of 4^n square populations in $[0,1]^2$. Additionally, we showed that the reduction of dimensionality using the mappings can be repeated, even if the starting neural field has a fractal structure. To help quantify how the mapping conserves the regularity between neural fields, we introduced a measure of locality, based on "average variation". Locality guarantees that the neural field equation on $[0,1]$ can be numerically approximated using classical grid-based methods.

Although, in this thesis, we have used simulations of simple low-rank networks for numerical illustrations, to demonstrate the generality of our result, future work would try to numerically study an example of neural field dynamics that does not have a low-rank structure. This would also avoid confusion between the linear dimensionality of the dynamics and the dimensionality of the embedding, which are the same in the context of low-rank networks. While we have presented some mathematical arguments for (1) the equivalence of the neural field dynamics in 2D and 1D, and (2) the convergence of the numerical approximation of the 1D neural field dynamics using locality, we could use these arguments to make rigorous proofs of both statements.

The use of the notion of regularity for the analysis of large-scale neuronal recordings is in its infancy [38]. This purely theoretical thesis gives a concrete example of an important conceptual fact: if we want to analyse large-scale neuronal recordings via nonlinear dimensionality reduction techniques (e.g. if we want to model neuronal activity as neural fields evolving on some abstract latent embedding space), the notion of dimensionality is meaningless without an associated notion of regularity. Indeed, our example shows that some neural field dynamics (neuronal population dynamics) can be equivalently expressed as dynamics on the embedding space $\mathbb{R}^2$ and the embedding space $[0,1]$ (but the regularity of the corresponding neural fields are different).

Acknowledgements

I would like to express my most sincere gratitude towards Dr. Valentin Schmutz, who supervised this thesis and provided key insights on the theoretical aspects, as well as invaluable feedback during the editing process.

My gratitude extends to the members of the LCN directed by Prof. Wulfram Gerstner, for their helpful comments during the midterm presentation, and for maintaining a welcoming atmosphere in the lab. Additionally, I would like to thank Louis Henry Pezon for his patience in sharing his knowledge on low-rank networks.

Thank you to my girlfriend Solène, for her unwavering love and support, and for teaching me about the Z-mapping, which proved to be essential in the writing of this thesis. To my friends and family for encouraging me and being there when I needed it most.

References

[1] V. B. Mountcastle, “Modality and topographic properties of single neurons of cat’s somatic sensory cortex,” *Journal of neurophysiology*, vol. 20, no. 4, pp. 408–434, 1957.

[2] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[3] H. R. Wilson and J. D. Cowan, “A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue,” *Kybernetik*, vol. 13, no. 2, pp. 55–80, 1973.

[4] P. L. Nunez, “The brain wave equation: A model for the EEG,” *Mathematical Biosciences*, vol. 21, no. 3–4, pp. 279–297, 1974.

[5] S. Amari, “Dynamics of pattern formation in lateral-inhibition type neural fields,” *Biological cybernetics*, vol. 27, no. 2, pp. 77–87, 1977.

[6] T. K. Sato, I. Nauhaus, and M. Carandini, “Traveling waves in visual cortex,” *Neuron*, vol. 75, no. 2, pp. 218–229, 2012.

[7] L. Muller, A. Reynaud, F. Chavane, and A. Destexhe, “The stimulus-evoked population response in visual cortex of awake monkey is a propagating wave,” *Nature communications*, vol. 5, no. 1, p. 3675, 2014.

[8] C. C. Petersen, T. T. Hahn, M. Mehta, A. Grinvald, and B. Sakmann, “Interaction of sensory responses with spontaneous depolarization in layer 2/3 barrel cortex,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 23, pp. 13638–13643, 2003.

[9] I. Ferezou, S. Bolea, and C. C. Petersen, “Visualizing the cortical representation of whisker touch: Voltage-sensitive dye imaging in freely moving mice,” *Neuron*, vol. 50, no. 4, pp. 617–629, 2006.

[10] D. Rubino, K. A. Robbins, and N. G. Hatsopoulos, “Propagating waves mediate information transfer in the motor cortex,” *Nature neuroscience*, vol. 9, no. 12, pp. 1549–1557, 2006.

[11] K. Takahashi *et al.*, “Large-scale spatiotemporal spike patterning consistent with wave propagation in motor cortex,” *Nature communications*, vol. 6, no. 1, p. 7169, 2015.

[12] E. V. Lubenov and A. G. Siapas, “Hippocampal theta oscillations are travelling waves,” *Nature*, vol. 459, no. 7246, pp. 534–539, 2009.

[13] J. Patel, S. Fujisawa, A. Berényi, S. Royer, and G. Buzsáki, “Traveling theta waves along the entire septotemporal axis of the hippocampus,” *Neuron*, vol. 75, no. 3, pp. 410–417, 2012.

[14] J. Patel, E. W. Schomburg, A. Berényi, S. Fujisawa, and G. Buzsáki, “Local generation and propagation of ripples along the septotemporal axis of the hippocampus,” *Journal of Neuroscience*, vol. 33, no. 43, pp. 17029–17041, 2013.

[15] L. Muller, F. Chavane, J. Reynolds, and T. J. Sejnowski, “Cortical travelling waves: Mechanisms and computational principles,” *Nature Reviews Neuroscience*, vol. 19, no. 5, pp. 255–268, 2018.

[16] M. Breakspear, “Dynamic models of large-scale brain activity,” *Nature neuroscience*, vol. 20, no. 3, pp. 340–352, 2017.

[17] M. Dipoppa, A. Ranson, M. Krumin, M. Pachitariu, M. Carandini, and K. D. Harris, “Vision and locomotion shape the interactions between neuron types in mouse visual cortex,” *Neuron*, vol. 98, no. 3, pp. 602–615, 2018.

[18] K. Ohki, S. Chung, Y. H. Ch’ng, P. Kara, and R. C. Reid, “Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex,” *Nature*, vol. 433, no. 7026, pp. 597–603, 2005.

[19] S. Bugeon *et al.*, “A transcriptomic axis predicts state modulation of cortical interneurons,” *Nature*, 2022.

[20] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky, “Theory of orientation tuning in visual cortex.” *Proceedings of the National Academy of Sciences*, vol. 92, no. 9, pp. 3844–3848, 1995.

[21] K. Zhang, “Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory,” *Journal of Neuroscience*, vol. 16, no. 6, pp. 2112–2126, 1996.

[22] W. Gerstner, “Time structure of the activity in neural network models,” *Phys. Rev. E*, vol. 51, no. 1, p. 738, 1995.

[23] J. Chevallier, A. Duarte, E. Löcherbach, and G. Ost, “Mean field limits for nonlinear spatially extended hawkes processes with exponential memory kernels,” *Stochastic Processes and their Applications*, vol. 129, no. 1, pp. 1–27, 2019.

[24] Z. Agathe-Nerine, “Multivariate hawkes processes on inhomogeneous random graphs,” *Stochastic Processes and their Applications*, vol. 152, pp. 86–148, 2022, doi: https://doi.org/10.1016/j.spa.2022.06.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304414922001545

[25] P.-E. Jabin, D. Poyato, and J. Soler, “Mean-field limit of non-exchangeable systems,” *arXiv preprint arXiv:2112.15406*, 2021.

[26] M. Jazayeri and S. Ostojic, “Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity,” *Current opinion in neurobiology*, vol. 70, pp. 113–120, 2021.

[27] V. Schmutz, J. Brea, and W. Gerstner, “Convergence of redundancy-free spiking neural networks to rate networks,” 2023 [Online]. Available: https://arxiv.org/abs/2303.05174

[28] M. Beiran, A. Dubreuil, A. Valente, F. Mastrogiuseppe, and S. Ostojic, “Shaping dynamics with multiple populations in Low-Rank recurrent networks,” *Neural Comput*, vol. 33, no. 6, pp. 1572–1615, May 2021.

[29] W. Gerstner and J. L. van Hemmen, “Associative memory in a network of 'spiking' neurons,” *Network: Computation in Neural Systems*, vol. 3, no. 2, p. 139, May 1992, doi: 10.1088/0954-898X/3/2/004. [Online]. Available: https://dx.doi.org/10.1088/0954-898X/3/2/004

[30] R. Veltz and O. Faugeras, “Local/global analysis of the stationary solutions of some neural field equations.” 2009 [Online]. Available: https://arxiv.org/abs/0910.2247

[31] D. J. Amit, *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1989.

[32] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[33] A. Kharazishvili, *Strange Functions in Real Analysis*, 3rd ed. New York: Chapman; Hall/CRC, 2017.

[34] H. Sagan, *Space-filling curves*. New York: Springer-Verlag, 1994.

[35] “Operations on measures and functions,” in *Measure theory*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 175–248 [Online]. Available: https://doi.org/10.1007/978-3-540-34514-5_3

[36] O. Faugeras, F. Grimbert, and J.-J. Slotine, “Absolute stability and complete synchronization in a class of neural fields models,” *SIAM Journal on applied mathematics*, vol. 69, no. 1, pp. 205–250, 2008.

[37] O. Faugeras, R. Veltz, and F. Grimbert, “Persistent neural states: Stationary localized activity patterns in nonlinear continuous n-population, q-dimensional neural networks,” *Neural computation*, vol. 21, no. 1, pp. 147–187, 2009.

[38] C. Stringer, M. Pachitariu, N. Steinmetz, M. Carandini, and K. D. Harris, “High-dimensional geometry of population responses in visual cortex,” *Nature*, vol. 571, no. 7765, pp. 361–365, Jul. 2019, doi: 10.1038/s41586-019-1346-5. [Online]. Available: https://doi.org/10.1038/s41586-019-1346-5

A Fixed point study in the network of neurons

We recall the equation for the low-rank network of neurons:

$$\dot{h}_i(t) = -h_i(t) + \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} F_{\mu i} G_{\mu j} \phi(h_j(t)).$$

The low-rank vectors are in general obtained by sampling a distribution $\boldsymbol{z} = (z_1, \cdots, z_p) \sim \rho$. Every neuron i samples this distribution independently, and low-rank vectors are obtained by passing the samples through functions $f_\mu(\boldsymbol{z})$ and $g_\mu(\boldsymbol{z})$.

In the main text, the distribution is a multivariate normal with unit variance, no covariance and zero mean, such that it factors out as $\rho(z_1, \cdots, z_p) = \prod_{\mu=1}^{p} \mathcal{N}(z_\mu)$. Furthermore, the functions are component-wise $f_\mu(z_1, \cdots, z_p) = z_\mu$ and $g_\mu(z_1, \cdots, z_p) = \tilde{\phi}(z_\mu) = \frac{\phi(z_\mu) - \langle \phi(z_\mu) \rangle}{\mathrm{Var}[\phi(z_\mu)]}$.

$$F_{\mu i} = f_\mu(z_{1i}, \cdots, z_{pi}),\ G_{\mu i} = g_\mu(z_{1i}, \cdots, z_{pi})$$

A.1 Derivation of fixed points

To find the fixed points, one has to solve in general $\dot{h}_i(t) = 0$ for all $i = 1, \cdots, N$. Because of the non-linearity $\phi(h_j(t))$, this is difficult, but we can guess the fixed points $h_i^\star$ and verify them by substituting into the equation that $\dot{h}_i(t) = 0$.

A.1.1 Zero fixed point

A trivial fixed point is $h_i^\star = 0$ for all i, which we prove to be a fixed point as $N \to \infty$.

$$\begin{aligned}
\dot{h}_i(t) &= \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} F_{\mu i}G_{\mu j}\phi(0) \\
&= \frac{\phi(0)}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} F_{\mu i}G_{\mu j} \\
&= \frac{\phi(0)}{N}\sum_{\mu=1}^{p}\sum_{j\neq i}^{N} F_{\mu i}G_{\mu j} + \underbrace{\frac{\phi(0)}{N}\sum_{\mu=1}^{p} F_{\mu i}G_{\mu i}}_{\mathcal{O}(\frac{1}{N})} \\
&= \phi(0)\sum_{\mu=1}^{p}\frac{1}{N}\sum_{j\neq i}^{N} F_{\mu i}G_{\mu j} + \mathcal{O}(\tfrac{1}{N}) \\
&= \phi(0)\sum_{\mu=1}^{p} F_{\mu i}\frac{N-1}{N}\frac{1}{N-1}\sum_{j\neq i}^{N} G_{\mu j} + \mathcal{O}(\tfrac{1}{N}) \\
&\overset{N\gg 1}{\approx} \phi(0)\sum_{\mu=1}^{p} F_{\mu i}\langle g_\mu(\boldsymbol{z})\rangle
\end{aligned}$$

To write the last line, we make a slight abuse of notation in order to avoid notation problems when taking the $N \to \infty$ limit of $F_{\mu i}$.

In the case of the network studied in this work, we find that

$$\langle g_\mu(\boldsymbol{z})\rangle = \langle\tilde{\phi}(z_\mu)\rangle = \langle\frac{\phi(z_\mu) - \langle\phi(z_\mu)\rangle}{\mathrm{Var}[\phi(z_\mu)]}\rangle = 0.$$

Thus, $h_i^\star = 0$ is a fixed point in the limit $N \to \infty$.

A.1.2 Pattern fixed point

Another set of fixed points is $h_i^\star = F_{\nu i}$ for all $\nu = 1, \cdots, p$. We refer to these as the "patterns", since in the context of learning the network "remembers" (converges to) these vectors. We verify the fixed point similarly as before.

$$
\begin{aligned}
\dot{h}_i(t) &= -F_{\nu i} + \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} F_{\mu i} G_{\mu j}\phi(F_{\nu j}) \\
&= -F_{\nu i} + \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j\neq i}^{N} F_{\mu i} G_{\mu j}\phi(F_{\nu j}) + \mathcal{O}(\tfrac{1}{N}) \\
&= -F_{\nu i} + \frac{1}{N}\sum_{j\neq i}^{N} F_{\nu i} G_{\nu j}\phi(F_{\nu j}) + \frac{1}{N}\sum_{\mu\neq\nu}^{p}\sum_{j\neq i}^{N} F_{\mu i} G_{\mu j}\phi(F_{\nu j}) + \mathcal{O}(\tfrac{1}{N}) \\
&\overset{N\gg 1}{\approx} -F_{\nu i} + F_{\nu i}\langle g_\nu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle + \sum_{\mu\neq\nu}^{p} F_{\mu i}\langle g_\mu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle
\end{aligned}
$$

The term $\langle g_\mu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle$ vanishes in the case of a factorizable distribution and when the functions g_μ are component-wise.

$$
\begin{aligned}
\langle g_\mu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle &= \int_{\mathbb{R}^p} g_\mu(z_1,\cdots,z_p)\phi(z_\nu)\rho(\mathrm{d}\boldsymbol{z}) \\
&= \int_{\mathbb{R}^p} g_\mu(z_\mu)\phi(z_\nu)\prod_{\gamma=1}^{p}\rho_\gamma(\mathrm{d}z_\gamma) \\
&= \int_{\mathbb{R}} g_\mu(z_\mu)\rho_\mu(\mathrm{d}z_\mu)\int_{\mathbb{R}}\phi(z_\nu)\rho_\nu(\mathrm{d}z_\nu) \\
&= \langle g_\mu(z_\mu)\rangle\langle\phi(z_\nu)\rangle
\end{aligned}
$$

As established in the previous section, in the case of the network studied in this work, $\langle g_\mu(z_\mu)\rangle = 0$.

The term $\langle g_\nu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle$ is more difficult, but we can prove that in our case it equals one.

$$
\begin{aligned}
\langle g_\nu(\boldsymbol{z_j})\phi(z_{\nu j})\rangle &= \langle\frac{\phi(z_{\mu j}) - \langle\phi(z_{\mu j})\rangle}{\mathrm{Var}[\phi(z_{\mu j})]}\phi(z_{\mu j})\rangle \\
&= \langle\frac{\phi(z)^2 - \langle\phi(z)\rangle\phi(z)}{\mathrm{Var}[\phi(z)]}\rangle \\
&= \frac{\langle\phi(z)^2\rangle - \langle\langle\phi(z)\rangle\phi(z)\rangle}{\mathrm{Var}[\phi(z)]} \\
&= \frac{\langle\phi(z)^2\rangle - \langle\phi(z)\rangle^2}{\mathrm{Var}[\phi(z)]} \\
&= \frac{\mathrm{Var}[\phi(z)]}{\mathrm{Var}[\phi(z)]} \\
&= 1
\end{aligned}
$$

Therefore, the set of $h_i^\star = F_{\nu i}$ for all $\nu = 1, \cdots, p$ are fixed points in the limit of large N, under the assumptions that the distribution is factorizable, the functions g_μ are component-wise, and $\langle g_\nu(z)\phi(z)\rangle = 1$, which is the case in the setup of this work.

A.2 Fixed points of the neural field

The derivation of the fixed points for the neural field equation follows the same arguments as the fixed points in the network of neurons.

As $N \to \infty$, the fixed points $h_i^\star$ become functions $h^\star(z_1, \cdots, z_p)$ on the $\mathbb{R}^p$ space:

- $h_i^\star = 0 \to h^\star(z_1, \cdots, z_p) = 0$
- $h_i^\star = F_{\nu i} = z_{\nu i} \to h^\star(z_1, \cdots, z_p) = z_\nu$ for all $\nu = 1, \cdots, p$

A.3 Stability of the fixed points

We now study the stability of the fixed points. In the evolution equation, we substitute $h_i(t) = h_i^\star + \delta_i(t)$, where $\delta_i(t)$ is a small perturbation.

$$\begin{aligned}
\dot{h}_i(t) &= \dot{\delta}_i(t) \\
&= -(h_i^\star + \delta_i(t)) + \sum_{j=1}^{N} J_{ij}\phi(h_j^\star + \delta_j(t)) \\
&= -(h_i^\star + \delta_i(t)) + \sum_{j=1}^{N} J_{ij}(\phi(h_j^\star) + \partial\phi(h_j^\star)\delta_j(t) + \mathcal{O}(\delta_j(t)^2)) \\
&= \underbrace{-h_i^\star + \sum_{j=1}^{N} J_{ij}\phi(h_j^\star)}_{=\,0 \text{ by definition of the fixed point}} -\delta_i(t) + \sum_{j=1}^{N} J_{ij}\partial\phi(h_j^\star)\delta_j(t) + \mathcal{O}(\|\boldsymbol{\delta}(t)\|^2) \\
&= -\delta_i(t) + \sum_{j=1}^{N} J_{ij}\partial\phi(h_j^\star)\delta_j(t) + \mathcal{O}(\|\boldsymbol{\delta}(t)\|^2) \\
&= \sum_{j=1}^{N} \underbrace{(-\mathrm{Id}_{ij} + \underbrace{J_{ij}\partial\phi(h_j^\star)}_{J'_{ij}})}_{K_{ij}}\delta_j(t) + \mathcal{O}(\|\boldsymbol{\delta}(t)\|^2)
\end{aligned}$$

In matrix notation $\boldsymbol{\delta}(t) = (\delta_1(t), \cdots, \delta_N(t))$, this becomes

$$\begin{aligned}\dot{\boldsymbol{\delta}}(t) &= K\boldsymbol{\delta}(t)\\ \implies \boldsymbol{\delta}(t) &= \exp(Kt)\boldsymbol{\delta}(0)\end{aligned}$$

The fixed point is stable if $\boldsymbol{\delta}(t) \to 0$ as $t \to \infty$, in other words, if all the eigenvalues of K are negative.

Before we study the spectrum of K, we write the following result: if λ is an eigenvalue of A, then $\lambda - 1$ is an eigenvalue of $A - \mathrm{Id}$. This follows simply from the equation for the eigenvalues λ_i and eigenvectors $\boldsymbol{v_i}$ of A. If $A\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i}$, then $(A - \mathrm{Id})\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i} - \boldsymbol{v_i} = (\lambda_i - 1)\boldsymbol{v_i}$. Therefore, to study K, it suffices to study the eigenvalues of J'.

Since J' is low-rank, its image is spanned by the vectors $\{\boldsymbol{F_\nu} = (F_{\nu 1}, \cdots, F_{\nu N}) | \nu = 1, \cdots, p\}$, which motivates the Ansatz of eigenvectors $\boldsymbol{v_\nu} = \boldsymbol{F_\nu}$ in the $N \to \infty$ limit.

$$\begin{aligned}(J'\boldsymbol{F_\nu})_i &= \frac{1}{N}\sum_{\mu=1}^{p}\sum_{j=1}^{N} F_{\mu i}\,\underbrace{G_{\mu j}\partial\phi(h_j^\star)}_{G'_{\mu j}}\,F_{\nu j}\\ &= \frac{1}{N}\sum_{\mu\neq\nu}^{p}\sum_{j=1}^{N} F_{\mu i}G'_{\mu j}F_{\nu j} + \frac{1}{N}\sum_{j=1}^{N} F_{\nu i}G'_{\nu j}F_{\nu j}\\ &\overset{N\gg 1}{\approx} \sum_{\mu\neq\nu}^{p} F_{\mu i}\langle g'_\mu(\boldsymbol{z})f_\nu(\boldsymbol{z})\rangle + F_{\nu i}\langle g'_\nu(\boldsymbol{z})f_\nu(\boldsymbol{z})\rangle\end{aligned}$$

We study each term separately. Additionally (and similarly as before), we assume that the functions g_μ and the functions f_μ are component-wise, and that the distribution ρ factorises.

$$\begin{aligned}\langle g'_\mu(\boldsymbol{z})f_\nu(\boldsymbol{z})\rangle &= \int_{\mathbb{R}^p} g'_\mu(\boldsymbol{z})f_\nu(\boldsymbol{z})\rho(\mathrm{d}\boldsymbol{z})\\ &= \int_{\mathbb{R}}\int_{\mathbb{R}} g'_\mu(z_\mu)f_\nu(z_\nu)\rho_\mu(\mathrm{d}z_\mu)\rho_\nu(\mathrm{d}z_\nu)\\ &= \int_{\mathbb{R}} g'_\mu(z_\mu)\rho_\mu(\mathrm{d}z_\mu)\int_{\mathbb{R}} f_\nu(z_\nu)\rho_\nu(\mathrm{d}z_\nu)\\ &= \langle g'_\mu(z_\mu)\rangle\langle f_\nu(z_\nu)\rangle\end{aligned}$$

In our case, $\langle f_\nu(z_\nu)\rangle = \langle z_\nu\rangle = \int_{\mathbb{R}} z_\nu \mathcal{N}(\mathrm{d}z_\nu) = 0$ and so the term vanishes. Therefore, we are left with

$$
\begin{aligned}
(J'\boldsymbol{F_\nu})_i &\overset{N\gg 1}{\approx} F_{\nu i}\langle g'_\nu(z_\nu) f_\nu(z_\nu)\rangle \\
\implies J'\boldsymbol{F_\nu} &= \langle g'_\nu(z_\nu) f_\nu(z_\nu)\rangle \boldsymbol{F_\nu} \\
\implies K\boldsymbol{F_\nu} &= (\underbrace{\langle g'_\nu(z_\nu) f_\nu(z_\nu)\rangle - 1}_{\text{eigenvalue } \lambda_\nu})\boldsymbol{F_\nu}
\end{aligned}
$$

The term $\langle g'_\nu(z_\nu) f_\nu(z_\nu)\rangle$ does not vanish in general because of the coupling between the components. The integral is too complicated to be analytically solved, but we provide a numerical estimation in our case ($\rho(z_1, \cdots, z_p) = \prod_{\mu=1}^{p} \mathcal{N}(z_\mu)$, $f_\mu(z_\mu) = z_\mu$, $g_\mu(z_\mu) = \tilde{\phi}(z_\mu)$) for fixed points $h_i^\star = 0$ and $h_i^\star = F_{\nu i}$.

$$
\begin{aligned}
\phi(h) &= \frac{1}{1 + \mathrm{e}^{-h}} \\
&\implies \begin{cases} h_i^\star = 0 & \implies \lambda_\nu = \langle \tilde{\phi}(z)\partial\phi(0) z\rangle - 1 \approx 0.19061 \pm 0.00014 \\ h_i^\star = F_{\nu i} & \implies \lambda_\nu = \langle \tilde{\phi}(z)\partial\phi(z) z\rangle - 1 \approx -0.28090\text{ś}0.00006 \end{cases}
\end{aligned}
$$

When ϕ is the logistic function, the pattern fixed points are stable, whereas the zero fixed point is unstable.

We now consider a Taylor expansion of the activation function $\phi(h)$ up to order 3, and cite results obtained from symbolic computation and known moments of the Gaussian distribution.

$$\phi(h) = c_0 + c_1 h,\ c_0, c_1 \in \mathbb{R}^2$$
$$\implies \lambda_\nu = \langle \frac{z}{c_1} c_1 z \rangle - 1 = 0 \text{ for any fixed point}$$
$$\phi(h) = c_0 + c_1 h + c_2 h^2,\ c_0, c_1, c_2 \in \mathbb{R}^3$$
$$\implies \begin{cases} h_i^\star = 0 & \implies \lambda_\nu = \langle \frac{c_1 z + c_2 z^2 - c_2}{c_1^2 + 2c_2^2}(c_1) z \rangle - 1 \\ & \quad = -\frac{2c_2^2}{c_1^2 + 2c_2^2} \\ h_i^\star = F_{\nu i} & \implies \lambda_\nu = \langle \frac{c_1 z + c_2 z^2 - c_2}{c_1^2 + 2c_2^2}(c_1 + 2c_2 z) z \rangle - 1 \\ & \quad = \frac{2c_2^2}{c_1^2 + 2c_2^2} \end{cases}$$
$$\phi(h) = c_0 + c_1 h + c_2 h^2 + c_3 h^3,\ c_0, c_1, c_2, c_3 \in \mathbb{R}^4$$
$$\implies \begin{cases} h_i^\star = 0 & \implies \lambda_\nu = \langle \frac{c_1 z + c_2 z^2 - c_2 + c_3 z^3}{c_1^2 + 6c_1c_3 + 2c_2^2 + 15c_3^2}(c_1) z \rangle - 1 \\ & \quad = \frac{-3c_1c_3 - 2c_2^2 - 15c_3^2}{c_1^2 + 6c_1c_3 + 2c_2^2 + 15c_3^2} \\ h_i^\star = F_{\nu i} & \implies \lambda_\nu = \langle \frac{c_1 z + c_2 z^2 - c_2 + c_3 z^3}{c_1^2 + 6c_1c_3 + 2c_2^2 + 15c_3^2}(c_1 + 2c_2 z + 3c_3 z^2) z \rangle - 1 \\ & \quad = \frac{2 \cdot (3c_1c_3 + c_2^2 + 15c_3^2)}{c_1^2 + 6c_1c_3 + 2c_2^2 + 15c_3^2} \end{cases}$$

What a mess! We summarize the results from the Taylor expansion:

- When ϕ is linear, we see that all the eigenvalues are zero, therefore the first order stability is inconclusive. Simulations show when ϕ is linear, the convergence to the fixed points is very slow (the spectrum is negative, but the eigenvalues λ_ν are weakly negative, and tend to zero when $N \to \infty$).
- When ϕ is quadratic, the zero fixed point is stable, and the pattern fixed point is unstable, regardless of the sign of the coefficients.
- When ϕ is cubic, the sign of the eigenvalues depends primarily on c_3 (negative c_1 breaks the assumption that ϕ must be monotonically increasing). The intuition is that we require c_1 to be large enough (in other words, the growth of the firing rate when h becomes positive) and $c_3 < 0$ such that $3c_1c_3 + 15c_3^2 < 0$. The intuition behind this is that if the slope c_1 is stronger, then the bump of $\partial\phi$ becomes "more concentrated", which helps lower the eigenvalues associated with the pattern fixed points. The magnitude of the second order term c_2 only contributes to adding instability to the pattern fixed point, and stability to the zero fixed point, which leads to the conclusion it is best set to $c_2 = 0$ if we want the network to be stable at the pattern fixed points.

The study of the Taylor expansion therefore instructs us that stability is improved when ϕ grows rapidly around $h = 0$, and that even powers (symmetries of ϕ around the y-axis) deteriorate the stability of the pattern fixed point. This helps us understand why the logistic function $\phi(h) = \frac{1}{1+\mathrm{e}^{-h}} = \frac{1}{2} + \frac{h}{4} - \frac{h^3}{48} + \mathcal{O}(h^5)$ has good convergence properties: we have that c_1 is large enough, c_3 is negative, and c_2 is zero.

We finish the study of eigenvalues by noting that in the $N \to \infty$ limit, the remaining $N - p$ eigenvalues of J' are zero. The corresponding eigenvectors span the orthogonal complement of the vectors $\boldsymbol{G'_\nu} = (G'_{\nu 1}, \cdots, G'_{\nu N})$, formally $\boldsymbol{v} \in \mathrm{span}(\boldsymbol{G'_1}, \cdots, \boldsymbol{G'_p})^\perp$.

In conclusion, the spectrum of K is composed of p eigenvalues $\lambda_\nu = \langle g_\nu(z_\nu) f_\nu(z_\nu) \partial\phi(h^\star) \rangle - 1$, and $N - p$ eigenvalues $\lambda_\perp = -1$. For the network considered in this work, the interpretation is that components of the field aligned with the patterns z_ν converge to the fixed point, and the orthogonal components decay as $t \to \infty$. The fixed point $h_i^\star = 0$ is therefore unstable, and the pattern fixed points $h_i^\star = z_{\nu i}$ are stable. We finally note that since the normal distribution is centred around zero and symmetric, $h_i^\star = -z_{\nu i}$ are also stable fixed points.

B Binned sampling of populations and visualization of kernels

B.1 Mean field derivation

Let us start by taking N samples of the distribution $\rho(z_1, z_2)$. As established in Section 2.2.1, this defines a network of neurons $i \in \{1, \cdots, N\}$ with connectivity matrix $J_{ij} = w_U(\boldsymbol{v_i}, \boldsymbol{v_j})$. We recall the evolution equation for the network of neurons:

$$\dot{h}_i(t) = -h_i(t) + \sum_{j=1}^{N} J_{ij}\phi(h_j(t)). \tag{B.1}$$

For now, we do not use the fact that in our setting, J_{ij} is low-rank. Applying the finite-n mapping S^n,[1] the neurons $\mathcal{A} = \{i_1, \cdots, i_{|\mathcal{A}|}\}$ will end up in the same 1D bin at position α. We stress that $|\mathcal{A}| \neq 2^n$ in general, because we sample the distribution, instead of applying a grid. There are, however, still 4^n 1D bins, each of length 4^{-n}.

As explained in the main text, we average the segment together in a bin, and we write the corresponding potential of the segment population at location α:

$$H_\alpha(t) = \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} h_i(t). \tag{B.2}$$

The segment populations interact via the connectivity kernel $\tilde{J}_{\alpha\beta}$, and we write the equation of evolution as:

$$\dot{H}_\alpha(t) = -H_\alpha(t) + \sum_{\beta=1}^{4^n} \tilde{J}_{\alpha\beta}\phi(H_\beta(t)). \tag{B.3}$$

[1]We are glossing over the fact that $S^n : [0,1]^2 \mapsto [0,1]$, but the positions $\boldsymbol{z_i} \in \mathbb{R}^2$. In the code, we can deal with this by either applying the inverse CDF (as described in Section 2.5), or considering a bounding box $[\min_i(z_{1,i}), \max_i(z_{1,i})] \times [\min_i(z_{2,i}), \max_i(z_{2,i})]$ of the sampled points, which is then rescaled to $[0,1] \times [0,1]$ (see implementation for `Box`). Both approaches are equivalent, in the sense that they give rise to the same dynamics as n becomes large, although the obtained connectivity matrix $\tilde{J}_{\alpha\beta}$ may not be the identical.

Since the neurons in $\mathcal{A}$ receive recurrent currents from neurons in $\mathcal{B}$, it might seem natural to take $\tilde{J}_{\alpha\beta}$ to be the average connectivity between the neurons in $\mathcal{A}$ and $\mathcal{B}$. Furthermore, the neurons in $\mathcal{A}$ receive recurrent currents from all the neurons in $\mathcal{B}$, which adds a weighing term $|\mathcal{B}|$:

$$\tilde{J}_{\alpha\beta} = |\mathcal{B}| \frac{1}{|\mathcal{B}||\mathcal{A}|} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{B}} J_{ij} = \frac{1}{|\mathcal{A}|} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{B}} J_{ij}. \tag{B.4}$$

We now verify that our intuition is correct. We substitute the definition of the mean field Equation B.2 and the our guess Equation B.4 into Equation B.3:

$$\begin{aligned}
\frac{1}{|\mathcal{A}|} \sum_{i\in\mathcal{A}} h_i(t) &= \frac{1}{|\mathcal{A}|} \sum_{i\in\mathcal{A}} h_i(t) + \sum_{\beta=1}^{4^n} \frac{1}{|\mathcal{A}|} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{B}} J_{ij}\phi(H_\beta(t)) \\
&\Longleftrightarrow \\
\sum_{i\in\mathcal{A}} h_i(t) &= \sum_{i\in\mathcal{A}} h_i(t) + \sum_{i\in\mathcal{A}} \underbrace{\sum_{\beta=1}^{4^n} \sum_{j\in\mathcal{B}}}_{=\sum_{j=1}^{N}} J_{ij}\phi(H_\beta(t)) \\
&= \sum_{i\in\mathcal{A}} h_i(t) + \sum_{i\in\mathcal{A}} \sum_{j=1}^{N} J_{ij}\phi(H_\beta(t)).
\end{aligned}$$

We now make the mean-field approximation by dropping $\sum_{i\in\mathcal{A}}$, that is, that all the neurons inside of each bin are the same (or at least, sufficiently similar so that the mean-field approximation is not a big mistake, in the sense that there is a lot of variation inside each bin, see Chapter 4). Doing so, we recover Equation B.1, which concludes the proof.

B.2 Sampling populations conserves the low-rank structure

In Equation B.4, we defined the connectivity of the binned segment populations as the average connectivity of the corresponding neurons sampled in the 2D embedding. In the case of low-rank J_{ij}, we can show the resulting $\tilde{J}_{\alpha\beta}$ also has a low-rank structure.

$$\begin{aligned}\tilde{J}_{\alpha\beta} &= \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} J_{ij} \\ &= \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}}\frac{1}{N}\sum_{\mu=1}^{p} F_{\mu i}G_{\mu j} \\ &= \frac{1}{|\mathcal{A}|N}\sum_{\mu=1}^{p}\left(\sum_{i\in\mathcal{A}} F_{\mu i}\right)\left(\sum_{j\in\mathcal{B}} G_{\mu j}\right)\end{aligned}$$

The form of the connectivity motivates the definition of *mean low-rank vectors*:

$$\tilde{F}_{\mu\alpha} = \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} F_{\mu i}, \quad \tilde{G}_{\mu\beta} = \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}} G_{\mu j}. \tag{B.5}$$

Additionally, we note that $N = \sum_{\beta'} |\mathcal{B}'|$, and substituting, we find that:

$$\tilde{J}_{\alpha\beta} = \frac{|\mathcal{B}|}{\sum_{\beta'} |\mathcal{B}'|}\sum_{\mu=1}^{p} \tilde{F}_{\mu\alpha}\tilde{G}_{\mu\beta}, \tag{B.6}$$

which shows that $\tilde{J}_{\alpha\beta}$ has a low-rank structure.[2]

Equation B.6 also nicely complements the intuition that the more neurons there are in each bin, the more "important" this bin should be. This is expressed through the weights $\frac{|\mathcal{B}|}{\sum_{\beta'} |\mathcal{B}'|}$.

We have hereby shown that sampling neurons in the 2D embeddings, and binning the correponding mappings (downsampling), yields the concept of "mean patterns" from Equation B.5.

[2]We note that if we exclude self-connections (as is done in [27]), that is we redefine $J_{ij} \mapsto (1-\mathrm{Id}_{ij})J_{ij}$, then we can show that $\tilde{J}_{\alpha\beta}$ becomes

$$\tilde{J}_{\alpha\beta} = \frac{|\mathcal{B}|}{\sum_{\beta'} |\mathcal{B}'|}\sum_{\mu=1}^{p} \tilde{F}_{\mu\alpha}\tilde{G}_{\mu\beta} - \mathrm{Id}_{\alpha\beta}\frac{|\mathcal{B}|}{\sum_{\beta'} |\mathcal{B}'|}\sum_{\mu=1}^{p}\sum_{i\in\mathcal{A}}\frac{F_{\mu i}}{|\mathcal{A}|}\frac{G_{\mu i}}{|\mathcal{A}|}.$$

We see that the correction is still of order $\mathcal{O}(\frac{1}{N})$, and therefore can be again ignored in the $N \to \infty$ limit.

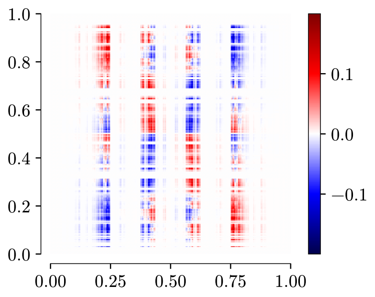
B.3 Numerical examples of methods for estimating the $[0, 1]$ connectivity kernel

For the sake of illustration, let us compare the connectivity kernels obtained using different methods in Figure B.1. These three approximations are equivalent, since they yield the same dynamics in the $N \to \infty$ limit.

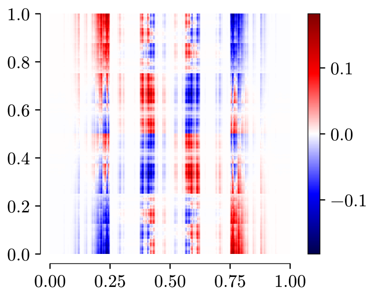
- Figure B.1a is the method described in this appendix, based on sampling N neurons in the PDF space $\mathbb{R}^2$, applying the mapping on the obtained positions $\boldsymbol{z_i}$, and finally applying the binning (this is coarse-graining). We see, through the empty rows in the connectivity, that due to finite-N effects, locations where ρ is small are not represented, and, as a consequence, no neurons are present in the corresponding 1D bin. When this happens, we set these rows to zero.
- To obtain Figure B.1b, we make the approximation that the support of $\rho(z_1, z_2)$ is compact in $[-4, 4] \times [-4, 4]$, which of course is not true, but in the case of a Gaussian distribution this is a reasonable approximation, because $\mathrm{CDF}(4) \approx 0.999968$. Defining $\tilde{\rho} = \rho \circ S^{-1}$, the grid is numerically renormalized so that despite the compact support approximation, the probabilities sum up to one, and the connectivity matrix is modified so that the columns are weighed by $\tilde{\rho}(\beta)$:

  $$\tilde{J}_{\alpha\beta} = \tilde{\rho}(\beta) \sum_{\mu=1}^{p} \tilde{F}_{\mu\alpha} \tilde{G}_{\mu\beta}.$$

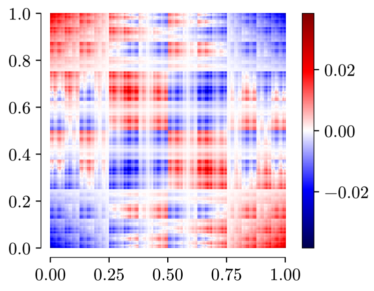
  We see that compared to Figure B.1a, using the grid method allows us to "sample" the locations where the density is small, and this results in less "whited-out" *rows*. Still, vertical bands of very fade colour correspond to the reweighing of the *columns* where $\tilde{\rho}(\beta)$ is small.
- Finally, Figure B.1c shows the kernel obtained from applying the grid method in the CDF space $[0, 1]^2$, as described in Section 2.5. We see that the connectivity seems much more "uniform", because each point of the grid has "equal weight" in the CDF space.



(a) Sampling neurons $\boldsymbol{z_i} \sim \rho(z_1, z_2)$ and binning the corresponding mappings.



(b) Grid method in a subset of the PDF space $[-4, 4] \times [-4, 4] \subset \mathbb{R}^2$.



(c) Grid method in the CDF space $[0, 1]^2$.

Figure B.1: Comparison of methods for computing the connectivity matrices $\tilde{J}_{\alpha\beta}$ approximating the kernel $\tilde{w}(\alpha, \beta)$.