Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Emil Alvar Myhre

Bayesian optimal experimental design for studying synaptic plasticity

Master's thesis in Applied Physics and Mathematics
Supervisor: Benjamin Adric Dunn
Co-supervisor: Claudia Battistin
June 2021

NTNU
Norwegian University of Science and Technology

Emil Alvar Myhre

Bayesian optimal experimental design for studying synaptic plasticity

Master's thesis in Applied Physics and Mathematics
Supervisor: Benjamin Adric Dunn
Co-supervisor: Claudia Battistin
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

The brain is the command center for the nervous system for humans, as well as for other species. *Neurons* are the fundamental cells of the brain, giving rise to the complex and powerful functioning brain that we possess by communicating through electrical and chemical signals. Learning and memory are often understood to be induced by changes in neural connections. This evolution of neural connectivity is referred to as *synaptic plasticity*. Insight and understanding of how these mechanisms are driven could be crucial within medical research for recognising and understanding neurological disorders. In this work, we approach synaptic plasticity from a mathematical perspective, aiming to devise a statistical inference framework to understand these dynamics.

This work extends on the framework presented by Linderman and coauthors [1] for studying synaptic plasticity, where the dynamics are believed to follow some underlying patterns, called *learning rules*. Specifically, employing spike-timing-dependent plasticity STDP learning rules, as suggested in [2]. This work presents a recently developed statistical inference method to infer the learning rule parameters. When applied to real data, the methods performance seems sensitive to external stimulation of the neurons and requires a lot of data to yield confident estimates, which might limit its applicability. Therefore, this work aims to develop a Bayesian optimal experimental design algorithm, which optimises the stimulation in order to minimise the amount of data required for obtaining adequate results. This is a novel approach for studying synaptic plasticity, which to our knowledge has not yet been explored.

Experiments on synthetic data show that our algorithm improves significantly on traditional stimulation protocols. The findings were formalised in an article, which could be of great interest to the plasticity community.

Sammendrag

Hjernen er senteret av nervesystemet til både oss mennesker og andre arter. *Nevroner* er de fundamentale cellene i hjernen, som gir opphav til den komplekse og kraftige hjernen som vi innehar ved å kommunisere med hverandre gjennom elektriske og kjemiske signaler. Læring og hukommelse er forstått å bli utløst av endringer i nevrale forbindelser. Slike forandringer er kjent som *synaptisk plastisitet*. Innsikt i hvordan disse mekanismene fungerer kan være essensielt innen medisinsk forskning for identisfisering og forståelse av nevrologiske lidelser. I dette arbeidet tar vi for oss synaptisk plastisitet fra et matematisk perspektiv, med mål om å utvikle et rammeverk for statistisk inferens av plastisitet i hjernen.

Dette arbeidet tar utgangspunkt i og videreutvikler et rammeverk presentert av Linderman og medforfattere [1] for å studere synaptisk plastisitet. Nevrale endringer antas å følge noen underliggende mønstre, kalt *læringsregler*. Nærmere bestemt, benytter vi såkalte STDP læringsregler, som ble foreslått i [2]. Dette arbeidet presenterer en nylig utviklet metode for statistisk inferens for å estimere parametere i læringsregelen. Imidlertig viser vi at metoden både er sensitiv til ekstern stimulering av nevronene og krever store datasett for å gi troverdige resultater, noe som setter spørsmålstegn ved anvendbarheten til metoden. Derfor sikter vi i dette arbeidet på å utvikle en Bayesisk optimal eksperiment design metode, som optimerer stimulering for å minimere datamengden som kreves for å oppnå advekate inferensresultater. Såvidt vi vet, er dette en ny tilnærming som ikke før har blitt utforsket med formål om å studere synaptisk plastisitet.

Eksperimenter gjort på syntetisk data viser at algoritmen presterer betydelig bedre enn andre normale stimuleringsprotokoller. Funnene ble også presentert som en forskningsartikkel, som kan være av stor interesse for forskningsfeltet rundt synaptisk plastisitet.

Preface

This work is my Master's Thesis - TMA4900 - in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU).

Since I was introduced to neuroscience two years ago, I really wanted to pursue my Master's within neuroscientific applications, combined with my interest in mathematics and statistics. I am really grateful for the opportunity of working on this project. I would like to express my appreciation for my supervisor Benjamin Adric Dunn, for having shaped such a good environment at the neural data science group at NTNU, and for his support and help in the process of conducting this thesis.

Moreover, this would never have been possible without my great co-supervisor Claudia Battistin. Her consistent effort of guiding me through this work, contributing with knowledge, creative ideas and help, has been crucial. I have really appreciated the collaboration, and I wish her, this project and the whole of the neural data science group nothing but success going forward.

Also, I would like to thank all the people who are important to me for continuous support. You know who you are!

Emil Alvar Myhre
June 2021

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

The study of dynamical and evolving networks is of great interest within various research fields. In general, one could imagine a network where internal components (or nodes) might interact and influence each other. At the same time, the network as a whole could be characterised by some state, which somehow describes the current network dynamics. These dynamics, however, might change over time due to either internal or external factors. These are flexible and complex systems that are applicable for many purposes and could describe many different dynamics that we face in real life.

Let's showcase this with a rather big and complex social example. Most countries in the world have a democracy of some sort [3]. Let us now consider this political system as a network, and briefly break down the dynamics. Well, we have a lot of different components in such a system: 1) politicians, 2) voters, 3) the press, 4) institutes/businesses, to mention a few. These are all allowed to interact, which leads to possible influence from each other, or voters could even be influenced by some external event in another country. If the dynamics within the population change enough, the political system could also change. That is, some other party could come to power, and the "state" of the network could change.

Similar networks might be constructed for other time-dependent systems, for instance, the stock market, the spread of a pandemic, social networks or infrastructural networks. However, in this work, we will focus on a super interesting biological application of such networks: **the brain!**

The brain consists of interconnected neural cells that communicate with each other, causing our body to function as it does. How these neural cells are connected can be considered a network. Furthermore, research has shown that such neural networks are non-stationary and that connectivity dynamics are induced by perceptual experiences[4]. In several areas of the brain, these dynamics have also been shown to follow some underlying patterns, i.e. functions on a given form[5]. Dependent on the state of the brain, these functions could look different and serve as a signature for neurological conditions. Hence, being able to characterise these patterns could potentially enable us to detect and understand neurological disorders or specific substances affecting the brain.

The first part of this work will present a statistical inference method for capturing these neural dynamics, motivated by Linderman and coauthors [1]. The method will be probed extensively on both real and synthetic data, from which we observe challenges this method faces in experimental settings. Prompting to optimise this method, our main contribution consists of a Bayesian experimental design approach, where we attempt to optimise the ex-

perimental setting in order to maximise the level of performance of the proposed inference method.

In chapter 2, we will introduce basic concepts in Neuroscience and relevant terminology in order to motivate the work and provide the reader a sufficient biological understanding. In chapter 3, the most relevant mathematical tools for this work will be defined and explained. In chapter 4, we will present the specific model and all the methods we employ with necessary implementation details. Chapter 5 will be dedicated to all of our results, and in chapter 6 our findings will be further discussed, limitations will be addressed and future prospects will be debated. We will concisely conclude our work and our findings in chapter 7. Additionally, this thesis consists of two appendices. The source code from which all of our simulations were conducted is found in appendix A. This includes both the inference method and the Bayesian experimental design. Lastly, the work has resulted in a preprint, which is available in appendix B.

Chapter 2

Concepts in Neuroscience

In this section, we will present some basic concepts within the field of neuroscience, giving insight into how the brain works on microscopic and macroscopic levels, which will be helpful for the reader in order to understand the work of this paper. Section 2.1 will consist of a biological presentation of the brain and the neural dynamics. In section 2.2, we will briefly present how the activity of neurons can be expressed, and touch on mathematical ways to model this activity. In seciton 2.3, we will discuss the underlying patterns that make new connections in the brain emerge, which is known as synaptic plasticity. Common experimental methods for studying these dynamics will also be presented. Lastly, in section 2.4, we will focus on one of many possible medical and societal applications, for which research on synaptic plasticity might be crucial, namely Alzheimer 's disease.

2.1 Neurons

A neural cell, from now on only referred to as a *neuron*, is the fundamental component of the brain. The human brain is believed to consist of ~ 100 billion neurons [6]. A single neuron consists of three main parts [7]:

- a cell body, also called *soma*
- *dendrites*
- an *axon*

As depicted in figure 2.1, the neurons are interconnected and can communicate with each other by sending electrical signals to other neurons. These signals are generated at the soma, and are referred to as *action potentials*, *spikes* or the neuron *firing*. Exactly when an action potential occurs in a neuron, depends on the *membrane potential* of the neuron. The membrane potential is defined as the difference in electrical potential between the inside and the outside of the cell, and is measured in voltage. If this voltage increases enough and hits a certain threshold, an action potential is triggered and the neuron spikes. Then this signal travels through the axon of the neuron, and connected neurons receive this signal through their dendrites. This connection between an axon of one neuron and a dendrite of another is called a *synapse*, and is illustrated in figure 2.1.

The neuron sending the signal is referred to as a *presynaptic neuron*, while the neuron receiving the signal is called a *postsynaptic neuron*. These synapses are one of the reasons for

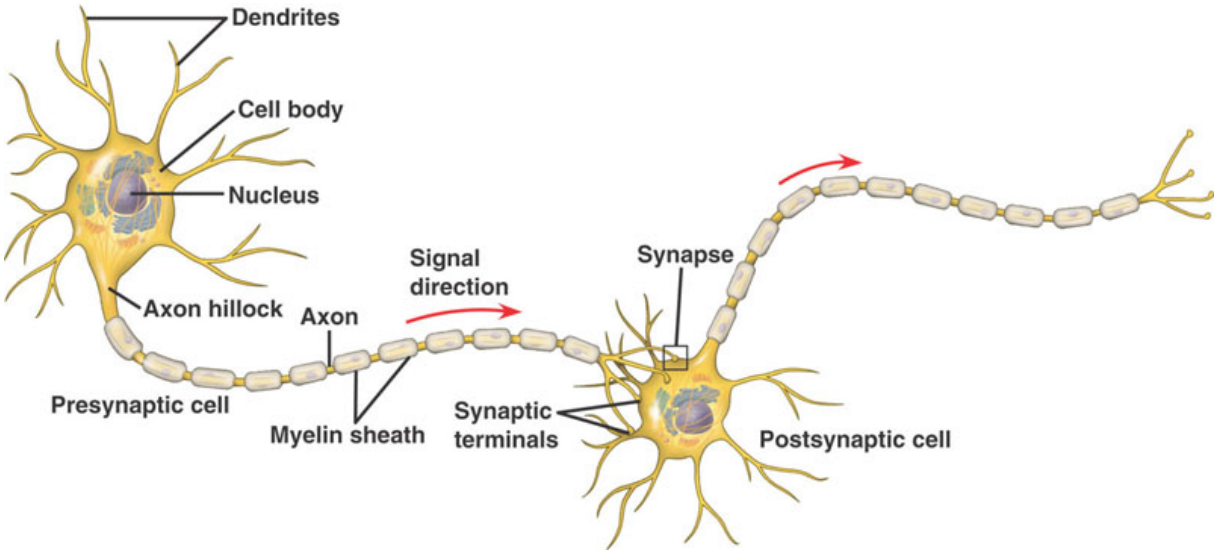

Figure 2.1: A picture showcasing two neurons and how they are connected through a synapse [8].

the constant evolution of the membrane potential in a neuron. An electrical signal following a spike from a presynaptic neuron affects the current voltage in the postsynaptic neuron. The membrane potential could either increase, in which case the synapse is called an *excitatory synapse*. However, the membrane potential could also decrease as a result of the spike, in which case the synapse is called an *inhibitory synapse*. Hence, if a neuron is exposed to a lot of excitatory inputs, the membrane potential will rise, which might result in an action potential if the voltage threshold is reached. Simultaneously, the neuron might be stimulated from other random sources, for instance other parts of the brain, which also could affect the membrane potential of the neuron. This is often referred to as *noise*. Another source of noise is the stochastic opening of ion channels on the neural membrane. Figure 2.2 shows how the membrane potential of a neuron could change because of some stimulus, either from connected neurons or from noise, and when the threshold is reached, a spike is triggered.

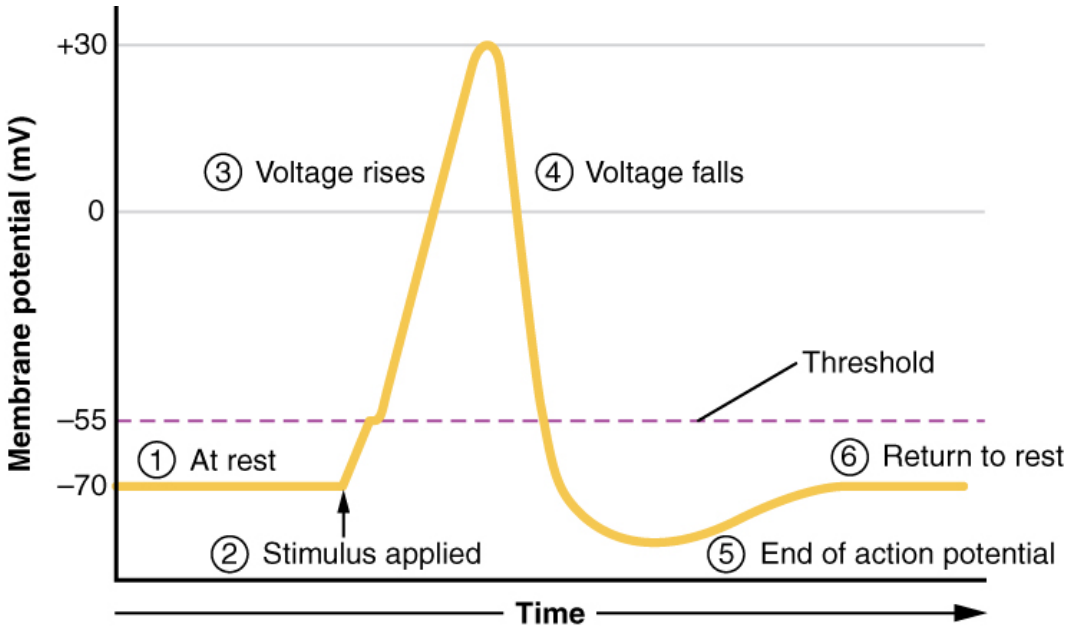

Figure 2.2: Dynamics of the membrane potential with a spike occuring [9].

2.2 Spike trains

The spiking events of a neuron, are essential when modelling and analysing the neural activity, as the time course of the membrane potential during an action potential is pretty much stereotyped. A functioning neuron will repeatedly fire, leading to a sequence of spikes. If we consider a certain time interval, all the spike times recorded from a neuron form what we refer to as a *spike train*. Assuming a neuron spikes n times within the time interval $[0, T]$, the spike train for this neuron can be expressed compactly as

$$\{t^{(i)}\}_{i=1}^{n} = \{t^{(1)}, t^{(2)}, ..., t^{(n)}\}, \quad t^{(i)} < t^{(i+1)} \; \forall i, \quad t^{(i)} \leq T \; \forall i$$

where $t^{(i)}$ is the spiking time of spike number i of the neuron. A visualisation of four spike trains is provided in figure 2.3. These spike trains are a selection of neurons from a data set of real neural recordings[10]. Here we observe neural activity of different nature, where neuron 3 has the significantly highest *firing rate*. Neuron 2 shows tendency of *burstiness*, meaning that in shorter time intervals it has a high firing rate, before being totally inactive for longer periods. These spiking times will be essential for this work. Having access to spike trains

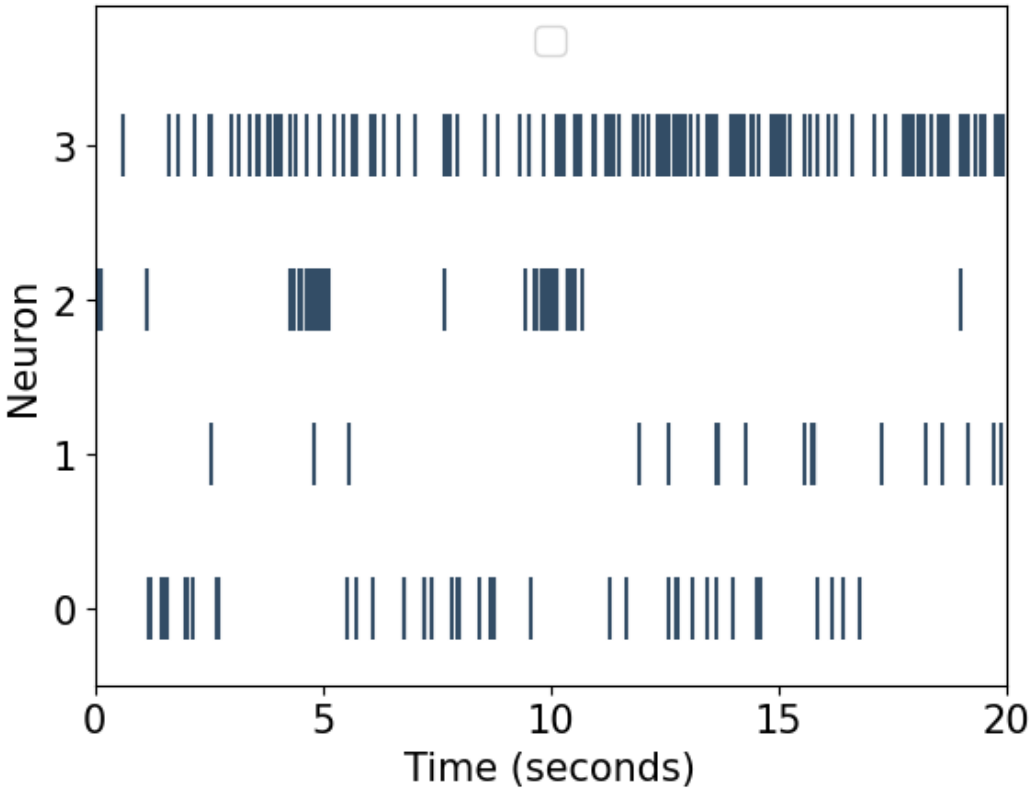

Figure 2.3: Four example spike trains where the blue lines illustrates the spike times.

from several connected neurons could be helpful to understand how they interact with each other. There exists many mathematical frameworks for modelling this activity. One common type of models are the flexible *integrate-and-fire (IF)* models[11]. Also, models can be of very different complexity, ranging from advanced methods modelling specific ionic channels to more simple binary models[12], which we will apply in this work.

2.3 Synaptic plasticity

Now that we know the basic dynamics of a single neuron, let us advance to the more interesting part: **a network of neurons**.

Initially in this chapter, the brain was presented as a complex and dynamic network, consisting of interconnected neurons that communicate with each other. A deeper understanding of these dynamics is what we strive for. As we know, neurons might be connected through either excitatory or inhibitory synapses. Furthermore, the strength of these synapses might be of different magnitude, and they might change over time. These changes are referred to as *synaptic plasticity*.

One century of research on synaptic plasticity indicates that these changes are activity-dependent and that they follow some patterns[13], which will be referred to as *learning rules*. These rules are dictating how the brain develops. If we would be able to identify or characterise these learning rules, we could have a much deeper understanding of the synaptic plasticity. In order to gain such knowledge, spike train information from several neurons is crucial. Imagine that the spike trains of two neurons are almost identical, that is, the neurons spike at almost the same times. Then it would be reasonable to assume some correlation between their activities, indicating a strong synaptic connection. In this work we will consider *spike-timing-dependent plasticity* (STDP) learning rules [14]. These rules use exactly such information about the spike times of connected neurons to determine the evolution of the network dynamics. Furthermore, STDP in particular has been shown to serve for storing sequences [15] in artificial neural networks, so it is becoming a popular topic in the machine learning community as well [16].

2.3.1 Experimental methods for studying plasticity

To study synaptic plasticity, electrophysiological stimulation has played an important part in being able to induce both long term potentiation (LTP) and long term depression (LTD) of synapses. Several different protocols have been developed and applied for this purpose. Already in 1973, Bliss and Lømo employed high-frequency stimulation of 100Hz of duration $1-5$ seconds to induce LTP[17]. In the work of Albensi and coauthors (2007)[18], a wide selection of common practices in experimental neurology is discussed. Whether one wants to study LTP or LTD, usually different protocols should be used. In general, the most common frequencies for *in vivo* stimulation range all the way up to 400Hz[18]. Even though one usually applies such stimulus in short bursts of a couple of seconds, experiments have been conducted where higher frequencies up to 200Hz have been used for longer recordings up to 1 minute[19]. In this thesis, we will develop frameworks that comply with these common practices, such that they could potentially be implemented in real life experiments.

2.4 Alzheimer's disease

A better understanding of synaptic plasticity could further be essential for the understanding of diseases like Alzheimer's. Synaptic plasticity is thought to be an essential mechanism for learning and for memory encoding [20] [21], functions which are dramatically impaired in Alzheimer's disease. The memory-related disorder progresses by causing brain cells to degenerate and die [22], which causes a decline in the ability to think and remember. Alzheimer's disease poses major challenges to both the diseased and society as a whole. As

the disease is closely related to age, we can expect it to become increasingly common, as the life expectancy increases and is not even approaching a plateau yet [23].

There is currently no treatment that can stop the development of Alzheimer's completely. However, there are methods for slowing the worsening of the symptoms, thereby improving the quality of life for both patients and their relatives. It would therefore be a great advantage if one could identify the disease as early as possible. The challenges Alzheimer's disease poses have inspired a ton of research, aiming to both prevent and discover the disease early on.

An Alzheimer's patient will experience significant brain shrinkage, caused by the loss of synapses and neurons. The exact cause of this loss is still being researched extensively, but are suggested to be caused by an accumulation of amyloid plaques [24], which in turn impairs the neural activity, interactions and then plasticity. By studying the synaptic plasticity properties of brains, it might be possible to distinguish between a healthy brain and the brain of an Alzheimer's patient.

In research, these studies are often conducted on rats or mice, as their brains have sufficient similarities to human brains. More specifically, the most heavily damaged cortex (the entorhinal cortex) [25] in a brain with Alzheimer's is phylogenetically conserved across species [26], so studying this cortex in rat brains can be beneficial to humans as well.

Chapter 3

Theory

In this chapter, we will provide some relevant mathematical concepts, all of them being important for the upcoming applications in this work. Along the way we will also conceptually connect these mathematical ideas to the situation of interest, namely synaptic plasticity in the brain.

In section 3.1, the idea of Markov chains is presented, this part is based on the book *Introduction to probability models* [27]. More specifically, we introduce a Hidden Markov Model (HMM) which describes the dynamics of connectivity in our network of neurons. In section 3.2, we define the concept of Generalised linear models (GLM) and how this looks like for the Bernoulli processes, which is used for modelling neural activity. In section 3.3 we discuss different approaches to estimate parameters, which is essential for uncovering the underlying learning rules. Furthermore, we describe a method for identifying putative connections between pairs of neurons in section 3.4, which is useful when working on real data. In section 3.5, we explain a sampling technique to estimate probability distributions, which is the basis of several approximations in our developed framework. Moreover, our main contribution of Bayesian optimal design is built on concepts within information theory, that are presented in section 3.6. Lastly, underlying theory and relevant literature of the Bayesian optimal design itself is covered in section 3.7.

3.1 Markov Chains

Let X_n be a stochastic process in discrete time. In principle, the process X_n can take either finite, countable or continuous values. For now, let's assume a countable state space for simplicity. We therefore denote these possible values with indices $\{0,1,2,...\}$, and call them *states*. If $X_n = i$, the process is in state i at time n. Then, given that the process is in state i at time n, there is a certain probability that the process is in state j at time $n+1$. This probability of transitioning from state i to j is called the *transition probability*, denoted by $P_{ij} \in [0,1]$. Transition probabilities exists for all set of state pairs i and j. The probability of remaining in the current state would be P_{ii}.

What characterises a Markov chain is the *Markov property*, which says that the next transition of the process only depends on the current state, regardless of the whole history of the

process. That is

$$P\Big(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0\Big) = P\Big(X_{n+1} = j | X_n = i\Big) = P_{ij}.$$

Since all transition probabilities are defined to be non-negative and the process evolves in time by necessarily making a transition, the sum of all transition probabilities from a given state i must be 1,

$$\sum_{j=0}^{\infty} P_{ij} = 1, \quad i = 0, 1, ...$$

All of these probabilities can be structured in a matrix $\boldsymbol{P}$, where the entry $\boldsymbol{P}[i, j]$ indicates the transition probability P_{ij}. An example of a 3-state Markov chain with transition probabilities is showcased in figure 3.1.

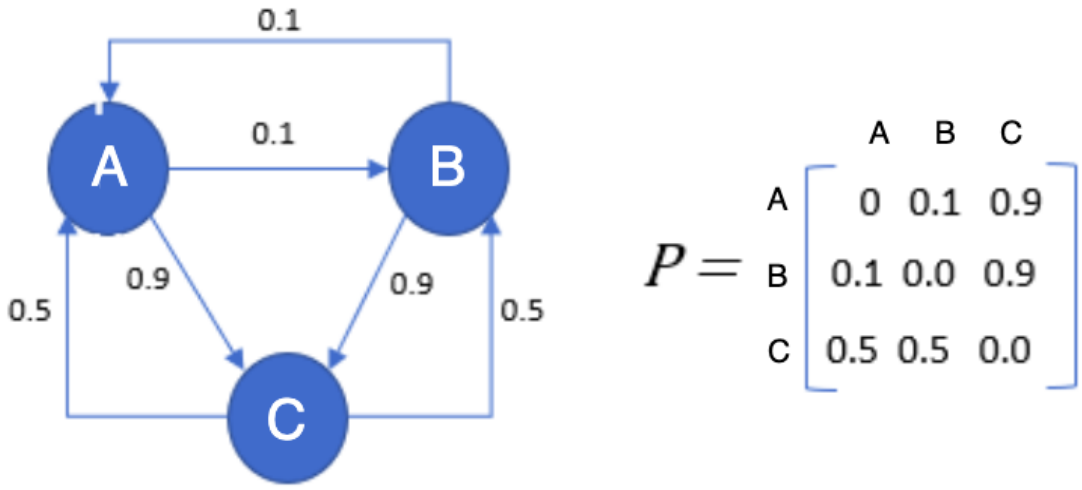

Figure 3.1: An example of a Markov chain with 3 possible states (left) and the corresponding transition probability matrix (right) [28].

3.1.1 Hidden Markov Model

Now we will introduce a more sophisticated model, which will be useful for modelling the dynamics of the connectivity in our network of neurons. Now let $X_n = \{X_1, X_2, ...\}$ be a Markov process as explained in section 3.1. In addition, let's now include another stochastic process, which is statistically dependent on the state of the Markov process X_n. Let's further assume, that we do not have any information about the states of the Markov process, hence the Markov Chain is called *hidden*. What we do have, however, is a set of observations $Y = \{Y_1, Y_2, ...\}$, from the second stochastic process. We also know, that they are dependent on X_n, so at time t, we consider the conditional probability for the observation, which follows the density

$$Y_t | X_t \sim P(Y_t | X_t). \tag{3.1}$$

We also know that X_n follows the Markov property, and is hence dependent on the previous state, that is, X evolves according to some density

$$X_{t+1} | X_t, X_{t-1}, ..., X_1 \sim P(X_{t+1} | X_t). \tag{3.2}$$

Such a model, where the sequence of underlying Markov chain states are unknown, while we have a set of correlated observations, is called a *Hidden Markov Model* (HMM). How would this correspond to the brain?

Brain interpretation

Well, in the biological system of neurons with non-stationary connectivity we are studying, imagine the following interpretation, using the notation of this section

- X_t would be the state of the neural network at time t. That is, how the neurons are connected: in terms of which ones are connected and how strong the synapses are.
- Y would be the observed spiking times of the neurons, see section 2.2. We say observed, because this is actually the data we have at our disposal.

The state of the network, X_t, is unknown in a real setting, but we assume that it changes over time according to the Markov property, corresponding to Eq. (3.2). Furthermore, when the neurons spike also depends on the connections between them, corresponding to Eq. (3.1).

Building on this concept of a general HMM, in the brain we assume that the density $P(X_{t+1}|X_t)$ is determined by our specific learning rule, which will be mathematically defined in section 4.1.2. As previously mentioned, this learning rule depends on the whole spike history of the neurons, being Y. In addition, it features further parameters, θ, that have to be inferred and determined. The state transition depends on the previous state, the spike times and the parameters, so the density can be expressed as

$$X_{t+1}|X_t \sim P(X_{t+1}|X_t, Y, \theta). \tag{3.3}$$

We will explain the model in more detail in section 4.1.1 and 4.1.2.

3.2 Generalised linear models

Now we will explore a generalisation of linear regression models, namely Generalised linear models, which we will refer to as GLMs. This section is based on the book *Regression* [29]. The standard linear regression for a response y and a vector of covariates $\boldsymbol{x}$ can be expressed through the normal distribution,

$$y \sim \mathcal{N}(\boldsymbol{x}^T\boldsymbol{\beta}, \sigma^2), \tag{3.4}$$

with $\boldsymbol{\beta}$ being the vector of regression coefficients and σ^2 being the variance of some Gaussian white noise.

However, if the response variable has some restrictions, the normality assumption for the response might be violated. The GLM framework is therefore useful, as it allows response variables to originate from different distributions than the normal one, and it allows for different functional dependencies of the mean of the response variable on the linear predictor, $\eta = \boldsymbol{x}^T\boldsymbol{\beta}$.

3.2.1 General GLM

The response variable y in the GLM framework can be drawn from a univariate exponential family with density function on the form

$$f(y|\theta) = \exp\left(\frac{y\theta - b(\theta)}{\phi} \cdot w + c(y,\phi,w)\right) \tag{3.5}$$

with $b(\theta)$ and $c(y,\phi,w)$ being known functions. θ is called the *canonical parameter*, ϕ is a so-called nuisance parameter and w is a weight function. The normal distribution on standard form can be retrieved from Eq. (3.5) by inserting $\theta = \mu$, $b(\theta) = \frac{\theta^2}{2}$, $\phi = \sigma^2$, $w = 1$ and $c(y,\phi,w) = -\frac{y^2}{2\phi} - \frac{1}{2}\ln\left(2\pi\phi\right)$.

Additionally, distributions like the Poisson, gamma and Bernoulli can also be written on the form of Eq. (3.5).

From Eq. (3.5), one can find the expected value and variance of the response variable y in the following way

$$\begin{aligned} \mathbb{E}[y] &= \mu = b'(\theta) \\ \mathrm{Var}[y] &= \sigma^2 = b''(\theta)\frac{\phi}{w}. \end{aligned} \tag{3.6}$$

For a normally distributed response, we see from Eq. (3.4), that the mean is equal to the linear predictor. However, in general we do not want to model the mean directly, but instead through a *link function*, $g(\mu)$, which maps the mean to the linear predictor

$$g(\mu) = \eta. \tag{3.7}$$

Since for the normal distribution $\mu = \eta$, the link function is the identity function, $g(\mu) = \mu$. If the link function also equals the canonical parameter θ, that is

$$g(\mu) = \theta,$$

then $g(\mu)$ is called a *canonical link function*. Other examples of canonical link functions are the logarithm function for Poisson distributions and the logit function for Bernoulli distributions, which we extend on in section 3.2.2.

3.2.2 GLM for a Bernoulli process

When we model the spikes of neurons, the event of spiking at each time step is a Bernoulli distributed response variable, receiving input stimulus from other neurons and background noise. This can be modelled as a GLM, with the input being modelled through a linear predictor.

A Bernoulli variable takes the values $\{0,1\}$ with probabilities $\{1-\mu,\mu\}$ respectively. The probability density function for a Bernoulli variable is

$$f(y|\mu) = \mu^y(1-\mu)^{1-y}, \tag{3.8}$$

which can be rewritten in the GLM framework given in Eq. (3.5) as follows

$$f(y|\mu) = \exp\left(y \cdot \ln\left(\frac{\mu}{1-\mu}\right) + \ln\left(1-\mu\right)\right). \tag{3.9}$$

Without proof, comparison with Eq. (3.5) gives that

$$\begin{aligned} \theta &= \ln\left(\frac{\mu}{1-\mu}\right) \\ b(\theta) &= \ln\big(1+\exp(\theta)\big), \end{aligned} \tag{3.10}$$

and $c(y,\phi,w) = 0$, $w = \phi = 1$. Recall the canonical link function, $g(\mu) = \theta$, which we can see from Eq. (3.10) has to be

$$g(\mu) = \ln\left(\frac{\mu}{1-\mu}\right), \tag{3.11}$$

which is called the *logit link funciton*. This also maps the expected value to the linear predictor, η,

$$\ln\left(\frac{\mu}{1-\mu}\right) = \eta = \boldsymbol{x}^T\boldsymbol{\beta}. \tag{3.12}$$

Solving for μ in Eq. (3.12), we obtain the expected value expressed through the linear predictor, with the *inverse logit function*

$$\mu = \frac{\exp(\eta)}{1+\exp(\eta)}. \tag{3.13}$$

3.3 Parameter estimation

An essential part of statistical analysis is the estimation of unknown model parameters given experimental data. Recall from section 3.1.1 that we are considering two simultaneously evolving processes, being 1) *the evolution of the network connectivity* and 2) *the spike history of the neurons*, respectively. Both of these processes are modelled with different parameters, that have to be inferred from only empirical neural data. In this work we will use two different approaches for estimating parameters.

3.3.1 Maximum likelihood estimation

Recall from section 3.2.2 that we model the neuronal spiking through a Bernoulli GLM. To estimate some of the parameters of this model, contained in the linear predictor, we use maximum likelihood estimation (MLE) [29].

The likelihood function for independent Bernoulli random variables y_i is the product of the likelihoods for the independent observations

$$L(\mu) = \prod_{i=1}^{n} L_i(\mu_i) = \prod_{i=1}^{n} f(y_i|\mu_i) = \prod_{i=1}^{n} \mu_i^{y_i}(1-\mu_i)^{1-y_i} \tag{3.14}$$

where μ_i is the expected value of the Bernoulli variable y_i. Now however, we would like to express this in the GLM framework. That is, our mean μ_i actually depends on the parameters, $\boldsymbol{\beta}$ through a linear predictor and a link function, seen from Eq. (3.13). We therefore write

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \mu_i(\boldsymbol{\beta})^{y_i}(1-\mu_i(\boldsymbol{\beta}))^{1-y_i}. \tag{3.15}$$

The idea is now to estimate the parameters in $\boldsymbol{\beta}$, by choosing those that maximise the probability of our observations, given by Eq. (3.15). Often it is more tractable to work with the logarithm of $L(\boldsymbol{\beta})$, and since the logarithm is a monotonically increasing function, $\ln(L(\boldsymbol{\beta}))$

will have the same maximum likelihood estimates as $L(\boldsymbol{\beta})$. Applying the logarithm to Eq. (3.15) yields

$$l(\boldsymbol{\beta}) = \ln(L(\boldsymbol{\beta})) = \sum_{i=1}^{n} l_i(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \ln\Big(\frac{\mu_i(\boldsymbol{\beta})}{1-\mu_i(\boldsymbol{\beta}}\Big) + \ln\big(1-\mu_i(\boldsymbol{\beta})\big). \tag{3.16}$$

As we derived in Eq. (3.13), the expected value, μ, can be expressed with $\boldsymbol{\beta}$ through the linear predictor. By replacing this expression into Eq. (3.16), we obtain

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_i(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \boldsymbol{x_i^T}\boldsymbol{\beta} - \ln\big(1+\exp(\boldsymbol{x_i^T}\boldsymbol{\beta})\big). \tag{3.17}$$

As mentioned, the goal is to find the $\boldsymbol{\beta}$'s maximising the log-likelihood. Since $l(\boldsymbol{\beta})$ is a strictly concave function [29], this can be done numerically with some gradient method. One of the most common methods, which we will use later, is the *Fisher scoring algorithm*. This makes use of the so-called *Score function* and *Fisher information matrix*, where the two provide information about the first and the second derivative of the log-likelihood, respectively.

For convenience, we now again write $\mu_i(\boldsymbol{\beta})$ instead of $\frac{\exp(\boldsymbol{x_i^T}\boldsymbol{\beta})}{1+\exp(\boldsymbol{x_i^T}\boldsymbol{\beta})}$. The score function is defined as

$$\boxed{s(\boldsymbol{\beta}) = \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} s_i(\boldsymbol{\beta}) = \sum_{i=1}^{n} \boldsymbol{x_i}\big(y_i - \mu_i(\boldsymbol{\beta})\big)}. \tag{3.18}$$

The Fisher information matrix is defined as

$$F(\boldsymbol{\beta}) = \mathbb{E}\bigg(-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^T}\bigg)$$

or equivalently

$$F(\boldsymbol{\beta}) = \mathrm{Cov}(s(\boldsymbol{\beta})) = \sum_{i=1}^{n} \mathrm{Cov}(s_i(\boldsymbol{\beta})),$$

where the sum is obtained by assuming independent observations. Note that $\mathrm{E}[s_i(\boldsymbol{\beta})] = 0$ as $\mathrm{E}[y_i] = \mu_i(\boldsymbol{\beta})$. Using this and the definition of covariance we obtain

$$\sum_{i=1}^{n} \mathrm{Cov}(s_i(\boldsymbol{\beta})) = \sum_{i=1}^{n} \mathbb{E}\Big[\Big(s_i(\boldsymbol{\beta}) - \mathbb{E}[s_i(\boldsymbol{\beta})]\Big)\Big(s_i(\boldsymbol{\beta}) - \mathbb{E}[s_i(\boldsymbol{\beta})]\Big)^T\Big] = \sum_{i=1}^{n} \mathbb{E}\big[s_i(\boldsymbol{\beta})\, s_i(\boldsymbol{\beta})^T\big],$$

which, by employing Eq. (3.18), leads to the following expression for the Fisher Information matrix:

$$F(\boldsymbol{\beta}) = \sum_{i=1}^{n} \mathbb{E}\big[s_i(\boldsymbol{\beta})\, s_i(\boldsymbol{\beta})^T\big] = \sum_{i=1}^{n} \boldsymbol{x_i}\boldsymbol{x_i^T}\mathbb{E}\big[\big(y_i - \mu_i(\boldsymbol{\beta})\big)^2\big].$$

And finally, by exploiting that $\mathbb{E}\Big[\big(y_i - \mu_i(\boldsymbol{\beta})\big)^2\Big]$ is the variance of y_i, we arrive at

$$\boxed{F(\boldsymbol{\beta}) = \sum_{i=1}^{n} \boldsymbol{x_i}\boldsymbol{x_i^T}\mu_i(\boldsymbol{\beta})\big(1-\mu_i(\boldsymbol{\beta})\big)}, \tag{3.19}$$

which explicits the dependence of the Fisher Information matrix on the model parameter and data. The Fisher scoring algorithm is defined by the parameter updates:

$$\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} + \left(F(\boldsymbol{\beta}^{(i)})\right)^{-1} s(\boldsymbol{\beta}^{(i)}). \tag{3.20}$$

The algorithm will converge towards the solution $s(\boldsymbol{\beta}) = \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$, because $F(\boldsymbol{\beta})$ is positive semi-definite, which is always the case for covariance matrices. Furthermore, because $l(\boldsymbol{\beta})$ is strictly concave, this will indeed converge towards the maximum likelihood estimate $\hat{\boldsymbol{\beta}}_{\text{MLE}}$.

The Fisher scoring algorithm is closely related to Newton's method, where we iteratively search for $\boldsymbol{\beta}$ maximising the log-likelihood. Note that for a Bernoulli GLM the algorithms are indeed identical since the Fisher Information Matrix (3.19) does not depend on the response variable.

3.3.2 Bayesian estimation

Bayesian estimation consists of a number of methods aiming at providing a point estimate of the model parameters, by minimising the average expected value of a chosen loss function over the posterior distribution [30]. Using Bayes' rule, the posterior distribution of the parameter $\boldsymbol{\beta}$ given the data y can be written as

$$f(\boldsymbol{\beta}|y) = \frac{f(y|\boldsymbol{\beta})f(\boldsymbol{\beta})}{f(y)} \tag{3.21}$$

where the denominator $f(y) = \int f(y,\boldsymbol{\beta})d\boldsymbol{\beta}$ is a normalisation factor. The estimator for our parameter becomes

$$\hat{\boldsymbol{\beta}} = \arg\min_{\hat{\boldsymbol{\beta}}} \mathbb{E}\left[C(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})\right] \tag{3.22}$$

where C is a non-negative cost function. Using that $\mathrm{E}(x) = \int x f(x) dx$, we can calculate

$$\hat{\boldsymbol{\beta}} = \arg\min_{\hat{\boldsymbol{\beta}}} \int_{-\infty}^{\infty} C(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) f(\boldsymbol{\beta}|y) d\boldsymbol{\beta}, \tag{3.23}$$

and insert a desired choice of cost function to obtain an estimate. For this work we will mainly be focusing on one particular estimate, namely the mean square error.

Mean square error

For the mean square error, the cost function is defined as

$$C(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \left|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right|^2. \tag{3.24}$$

To derive the corresponding estimator, we insert this cost function into Eq. (3.23),

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\hat{\boldsymbol{\beta}}} \int_{-\infty}^{\infty} \left|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right|^2 f(\boldsymbol{\beta}|y) d\boldsymbol{\beta} \\
&\implies \frac{d}{d\hat{\boldsymbol{\beta}}} \int_{-\infty}^{\infty} \left(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right)^2 f(\boldsymbol{\beta}|y) d\boldsymbol{\beta} = 0 \\
&\implies 2\hat{\boldsymbol{\beta}} \int_{-\infty}^{\infty} f(\boldsymbol{\beta}|y) d\boldsymbol{\beta} = 2 \int_{-\infty}^{\infty} \boldsymbol{\beta} f(\boldsymbol{\beta}|y) d\boldsymbol{\beta} \\
&\implies \hat{\boldsymbol{\beta}} = \int_{-\infty}^{\infty} \boldsymbol{\beta} f(\boldsymbol{\beta}|y) d\boldsymbol{\beta} = \mathbb{E}(\boldsymbol{\beta}),
\end{aligned}$$

which leads to our estimator being the mean of the posterior distribution.

Other common cost functions are the absolute error[30] and the 0-1 loss function[31], which yield the median and the mode of the posterior as estimators, respectively.

3.4 Cross-correlation estimation

A *time series* can be defined as a sequence of random variables indexed according to the order they are obtained in time [32]. Hence, the spike train of a neuron can ultimately be considered as a time series. When working with real data, our first task would be to identify neuron pairs that actually seem to be connected, before applying the method that we propose for studying the synaptic plasticity rule. It could be wasteful to analyse each pair of neurons comprehensively, without even having an indication whether they might be connected or not. For rather quickly detecting possible connections, we will make use of cross-correlation estimation of spike trains of pairs of neurons, where we apply the framework presented by Haugh [33].

The idea is to consider small time lags, denoted by k, for one of the series i.e. spike trains. If the neurons are connected, we would experience correlation between the spike trains of the given neurons for certain time lags, indicating the required time for one response to influence the other. This would in our case be the time a signal needs to travel from the presynaptic neuron to the postsynaptic one.

Denote two time series s_1 and s_2 for some time units $t = 0, 1, ..., N$ and let $\hat{r}_{s1,s2}(k)$ be the estimated correlation function for some time lag k, being a discrete amount of time steps. The estimated correlation is defined as

$$\hat{r}_{s1,s2}(k) = \begin{cases} \frac{\frac{1}{N}\sum_{t=0}^{N-k}(s_1(t+k)-\bar{s}_1)\cdot(s_2(t)-\bar{s}_2)}{\sigma_{s_1}\sigma_{s_2}}, & \text{for } k > 0 \\ \frac{\frac{1}{N}\sum_{t=0-k}^{N}(s_1(t+k)-\bar{s}_1)\cdot(s_2(t)-\bar{s}_2)}{\sigma_{s_1}\sigma_{s_2}}, & \text{for } k < 0 \end{cases} \tag{3.25}$$

where N is the number of time steps, $\bar{s}_1$ and $\bar{s}_2$ are the means of the two time series, and σ_{s_1} and σ_{s_2} are the standard deviations of the s_1 and s_2.

From this we can obtain estimates for the correlation at different time lags. Furthermore, we would like to say something about the significance of the observed correlation, which we do in the following fashion.

Let's assume no autocorrelation for both time series, which basically means that the internal events of the individual series are independent of each other. Then we create a hypotheses test for the estimation for some time lag k, namely

$$H_0 : \ \hat{r}_{s1,s2}(k) = 0 \qquad \text{vs} \qquad H_1 : \ \hat{r}_{s1,s2}(k) \neq 0. \tag{3.26}$$

Under the assumption of H_0, it is shown in the article [33] that the estimator asymptotically follows an approximate normal distribution

$$\hat{r}_{s1,s2}(k) \sim \mathcal{N}\Big(0, \frac{1}{N-|k|}\Big), \tag{3.27}$$

where $|k|$ is the absolute value of the considered time lag. Using this we can easily calculate a confidence interval for the estimator under the assumption of H_0, based on the normal distribution, for some desired significance value α. If the observed correlation lies outside this interval, we might reject the null hypothesis, and conclude that the correlation seems

significant, that is, $\hat{r}_{s1,s2}(k) \neq 0$. The significant correlation will constitute for us an indication of a putative directed synaptic connection between the neural pair.

From this, one could also construct more sophisticated and accurate test-statistics, being for instance chi-squared distributed or F-distributed. We will however apply this more simple framework, as we consider this to be accurate enough for our purpose. For more developed methods, we can recommend the interested reader to the paper "*Tests for non-correlation of two multivariate time series: a nonparametric approach*" [34].

3.5 Importance sampling

Importance sampling is a Monte Carlo method for approximating a desired probability distribution, say $f(x)$ (that is difficult to sample from), by sampling from another distribution $g(x)$ that we can sample from. If we want an estimate of the mean of some other function $h(x)$, that is, $\mathbb{E}_f(h(x)) = \int h(x)f(x)dx$, we can rewrite this to be

$$\mathbb{E}_f(h(x)) = \int h(x)\frac{f(x)}{g(x)}g(x)dx, \tag{3.28}$$

meaning that we may try to estimate the mean with

$$\mathbb{E}_f(h(x)) = \mathbb{E}_g\left(\frac{h(x)f(x)}{g(x)}\right) \tag{3.29}$$

instead. A Monte Carlo estimate of the expectation of $h(x)$ with respect to the probability distribution $f(x)$, denoted $\hat{\mu}_{\mathrm{MC}}$, as described in [35], can be obtained by drawing N samples from our distribution $f(x)$, thereby producing the sample average

$$\hat{\mu}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} h(x_i).$$

In the case described above, where $f(x)$ is difficult to sample from directly, we use the rewritten version that has been multiplied and divided by $g(x)$, recall Eq. (3.28) and (3.29), so that our estimator becomes

$$\hat{\mu}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} h(x_i)\,v(x_i),$$

where these $v(x_i) = f(x_i)/g(x_i)$ are referred to as the (unnormalised) importance weights of the samples. The normalised importance weights, $\tilde{v}$, are obtained by dividing all the weights by their sum,

$$\tilde{v}(x_i) = \frac{v(x_i)}{\sum_{i=1}^{N} v(x_i)}. \tag{3.30}$$

In other words, importance sampling approximates $f(x)$ by using a discrete distribution with mass $v(x_i)$ for each of the observed points. In [36] it was shown that approximating using this sampling method will converge to the target distribution $f(x)$ as $N \to \infty$.

3.6 Information theory

A concept that has been fundamental for modern science, is the idea of defining and measuring information. This field of science is often referred to as *Information theory*, which is

largely based on the fundamental work by Claude Shannon, which was formalised and published in 1948 [37]. These concepts have also proved to be valuable in numerous fields such as neuroscience [38], biology [39], economics [40], machine learning [41] and cognitive science[42]. It will also be essential for the contributions presented in this work. The theory in this section is based on the book *Elements of Information Theory*[43].

3.6.1 Shannon entropy

For a random variable X, *entropy* is a measure of the uncertainty associated with X. In the discrete case, let $X \in \mathcal{X}$ be a random variable on some probability space with probability function $p_X(x) = P(X = x)$. Then the entropy $H_b(X)$ is defined as

$$H_b(X) = -\sum_{x\in\mathcal{X}} p_X(x)\log_b p_X(x) \tag{3.31}$$

where b denotes the choice of logarithm base. The most common choices for b are 2 or e. These choices would yield the units *bits* and *nats* for the entropy, respectively[43]. Note that in Eq. (3.31) we assume $0 \cdot \log_b(0) = 0$. Using Jensen inequality, one can easily see that the Shannon entropy of discrete random variables (3.31) is semi-positive defined and bounded from above by the $\log_b|\mathcal{X}|$, where $|\mathcal{X}|$ is the cardinality of $\mathcal{X}$.

An intuitive understanding of entropy can be obtained from the following constructed example. Assume a random variable $X \in \mathcal{X}$, where $p_X(\hat{x}) = 1$ for some event $\hat{x} \in \mathcal{X}$, then $p_X(x) = 0$ for all $x \in \mathcal{X}\backslash\{\hat{x}\}$. In this case, there will be no uncertainty associated with the outcome of X, since the outcome will always be $\hat{x}$. Hence, us observing the outcome doesn't convey any new information which we didn't already know. Calculating $H_b(X)$ from Eq. (3.31) would clearly yield $H_b(X) = 0$. In contrast, the entropy is maximised for a random variable $X \in \mathcal{X}$ where $p_X(x) = \frac{1}{|X|}$ for all $x \in \mathcal{X}$, where $|X|$ is the cardinality of X. That is, all the events are equally likely, yielding maximum uncertainty of the outcome of X. So the lower the entropy is, the less uncertainty and less information gain is associated with the outcome of the random variable.

This concept can further be extended to the more complex and less intuitive case where our random variable X is continuous. We refer to it as *differential entropy*.

Differential entropy

Suppose X is a continuous random variable. If $\int_{-\infty}^{\infty} f(x)\mathrm{dx} = 1$, $f(x)$ is a *probability density function* for X. Let $\mathcal{S}$ be the *support set* of X, defined as $\mathcal{S} = \{x \in \mathcal{X} : f(x) > 0\}$. Then the differential entropy $H_b(X)$ is defined as

$$H_b(X) = -\int_{\mathcal{S}} f(x)\log_b f(x)\mathrm{dx}. \tag{3.32}$$

In contrast to the discrete case, the differential entropy can also take negative values. The definition of entropy in both the discrete and continuous case can also be extended to several random variables and multivariate distributions. Important for this work will be the multivariate normal distribution, so let us derive the corresponding differential entropy.

Multivariate normal distribution

Let $\boldsymbol{x} \in \mathbb{R}^n$ have a multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. The probability density function is defined as

$$f(\boldsymbol{x}) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}}, \tag{3.33}$$

where $|\Sigma|$ denotes the determinant of Σ. By inserting $f(\boldsymbol{x})$ into Eq. (3.32) and applying the natural logarithm, we obtain

$$H_e(\boldsymbol{x}) = -\int f(x)\Big[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) - \frac{1}{2}\ln\big((2\pi)^n |\Sigma|\big)\Big]\mathrm{dx}. \tag{3.34}$$

By canceling the minus signs, splitting the terms and considering them seperately, we have

$$H_e(\boldsymbol{x}) = \int f(x)\Big(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big)\mathrm{dx} + \underbrace{\int f(x)\Big(\frac{1}{2}\ln\big((2\pi)^n |\Sigma|\big)\Big)\mathrm{dx}}_{\frac{1}{2}\ln\left((2\pi)^n |\Sigma|\right)}, \tag{3.35}$$

where the second term can easily be simplified since the constant term can be put outside of the integral and the remaining integral yields 1 by definition of the density. The first term can be simplified by using the expected value of functions of random variables

$$\frac{1}{2}\int f(x)\underbrace{\Big((\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big)}_{g(\boldsymbol{x})}\mathrm{dx} = \frac{1}{2}\mathbb{E}\big[g(\boldsymbol{x})\big] \tag{3.36}$$

and further simplified by applying properties of the trace operator

$$\begin{aligned}
\mathbb{E}\Big[(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big] &= \mathbb{E}\Big[\mathrm{tr}\big((\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\big)\Big] && (3.37)\\
&= \mathbb{E}\Big[\mathrm{tr}\big(\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{x}-\boldsymbol{\mu})\big)\Big] && (3.38)\\
&= \mathrm{tr}\Big(\Sigma^{-1}\mathbb{E}\big[(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{x}-\boldsymbol{\mu})\big]\Big) && (3.39)\\
&= \mathrm{tr}\big(\Sigma^{-1}\Sigma\big) && (3.40)\\
&= \mathrm{tr}\big(\boldsymbol{I}_n\big) && (3.41)\\
&= n. && (3.42)
\end{aligned}$$

Merging the two terms yields us the final expression for the differential entropy of the multivariate normal distribution

$$\begin{aligned}
H_e(\boldsymbol{x}) &= \frac{n}{2} + \frac{1}{2}\ln\big((2\pi)^n |\Sigma|\big) && (3.43)\\
&= \frac{1}{2}\ln\big((2\pi e)^n |\Sigma|\big) \quad \text{nats} && (3.44)
\end{aligned}$$

3.6.2 Mutual information

Whereas entropy quantifies the information associated with a random variable, mutual information expands on this concept to quantify how much information one random variable contains about another one. In other words, how much does the uncertainty of a random variable decrease, given the knowledge of another one.

Discrete case

Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be random variables with joint probability mass function $p_{XY}(x,y)$, while $p_X(x)$ and $p_Y(y)$ are the respective marginal probability mass functions. The mutual information $I(X;Y)$ is defined as

$$I_b(X;Y) = \sum_{X\in\mathcal{X}} \sum_{Y\in\mathcal{Y}} p_{XY}(x,y) \log_b \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}. \tag{3.45}$$

Exploiting basic formulas for conditional probability and logarithmic properties allows us to write

$$\begin{aligned} I_b(X;Y) &= \sum_{x,y} p_{XY}(x,y) \log_b \frac{p(x|y)}{p_X(x)} && (3.46)\\ &= -\sum_{x,y} p_{XY}(x,y) \log_b p_X(x) + \sum_{x,y} p_{XY}(x,y) \log_b p(x|y) && (3.47)\\ &= \underbrace{-\sum_{x} p_X(x) log_b p_X(x)}_{=H_b(X)} \underbrace{-\Big(-\sum_{x,y} p_{XY}(x,y) \log_b p(x|y)\Big)}_{=H_b(X|Y)}, && (3.48) \end{aligned}$$

where the second term is the definition of conditional entropy, defined more in detail in[43]. So the mutual information can be expressed as a difference of two entropies, $H_b(X) - H_b(X|Y)$, which is indeed the reduction in uncertainty of X, given knowledge about the outcome of Y. This allows for an intuitive understanding, in that if X and Y are independent, $H_b(X|Y) = H_b(X)$, which in turn would yield no mutual information, $I_b(X;Y) = 0$. Oppositely, the mutual information is maximised if $H_b(X|Y) = 0$, that is, if given the outcome of Y, there is no uncertainty associated with the outcome of X. In this case, all information about X is contained in Y.

Also note that by a minor reformulation in Eq. (3.46), we can prove symmetry, in that the mutual information also can be expressed as $I_b(X;Y) = H_b(Y) - H_b(Y|X)$.

Continuous case

The extension to continuous random variables is pretty straight forward, where we again will have to exchange the sum with an integral. Assume $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ are continuous random variables with probability density functions $f(x)$ and $f(y)$. Also let their joint density be denoted by $f(x,y)$. The mutual information between X and Y is then defined as

$$I_b(X;Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} f(x,y) \log_b \frac{f(x,y)}{f(x)f(y)} \mathrm{dxdy}. \tag{3.49}$$

In general, $I_b(X;Y)$ possesses the same properties as in the discrete case. Note that in the continuous case, $I_b(X;Y)$ can also take negative values.

3.7 Bayesian experimental design

The work of lots of people in both science and industry consists of planning, executing and analysing experiments with the intention to gain information and understanding of some desired phenomena. However, an important challenge is how to optimally conduct these

experiments with limited experimental resources. *Design of Experiments* is a field withing applied statistics which has emerged to address this problem. Methods can be traced back to the work by Ronald Fisher in the 1930's[44][45]. Since then, numerous ideas, methods and frameworks have been developed[46]. However, among the classical methods, there might be limitations[47], and they might not be best suited for all sorts of problems.

Recently, a more sophisticated approach has increased in popularity, which exploits Bayesian theory, therefore called *Bayesian experimental design*. This sort of approach seems to have been applied in practice first by Flournoy in 1993[48]. A Bayesian framework might be clever because the sample is not yet observed, hence incorporation of uncertainty and averages for experimental parameters in the design can be beneficial[49][50]. Also, a Bayesian approach is suited for an adaptive design where the design itself is optimised in parallel with data collection, which we will explore more in detail later. Chaloner and Verdinelli present a detailed overview of the literature within the field, as well as tasks where the Bayesian approach might outperform more classical methods[50]. Such frameworks have also been developed and applied to different experiments within neuroscience[51][52]. This will also be our approach when doing experimental design in this thesis.

3.7.1 Utility function

The main idea is to formulate an optimisation problem where the design should maximise the expectancy of some utility function, which is appropriate for the considered experiment. Denote the data to be collected $Y \in \mathcal{Y}$, define some model parameters $\theta \in \Omega$ and let $X \in \mathcal{X}$ be some experimental design. Then a Bayesian optimal design aims to find a design X^* which maximises some utility function, $u : \mathcal{Y} \times \Omega \times \mathcal{X} \longrightarrow \mathbb{R}$

$$X^* = \underset{X \in \mathcal{X}}{\operatorname{argmax}} \mathbb{E}[u(Y, \theta, X)]. \tag{3.50}$$

The choice of utility function should depend on the purpose of the experiment. Chaloner and Verdinelli discuss different choices of $u(Y, \theta, X)$[50]. In this work and in statistics in general, solving statistical inference problems is often either part of - or indeed the main task. When doing this in a Bayesian manner, the natural choice for the utility function is the expected mutual information between the parameters and the data, discussed in detail by Kenneth Ryan[49]. This utility function might be written on the form

$$u(Y, \theta, X) = \int_\Omega \int_{\mathcal{Y}} p(Y, \theta | X) \log_b p(\theta | Y, X) \mathrm{d}y \mathrm{d}\theta \quad + \text{const} \tag{3.51}$$

$$= -\int_{\mathcal{Y}} p(Y|X) H_b(\theta | Y, X) \mathrm{d}y \quad + \text{const}, \tag{3.52}$$

where $H_b(\theta|Y, X)$ is the Shannon entropy of θ as defined in Eq. (3.32). The constant term is independent of the design X, and is therefore irrelevant for solving Eq. (3.50). This is a general problem formulation, which will be rewritten to suit our specific problem. In section 4.4.2, it becomes clear that we essentially employ the same utility function as Eq. (3.52). Therefore, this literature by Ryan, from which we were inspired, serves well as a theoretical background for our work.

The utility function in Eq. (3.51) often proves to be challenging to calculate and to maximise with respect to X. Firstly, because the posterior $p(\theta|Y, X)$ usually has to be approximated, as the analytical form might be infeasible to obtain. Secondly, the parameter space Ω and the data domain $\mathcal{Y}$ might be of high dimensionality or in general intractable to integrate over.

3.7.2 Approximating the utility

Ryan suggests that $u(Y,\theta,X)$ can be approximated by combining MCMC techniques and importance sampling, being methods commonly used in Bayesian statistics. Although estimators proposed by Ryan are not identical to how we will solve our specific problem, they apply the same techniques, and consequently we don't doubt the relevancy of the theory presented by Ryan.

If $p(\theta|Y,X)$ can not be expressed in closed form, it can be expressed as

$$p(\theta|Y,X) = \frac{p(Y|\theta,X)\,p(\theta)}{p(Y|X)}, \tag{3.53}$$

which when combined with Eq. (3.51) yields an MCMC estimate of $u(Y,\theta,X)$ on the form

$$\hat{u}(Y,\theta,X) = \frac{1}{L}\sum_{i=1}^{L} \log\big[p(Y_i|\theta_i,X)\,p(\theta_i)\big] - \log\hat{p}(Y_i|X), \tag{3.54}$$

where (Y_i,θ_i) for $i = 1,...,L$ is an MCMC sample from $p(\theta,Y|X)$. Furthermore, the second term in Eq. (3.54) requires an estimate $\hat{p}(Y_i|X)$ for the data distribution, which can be obtained by importance sampling

$$\hat{p}(Y_i|X) = \frac{1}{M}\sum_{j=1}^{M} p(Y_i|\theta_i^{(j)},X), \tag{3.55}$$

by drawing $j = 1,...,M$ samples at fixed parameter values $\theta_i^{(j)}$ drawn from $p(\theta)$.

Properties of the estimator

Estimators obtained from sampling techniques generally show nice asymptotic properties[53][54]. The estimator $\hat{u}(Y,\theta,X)$ in Eq. (3.54) can also be shown to have satisfactory asymptotic properties[49].

Suppose that $\{(\theta_i,Y_i)\}_{i=1}^{L}$ is an independently and identically distributed (iid) sequence from the density $p(\theta|Y,X)$ and that $\{\theta_i^{(j)}\}_{i=1}^{L}$ are iid sequences from $p(\theta)$. It can then be shown that the variance of $\hat{u}(Y,\theta,X)$ is of the order of L^{-1}

$$\mathrm{Var}\Big(\hat{u}(Y,\theta,X)\Big) \propto \frac{1}{L}. \tag{3.56}$$

Also, the expected bias can be shown to be positive and can be approximated by

$$\mathbb{E}\Big[\hat{u}(Y,\theta,X) - u(Y,\theta,X)\Big] \approx \frac{C(X)}{M}, \tag{3.57}$$

where $C(X)$ is a function of the design X. Detailed derivations of these properties can be found in the work by Ryan[49].

The computational complexity of approximating $\hat{u}(Y,\theta,X)$ is of the order $\mathcal{O}(LM)$. Therefore, if we want to fix the computational cost at some feasible level, it becomes a trade-off between L, which affects the variance, and M, which affects the bias. Under the assumption that $C(X)$ will be approximately constant over $X \in \mathcal{X}$

$$C(X) \approx \text{const}, \tag{3.58}$$

the bias will be equal across different designs, regardless of the choice of M. In this case one could argue that for comparing designs, determining X^* and solving the optimisation problem Eq. (3.50), lowering the variance should be prioritised. Hence an argument for increasing L and decreasing M is reasonable.

Chapter 4

Methodology

This chapter will be dedicated to explaining our experimental setup, the detailed model we consider, and all of the developed methods for inference and Bayesian experimental design.

In section 4.1 we introduce the modelling framework for neural activity as a generalised HMM, including both stochastic processes we are using for the model.

In section 4.2, we present the types of data we are working with. This includes synthetic data and choices for all relevant hyperparameters to obtain the data, as well as a presentation of a real data set we are using.

In section 4.3, we provide more details on a composite MCMC inference algorithm for uncovering the learning rule parameters in Bayesian fashion. Implementation details for the corresponding algorithm are also presented.

With these things sorted, we proceed to the main goal of this work, which is to develop a framework for optimal stimulation to infer the learning rule parameters. In section 4.4 we explain how we approach the problem, we will define our optimisation problem and how we use approximations to solve it, and lastly the implementation details and relevant algorithms are presented.

4.1 Model description

We are modelling the dynamics in the brain by two stochastic processes, which also depend on each other, recall section 3.1.1. The first one describes the neural activity, more specifically the spiking times of considered neurons. The second one models the evolution of the synaptic connectivity in an interconnected network. A hierarchical diagram showcasing the dependencies is presented in figure 4.1C).

4.1.1 Spiking model

The spike trains of the neurons are defined according to section 2.2. The time domain is binned into T bins of some size δ, where the bins are indexed by $t \in \{1, ..., T\}$. Furthermore, we define the spike trains $s_i^{1:T}$ for neuron i. The synaptic connectivity strength between a pair of neurons i and j at time t is denoted by w_{ij}^t. Spiking events are modelled with a Bernoulli GLM as defined in section 3.2.2, hence $s_i^t \in \{0, 1\}$, where $s_i^t = 1$ denotes at least one spike of neuron i in time bin t. The conditional spiking probability for neuron n in time bin

$t+1$, λ_n^{t+1} is then defined as

$$\lambda_n^{t+1} = g\Big(b_n^{t+1} + \sum_{n'=1, n'\neq n}^{N} w_{n'n}^t s_{n'}^t\Big), \tag{4.1}$$

for a network of N neurons, where $g(\cdot)$ is the inverse logit function in Eq. (3.13) defined in section 3.2.1. This is a general model definition, however, in this work we will consider a simplified model where $N = 2$, and where the neuron pair is connected through a mono-directional synapse, as is illustrated in figure 4.1A).

The parameter b_n^t denotes an external time-dependent field, which the neurons are exposed to. This can be decomposed into two parts. One being a stationary input, b_n, which accounts for noise and sources of input which is not explicitly modelled. In addition, we have a second non-stationary component, $b_{n,ext}^t$, which represents the external stimulation delivered by the experimentalist to the neuron. In the absence of external stimulation, $b_{n,ext}^t = 0$, the resulting firing rate (the expected number of spikes per second) is referred to as the baseline firing rate. In this work, we will consider regimes where only the presynaptic neuron is externally stimulated. That is, $b_{2,ext}^t = 0$, throughout this thesis.

4.1.2 Synaptic plasticity model

Now, we will define the STDP learning rule, that we discussed in 2.3. This was first proposed by Abott and coauthors [2]. The learning rule considers the time between spikes, interspike intervals, of the neurons to determine the update of the synaptic weight. The weight is assumed to follow a Markov process and evolves according to the density

$$w^t \sim \mathcal{N}\Big(w^{t-1} + l(s_1^{1:t-1}, s_2^{1:t-1}, \theta), \sigma\Big). \tag{4.2}$$

where $\epsilon(\sigma)$ is a Gaussian white noise, while $l(s_1^{1:t-1}, s_2^{1:t-1}, \theta)$ is the STDP learning rule for parameters $\theta = \{A_+, A_-, \tau_+, \tau_-\}$, which is defined as

$$\begin{aligned} l(s_1^{1:t}, s_2^{1:t}, \theta) &= l_+(s_1^{1:t}, s_2^{1:t}, A_+, \tau_+) - l_-(s_1^{1:t}, s_2^{1:t}, A_-, \tau_-), \\ l_+(s_1^{1:t}, s_2^{1:t}, A_+, \tau_+) &= s_2^t \sum_{t'=\gamma_t}^{t} s_1^{t'} A_+ \exp\Big(\frac{t'-t}{\tau_+}\Big), \\ l_-(s_1^{1:t}, s_2^{1:t}, A_-, \tau_-) &= s_1^t \sum_{t'=\gamma_t}^{t} s_2^{t'} A_- \exp\Big(\frac{t'-t}{\tau_-}\Big). \end{aligned} \tag{4.3}$$

Here we have made a modification to the learning rule defined in the literature by introducing γ_t, defined $\gamma_t = t - 10\tau_+$, which denotes the first time bin of previous history to be considered. Compared to doing the sum over all of the history, this reduces the computational complexity, is more biologically plausible and has a negligible impact on the weight trajectory. This was shown in a preliminary project which we concluded prior to this work[55].

The learning rule for two sets of parameters θ are shown in figure 4.1B). The parameters A_+ and A_- can be interpreted as the scale of the weight update. For really small interspike intervals, the value of the learning rule approaches maximum values A_+ and A_- for either a positive or negative update. τ_+ and τ_- are describing the domain of relevant interspike intervals. The bigger these parameters are, the slower the contributions from larger interspike intervals decay to zero.

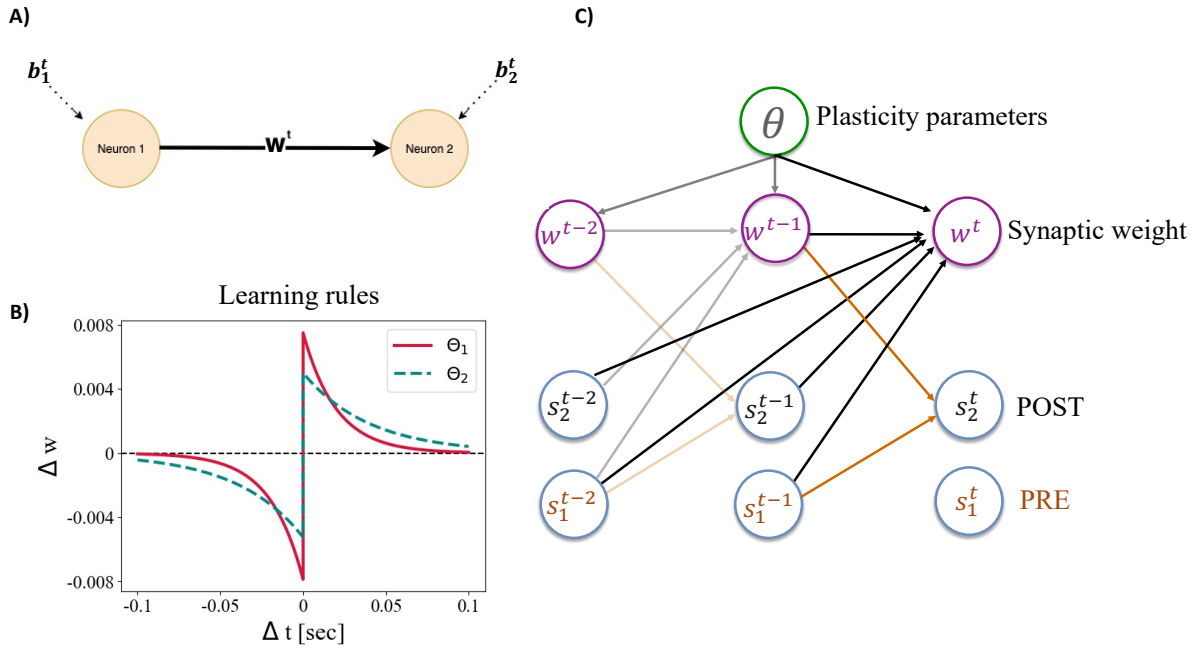

Figure 4.1: Spiking and STDP model. A): Monodirectionally connected neuron pair with non-stationary synaptic weight w^t. B): STDP learning rules with paramaters $\theta_1 = \{0.0075, 0.0079, 0.02, 0.02\}$ and $\theta_2 = \{0.0050, 0.0053, 0.04, 0.04\}$. C): Probabilistic graphical representation of the full Bayesian hierarchical model.

4.2 Data sets

In this work we will probe our methods on both synthetic and real data. In this section we will briefly explain the implementation details for the synthetic data and argue for the choices of hyperparameters we made, and we will introduce the real data set which we are using.

4.2.1 Synthetic data

All of the synthetic data is simulated according to the models in section 4.1.1 and 4.1.2. Though there are many hyperparameters that need to be chosen. We denote the bin size by δ and the generative noise by σ_{gen}. They are chosen to be 2ms and 0.0001, respectively. These values are both based on extensive analysis in our previous project[55]. Furthermore, we are studying an excitatory synapse, which is the most common synapse type[56], hence the initial value for the connectivity, $w^{t=1}$, is set to a small positive value. The values for the generative learning rule parameters θ, are inspired by Abott and coauthors [2]. Moreover, Linderman and coauthors use a baseline firing rate of 20Hz[1], which we used as motivation to find suitable values of the stationary external fields. The parameter setting is displayed in Table 4.1.

Since the goal of our main contribution is to study stimulation of different strengths, an important parameter is the external stimulation $b_{n,ext}^t$, which together with the stationary contribution b_n makes up the external field, b_n^t in Eq. (4.1). To adjust b_n^t, the external stimulation contribution can be constructed in two different ways. Either it can be a positive contribution, similar to b_n^t, such that $b_n^t > b_n$, which in turn leads to a stochastic firing rate which is greater than the baseline firing rate. The other option, is to implement a tetanic stimulation, being some short sporadic stimulation pulses, that are assumed to determinis-

parameter	value
A_+	0.005
A_-	0.00525
τ_+	0.02
τ_-	0.02
σ_{gen}	0.0001
δ	2ms
b_1	-3.1
b_2	-3.1
$w^{t=1}$	1

Table 4.1: Parameter setting for synthetic data.

tically induce a spike, meaning that the contribution of b_n^t leads to a deterministic firing rate of some desired frequency, in addition to the stochastic baseline firing rate.

We also assume that the excitatory synapse can not change to an inhibitory synapse, i.e. w^t can not change sign. This is known as Dale's Law[57], and is implemented in the data generating process.

4.2.2 Real data

Real data used in this work is collected by McKenzie and coauthours[10]. The data is collected via electrophysiological techniques from the CA1 region of the brain from mice and rats, where spiking activity was recorded from a population of neurons. For each session, one of these neurons was artificially stimulated with juxtacellular stimulation, this neuron will be used as the presynaptic neuron for our applications. A suitable mono-synaptic synapse will be detected according to the theory of cross-correlation presented in 3.4. The data consists of spike trains for all recorded neurons and exact time intervals for the juxtacellular stimulation.

4.3 Inference

Since we consider a HMM with two stochastic processes, parameters of both processes will need to be inferred. This section will concisely present the methods used for this purpose, and some theoretical details and derivations are omitted, as this is not the main contribution of this work. These methods were developed and explained in a more detailed and comprehensive manner in our previous project[55]. More details are also always available in the cited literature.

4.3.1 Learning rule parameters

Inference in this work aims first and foremost to recover the underlying learning rule parameters, for which we employ a Bayesian approach, as explained in section 3.3.2, using the mean estimates obtained from the mean squared error cost function. By using the values proposed in table 4.1, we have fixed relations between the parameters

$$A_- = 1.05A_+, \qquad \tau_+ = \tau_-. \tag{4.4}$$

This enables us to reduce the parameter space to two dimensions and only infer A_+ and τ_+, from which we find the other parameters from Eq. (4.4) afterwards. Going forward, τ_+ will be denoted by τ.

Particle Metropolis Hastings

The posterior of the parameters can be expressed in terms of the prior, $P(\theta)$, and the likelihood, $P(s_2^{1:T})$, as

$$P(\theta|s_2^{1:T}) = \frac{P(\theta)P(s_2^{1:T}|\theta)}{\int_\Theta p(\theta')P(s_2^{1:T}|\theta')d\theta'}, \tag{4.5}$$

where $s_2^{1:T}$ represents the whole spike train history of neuron 2. Notice that the dependence of the posterior and of the likelihood $P(s_2^{1:T}|\theta)$ on $s_1^{1:T}$ is here omitted for the sake of a lighter notation. One of the main difficulties with Bayesian estimation is that the denominator in Eq. (4.5) might be challenging or impossible to calculate. Markov Chain Monte Carlo (MCMC) methods are strong tools circumventing the calculation of the denominator. A common MCMC algorithm is the Metropolis-Hastings (M-H) algorithm[58], from which we have developed a more sophisticated algorithm suitable for our problem, inspired by [1][59].

The Metropolis-Hastings algorithm, which we employ to provide an empirical posterior for the learning rule parameters $P(\theta|s_2^{1:T})$, requires an estimate of the likelihood $P(s_2^{1:T}|\theta)$, in the numerator of Eq. (4.5). Notice that we can factorise the likelihood in time in the following way

$$P(s_2^{1:T}|\theta) = P(s_2^1|\theta)\prod_{t=2}^{T} P(s_2^t|s_2^{1:t-1},\theta), \tag{4.6}$$

where every factor can be written as an integral over the synaptic weight trajectory up to time t

$$P(s_2^t|s_2^{1:t-1},\theta) = \int P(s_2^t|w^{t-1},\theta)P(w^t|w^{t-1},s_2^{1:t},\theta)P(w^{1:t-1}|s_2^{1:t-1},\theta)dw^{1:t}. \tag{4.7}$$

Under the integral sign one recognises two distributions which are defined in our model: the spiking probability $P(s_2^t|w^{t-1},\theta)$ and the weight update $P(w^t|w^{t-1},s_2^{1:t},\theta)$ from Eq. (4.2) given by the learning rule in Eq. (4.3). The third factor, $P(w^{1:t-1}|s_2^{1:t-1},\theta)$, is the posterior of the weights up to time $t-1$, which by applying Bayes rule and the Markov property, can be expressed as

$$P(w^{1:t-1}|s_2^{1:t-1},\theta) = P(w^{t-1}|w^{t-2},s_2^{1:t-1},\theta)\frac{P(s_2^{t-1}|w^{t-2},\theta)}{P(s_2^{t-1}|s_2^{1:t-2},\theta)}P(w^{1:t-2}|s_2^{1:t-2},\theta), \tag{4.8}$$

This recursive relation for the synaptic weights posterior makes it suited for sampling via particle filtering methods. Particle filtering methods extend standard Monte Carlo approaches (in our case Importance sampling, see section 3.5) to processes/sequences[59][60]. The idea is that P realisations of our target weight trajectory up to time t are drawn given the observed data, which are referred to as *particles*, $\{w_{(p)}^{1:t}\}_{p=1}^{P}$. Each particle $p = 1,..,P$ gets associated a particle weight $v_{(p)}^t$, and together they constitute an importance sampling estimate for the target weight posterior $P(w^{1:t-1}|s_2^{1:t-1},\theta)$, as explained in section 3.5. The empirical posterior $\hat{P}(w^{1:t}|s_2^{1:t},\theta)$ can then be used to approximate the marginals $P(s_2^t|s_2^{1:t-1},\theta)$ in Eq. (4.8) and ultimately estimating the target likelihood $P(s_2^{1:T}|\theta)$, as in Eq. (4.6).

It can be shown that our estimate for the likelihood, $\hat{P}(s_2^{1:T}|\theta)$, obtained by particle filtering, is unbiased and that $\mathrm{Var}(\hat{P}(s_2^{1:T}|\theta)/P(s_2^{1:T}|\theta)) \propto T/P$ [61], where T is the number of time

points. So the variance is expected to increase linearly with the number of time points, as well as decrease linearly with the number of particles, P.

So, estimating $P(s_2^{1:T}|\theta)$ by embedding a particle filtering procedure in the MCMC algorithm, yields a Particle Metropolis-Hastings algorithm which provides an empirical distribution for the desired $P(\theta|s_2^{1:T})$. The embedded particle filtering procedure is presented in Algorithm 1, while the full Particle MCMC (PMCMC) algorithm is presented in Algorithm 2.

We also implement a number of burn-in iterations, B, which the algorithm uses to converge towards the higher density area of the parameter space. B is chosen for all of the experiments to be 300. We have two other hyperparameters to be chosen, namely P and a noise level σ used for sampling $P(w^{1:t-1}|s_2^{1:t-1},\theta)$ in the particle filtering. These were chosen based on numerical experiments and are discussed in section 5.1.

Resampling

The particle filtering is further improved by implementing a resampling procedure. This is done sporadically, based on the distribution of generated particles. By resampling, we aim to concentrate the particle distribution around the "promising" regions of the state space. It is effective at counteracting the tendency of particle filtering to reach a state with a few dominating particles, while the rest being negligible in terms of importance weights, referred to as particle degeneracy [62]. A natural measure of degeneracy is the *perplexity* of the sample, defined as $\text{per}(v) = \exp(H(v))/P$ [63], where $H(v)$ denotes the Shannon Entropy of the sample, as defined in section 3.6. We resample whenever the perplexity falls below a certain threshold. The threshold was set following Martino and coauthors [64], who suggest adopting the mean of the perplexity under a uniform distribution over the P-dimensional simplex (~ 0.66), as a threshold.

Proposal distribution

The Metropolis Hastings algorithm makes use of a proposal distribution which proposes new values in the parameter space[58]. The choice of this distribution is important for the convergence of the algorithm. In our model, the parameter space for θ is restricted to positive values. Hence we opted for a gamma distribution, for the proposal of both A_+ and τ. In each step, the two parameters are independently drawn from gamma distributions, such that the proposal distribution becomes a product of two gammas. The gamma distribution has a density function given by

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}, \qquad \text{for } x \in (0,\infty) \tag{4.9}$$

where α is the *shape* parameter, and β is the *scale* parameter. When the shape parameter gets sufficiently large, the gamma distribution approaches the standard distribution, which we consider being a reasonable assumption when nothing is known about the target distribution, being the learning rule posteriors. The mean and variance of the gamma distribution are given by

$$\text{Mean} = \alpha\beta \qquad \text{Variance} = \alpha\beta^2. \tag{4.10}$$

In the M-H sampler, after initialisation at the gamma prior hyperparameter values, see upcoming subsection 'Prior distribution', β for the proposal distribution is updated at each iteration based on the sample history, in order for the mean of the gamma in Eq. (4.10) to

equal the parameter values sampled at the previous Metropolis-Hastings iteration, while the variance is set as explained in the following paragraph. The Particle Metropolis-Hastings algorithm thus performs a random walk in the parameter space with non-stationary variance.

Adaptive variance

We also improve the algorithm by occasionally adjusting the variance of the proposal distribution, which is inspired by Haario and coauthors[65]. This serves the purpose of finding an adequate trade-off between exploration and exploitation of the algorithm. We choose an update frequency, $U = 100$. Both the shape parameter α and the scale parameter β are updated every U iteration to match the mean and the variance of the algorithm for the previous U iterations. Variance$(\theta)^U$ and Mean$(\theta)^U$ are computed, being the variance and mean of the last U samples, and α and β are found subsequently by solving the set of equations

$$\text{Mean}(\theta)^U = \alpha\beta \qquad \text{Variance}(\theta)^U \cdot c_d^2 = \alpha\beta^2 \tag{4.11}$$

and α is kept fixed for the next U iterations. This procedure is performed separately for A_+ and τ. c_d is a constant which scales with the dimensionality of our parameter space: for the one-dimensional case [65] recommends $c_d = 2.4$.

Prior distribution

We used a gamma for the prior distribution of A_+ and τ. In our work we set the scale and shape parameters to be $\boldsymbol{\alpha} = [4,5]$, $\boldsymbol{\beta} = [0.02, 0.01]$ respectively, see Eq. (4.9). Notice that our choice does not match the mean of the priors to the generative values of θ for the synthetic data, see Table 4.1; our gamma priors have also larger variance than those used by [1], allowing broader exploration of the parameter space, which we consider important for smaller data sizes which we are interested in. That is, for fewer data, the likelihood $P(s_2^{1:T}|\theta)$ will be less distinguishable between different values for θ. Then if the prior is too peaked, the algorithm will more easily be driven by the prior rather than the likelihood.

Algorithm 1: Particle filtering

```
Sample w_(p)^{t=1} ~ δ(ŵ^{t=1});
Set particle weights v_(p)^1 = 1;
for t = 2,3,4,... do
    Sample w_(p)^t ~ p(w_(p)^t | w_(p)^{t-1}, θ);
    Compute particle weights v_(p)^t = p(s_2^t | w_(p)^t, θ) ṽ_(p)^{t-1} ;
    Compute perplexity, per(v) = exp(H(v))/P ;
    if perplexity ≤ 0.66 then
        Normalise the particle weights;
        Resample;
        Set weights v_(p)^t = 1/P
    end
end
```

Algorithm 2: Metropolis Hastings sampler with embedded Particle filtering, PMCMC

Initialise θ^0;
Run particle filter, targeting $P(s_2^{1:T}|\theta^0)$;
for $i = 1, 2, \ldots$ **do**
  if i mod $U = 0$ **then**
    Adjust variance;
  end
  Sample $\theta^* \sim Q(\cdot|\theta^{i-1})$;
  Run particle filter, targeting $P(s_2^{1:T}|\theta^*)$;
  Compute $\alpha(\theta^*) = \frac{P(\theta^*)\cdot P(s_2^{1:T}|\theta^*)\cdot Q(\theta^{i-1}|\theta^*)}{P(\theta^{i-1})\cdot P(s_2^{1:T}|\theta^{i-1})Q(\theta^*|\theta^{i-1})}$;
  Accept θ^* with probability $\min\{1, \alpha(\theta^*)\}$;
  if θ^* *accepted* **then**
    $\theta^i = \theta^*$;
  else
    $\theta^i = \theta^{i-1}$;
  end
end

4.3.2 GLM parameters

For the PMCMC to yield accurate estimates, we also need estimates of the parameters contained in the GLM spiking model. Notice from Algorithm 1, that we need to initialise the particle weights, as well as calculating the spiking probability of the postsynaptic neuron. Therefore, we also need estimates of the initial weight $w^{t=1}$ and the external field of the postsynaptic neuron, b_2^t, which determines the spike rate in the absence of a presynaptic spike. For this, we are implementing the Fisher Scoring Algorithm to find MLE estimators for these parameters, according to the theory in section 3.3.1. However, this means that we are assuming that the coefficients in the linear predictor, recall Eq. (3.12), are stationary, which is not true for w^t in our modelling framework. Therefore, we only apply the Fisher Scoring Algorithm on a smaller subset of the data from the start of the experiment, and assume that $w^t \approx$ const within this subset. Recall from Eq. (3.20), that the algorithm is converging towards $s(\boldsymbol{\beta}) = 0$. A stopping criterion of $s(\boldsymbol{\beta}) < 10^{-10}$ for the algorithm is implemented, such that convergence is assumed when all elements of the score function are smaller than this threshold.

4.4 Bayesian optimal experimental design

Regarding the main research question of this work, we wish to study how we can optimally stimulate the presynaptic neuron, such that we can obtain adequate inference with as few data as possible. Our approach is based on the theory on Bayesian experiment design in section 3.7. An experimental design in our case, corresponds to a tetanic external stimulation of the presynaptic neuron with a certain frequency, as explained in 4.1.1. In order to find an optimum, we first construct a space of frequencies, based on commonly used stimulation protocols in experiments, see section 2.3. In this space, we only consider frequencies between 10Hz and 250Hz. We employ a heuristic approach by pre-defining a grid of selected frequencies in this space and searching for the optimal stimulation within the grid. Now, let us present the mathematical details to find this optimum.

4.4.1 Active learning approach

Inspired by the work of Shababo, Paige and coauthors (2013)[51], we implement an active learning procedure. This means, that we are adaptively adjusting and tuning the stimulation during the experiment. The idea is to split the experiment into N smaller intervals, referred to as *trials*. Data is then collected trial after trial in a sequential manner, and between every trial, the design, in our case the stimulus, for the next trial will be optimised given the already gathered data. Together, these trials then constitute a continuous data set, and the aim is to minimise the number of trials N, in order to achieve sufficiently good inference. After every trial, our inference method from section 4.3.1 will be applied to all of the currently gathered data to measure the inference accuracy as a function of the number of trials collected, N.

4.4.2 Utility function

In order for this active learning approach to be useful, we need to find an appropriate utility function for our specific problem to solve an optimisation problem on the form of Eq. (3.50), as explained in section 3.7.

Let X^n be an experimental design, i.e. stimulation frequency for trial n. For a lighter notation, let $\mathcal{D}^n = \{\boldsymbol{s}_1^n, \boldsymbol{s}_2^n\}$ be the data i.e. the spike trains collected in trial n. Here, we have defined $\boldsymbol{s}_i^n = s_i^{a_n:b_n}$, where a_n and b_n are the first and the last time points, respectively, in trial n. Furthermore, let $\mathcal{D}_+^n = \bigcup_{k=1}^n \mathcal{D}^k$ be the data from all trials up until and including trial n. Since our goal is to accurately estimate the learning rule parameters θ, we want the data to contain information about θ, and vice versa. Motivated by the theoretical foundation in section 3.7, we choose our utility function, u, to be the mutual information between the data, $\mathcal{D}^n$, and the parameters, θ, conditioned on the design X^n and the previously collected data $\mathcal{D}_+^{n-1}$

$$u(\mathcal{D}, \theta, X) = I_e(\theta, \mathcal{D}^n | X^n, \mathcal{D}_+^{n-1}). \tag{4.12}$$

Note that $\mathcal{D}$ denotes the data, while in section 3.7 the data was denoted by Y. Also notice the dependency on $\mathcal{D}_+^{n-1}$, being the already collected data, since we are optimising in a sequential manner. The mutual information is defined as in section 3.6, which in our notation and framework results in:

$$I_e(\theta, \mathcal{D}^n | X^n, \mathcal{D}_+^{n-1}) = \sum_{\mathcal{D}^n} \int P(\theta, \mathcal{D}^n | X^n, \mathcal{D}_+^{n-1}) \ln\Big(\frac{P(\theta, \mathcal{D}^n | X^n, \mathcal{D}_+^{n-1})}{P(\theta | X^n, \mathcal{D}_+^{n-1}) P(\mathcal{D}^n | X^n, \mathcal{D}_+^{n-1})} \Big) d\theta, \tag{4.13}$$

where we have used the natural logarithm. The expression in Eq. (4.13) can be simplified to

$$I_e(\theta, \mathcal{D}^n | X^n, \mathcal{D}_+^{n-1}) = -\sum_{D^n} P(D^n | X^n, \mathcal{D}_+^{n-1}) \cdot H_e(\theta | \mathcal{D}^n, X^n, \mathcal{D}_+^{n-1}) \quad + \quad \text{const}, \tag{4.14}$$

where $H_e(\theta | \mathcal{D}^n, X^n, \mathcal{D}_+^{n-1})$ is the posterior entropy of θ, defined as in Eq. (3.32). Now notice that Eq. (4.14) is the same as Eq. (3.52), with the exception that we have substituted the integral with the sum since the spike trains are discrete random variables, and we are conditioning here on the history $\mathcal{D}_+^{n-1}$ due to our active learning approach.

Approximation

As discussed in section 3.7, we also exploit a combination of MCMC techniques and importance sampling to get an estimation of the utility function, in our case Eq. (4.14). Our approach is just slightly different, as we approximate $H_e(\theta | \mathcal{D}^n, X^n, \mathcal{D}_+^{n-1})$ directly, by constructing an MCMC sample with our inference procedure, which yields an empirical distribution

for $P(\theta|\mathcal{D}^n, X^n, \mathcal{D}_+^{n-1})$. We make an assumption that $P(\theta|\mathcal{D}^n, X^n, \mathcal{D}_+^{n-1})$ is approximately normally distributed and fit a multivariate normal distribution to the sample

$$P(\theta|\mathcal{D}^n, X^n, \mathcal{D}_+^{n-1}) \approx \mathcal{N}(\boldsymbol{\mu}, \Sigma), \tag{4.15}$$

where $\boldsymbol{\mu}$ and Σ are the mean and the covariance matrix of the MCMC sample of size L.

Moreover, our final step is to apply importance sampling to obtain an eventual estimator of Eq. (4.14). By drawing M realisations of the data D^n according to $P(D^n|X^n, \mathcal{D}_+^{n-1})$, our utility function can be estimated by importance sampling as in section 3.5

$$I_e(\theta, \mathcal{D}^n|X^n, \mathcal{D}_+^{n-1}) \approx -\frac{1}{M}\sum_{j=1}^{M} H_e(\theta|\mathcal{D}_{(j)}^n, X^n, \mathcal{D}_+^{n-1}), \tag{4.16}$$

where j labels the realisation from the spike train space and $H_e(\theta|\mathcal{D}_{(j)}^n, X^n, \mathcal{D}_+^{n-1})$ is the differential entropy of the multivariate normal distribution as derived in section 3.6. More precisely, this is done by drawing realisations from $P(D^n|\hat{\theta}, X^n, \mathcal{D}_+^{n-1})$, where $\hat{\theta}$ is the mean estimates of the learning rule parameters obtained from inference on the already collected data, $\mathcal{D}_+^{n-1}$. So before generating the actual data for trial n, we simulate M realisations from the spike train space and construct an MCMC sample of size L for each realisation using the particle Metropolis-Hastings algorithm described in section 4.3.1. These realisations together yield an estimate of our utility function by Eq. (4.16).

Choice of hyperparameters for the MCMC approximation and the importance sampling are inspired by the discussion in section 3.7. Using similar sampling techniques as in the theory, we make the assumption of similar asymptotic properties of our estimator, recall Eq. (3.56) and (3.57). Because we don't know the form of $C(X)$ from Eq. (3.57), we make an assumption of $C(X) \approx$ const, for simplification. This supports parameters where $L >> M$. For our experiments L is chosen to be 1200, while M is 15. This way, we assume lower variance of our estimator, which we believe will make the comparison between designs more robust.

4.4.3 Algorithm

The algorithm is constructed by defining a grid in the space of frequencies, $\boldsymbol{X} = \{X_1, X_2, ...\}$, over which we search for the optimal stimulation. This grid is fixed for the all the trials. Which design i.e. frequency is used to collect data for trial n, is decided by estimating the utility function by Eq. (4.16) for all of the considered frequencies in our grid, and choosing the one maximising Eq. (4.16).

As explained in section 4.4.2, estimating the utility function involves performing inference of the learning rule parameters via the particle Metropolis-Hastings sampler. For this purpose, a couple of modifications are being made to the Metropolis-Hastings procedure from section 4.3.1. We want to approximate Eq. (4.16) conditioned on all the collected data, $\mathcal{D}_+^{n-1}$. However, the MCMC sample constituting (4.15) is obtained from only realisations of $\mathcal{D}^n$. The way we condition on $\mathcal{D}_+^{n-1}$ is by adjusting the prior distribution. We set the prior in trial n to be a multivariate Gaussian with mean and variance based on our inference on $\mathcal{D}_+^{n-1}$,

$$P_{\text{prior}}^n(\theta) = \mathcal{N}(\hat{\theta}, \hat{\Sigma}), \tag{4.17}$$

where $\hat{\theta}$ is the mean estimator from the MCMC sample on $\mathcal{D}_+^{n-1}$ and $\hat{\Sigma}$ is the corresponding covariance matrix. This way, the information from the already gathered data $\mathcal{D}_+^{n-1}$ is incorporated in the inference to obtain Eq. (4.15), although only applying the PMCMC to data realisations from trial n.

To boost the efficiency of the algorithm, we adjust the initial proposal distribution in the Metropolis Hastings to a gamma with the same mean and variance,

$$P^n_{\text{prop}}(\theta) \sim \text{Gamma}(\hat{\theta}, \hat{\Sigma}). \tag{4.18}$$

This way, the PMCMC algorithm is initialised in what we believe is the interesting area of the parameter space, based on inference on $\mathcal{D}^{n-1}_+$. Note that this is done similarly to how the proposal distribution was constructed in section 4.3.1, namely separately for A_+ and τ such that the proposal is a product of two gammas.

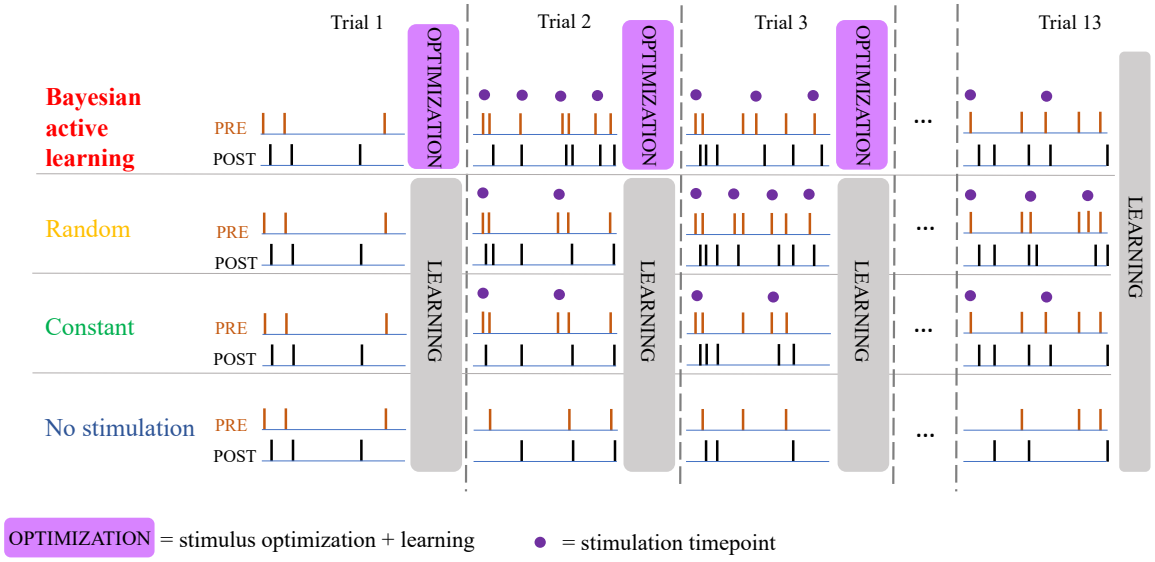

Figure 4.2: Experimental setting and inference for the considered stimulation protocols. The Bayesian active learning procedure optimises the stimulus for the next trial based on the inference of the STDP parameters from the history of spike trains.

We compare the performance of our inference when using Bayesian active learning with other stimulation protocols. These being *no stimulation*, *20Hz stimulation* and *random stimulation*. For the random stimulation, we use the same frequency grid as we optimise over, but then at every trial, the stimulation is drawn randomly from the grid from a uniform distribution. For comparison purposes, we do a couple of adjustments. Firstly, we let all of the protocols start with the exact same data for only the first trial. Secondly, the stationary parameters of interest, $w^{t=1}$ and b_2, are not being learned for any of the protocols, to avoid potential variance and bias (which we showed in [55]) affecting the inference. An illustrative cartoon of the compared protocols is presented in figure 4.2.

The design optimisation algorithm is presented in Algorithm 3. The full algorithm for collecting all the data and doing inference, with the embedded design optimiser, is summarised in Algorithm 4.

Algorithm 3: StimuliOptimiser. Function for optimising the upcoming stimulation. PMCMC*: PMCMC algorithm with updated prior and proposal distributions according to Eq. (4.17) and (4.18).

Define frequency grid $\boldsymbol{X} = [X_1, X_2, ...]$;
for X *in* $\boldsymbol{X}$ **do**
  for $j = 1, 2, ..., M$ **do**
    Generate $\mathcal{D}^n_{(j)} \sim P(\mathcal{D}^n | \hat{\theta}^n, X, \mathcal{D}^{n-1}_+)$;
    Run PMCMC* to estimate $P(\theta | \mathcal{D}^n_{(j)}, X^n, \mathcal{D}^{n-1}_+)$;
    Calculate entropy of the approximation Eq. (4.15);
  end
  Estimate mutual information Eq. (4.16)
end
Return X in $\boldsymbol{X}$ maximising Eq. (4.16)

Algorithm 4: Full Bayesian optimal design

Generate initial data $\mathcal{D}^1$;
Run PMCMC to estimate $P(\theta | \mathcal{D}^1)$;
Calculate $\hat{\theta}^1$, $\hat{\Sigma}^1$;
for $n = 2, 3, ...$ **do**
  Update $P^n_{\text{prior}}(\theta)$ Eq. (4.17) and $P^n_{\text{prop}}(\theta)$ Eq. (4.18);
  X^n = StimuliOptimiser (Algorithm 3);
  Generate $\mathcal{D}^n \sim P(\mathcal{D}^n | \theta_{\text{true}}, X^n, \mathcal{D}^{n-1}_+)$;
  Run PMCMC to estimate $P(\theta | \mathcal{D}^n, X^n, \mathcal{D}^{n-1}_+)$;
  Calculate $\hat{\theta}^n$, $\hat{\Sigma}^n$
end

Chapter 5

Results

In this chapter, our methods will be probed in several different settings to analyse their performance and investigate their behaviour.

In section 5.1 we discuss the remaining hyperparameters for the inference procedure, and justify our choices with experimental analysis. In section 5.2, we study our inference method in a real neural setting when applied to the data presented in 4.2.2. These explorations will serve as motivation for developing a Bayesian optimal design method. Furthermore, we will elaborate on and support these insights on real data by studying synthetic data in section 5.3.

Then, proceeding to our proposed Bayesian experimental design, we first extensively tested the PMCMC on single trial experiments, to explore the prospects of optimising the frequency based on single trial data. Also, this enabled us to make an educated decision for choosing the size of the trials used in the Bayesian optimal design. All of this is presented in section 5.4. Lastly, we present our results from the Bayesian design algorithm, where we compare our optimal design to other stimulation protocols and investigate the behaviour, the choices it makes and how it affects the exploration of the plasticity dynamics.

All of the results were acquired systematically to strengthen the robustness and validity of our analysis. To avoid specifying this several times, all of the results involving error bars or shaded error areas are obtained by conducting 20 unique experiments. The only exception is figure 5.10, which will be described separately.

Shaded areas corresponding to confidence intervals (see figure 5.9, 5.12 and 5.13) are calculated with built in functions of the Seaborn library in Python [66], which utilises bootstrapping techniques.

On synthetic data, the performance will often be measured according to the Root Mean Squared Error (RMSE)[67] measure between estimated learning rule parameters or learning rule functions, and the true parameters and corresponding function.

The computations are performed on resources provided by the IDUN/EPIC computing cluster [68].

5.1 Hyperparameters for inference

From the inference method described in 4.3.1, there are still some hyperparameters yet to have been discussed and chosen, being the level of noise used in the inference, σ, and the number of particles, P, employed by the particle filtering algorithm in section 4.3.1 to estimate the posterior of the weight trajectories. To determine these, the method was extensively probed, and a trade-off between adequate performance and computational complexity was made.

5.1.1 Choice of noise

The noise parameter σ enters in Eq. (4.2), where the weight connectivity is updated, which is used for sampling the particles in the particle filtering from section 4.3.1. In an experimental setting, one would not know the size of the underlying generative noise in the data, σ_{gen}. Therefore, we wish to either be able to infer or set this value independently of the generative value. Note that we previously tried to use our PMCMC method to infer the generative noise level[55]. However, as we concluded in that work, it was unsuccessful with the current method. Therefore, to instead determine a good choice for σ, the performance of the method was tested in different scenarios. Firstly, when the underlying noise in the generative process was low, and secondly, when the data was generated with a high noise level. Under both of these generative noise regimes, the inference method was tested at different levels of noise in the inference, and the most informative results were obtained by analysing the inference of A_+. The results are illustrated in figure 5.1, where the mean of the posterior mean estimates of A_+ across the experiments is plotted for different values of σ, along with the standard deviation across means.

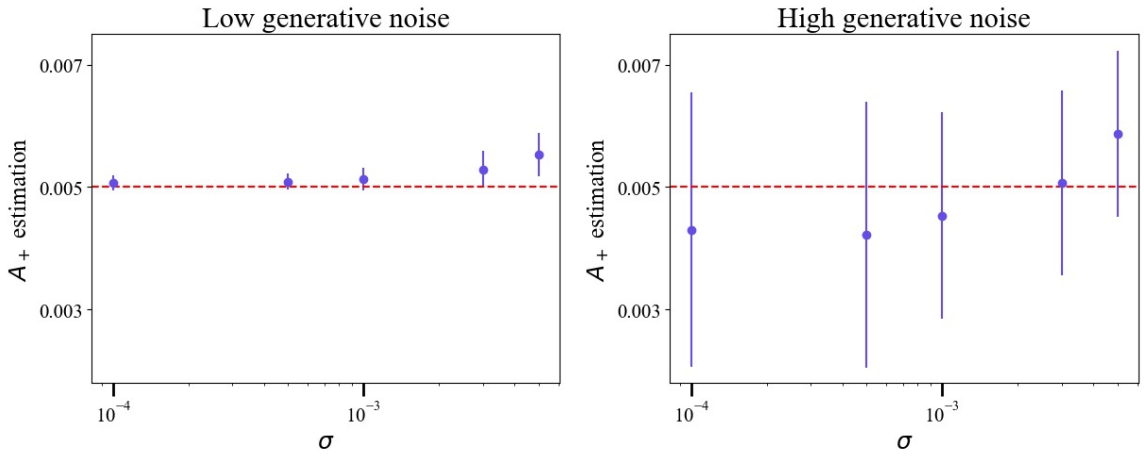

Figure 5.1: Estimation of A_+ as a function of noise level, σ, used in the inference. Dots show the average posterior mean across 20 data sets and error bars are one standard deviation of the average. Data sizes for inference were 120 seconds. Left: low generative noise, $\sigma_{\text{gen}} = 10^{-4}$. Right: high generative noise, $\sigma_{\text{gen}} = A_+ = 5 \times 10^{-3}$. The red dashed line marks the A_+ generative value.

The results suggest, that the estimates are being shifted in the same direction for both generative noise regimes, when increasing the inference noise. On the left plot, the generative noise, σ_{gen} equals the smallest one of those used for inference, σ: here we see that the accuracy slowly worsens when increasing the inference noise parameter. On the right plot, the generative noise is equal to the biggest one for inference. Here we see that the variance seems to decrease when the inference noise gets closer to the generative value. We believe, that this is due to the high generative noise leading to a lot more variation across the data sets, which again is being better captured when using a small noise in the inference. Moreover, for both regimes, we observe that the mean of the estimated A_+ increases with the inference noise, σ. We believe this could be due to a trade-off between A_+ and σ. When σ increases, the algorithm experiences bigger updates in weight trajectories from Eq. (4.2), which it will compensate for by overestimating A_+.

We interpret the data such that for low inference noise, the nature of the data is better captured. In the case of big inference noise, we get biased estimators regardless of the generative noise. Based on these plots, we conclude that the best option is to use a small noise for inference, when not knowing the underlying generative noise. This choice will be used for the rest of the results.

5.1.2 Choice of particles

As discussed in section 4.3.1, increasing the number of particles, P, is expected to decrease the variance of the particle filtering approximation. At the same time, increasing the particles leads to a linear increase in computational complexity. Therefore, we wish to find a particle number which serves good inference, but still keeps the computational cost as low as possible. For this purpose, we probed the inference method for a range of particle values, to get an empirical estimate of the variance associated with different values of P. For all of the particle numbers, the method was employed for the data sets, for better comparison. Mean of posterior means across experiments are showcased for A_+, with error bars indicating the standard deviation. From figure 5.2, it seems like we achieve satisfactory results already for $P = 50$, with a RMSE of $1-2\%$ and variance of the same order of magnitude as for $P = 5000$. This value was chosen for all the remaining experiments. Note that the minimal bias might be due to some variance in the estimation of $w^{t=1}$, which we discussed in more detail in our previous work[55].

5.2 Inference on real data

One of the most exciting explorations when developing a method, is to experience how it performs on real data. In this section, results from our inference method will be presented, when applied to real data. The method was tested for different data sizes, and both in stimulation and non-stimulation regimes. The results suggest that the method performs significantly worse with smaller data sets, and that the presence of stimulation might be important for uncovering learning rule parameters. This will serve as a motivation for why we want to develop a more sophisticated framework for optimising the stimulation protocol. The results which will be presented, are all from the same neuron pair from the real data explained in 4.2.2.

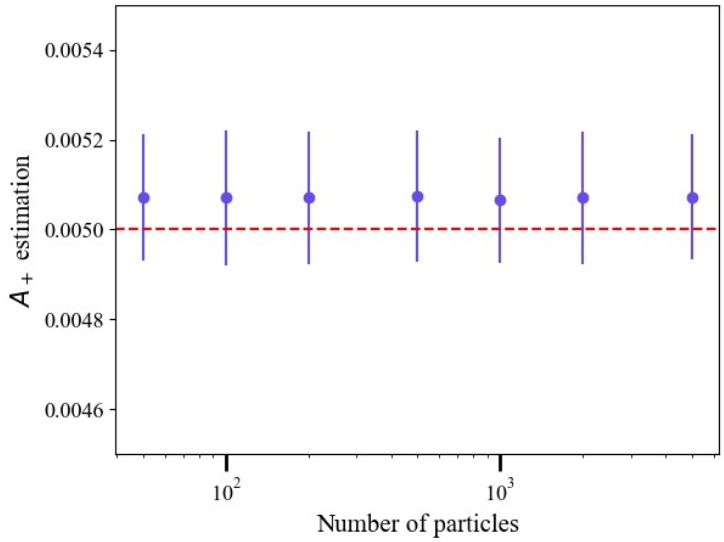

Figure 5.2: Estimation of A_+ as a function of number of particles, P, used in the inference. Dots show average posterior mean of A_+ across 20 experiments with error bars showing corresponding to the standard deviation. Data sizes were 120s. Red dotted line is generative A_+ value.

5.2.1 Detecting a monodirectional synapse

Before applying the inference method to real data, it is essential to find a good neuron pair which fulfills our model assumptions. That is, a neuron pair with a monosynaptically directed connection. Moreover, we want to investigate the effect of stimulation on the presynaptic neuron. Since the stimulated neuron is pre-defined in the data, we need to find a second neuron among the population which seems to be connected in the above mentioned manner. To do this, we analyse the cross-correlation between the spiking activity of neuron pairs, as explained in section 3.4. Experiments were run on smaller data sets (20 seconds) and bigger data sets (100 seconds). The data consists of around 2000 seconds of recordings in total, and we expect significant variability within the data. Therefore, we want to pick subsamples close to each other, such that the system is as stationary as possible within the considered time frame. Two 100 second subsamples were picked, such that they were disjoint but contiguous in time. The first sample consists of the last 100 seconds before delivering the stimulation, and the other sample of the first 100 seconds of stimulation. Smaller data sets (20 seconds) were picked to be again subsampled from these 100 second samples. In figure 5.3, cross-correlograms from the picked intervals of the selected neuron pair are shown.

The cross-correlations are constructed by shifting the spike train of the stimulated neuron for certain time lags. As it becomes clear from the plot, in general we seem to have the most significant correlation for negative time lags of 2-4ms of the stimulated neuron, suggesting that this would serve as the presynaptic neuron in the coupling. The significant cross-correlation at biologically plausible time lags for a monosynaptic connection[13], makes this a good candidate neurons pair. Significance is measured by the red line in the figures, which is calculated according to the hypothesis test defined in 3.4, and corresponds to the 99% confidence interval under H_0, which assumes no correlation. Based on this, we considered this a reasonable neuron pair for the method to be applied to.

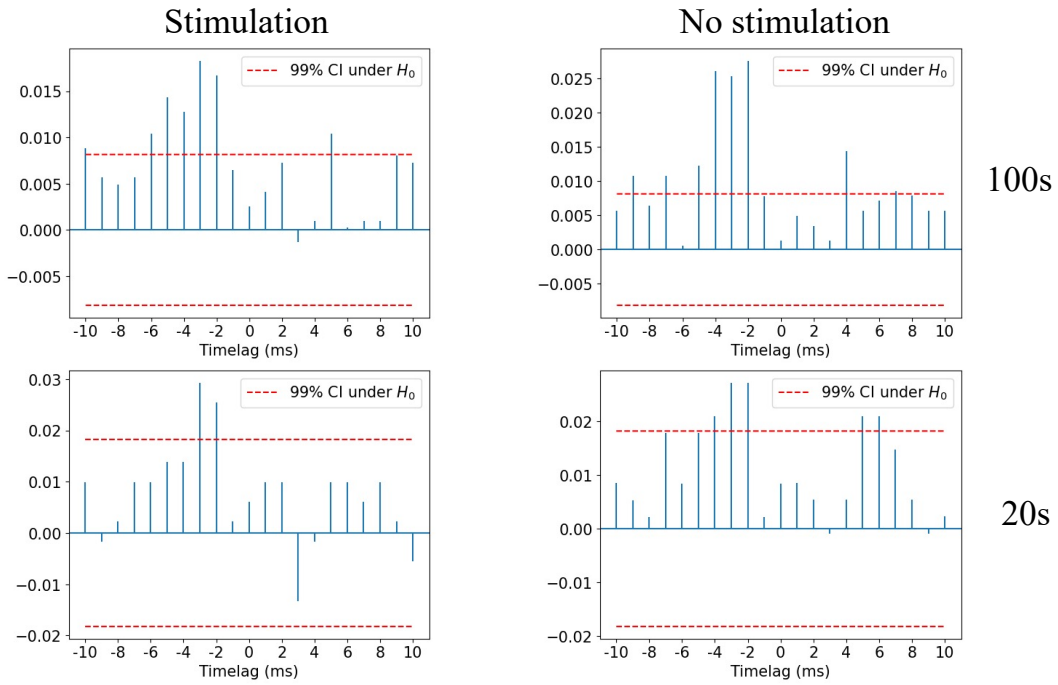

Figure 5.3: Estimated cross-correlation as a function of time lags of the presynaptic neuron. Top row: Cross-correlograms for 100 second data sets. Bottom row: Cross-correlograms for 20 second data sets. Left column: Stimulation of presynaptic neuron. Right column: No stimulation of presynaptic neuron.

5.2.2 Real data posteriors

Applying the method on data from this pair of neurons, yields us posterior distributions for both of our parameters, A_+ and τ. To measure the performance on the real data, where we do not know the underlying values, we mainly consider two things. Firstly, we want the entropy of the posteriors to be as small as possible, which indicates that the uncertainty connected to the estimates is small. Secondly, we also want the method to reproduce the same results, when applied to a different data set, assuming that the underlying parameters still are the same.

In figure 5.4, posterior distributions of A_+ are shown. We can clearly observe that the inference accuracy is superior when using 100 second recordings (A,B,C,D), compared to 20 second recordings (E,F,G,H), based on the dispersion of the distributions. Moreover, we see the improvement for the 100 second recordings, when an external stimulus is applied. The posteriors are in this case even more peaked, and they seem to be better reproduced for a second data set. For shorter recordings of 20 seconds, most of the posteriors seem to be strongly dependent on the prior distribution, and the presence of stimulation does not seem to improve the performance.

Figure 5.5 shows posteriors of τ obtained from the same data. We see that the results are not particularly good for either of the data sizes. The posteriors mostly seem to be dependent on the prior, and the presence of stimulation does not seem to make a difference. We have previously discovered, that the inference of τ seems to demand a smaller bin size δ to be adequate[55], compared to A_+. Although we showed that on synthetic data $\delta = 2$ms yields good results, we hypothesise that a bin size of $\delta = 2$ms might be too big to be able to characterise the features of τ for some certain underlying values, which might be a reason for these results.

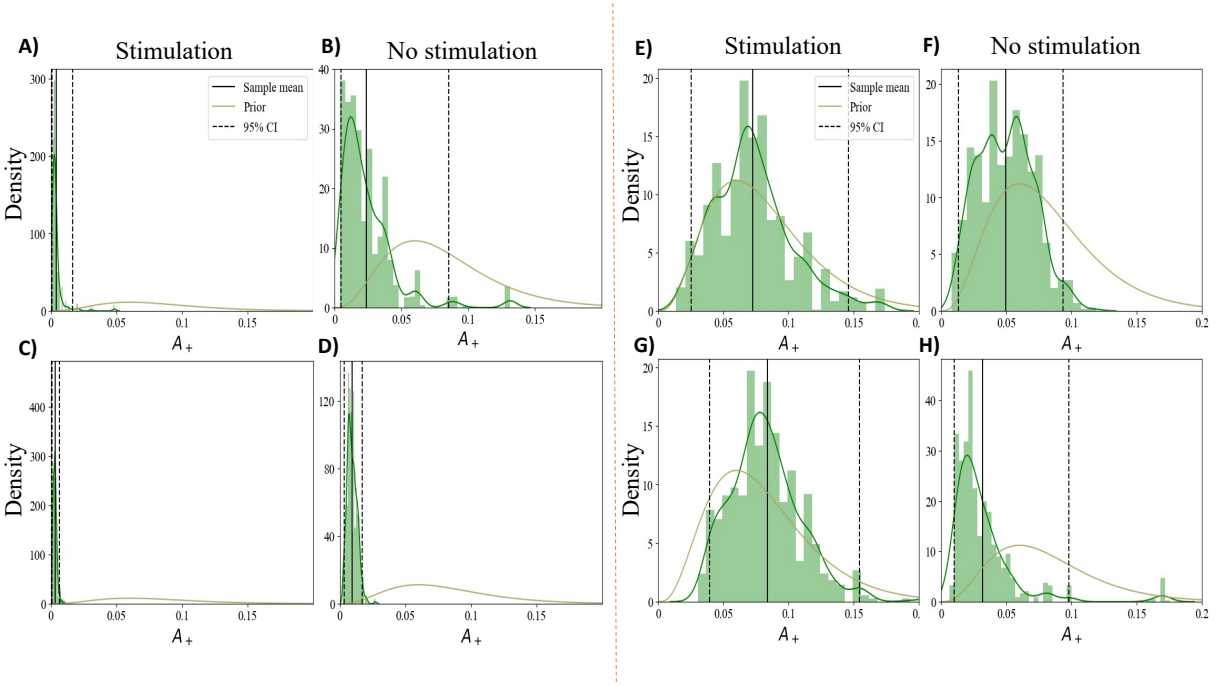

Figure 5.4: Posterior distributions for A_+. Dotted lines are 95% credible intervals. The yellow line is the prior distribution. (A,B,C,D): 100 second data sets. (E,F,G,H): 20 second data sets. (A,C,E,G): With stimulation. (B,D,F,H): Without stimulation. Solid green lines were obtained by Gaussian kernel density estimation on the sample posteriors, using Seaborn[66].

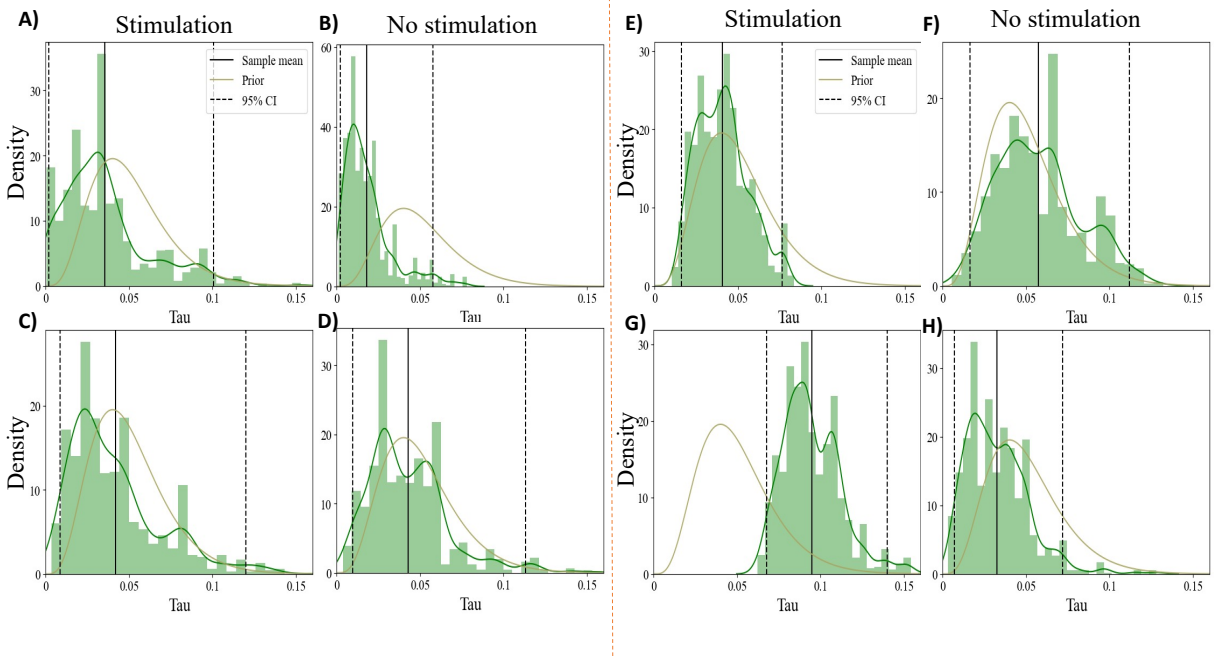

Figure 5.5: Posterior distributions for τ. Dotted lines are 95% credible intervals. The yellow line is the prior distribution. (A,B,C,D): 100 second data sets. (E,F,G,H): 20 second data sets. (A,C,E,G): With stimulation. (B,D,F,H): Without stimulation. Solid green lines were obtained by Gaussian kernel density estimation on the sample posteriors, using Seaborn[66].

5.2.3 Reproducibility

In addition to studying the distributions of the parameters separately, we can study the resulting learning rule. After all, this is the most interesting feature, and the one we want to be able to characterise. In figure 5.6 we show the estimated learning rules for all the considered paradigms. For all of the settings, i.e. same data size and stimulation, we show two learning rules obtained from disjoint data sets.

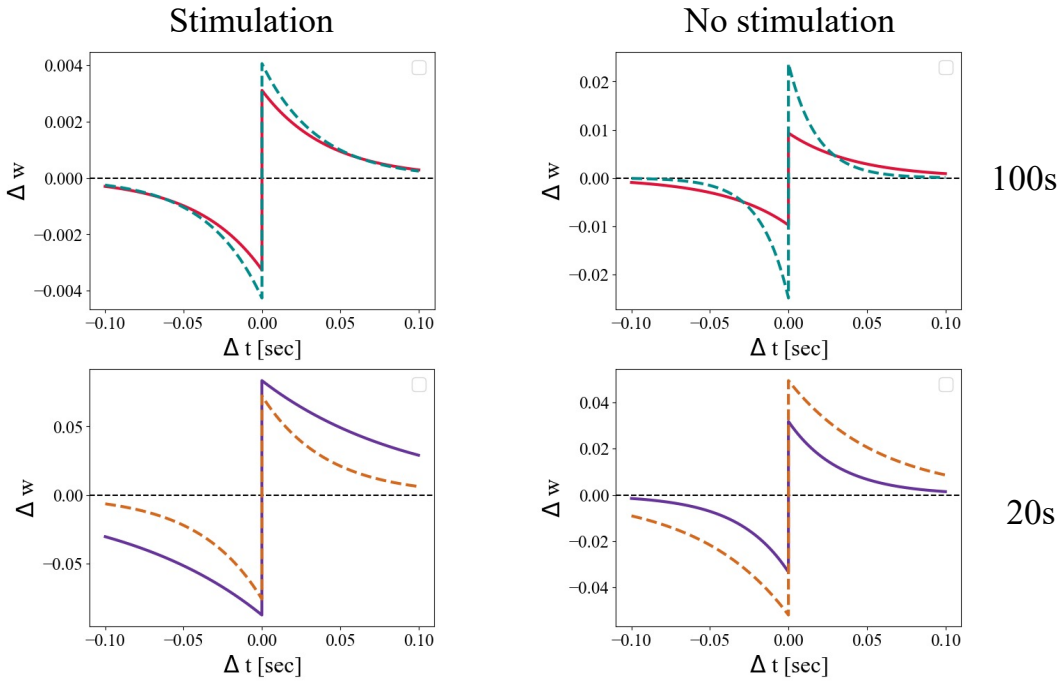

Figure 5.6: Learning rules based on mean estimates of A and τ. The two lines in the same plot are from different data sets with identical data sizes and stimulation regimes to measure reproducibility. Top row: 100 second data sizes. Bottom row: 20s. First column: With stimulation. Second column: Without stimulation.

We see that the learning rule is best retrieved when we have long recordings and external stimulation.

Based on these results, the method seems to require relatively long recordings when applied real data, which might question its applicability. Also, it seems evident that external stimulation might be important for reconstructing the learning rule from real data. This motivates the idea of developing an optimal design framework to make the method more applicable on real neural recordings. To study the effect of the data length and stimulation more extensively, we turn to similar experiments on synthetic data.

5.3 Data size analysis on synthetic data

For the synthetic data, we first study how accurately the method recovers A_+ and τ, as a function of the data length. For these experiments, only one parameter is being inferred at a time, and the other parameter is being fixed in the inference. Figure 5.7 shows the average posterior means across experiments with standard deviation, for both A_+ (left) and τ (right), in the absence of stimulation. We observe that the accuracy clearly improves for both parameters when the data length increases. However, this also increases the computational complexity.

Therefore, we wish to study other stimulation regimes, where the estimates could hopefully converge faster towards the true value. Stimulation is always applied to the presynaptic neuron.

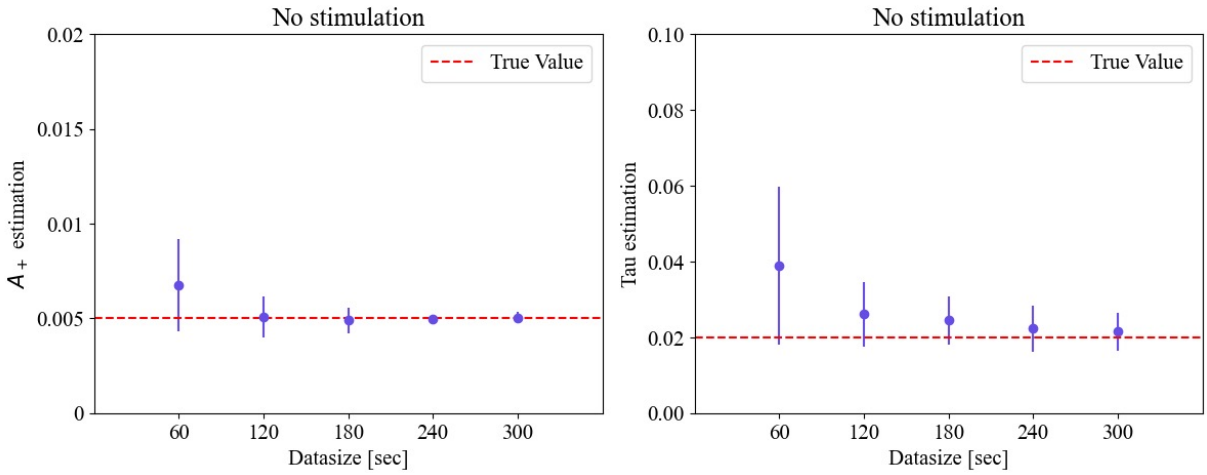

Figure 5.7: Estimation of A_+ (left) and τ (right) as a function of data size, without external stimulation on the presynaptic neuron, $b^t_{1,ext} = 0$. The resulting total external field is $b^t_1 = -3.1$. Red dotted lines are generative parameter values.

In figure 5.8, a similar analysis is presented, but where we add an external stimulation on the presynaptic neuron, such that the external field the parameter $b^t_1 = -1.5$. This increases the stochastic firing rate to $\approx$ 100Hz, which is a common stimulation protocol used for plasticity studies, recall section 2.3. From the plots, it becomes clear that for this stimulus, the accuracy improves, in particular for the shortest data sizes. This further suggests, that if we want to minimise the data length, choosing a good stimulus seems to be essential. But then the question arises, what is a good stimulus? Is it such that the stronger the stimulation is, the better the method performs? To try to answer this question, we study how well the learning rule is recovered, for three different stimulation protocols. Now, both parameters A_+ and τ are being inferred, and the performance is measured by calculating the RMSE between the resulting estimated learning rule and the true learning rule. The protocols which are compared, are *no stimulation*, 100Hz stimulation and 250Hz stimulation. In these experiments, the stimulation is tetanic, recall section 4.1.1. Results are plotted as the mean RMSE estimate across the experiments, with a corresponding 95% confidence interval to the estimate.

From the results in figure 5.9, we observe, that the highest stimulation is not the best choice. As expected from figure 5.7, without external stimulation, the accuracy is bad for low data sizes, but converges pretty consistently when accumulating more data. As expected from figure 5.8, the method is performing significantly better when delivering a 100Hz stimulation to the presynaptic neuron. However, we see that if we stimulate at an even higher frequency (250Hz), the performance is degraded compared to 100Hz. This suggests that the stimulation could be tuned and that there could exist some optimal stimulation between the extremes of no stimulation and 250Hz stimulation.

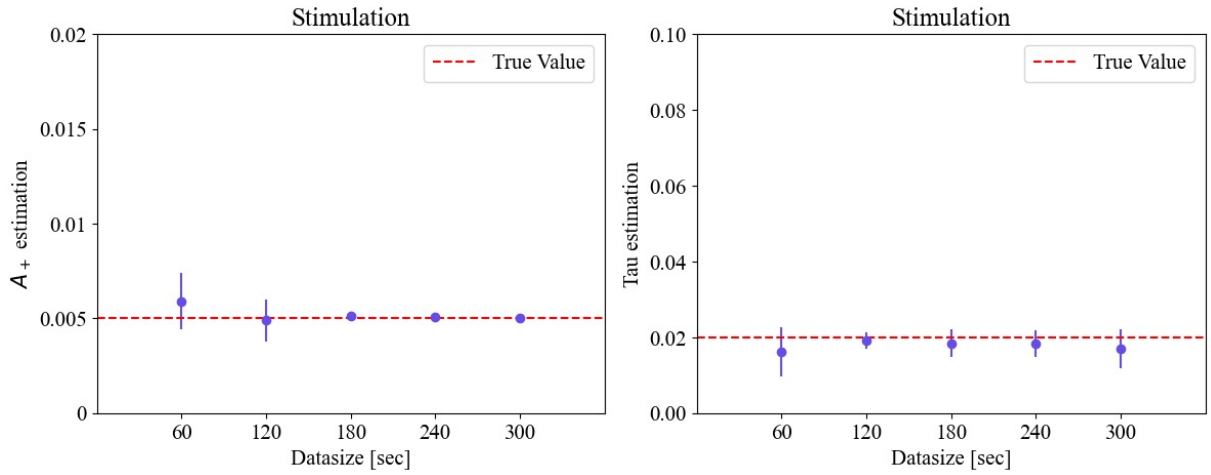

Figure 5.8: Estimation of A_+ (left) and τ (right) as a function of data size, with external stimulation on the presynaptic neuron, $b^t_{n,ext} = 1.6$. The resulting total external field is $b^t_1 = -1.5$. Red dotted lines are generative parameter values.

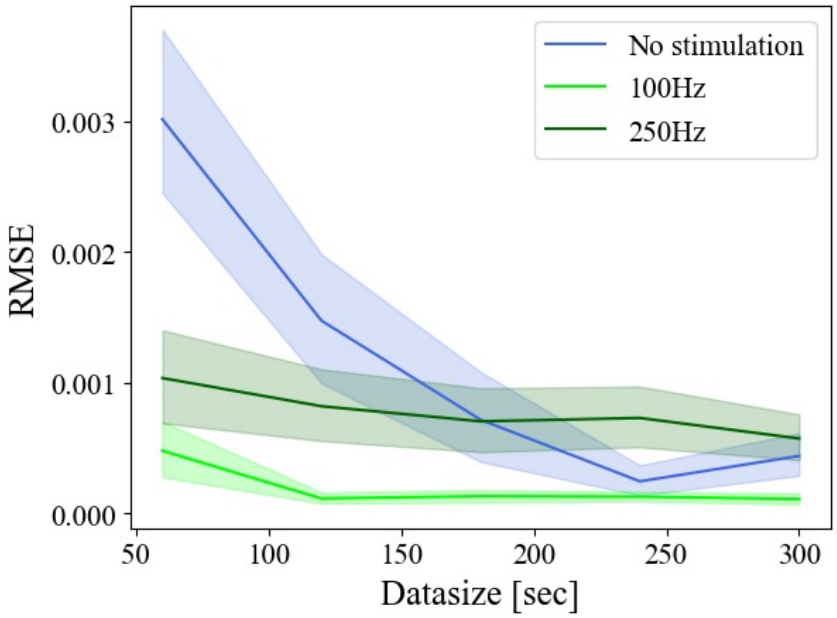

Figure 5.9: RMSE of learning rule versus the data size, for three different stimulation regimes. Lines are average RMSE across experiments, while shaded regions are 95% confidence intervals.

5.4 Single trial experimenting

Before probing the full Bayesian design method, we did extensive preparation in terms of probing the method on single trial experiments. Firstly, to see to what extent stimulation is

important on such short recordings. If the performance accuracy is easily distinguishable for different stimulation protocols also on really short data, we believe that our optimisation framework in section 3.7 could also identify the optimal protocols, only based on single trial data. Secondly, an important hyperparameter is the length of the trials, which needs to be determined, and the behaviour of the algorithm for different trial sizes would be important to make this choice. We considered trial sizes of 1s, 5s and 10s.

5.4.1 General performance and stimulation

To study the behaviour of our method on single trial level, we simulated 120s of data, which was divided into disjoint trials of different sizes. The method was then probed on these disjoint trials, for different strengths of stimulation. In figure 5.10, the performance of the method in terms of estimating A_+ is illustrated for trials sizes of 1s and 5s, and for *no stimulation* and 100Hz stimulation. Note that the underlying data is identical for the same stimulation, before being subsampled into different trial sizes. The top row shows the mean estimator of A_+ on each of the chronologically ordered disjoint trials. Note that in this plot, every estimation is only from one trial. The error bars correspond to the standard deviation of the respective posterior distribution, while the bottom row shows the corresponding weight trajectory and postsynaptic spike rate for the data in the course of the entire experiment.

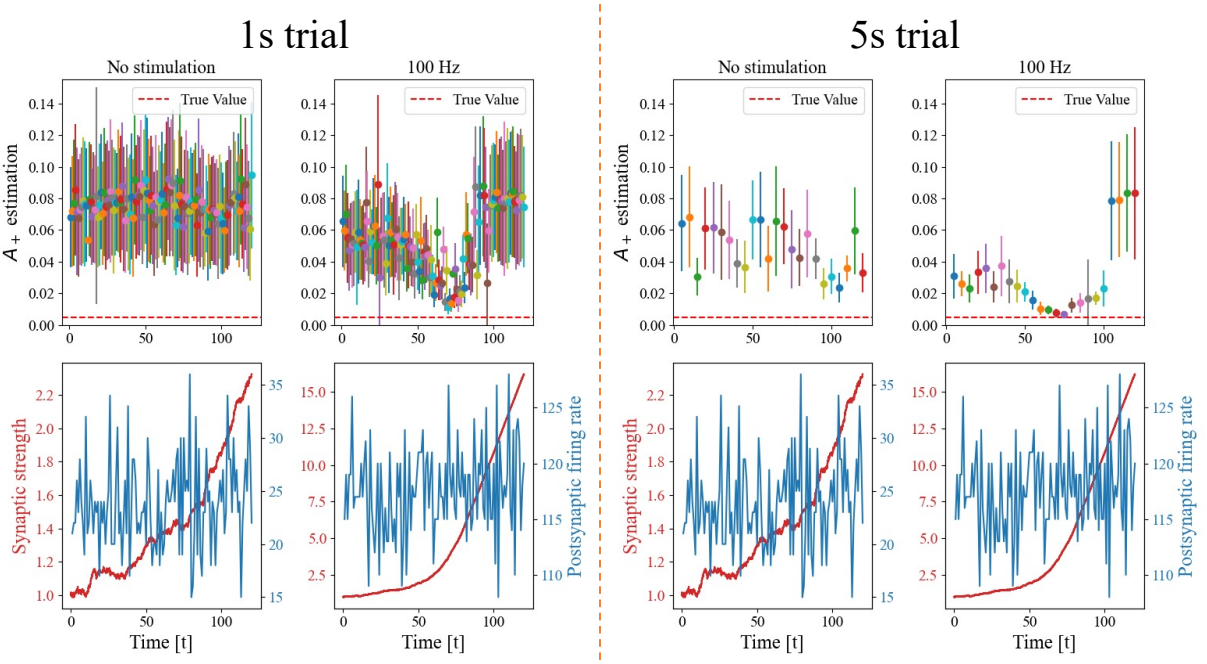

Figure 5.10: Estimation of A_+ as a function of time for disjoint trials. Top row: Estimates of A_+ on single trial experiments as a function of time. Bottom row: postsynaptic spike rates (blue) and weight trajectories (red). Column 1,3: No stimulation. Column 2,4: 100Hz stimulation. Column 1,2: 1 second trial size. Column 3,4: 5 second trial size.

We observe that even for such small data lengths, the performance seems to be significantly improved by applying stimulation. This is promising for the prospect of applying the Bayesian optimal experimental design. However, notice the variability of performance also within the same data size and stimulation regime, in particular for the 100Hz experiments. The performance generally seems to improve from trial to trial as the experiment progresses,

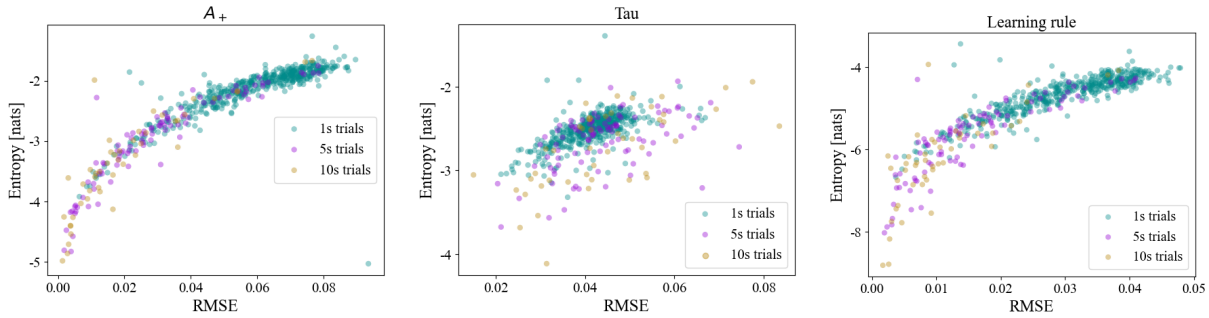

Figure 5.11: Entropy of the posterior distribution for A_+ (left), τ (middle) and $[A_+, \tau]$ in 2D space (right) as a function of the RMSE of the estimates for A_+ (left), τ (middle) and learning rule (right). Color codes indicate trial sizes used for the samples.

but at some point, the performance heavily degrades for the last set of trials. From the bottom row we observe that there is low variation of spike rates across the trials, but we see that the weight strength is increasing, and at some point escalating. Therefore, the accuracy is assumed to also be heavily dependent on the connectivity strength, which makes us believe that the optimal stimulation may differ for different connectivity settings. This hypothesis is explored in section 5.4.2.

From the first column in figure 5.10, it also seems evident that the standard deviation, and in turn also the posterior entropy, looks smaller when the estimates are closer to the true value. To study this interesting correlation, inference results from all of the experiments for the three different trial sizes and two more stimulation regimes which are not showcased (20Hz and 50Hz), were gathered in figure 5.11. On the x-axis, we have the RMSE of the estimated A_+ (left), τ (middle) and resulting learning rule (right). The y-axis is the calculated posterior entropy of A_+ (left), τ (middle) and $[A_+, \tau]$ in the 2D parameter space.

The plots suggest some clear correlation between the entropy of the MCMC samples and the corresponding error of our estimates. For A_+, the correlation looks logarithmic, while for τ it looks more linear. Together they yield a logarithmic looking correlation on the desired learning rule. Recall from section 3.7 that our utility function can be written as the negative expected posterior entropy with respect to the data. Hence, our optimisation aims to minimise the posterior entropy. Therefore, these plots also suggest that the optimisation would in turn minimise the RMSE of our estimates, which is encouraging. Also, we observe that the lowest entropies and RMSE are obtained from trial sizes of 5 and 10 second trials. To develop a flexible design algorithm, we would like to have a small trial size, such that the algorithm can adapt the stimulation more often given the nature of the data. However, this also increases the computational complexity, since it leads to more trials and we are doing inference on all the gathered data after every trial.

5.4.2 Optimal stimulation sensitivity to weight strength and generative learning rule

As discussed in the previous section, based on figure 5.10, we think that the optimal stimulation may depend on the underlying weight. In this section, we study to what extent this is true, and we investigate whether the generative learning rule also affects the optimal stimulation. In figure 5.9, the inference method was applied to single trial data with different initial values of $w^{t=1}$, experiments were done for the three different trial sizes and for two different learning rules. The figures show the mean RMSE of the estimated learning across experiments with associated 95% confidence intervals.

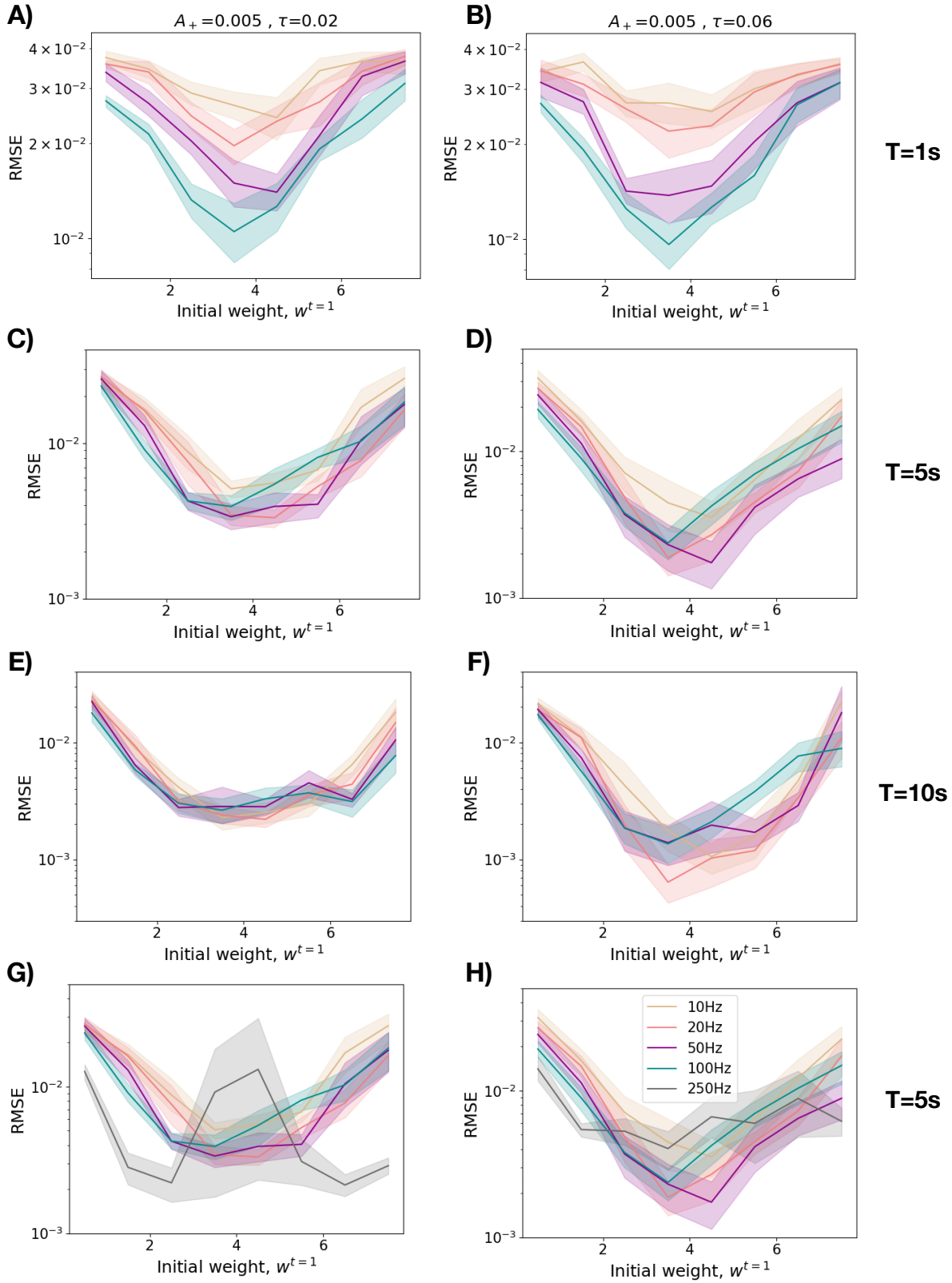

Figure 5.12: RMSE of the learning rule on single trial experiments as a function of the initial weight. (A,C,E,G): $A_+ = 0.005$, $\tau = 0.02$. (B,D,F,H): $A_+ = 0.005$, $\tau = 0.06$. (A,B): 1s trials. (C,D,G,H): 5s trials. (E,F): 10s trials. Color codes for different stimulation protocols are consistent in all plots. Labels are found in H). Shaded areas are 95% confidence intervals of the mean estimate across 20 experiments.

We observe that for trial sizes of 1s, the stronger stimulation seems to perform best regardless of the initial weight, for the weight strengths we considered. In plots (C,D) we see that for 5s trials the stronger stimulation seems to perform best for low weight strengths, but is then outperformed by both 50Hz and 20Hz for $w^{t=1} > 3$. We also see from C) that 20Hz seems to be the best choice for $A_+ = 0.005$, $\tau = 0.02$, while from D) it seems that 50Hz is the best choice for $A_+ = 0.005$, $\tau = 0.06$. (G,H) is equal to (C,D), except here the RMSE from data with stimulation of 250Hz is also added for comparison. From (G,H) we see that the performance of the 250Hz stimulation also shows variability between different learning rules. (E,F) also support our theory of a dynamic optimum across different underlying parameters. Interestingly, from the column 2, we also observe that there seems to be a sweet spot around $w^{t=1} = 3$, for which the RMSE is lowest for all the protocols. This also seems to be dependent on the learning rule, as the same conclusion can not be drawn from column 1, in particular we observe an opposite behaviour for 250Hz in G).

Based on the performance for different trial sizes and on the discussion in the previous section, we settled with a trial size of 5 seconds for the Bayesian optimal experimental design.

5.5 Bayesian optimal design

In this section, the developed Bayesian optimal experimental design algorithm from section 3.7 will be explored, analysed and compared to other stimulation protocols. The results are all obtained from synthetic data.

5.5.1 Main result

Finally, the performance of our full experimental design method will be explored. The optimisation was done for two different ranges of stimulation frequencies. The results from the four different stimulation protocols, which all are simulated according to figure 4.2, are shown in figure 5.13. The first considered frequency grid was $\{20, 50, 100, 250\}$Hz (left plot), while the second was $\{10, 20, 50, 100\}$Hz (right plot). The plots show the average RMSE of the estimated learning rule across experiments as a function of the number of trials of collected data. Shaded areas correspond to 95% confidence intervals of the mean RMSE estimator.

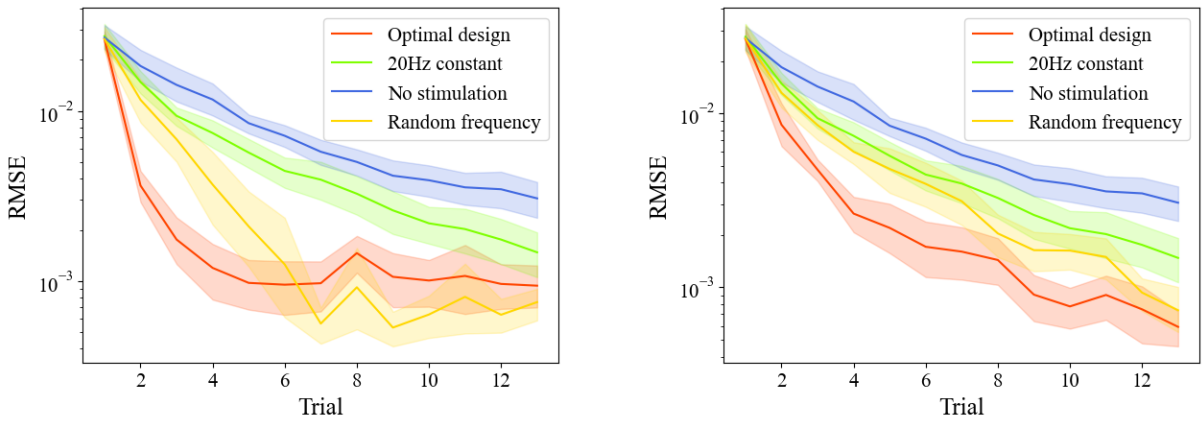

Figure 5.13: Average RMSE as a function of trials for different stimulation protocols, across experiments. Shaded regions correspond to 95% confidence intervals of the mean RMSE. Left: Optimisation on the frequency grid $\{20, 50, 100, 250\}$Hz. Right: Optimisation on the frequency grid $\{10, 20, 50, 100\}$Hz.

In both frequency domains, we observe that the optimised design method outperforms the other protocols in the beginning of the experiment. On the frequency space $\{10,20,50,100\}$Hz, the optimal design is superior for the whole experiment. However, on the $\{20,50,100,250\}$Hz frequency space, we see that the performance saturates, and is outperformed by the random design for the second half of the experiment. This behaviour will be further discussed in section 5.5.3. All things considered, we think these results are promising for this novel framework for studying plasticity. The optimal design algorithm seems to indeed manage to choose a stimulation at each trial, which leads to better data and better inference compared to standard protocols.

5.5.2 Mutual information

In addition to testing its performance, we wish to understand how exactly the optimal design behaves. What frequencies does it choose, and how does it affect the neuron coupling?

Firstly, we study the utility function which is used for optimisation. In figure 5.14, we have calculated the mutual information for a much finer grid of frequencies, at different trials of the experiment. In the left plot, the results are shown for realisations in trial number $2,3,4$ and 5, i.e. in the beginning of the experiment. In the right plot instead, we showcase trials from the later parts of the experiments, from trial number $7,9,11$ and 13. The x-axis shows increasingly ordered frequencies, while the normalised mutual information is plotted as bars, where different color shadings represent different trials. From the left plot we observe a clear trend, that the mutual information seems to increase with increased stimulation. However, on the right plot, we see that the mutual information seems to be a lot more equally distributed. Most of the peeks actually occur for the lowest frequencies. But notice the values on the y-axis, which indicates that the mutual information only deviates less than 1% across the different frequencies. So it seems that the algorithm does not clearly favour any of the frequencies for the later parts of the experiment.

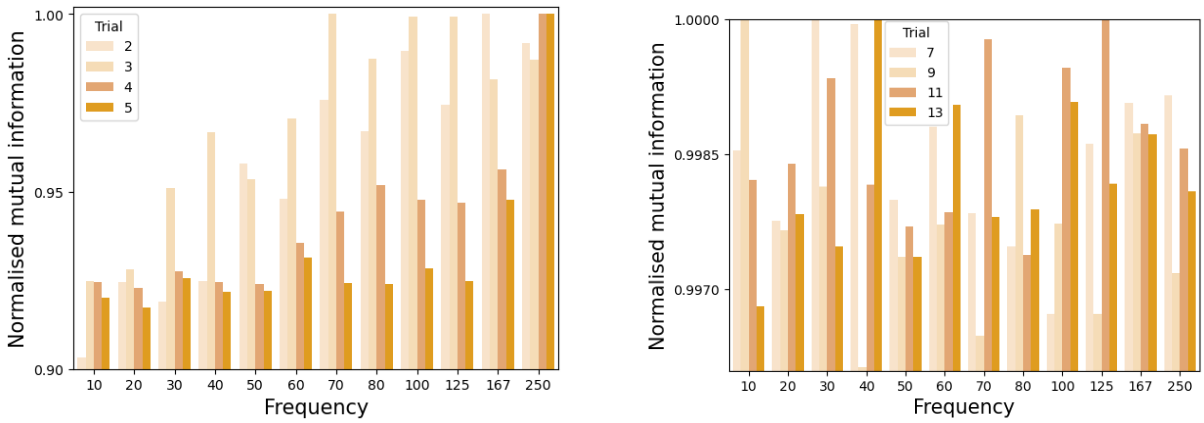

Figure 5.14: Normalised mutual information for a fine range of frequencies. X-axis indicates different stimulation frequencies. Colors indicate different trial numbers. Left: Early trials. Right: Late trials.

The same story is being told in figure 5.15. Here we present a heat map of which frequencies were chosen by the Bayesian framework, for the results shown in figure 5.13. We observe that for both frequency domains, the highest frequency is mostly chosen in the beginning, although not always. As the experiment progresses, the distribution of chosen frequencies becomes more uniform. However, we observe that in the last trial, the smallest stimulation is chosen most often in both grids.

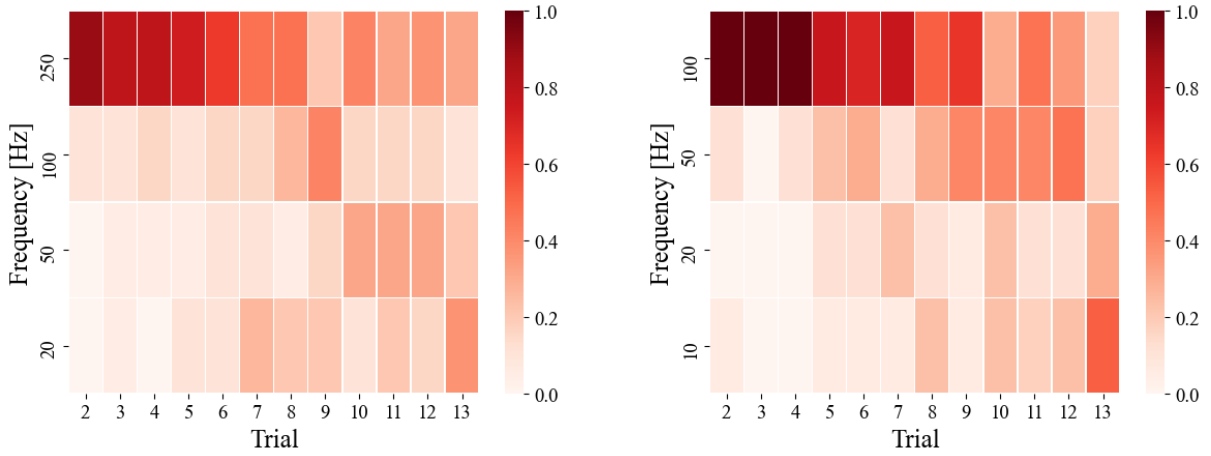

Figure 5.15: Heat map of chosen frequencies based on the Bayesian optimal design algorithm as a function of trials. Left: Optimisation on the frequency grid $\{20, 50, 100, 250\}$Hz. Right: Optimisation on the frequency grid $\{10, 20, 50, 100\}$Hz.

5.5.3 Weight trajectories

To gain a broader understanding of the performance of the different protocols, we analyse the weight trajectories of the experiments which yielded our main results in figure 5.13. In figure 5.16, the average weight trajectories for the four protocols from the 20 experiments are plotted, with corresponding error shadings being the standard deviation. In figure 5.17 we have plotted all of the weight trajectories for the two protocols with non-stationary stimulation, the optimal design and the randomised design. The trajectories are of different transparency, which indicates the inference performance of the individual experiments. Low transparency is used for experiments with lower RMSE of the estimated learning rule after all trials of data were collected.

Recall from our parameter setting from section 4.2, that $A_- = 1.05A_+$, i.e. our model actually emphasises synaptic depression over potentiation. In the left columns of figure 5.16 and 5.17, resulting from the frequency space including 250Hz, we observe that the weight trajectories for the optimal design converge to zero. This is due to a combination of our model being biased towards depression and our optimisation mainly choosing 250Hz for the first part of the experiment. Our bin size of $\delta = 2$ms means that for a 250Hz firing rate, the interspike intervals will be nearly deterministic and equal for both positive and negative time delays, which in turn leads to depression of the synaptic weight due to our generative model choice. Notice from figure 5.13, that the optimal design for this frequency grid actually saturates around halfway through the experiment, which indeed turns out to be approximately where the synaptic weights become 0 or close to 0. So in practice, when the neurons are not connected ($w = 0$), the plasticity parameters θ becomes difficult to infer, as under this condition, the activity of the postsynaptic neuron is independent of the presynaptic one.

From the right columns of figure 5.16 and 5.17, we have the same plots, but for the grid with maximum 100Hz frequency. All of these frequencies seem to generally increase the weight strength. We observe that the trajectories for all of the protocols are pretty similar and stationary in the beginning of the experiment, although the frequencies might differ a lot. This is also an assumption we make, recall section 4.3.2, that there is approximately no learning in the beginning of the experiment. In general, the average weight trajectories don't look too dissimilar, and since the optimal distribution of frequencies becomes more

uniform, the generated weight trajectories seem to be pretty similar to the random ones on average. Yet, we observed a superior performance by the optimal design in figure 5.13 on the $\{10, 20, 50, 100\}$Hz grid. This again emphasises the importance of choosing the correct stimulation given the exact data and synaptic strength, although the weight trajectories on average might share similarities. Our algorithm seems to accomplish sophisticated optimisation for short data recordings.

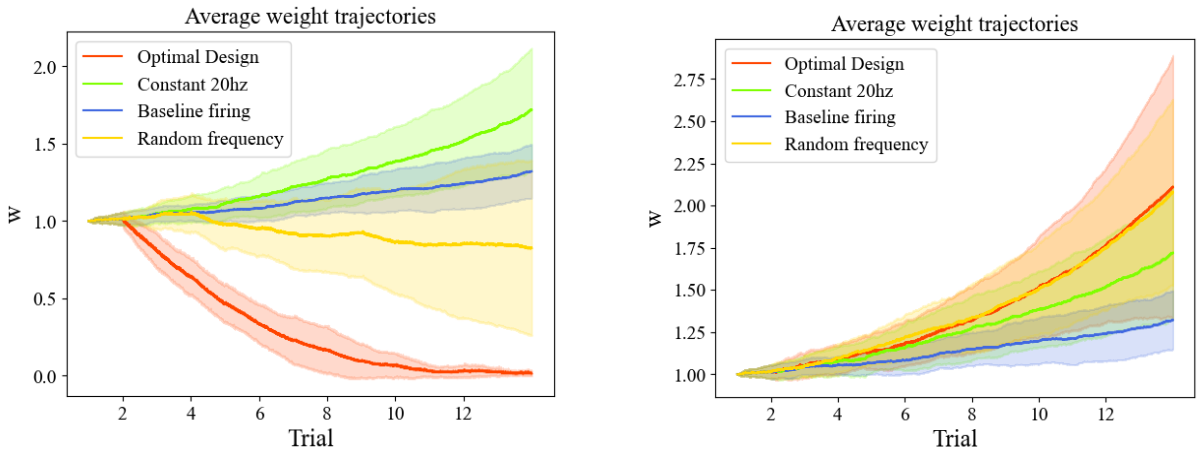

Figure 5.16: Average weight trajectories for the four different protocols across experiments as a function of trials. Shadings correspond to one standard deviation. Left: Optimisation on the frequency grid $\{20, 50, 100, 250\}$Hz. Right: Optimisation on the frequency grid $\{10, 20, 50, 100\}$Hz.

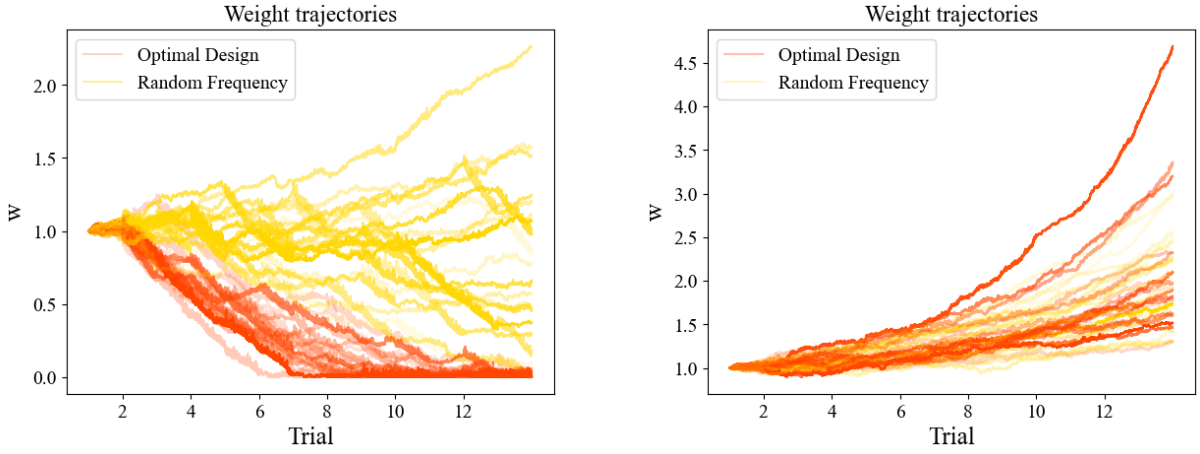

Figure 5.17: Weight trajectories for optimal (red) and random (yellow) protocols, as a function of trials. Transparency represents inference accuracy, stronger lines yielded better inference. Left: Optimisation on the frequency grid $\{20, 50, 100, 250\}$Hz. Right: Optimisation on the frequency grid $\{10, 20, 50, 100\}$Hz.

Chapter 6

Discussion

Our developed experimental design framework shows promising preliminary results. In this section, we will firstly elaborate on some interesting observations of our framework and discuss its advantages. Secondly, we consider the framework to have a lot of interesting prospects for the future, with several interesting mathematical possibilities, as well as other biological applications. Initial thoughts on these future directions will be discussed.

6.1 Observations and advantages

First of all, the Bayesian optimal experimental design seems to lead to significantly better inference accuracy for small data recordings. The optimised stimulation protocol was compared to three other protocols, being *no stimulation*, 20*Hz constant stimulation* and a *randomised stimulation*. For small data sets, i.e. just a few trials, the optimal design seems to achieve similar accuracy as the other protocols in approximately half the amount of trials, based on figure 5.13. Another interesting observation is the performance of the random stimulation. Based on the literature presented in section 2.3, applying a random stimulation whose frequency changes during the experiment is not a common practice in neuroscience. However, a randomised frequency and thus a randomly varying spike rate, probably leads to a bigger variety of sampled spiking delays, which might allow the algorithm to better explore different features of the parameter space, which is supported by the good performance displayed in figure 5.13.

This framework is useful, because even though one could argue based on the chosen frequencies in figure 5.15, that a constant stimulation of 100Hz would be equally good, this is not something one would know before the experiment in a practical setting. We have indeed shown in figure 5.9 that for different parameter settings the optimal frequency might be totally different. This algorithm would be able to detect which stimulation is indeed the best one for any parameter setting.

Furthermore, being able to significantly decrease the amount of required data, could open doors to other research questions and ways to study synaptic plasticity. The brain is a very complicated system, where plasticity can be induced and affected by several different factors. Examples could be neuromodulators such as dopamine[69], hormons[70], the effect of drugs[71] or sleep[72]. Such parameters are difficult to account for in a modelling framework, meaning they would instead add noise to the data, making it harder to correctly infer the plasticity dynamics. Therefore, data sizes on a time scale where the effect of these fac-

tors can be neglected, could allow for data giving rise to a better understanding of plasticity. Additionally, it could enable us to measure the impact of different short-term drugs in the context of plasticity and learning, or study plasticity during sleep.

Furthermore, the method could have an influential impact on addressing ethical and sustainability issues within computational neuroscience research. Research on synaptic plasticity often relies on experiments conducted on living species, where less required experimental time addresses ethical dilemmas associated with such methods. Also, computationally expensive data analysis has negative environmental impacts, wherefore our contribution is also beneficial for sustainability considerations.

6.2 Limitations and prospects

Even though the results are encouraging, we still see possible improvements in the developed framework. Firstly, the developed algorithm only chooses the optimal stimulation based on solving an optimisation problem for the next trial. In this sense, the algorithm can be considered *greedy*. This is most evidently reflected for the optimisation on the $\{20, 50, 100, 250\}$Hz frequency grid. The algorithm is choosing a high concentration of 250Hz stimulation in the beginning, without being able take the long term depression into consideration, which ultimately leads to a weight connectivity of zero and in turn a saturation of the inference performance. So the 250Hz, which was deemed optimal based on single trials in the beginning, was probably not the best choice in the long run. However, greedy Bayesian optimal design methods have still been shown to be superior to classical design methods under certain consistency conditions[73].

In terms of future applications of this method, we still consider improvements necessary for this to be used in an actual experimental setting. First and foremost, regarding the computational complexity. Although our discussed optimisation algorithm is well suited for parallelisation, which would save a lot of computational time, it would still be too slow. This issue is also discussed in [51], where a variational inference approximation was applied instead of an MCMC method to calculate desired posteriors. This reduces the computations significantly, which actually could make it feasible for application in a live data collection setting.

Furthermore, extending the application from just a neuron pair to a bigger network of neurons could be an exciting prospect. This would cause more computations, which in turn would demand previously mentioned method improvements. However, this could allow for explorations of other interesting experimental questions, such as deciding how many and which neurons should be stimulated to study plasticity on a bigger scale.

We believe, that our study has potential for future applications of big societal impact. In section 2.4 we introduced Alzheimer's disease, being a disorder that more and more people will suffer from. In an aging society, the number of people affected by dementia worldwide is estimated to rise from 46 million in 2015 to 131.5 million in 2050[74]. In the long term, our study may contribute to addressing this imperative health challenge and inform treatments targeting these memory related diseases, by uncovering dysfunctionalities at the phenomenological level pointing out dysfunctionalities at the molecular level.

Chapter 7

Conclusion

In this work, synaptic plasticity has been studied by employing a recently developed statistical inference method for uncovering underlying model parameters. Spiking activity was modelled by a binary Bernoulli GLM, while plasticity was modelled as a Markovian process dependent on the activity and driven by the desired learning rule function. A particle Markov Chain Monte Carlo algorithm was implemented to learn plasticity parameters. When applied to real data, the method proved to have limitations. Long recordings were required to infer the learning rule parameters with adequate confidence and stimulation regimes seemed to heavily affect the performance. Motivated by these challenges, we proposed a Bayesian online experimental design algorithm, which aims to answer the question: what stimulation protocol should be used to ensure optimal inference results? The algorithm employs an online approach, where the stimulation is optimised and chosen in parallel to data collection in order to maximise the expected mutual information between the learning rule parameters and the data. This approach is, to our knowledge, unprecedented for studying synaptic plasticity, and on synthetic data, we show that our developed method significantly improves on other standard stimulation protocols. The algorithm is effective at minimising the amount of required data for satisfactory parameter reconstruction. This allows for better circumstances to study synaptic plasticity and might open doors to new research questions within the field.

References

[1] S. W. Linderman, C. H. Stock, and R. P. Adams. A framework for studying synaptic plasticity with neural spike train data, 2014.

[2] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity:919–926, 2000. URL: https://doi.org/10.1038/78829.

[3] D. Desilver. Despite global concerns about democracy, more than half of countries are democratic, 2019.

[4] D. O. Hebb. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

[5] N. Caporale and Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[6] S. Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 3:31, 2009. ISSN: 1662-5161. DOI: 10.3389/neuro.09.031.2009. URL: https://www.frontiersin.org/article/10.3389/neuro.09.031.2009.

[7] A. Woodruff. What is a neuron? Aug. 2019. URL: https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron.

[8] Neurons, nerve tissues, the nervous system. URL: http://biomedicalengineering.yolasite.com/neurons.php.

[9] The action potential. URL: https://opentextbc.ca/anatomyandphysiology/chapter/12-4-the-action-potential/.

[10] D. F. English, S. McKenzie, T. Evans, K. Kim, E. Yoon, and G. Buzsaki. Pyramidal cell-interneuron circuit architecture and dynamics in hippocampal networks, 2017.

[11] W. Gerstner, W. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics*. Cambridge university press, 2014.

[12] P. Dayan and L. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, volume 15. Jan. 2001.

[13] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A.-S. LaMantia, J. O. McNamara, and L. E. White (Eds.) *Neuroscience*. Sinauer Associates, 4th edition, 2008.

[14] B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *J. Neurophysiol*, 76:3460, 1996.

[15] N. Yusoff and A. Grüning. Biologically inspired temporal sequence learning. *Procedia Engineering*, 41:319–325, 2012. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2012.07.179. URL: http://www.sciencedirect.com/science/article/pii/S1877705812025659. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[16] A. Vigneron and J. Martinet. A critical survey of stdp in spiking neural networks for pattern recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[17] T. V. Bliss and T. Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356, 1973.

[18] B. C. Albensi, D. R. Oliver, J. Toupin, and G. Odero. Electrical stimulation protocols for hippocampal synaptic plasticity and neuronal hyper-excitability: are they effective or relevant? *Experimental neurology*, 204(1):1–13, 2007.

[19] Z. Wang, Z. Feng, and X. Wei. Axonal stimulations with a higher frequency generate more randomness in neuronal firing rather than increase firing rates in rat hippocampus. *Frontiers in Neuroscience*, 12:783, 2018. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00783. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00783.

[20] E. I. Moser, K. A. Krobert, M.-B. Moser, and R. G. Morris. Impaired spatial learning after saturation of long-term potentiation. *Science*, 281(5385):2038–2042, 1998.

[21] J. R. Whitlock, A. J. Heynen, M. G. Shuler, and M. F. Bear. Learning induces long-term potentiation in the hippocampus. *science*, 313(5790):1093–1097, 2006.

[22] B. A. Yankner. Mechanisms of neuronal degeneration in alzheimer's disease. *Neuron*, 16(5):921–932, 1996.

[23] J. W. Vaupel. Biodemography of human ageing. *Nature*, 464(7288):536–542, 2010.

[24] J. A. Hardy and G. A. Higgins. Alzheimer's disease: the amyloid cascade hypothesis. *Science*, 256(5054):184–186, 1992.

[25] G. W. Van Hoesen, B. T. Hyman, and A. R. Damasio. Entorhinal cortex pathology in alzheimer's disease. *Hippocampus*, 1(1):1–8, 1991.

[26] A. Kobro-Flatmoen and M. P. Witter. Neuronal chemo-architecture of the entorhinal cortex: a comparative review. *European Journal of Neuroscience*, 50(10):3627–3662, 2019.

[27] S. Ross. *Introduction to probability models*. Elsevier, pp.191-195, 269-275, 2010.

[28] H. Scheepers. Markov chain analysis and simulation using python. Nov. 2019. URL: https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e.

[29] L. Ross, T. Kneib, S. Land, and B. Marx. *Regression*. Springer, pp.269-285, 2013.

[30] S. V. Vaseghi. *Advanced Digital Signal Processing and Noise Reduction, Second Edition*. John Wiley & Sons, pp.105-109, 2008.

[31] R. Bassett and J. Deride. Maximum a posteriori estimators as a limit of bayes estimators. *Mathematical Programming*, 174(1-2):129–144, Jan. 2018. ISSN: 1436-4646. DOI: 10.1007/s10107-018-1241-0. URL: http://dx.doi.org/10.1007/s10107-018-1241-0.

[32] R. Shumway and D. Stoffer. *Time Series Analysis and Its Applications With R Examples*, volume 9. Jan. 2011, page 10. ISBN: 978-1-4419-7864-6. DOI: 10.1007/978-1-4419-7865-3.

[33] L. D. Haugh. Checking the independence of two covariance-stationary time series: a univariate residual cross-correlation approach. *J. Amer. Statist. Assoc.*, 7(354):378–385, 1976. ISSN: 0162-1459. URL: http://links.jstor.org/sici?sici=0162-1459(197606)71:354%3C378:CTIOTC%3E2.0.CO;2-Q&origin=MSN.

[34] K. E. Himdi and R. Roy. Tests for non-correlation of two multivariate time series: a nonparametric approach, 2003.

[35] G. Givens and J. Hoeting. Simulation and monte carlo integration. In 2013.

[36] D. Rubín and D. Rubin. Using the sir algorithm to simulate posterior distributions. In 1988.

[37] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[38] A. Dimitrov, A. Lazar, and J. Victor. Information theory in neuroscience. *Journal of Computational Neuroscience*, 30:1–5, 2011.

[39] C. Adami. Information theory in molecular biology. *Physics of Life Reviews*, 1(1):3–22, 2004. ISSN: 1571-0645. DOI: https://doi.org/10.1016/j.plrev.2004.01.002. URL: https://www.sciencedirect.com/science/article/pii/S157106450400003X.

[40] E. Maasoumi. A compendium to information theory in economics and econometrics. *Econometric Reviews*, 12:137–181, Feb. 1993. DOI: 10.1080/07474939308800260.

[41] D. MacKay. *InformationTheory, Inference, and Learning Algorithms*, volume 50. Jan. 2003. ISBN: 978-0-521-64298-9. DOI: 10.1109/TIT.2004.834752.

[42] F. I. Dretske. *Knowledge and the Flow of Information*. MIT Press, 1981.

[43] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., pp.13-21, 243-251, 2016.

[44] R. Fisher. *Statistical Methods for Research Workers*. Oliver & Boyd, 1930.

[45] R. Fisher. *The Design of Experiments*. The Design of Experiments. Oliver & Boyd, 1935. URL: https://books.google.no/books?id=-EsNAQAAIAAJ.

[46] K.-M. Tay and C. Butler. Methodologies for experimental design: a survey, comparison, and future predictions. *Quality Engineering*, 11(3):343–356, 1999. DOI: 10.1080/08982119908919250. eprint: https://doi.org/10.1080/08982119908919250. URL: https://doi.org/10.1080/08982119908919250.

[47] M. Tanco, E. Viles, and L. Pozueta. *Comparing different approaches for design of experiments (doe)*. In *Advances in Electrical Engineering and Computational Science*. S.-I. Ao and L. Gelman, editors. Springer Netherlands, Dordrecht, 2009, pages 611–621. ISBN: 978-90-481-2311-7. DOI: 10.1007/978-90-481-2311-7_52. URL: https://doi.org/10.1007/978-90-481-2311-7_52.

[48] N. Flournoy. A clinical experiment in bone marrow transplantation: estimating a percentage point of a quantal response curve. In C. Gatsonis, J. S. Hodges, R. E. Kass, and N. D. Singpurwalla, editors, *Case Studies in Bayesian Statistics*, pages 324–336, New York, NY. Springer New York, 1993. ISBN: 978-1-4612-2714-4.

[49] K. J. Ryan. Estimating expected information gains for experimental designs with application to the random fatigue-limit model. *Journal of Computational and Graphical Statistics*, 12(3):585–603, 2003. DOI: 10.1198/1061860032012. eprint: https://doi.org/10.1198/1061860032012. URL: https://doi.org/10.1198/1061860032012.

[50] K. Chaloner and I. Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273–304, 1995. DOI: 10.1214/ss/1177009939. URL: https://doi.org/10.1214/ss/1177009939.

[51] B. Shababo, B. Paige, A. Pakman, and L. Paninski. Bayesian inference and online experimental design for mapping neural microcircuits. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/17c276c8e723eb46aef576537e9d56d0-Paper.pdf.

[52] M. Park and J. Pillow. Bayesian active learning with localized priors for fast receptive field characterization. *Advances in neural information processing systems*, 25:2348–2356, 2012.

[53] W. Sandmann. Efficiency of importance sampling estimators. *Journal of Simulation*, 1(2):137–145, 2007. DOI: 10.1057/palgrave.jos.4250011. URL: https://doi.org/10.1057/palgrave.jos.4250011.

[54] P. Glynn and D. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35:1367–1392, Nov. 1989. DOI: 10.1287/mnsc.35.11.1367.

[55] E. A. Myhre and M. Rødsmoen. Inferring the learning rule with particle metropolis hastings. 2020. URL: https://github.com/amemil/MasterThesisRaw/blob/main/Preliminary-work.pdf.

[56] M. Megias, Z. Emri, T. Freund, and A. Gulyas. Total number and distribution of inhibitory and excitatory synapses on hippocampal ca1 pyramidal cells. *Neuroscience*, 102(3):527–540, 2001. ISSN: 0306-4522. DOI: https://doi.org/10.1016/S0306-4522(00)00496-6. URL: https://www.sciencedirect.com/science/article/pii/S0306452200004966.

[57] D. R. Mosier. Chapter 1 - clinical neuroscience. In L. A. Rolak, editor, *Neurology Secrets (Fifth Edition)*, pages 7–17. Mosby, Philadelphia, fifth edition edition, 2010. ISBN: 978-0-323-05712-7. DOI: https://doi.org/10.1016/B978-0-323-05712-7.00001-5. URL: https://www.sciencedirect.com/science/article/pii/B9780323057127000015.

[58] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm, 1995.

[59] C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[60] A. Doucet, N. de Freitas, and N. Gordon. *An Introduction to Sequential Monte Carlo Methods*. Springer, 2001.

[61] N. Whiteley et al. Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6):2500–2537, 2013.

[62] O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Science & Business Media, 2006.

[63] O. Cappé, R. Douc, A. Guillin, J.-M. Marin, and C. P. Robert. *Adaptive Importance Sampling in General Mixture Classes*. 2008. URL: https://arxiv.org/abs/0710.4242.

[64] L. Martino, V. Elvira, and F. Louzada. *Effective Sample Size for Importance Sampling based on discrepancy measures*. Signal Processing 131, 386–401, 2017.

[65] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk metropolis algorithm, 1999.

[66] M. L. Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[67] J. Moody. What does rmse really mean? Sept. 2019. URL: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e.

[68] M. Själander, M. Jahre, G. Tufte, and N. Reissmann. EPIC: an energy-efficient, high-performance GPGPU computing research infrastructure, 2019. arXiv: 1912.05848 [cs.DC].

[69] E. Edelmann and V. Lessmann. Dopamine modulates spike timing-dependent plasticity and action potential properties in ca1 pyramidal neurons of acute rat hippocampal slices. *Frontiers in synaptic neuroscience*, 3:6, 2011.

[70] P. R. Moult and J. Harvey. Hormonal regulation of hippocampal dendritic morphology and synaptic plasticity. *Cell Adhesion & Migration*, 2(4):269–275, 2008. DOI: 10.4161/cam.2.4.6354. eprint: https://doi.org/10.4161/cam.2.4.6354. URL: https://doi.org/10.4161/cam.2.4.6354. PMID: 19262152.

[71] C. Lüscher and R. Malenka. Drug-evoked synaptic plasticity in addiction: from molecular changes to circuit remodeling. *Neuron*, 69:650–63, Feb. 2011. DOI: 10.1016/j.neuron.2011.01.017.

[72] G. Wang, B. Grone, D. Colas, L. Appelbaum, and P. Mourrain. Synaptic plasticity in sleep: learning, homeostasis and disease. *Trends in neurosciences*, 34:452–63, Aug. 2011. DOI: 10.1016/j.tins.2011.07.005.

[73] L. Paninski. Asymptotic theory of information-theoretic experimental design. *Neural Computation*, 17(7):1480–1507, 2005.

[74] M. Prince, A. Wimo, M. Guerchet, G.-C. Ali, W. Yu-Tzu, and M. Prina. World alzheimer report 2015: global impact of dementia, 2015.

Appendix A

Source code in Python

```
import numpy as np
import matplotlib.pyplot as plt
import math
from tqdm import tqdm
from scipy.stats import gamma
from scipy.stats import multivariate_normal
from scipy.stats import norm
from numba import njit
@njit

def learning_rule(s1,s2,Ap,Am,taup,taum,t,i,binsize):
    '''
    s1,s2 : binary values for the different time bins for neuron 1 and 2 respectively, 1:spike, 0:no spike
    i : current iteration/timebin for the numerical approximation
    '''
    l = i - np.int(np.ceil(10*taup / binsize))
    return s2[i-1]*np.sum(s1[max([l,0]):i]*Ap*np.exp((t[max([l,0]):i]-max(t))/taup)) - s1[i-1]*np.sum(s2[max([l,0]):i]*Am*np.exp((t[max([l,0]):i]-max(t))/taum))

def logit(x):
    return np.log(x/(1-x))

def inverse_logit(x):
    return np.exp(x)/(1+np.exp(x))


class SimulatedData():
    '''
    Ap, Am, tau : learning rule parameters
    b1,b2 : background noise constants for neuron 1 and neuron 2, determnining their baseline firing rate
    w0 : start value for synapse strength between neuron 1 and 2.
    '''
    def __init__(self,Ap=0.005, tau=0.02, std=0.001,b1=-2.0, b2=-2.0, w0=1.0,sec = 120, binsize = 1/200.0,freq = 50):
        self.Ap = Ap
        self.tau = tau
        self.std = std
        self.Am = 1.05*self.Ap
        self.b1 = b1
        self.b2 = b2
```

```
        self.w0 = w0
        self.sec = sec
        self.binsize = binsize
        self.freq = freq

    def set_Ap(self,Ap):
        self.Ap = Ap
    def set_tau(self,tau):
        self.tau = tau
    def set_std(self,std):
        self.std = std
    def set_b1(self,b1):
        self.b1 = b1
    def set_b2(self,b2):
        self.b2 = b2
    def set_w0(self,w0):
        self.w0 = w0
    def set_sec(self,sec):
        self.sec = sec
    def set_binsize(self,binsize):
        self.binsize = binsize

    def get_Ap(self):
        return self.Ap
    def get_tau(self):
        return self.tau
    def get_std(self):
        return self.std
    def get_b1(self):
        return self.b1
    def get_b2(self):
        return self.b2
    def get_w0(self):
        return self.w0
    def get_sec(self):
        return self.sec
    def get_binsize(self):
        return self.binsize

    def create_data(self):
        iterations = np.int(self.sec/self.binsize)
        t,W,s1,s2 = np.zeros(iterations),np.zeros(iterations),np.zeros(
    iterations),np.zeros(iterations)
        W[0] = self.w0
        s1[0] = np.random.binomial(1,inverse_logit(self.b1))
        for i in range(1,iterations):
            lr = learning_rule(s1,s2,self.Ap,self.Am,self.tau,self.tau,t,i,self.
    binsize)
            step = W[i-1] + lr + np.random.normal(0,self.std)
            if step > 0:
                W[i] = step
            else:
                W[i] = 0
            s2[i] = np.random.binomial(1,inverse_logit(W[i]*s1[i-1]+self.b2))
            s1[i] = np.random.binomial(1,inverse_logit(self.b1))
            t[i] = self.binsize*i
        self.s1 = s1
        self.s2 = s2
        self.t = t
        self.W = W

    def create_freq_data(self):
```

```
99          iterations = np.int(self.sec/self.binsize)
100         t,W,s1,s2 = np.zeros(iterations),np.zeros(iterations),np.zeros(
    iterations),np.zeros(iterations)
101         W[0] = self.w0
102         s1[0] = 1
103         for i in range(1,iterations):
104             lr = learning_rule(s1,s2,self.Ap,self.Am,self.tau,self.tau,t,i,self.
    binsize)
105             step = W[i-1] + lr + np.random.normal(0,self.std)
106             if step > 0:
107                 W[i] = step
108             else:
109                 W[i] = 0
110             s2[i] = np.random.binomial(1,inverse_logit(W[i]*s1[i-1]+self.b2))
111             s1[i] = [np.random.binomial(1,inverse_logit(self.b1)),1][i % int((1/
    self.binsize)/self.freq) == 0]
112             t[i] = self.binsize*i
113         self.s1 = s1
114         self.s2 = s2
115         self.t = t
116         self.W = W
117
118     def get_data(self):
119         return self.s1,self.s2,self.t,self.W
120
121
122 class ParameterInference():
123     '''
124     Class for estimating b1,b2,w0,Ap,Am,tau from SimulatedData, given data s1,s2
    .
125     '''
126     def __init__(self,s1,s2,P = 100, Usim = 100, Ualt = 200,it = 1500, infstd
    =0.0001, N = 2\
127                  , shapes_prior = np.array([4,5]), rates_prior = np.array
    ([50,100]),sec=120\
128                      ,binsize = 1/200.0,taufix = 0.02,Afix = 0.005,b1est = -3.1,
    b2est = -3.1,w0est = 1):
129         self.s1 = s1
130         self.s2 = s2
131         self.infstd = infstd
132         self.P = P
133         self.Usim = Usim
134         self.Ualt = Ualt
135         self.it = it
136         self.N = N
137         self.shapes_prior = shapes_prior
138         self.rates_prior = rates_prior
139         self.sec = sec
140         self.binsize = binsize
141         self.Afix = Afix
142         self.taufix = taufix
143         self.b1est = b1est
144         self.b2est = b2est
145         self.w0est = w0est
146
147     def get_sec(self):
148         return self.sec
149
150     def set_std(self,std):
151         self.std = std
152     def set_P(self,P):
153         self.P = P
```

```
    def set_s1(self,s1):
        self.s1 = s1
    def set_s2(self,s2):
        self.s2 = s2
    def set_sec(self,sec):
        self.sec = sec
    def set_shapes_prior(self,shapes_prior):
        self.shapes_prior = shapes_prior
    def set_rates_prior(self,rates_prior):
        self.rates_prior = rates_prior
    def set_w0est(self,w0est):
        self.w0est=w0est
    def set_N(self,N):
        self.N = N


    def b1_estimation(self):
        self.b1est = logit(np.sum(self.s1)/len(self.s1))
        return self.b1est

    def normalize(self,vp):
        return vp/np.sum(vp)

    def perplexity_func(self,vp_normalized):
        h = -np.sum(vp_normalized*np.log(vp_normalized))
        return np.exp(h)/self.P

    def resampling(self,vp_normalized,wp):
        wp_new = np.copy(wp)
        indexes = np.linspace(0,self.P-1,self.P)
        resampling_indexes = np.random.choice(indexes,self.P,p=vp_normalized)
        for i in range(self.P):
            wp_new[i] = np.copy(wp[resampling_indexes.astype(int)[i]])
        return wp_new

    def likelihood_step(self,s1prev,s2next,wcurr):
        return inverse_logit(wcurr*s1prev + self.b2est)**(s2next) * (1-
inverse_logit(wcurr*s1prev + self.b2est))**(1-s2next)

    def parameter_priors(self):
        return np.array([(np.random.gamma(self.shapes_prior[i],1/self.
rates_prior[i])) for i in range(self.N)])

    def parameter_priors_ns(self,mean,cov):
        return multivariate_normal.rvs(mean,cov,1)

    def proposal_step(self,shapes,theta):
        return np.array([(np.random.gamma(shapes[i],theta[i]/shapes[i])) for i
in range(self.N)])

    def adjust_variance(self,theta,shapes):
        means = theta[-self.Usim:].mean(0)
        var_new = np.array([0,0])
        u_temp = self.Usim
        while (any(i == 0 for i in var_new)):
            var_new = theta[-u_temp:].var(0)*(2.4**2)
            u_temp += 50
            if u_temp > self.it:
                return shapes, np.array([(np.random.gamma(shapes[i],theta[-1][i
]/shapes[i])) for i in range(self.N)])
        new_shapes = np.array([((means[i]**2) / var_new[i]) for i in range(self.
N)])
```

```
211         proposal = np.array([(np.random.gamma(new_shapes[i],theta[-1][i]/
    new_shapes[i])) for i in range(self.N)])
212         return new_shapes,proposal
213 
214     def ratio(self,prob_old,prob_next,shapes,theta_next,theta_prior):
215         spike_prob_ratio = prob_next / prob_old
216         prior_ratio, proposal_ratio = 1,1
217         for i in range(self.N):
218             prior_ratio *= gamma.pdf(theta_next[i],a=self.shapes_prior[i],scale
    =1/self.rates_prior[i])/\
219             gamma.pdf(theta_prior[i],a=self.shapes_prior[i],scale=1/self.
    rates_prior[i])
220             proposal_ratio *= gamma.pdf(theta_prior[i],a=shapes[i],scale=
    theta_next[i]/shapes[i])/\
221             gamma.pdf(theta_next[i],a=shapes[i],scale=theta_prior[i]/shapes[i])
222         return spike_prob_ratio * prior_ratio * proposal_ratio
223 
224     def ratio_g(self,prob_old,prob_next,shapes,theta_next,theta_prior,mean,cov):
225         spike_prob_ratio = prob_next / prob_old
226         proposal_ratio = 1
227         prior_ratio = multivariate_normal.pdf(theta_next,mean,cov) /
    multivariate_normal.pdf(theta_prior,mean,cov)
228         for i in range(self.N):
229             proposal_ratio *= gamma.pdf(theta_prior[i],a=shapes[i],scale=
    theta_next[i]/shapes[i])/\
230             gamma.pdf(theta_next[i],a=shapes[i],scale=theta_prior[i]/shapes[i])
231         return spike_prob_ratio * prior_ratio * proposal_ratio
232 
233 
234 
235     def scaled2_spike_prob(self,old,new):
236         return np.exp(old - min(old,new)),np.exp(new - min(old,new))
237 
238     def b2_w0_estimation(self):
239         '''
240         Fisher scoring algorithm
241         Two in parallell, since w0 is estimated with a subset of the data
242         '''
243         s1short,s2short = self.s1[:int((self.sec/10)/(self.binsize))],self.s2[:
    int((self.sec/10)/(self.binsize))]
244         beta,beta2 = np.array([0,0]),np.array([0,0])
245         x,x2 = np.array([np.ones(len(self.s1)-1),self.s1[:-1]]),np.array([np.
    ones(len(s1short)-1),s1short[:-1]])
246         i = 0
247         score,score2 = np.array([np.inf,np.inf]), np.array([np.inf,np.inf])
248         while(i < 1000 and any(abs(i) > 1e-10 for i in score) and any(abs(j) > 1
    e-10 for j in score2)):
249             eta,eta2 = np.matmul(beta,x),np.matmul(beta2,x2) #linear predictor
250             mu,mu2 = inverse_logit(eta),inverse_logit(eta2)
251             score,score2 = np.matmul(x,self.s2[1:] - mu),np.matmul(x2,s2short
    [1:] - mu2)
252             hessian_u,hessian_u2 = mu * (1-mu), mu2 *(1-mu2)
253             hessian,hessian2 = np.matmul(x*hessian_u,np.transpose(x)),np.matmul(
    x2*hessian_u2,np.transpose(x2))
254             delta,delta2 = np.matmul(np.linalg.inv(hessian),score),np.matmul(np.
    linalg.inv(hessian2),score2)
255             beta,beta2 = beta + delta, beta2 + delta2
256             i += 1
257         self.b2est = beta[0]
258         self.w0est = beta2[1]
259         return self.b2est,self.w0est
260 
```

```
261
262     def particle_filter(self,A,tau):
263         '''
264         Particle filtering
265         '''
266         timesteps = np.int(self.sec/self.binsize)
267         t = np.zeros(timesteps)
268         wp = np.full((self.P,timesteps),np.float(self.w0est))
269         vp = np.ones(self.P)
270         log_posterior = 0
271         for i in range(1,timesteps):
272             v_normalized = self.normalize(vp)
273             perplexity = self.perplexity_func(v_normalized)
274             if perplexity < 0.66:
275                 wp = self.resampling(v_normalized,wp)
276                 vp = np.full(self.P,1/self.P)
277                 v_normalized = self.normalize(vp)
278             lr = learning_rule(self.s1,self.s2,A,A*1.05,tau,tau,t,i,self.binsize
    )
279             ls = self.likelihood_step(self.s1[i-1],self.s2[i],wp[:,i-1])
280             vp = ls*v_normalized
281             step = wp[:,i-1] + lr + np.random.normal(0,self.infstd,size = self.P
    )
282             step[step<0] = 0
283             wp[:,i] = step
284             t[i] = i*self.binsize
285             log_posterior += np.log(np.sum(vp)/self.P)
286         return wp,t,log_posterior
287
288
289
290     def standardMH(self):#,w0est,b1,b2):
291         '''
292         Monte Carlo sampling with particle filtering, Metropolis Hastings
    algorithm
293         '''
294         theta_prior = self.parameter_priors()
295         theta = np.array([theta_prior])
296         shapes = np.copy(self.shapes_prior)
297         _,_,old_log_post = self.particle_filter(theta_prior[0],theta_prior[1])
298         for i in range(1,self.it):
299             if (i % self.Usim == 0):
300                 shapes, theta_next = self.adjust_variance(theta,shapes)
301             else:
302                 theta_next = self.proposal_step(shapes,theta_prior)
303             _,_,new_log_post = self.particle_filter(theta_next[0],theta_next[1])
304             prob_old,prob_next = self.scaled2_spike_prob(old_log_post,
    new_log_post)
305             r = self.ratio(prob_old,prob_next,shapes,theta_next,theta_prior)
306             choice = np.int(np.random.choice([1,0], 1, p=[min(1,r),1-min(1,r)]))
307             theta_choice = [np.copy(theta_prior),np.copy(theta_next)][choice ==
    1]
308             theta = np.vstack((theta, theta_choice))
309             theta_prior = np.copy(theta_choice)
310             old_log_post = [np.copy(old_log_post),np.copy(new_log_post)][choice
    == 1]
311         return theta
312
313
314     def standardMH_mv(self,mean,cov):#,ir):
315         '''
```

```
316         Monte Carlo sampling with particle filtering, Metropolis Hastings
    algorithm
317         '''
318         theta_prior = self.parameter_priors()
319         #print('prior:',theta_prior)
320         theta = np.array([theta_prior])
321         shapes = np.copy(self.shapes_prior)
322         _,_,old_log_post = self.particle_filter(theta_prior[0],theta_prior[1])
323         for i in range(1,self.it):
324             if (i % self.Usim == 0):
325                 shapes, theta_next = self.adjust_variance(theta,shapes)
326             else:
327                 theta_next = self.proposal_step(shapes,theta_prior)
328             _,_,new_log_post = self.particle_filter(theta_next[0],theta_next[1])
329             prob_old,prob_next = self.scaled2_spike_prob(old_log_post,
    new_log_post)
330             r = self.ratio_g(prob_old,prob_next,shapes,theta_next,theta_prior,
    mean,cov)
331             choice = np.int(np.random.choice([1,0], 1, p=[min(1,r),1-min(1,r)]))
332             theta_choice = [np.copy(theta_prior),np.copy(theta_next)][choice ==
    1]
333             theta = np.vstack((theta, theta_choice))
334             theta_prior = np.copy(theta_choice)
335             old_log_post = [np.copy(old_log_post),np.copy(new_log_post)][choice
    == 1]
336         return theta
337
338 ### functions for optimal experimental design
339
340
341 class ExperimentDesign():
342     def __init__(self,freqs_init=np.array([20,50,100,200]),maxtime=120,trialsize
    =5\
343                  ,Ap=0.005, tau=0.02, genstd=0.0001,b1=-3.1, b2=-3.1, w0=1.0,
    binsize = 1/500.0,reals = 20, longinit = 60,s1init = 1,s2init =1,Winit =1,W
    =0):
344         self.maxtime = maxtime
345         self.freqs_init = freqs_init
346         self.Ap = Ap
347         self.tau = tau
348         self.genstd = genstd
349         self.trialsize = trialsize
350         self.Am = 1.05*self.Ap
351         self.binsize = binsize
352         self.b1 = b1
353         self.b2 = b2
354         self.w0 = w0
355         self.b2est = b2
356         self.b1est = b1
357         self.w0est = w0
358         self.W = W
359         self.reals = reals
360         self.longinit = longinit
361         self.s1init = s1init
362         self.s2init = s2init
363         self.Winit = Winit
364
365
366     def NormEntropy(self,sigma):
367         return 0.5 * np.log(np.linalg.det(2*np.pi*np.exp(1)*sigma))
368
369
```

```
    def datasim(self,freq,a,tau,init,optim,l):
        iterations = [np.int(self.trialsize/self.binsize),np.int(self.longinit/
    self.binsize)][l==True]
        t = np.zeros(iterations)
        s1,s2,W = np.zeros(iterations),np.zeros(iterations),np.zeros(iterations)
        s1[0] = 1
        if init == True:
            W[0] = self.w0
        else:
            W[0] = self.W[-1]
        for i in range(1,iterations):
            lr = learning_rule(s1,s2,a,1.05*a,tau,tau,t,i,self.binsize)
            step = W[i-1] + lr + np.random.normal(0,self.genstd)
            if step > 0:
                W[i] = step
            else:
                W[i] = 0
            s2[i] = np.random.binomial(1,inverse_logit(W[i]*s1[i-1]+self.b2))
            s1[i] = [np.random.binomial(1,inverse_logit(self.b1)),1][i % int((1/
    self.binsize)/freq) == 0]
            t[i] = self.binsize*i
        if optim == False:
            if init == True:
                self.s1,self.s2,self.W = s1,s2,W
            else:
                self.s1 = np.hstack((self.s1,s1))
                self.s2 = np.hstack((self.s2,s2))
                self.W = np.hstack((self.W,W))
        else:
            return s1,s2,W

    def datasim_const(self,a,tau,init=False,optim = False,l = False):
        iterations = [np.int(self.trialsize/self.binsize),np.int(self.longinit/
    self.binsize)][l==True]
        s1,s2,W = np.zeros(iterations),np.zeros(iterations),np.zeros(iterations)
        if init == True:
            W[0] = self.w0
        else:
            W[0] = self.W[-1]
        t = np.zeros(iterations)
        for i in range(1,iterations):
            lr = learning_rule(s1,s2,a,1.05*a,tau,tau,t,i,self.binsize)
            step = W[i-1] + lr + np.random.normal(0,self.genstd)
            if step > 0:
                W[i] = step
            else:
                W[i] = 0
            s2[i] = np.random.binomial(1,inverse_logit(W[i]*s1[i-1]+self.b2))
            s1[i] = np.random.binomial(1,inverse_logit(self.b1))
            t[i] = self.binsize*i
        if optim == False:
            if init == True:
                self.s1,self.s2,self.W = s1,s2,W
            else:
                self.s1 = np.hstack((self.s1,s1))
                self.s2 = np.hstack((self.s2,s2))
                self.W = np.hstack((self.W,W))
        else:
            return s1,s2,W


    def adjust_proposal(self,means,sample):
```

```
429         new_shapes = np.array([((means[i]**2) / np.var(sample[300:,i])) for i in
     range(2)])
430         new_rates = np.array([((new_shapes[i]) / means[i]) for i in range(2)])
431         return new_shapes,new_rates
432 
433     def freq_optimiser(self,means,cov,init,optim,l,inference):
434         entropies = []
435         for j in range(len(self.freqs_init)):
436             entropies_temp = []
437             for k in range(self.reals):
438                 s1temp,s2temp,_ = self.datasim(self.freqs_init[j],means[0],means
    [1],init = init, optim = optim,l = l)
439                 inference.set_s1(s1temp)
440                 inference.set_s2(s2temp)
441                 sample_temp = inference.standardMH_mv(means,cov)
442                 cov_temp = np.cov(np.transpose(sample_temp[300:,:]))
443                 entropies_temp.append(self.NormEntropy(cov_temp))
444             entropies_temp_clean = [x for x in entropies_temp if math.isnan(x)
    == False]
445             entropies.append(np.mean(entropies_temp_clean))
446         return self.freqs_init[np.where(entropies == np.amin(entropies))[0][0]],
    entropies
447 
448     def onlineDesign_wh(self,nofreq = False, constant = False, random = False,
    optimised = True):
449         freq_const = self.freqs_init[0]
450         optimal_freqs = []
451         trials = np.int(self.maxtime / self.trialsize)
452         init = False
453         self.s1 = self.s1init
454         self.s2 = self.s2init
455         self.W = self.Winit
456         #self.datasim(freq_const,self.Ap,self.tau,init = init, optim = False,l=
    False)
457         inference_whole = ParameterInference(self.s1,self.s2,P = 50, Usim = 100,
     Ualt = 200,it = 1500, infstd=0.0001, N = 2\
458                                              , shapes_prior = np.array([4,5]),
    rates_prior = np.array([50,100]),sec=self.trialsize\
459                                                  ,binsize = 1/500.0,taufix = 0.02,
    Afix = 0.005)
460 
461         sample = inference_whole.standardMH()
462         posts = [sample]
463         means, cov = [np.mean(sample[300:,0]),np.mean(sample[300:,1])], np.cov(
    np.transpose(sample[300:,:]))
464         ests, entrs = np.array([means]), np.array([self.NormEntropy(cov)])
465         new_shapes, new_rates = self.adjust_proposal(means,sample)
466         if optimised == True:
467             inference_optim = ParameterInference(1,1,P = 50, Usim = 100, Ualt =
    200,it = 1500, infstd=0.0001, N = 2\
468                                                  , shapes_prior = new_shapes,
    rates_prior = new_rates,sec=self.trialsize\
469                                                      ,binsize = 1/500.0,taufix =
     0.02,Afix = 0.005)
470             mutinfs = []
471         for i in range(trials):
472             if optimised == True:
473                 inference_optim.set_w0est(self.W[-1])
474                 opts_temp,mutinfs_temp = self.freq_optimiser(means,cov,init =
    init, optim = True,l=False,inference = inference_optim)
475                 optimal_freqs.append(opts_temp)
476                 mutinfs.append(mutinfs_temp)
```

```
                self.datasim(optimal_freqs[-1],self.Ap,self.tau,init=init,optim
    = False,l=False)
            elif random == True:
                freq_temp = np.random.choice(self.freqs_init)
                self.datasim(freq_temp,self.Ap,self.tau,init=init,optim = False,
    l=False)
            elif constant == True:
                self.datasim(freq_const,self.Ap,self.tau,init=init,optim = False
    ,l=False)
            elif nofreq == True:
                self.datasim_const(self.Ap,self.tau,init=init,optim = False,l=
    False)
            inference_whole.set_s1(self.s1)
            inference_whole.set_s2(self.s2)
            inference_whole.set_sec(np.int(len(self.s1)*self.binsize))
            sample = inference_whole.standardMH()
            posts.append(sample)
            means = [np.mean(sample[300:,0]),np.mean(sample[300:,1])]
            cov = np.cov(np.transpose(sample[300:,:]))
            ests = np.vstack((ests, means))
            entrs = np.vstack((entrs,self.NormEntropy(cov)))
            new_shapes, new_rates = self.adjust_proposal(means,sample)
            if optimised == True:
                inference_optim.set_shapes_prior(new_shapes)
                inference_optim.set_rates_prior(new_rates)
        if optimised == True:
            return ests,entrs,optimal_freqs,mutinfs,self.W,posts
        else:
            return ests,entrs,self.W,posts
```

Listing A.1: Python implementation with all the functions being applied for both inference and Bayesian optimal design

Appendix B

Preprint: Bayesian active learning for inferring synaptic plasticity rules

Bayesian active learning for inferring synaptic plasticity rules

Claudia Battistin
Kavli Institute for Systems Neuroscience
NTNU
19 Olav Kyrres gate, 7030 Trondheim
claudia.battistin@ntnu.no

Emil A. Myhre
Department for Mathematical Sciences
NTNU
1 Alfred Getz' vei, 7034 Trondheim
emilalvar.myhre@gmail.com

Samuel A. McKenzie
Department of Neurosciences
UNM School of Medicine
2500 Marble Ave NE, NM 87106 Albuquerque
SAMcKenzie@salud.unm.edu

Benjamin Dunn
Department for Mathematical Sciences
NTNU
1 Alfred Getz' vei, 7034 Trondheim
benjamin.dunn@ntnu.no

Abstract

Recently a statistical inference method has been introduced to learn the spike timing dependent plasticity (STDP) rule from spike train data. In this paper we observe that when applied to real data, this method requires long recordings to estimate the learning rule with adequate confidence. This challenges its applicability in experiments. Efficiency improves in the presence of stimulation, providing a stable reconstruction of STDP parameters with fewer data. Although several stimulation protocols exist for inducing plasticity, it is not clear whether any of them are optimal for inferring the learning rule. We, therefore, propose a Bayesian active learning procedure, which adaptively tunes the stimulus during the experiment to maximise the expected mutual information between the data to be collected and the STDP rule parameters. On synthetic data we show that Bayesian active learning can recover the underlying learning rule with roughly half of the data compared to standard stimulation protocols. Our method is useful whenever it is crucial to optimise resources, either for computational reasons or for probing the system within a time frame where it can be considered stationary, e.g. at shorter time scales than those at which neuromodulators or homeostatic plasticity operate.

1 Introduction

Synaptic plasticity is arguably one of the most fascinating properties of the brain, which is thought to underlie our ability to change as a result of experience [1, 2]. The Canadian psychologist Donald Hebb [3] postulated that perceptual experiences drive plasticity by impinging on single synapses through the correlated activity of pre- and postsynaptic neurons. Several decades of research have revealed that activity-dependent changes in synaptic strength are a widespread phenomenon in the brain, and that they are expressed at both excitatory and inhibitory synapses [4]. Dual pairing experiments in the 90s uncovered a tight contingency time window and a role of the temporal ordering of the activity of the two neurons in order for a synaptic modification to occur [5, 6], now termed spike timing-dependent plasticity (STDP). STDP has been reported in a variety of brain regions [7], both *in-vivo* in animal models [8] and in human brain slices [9], although the temporal dependence of the synaptic modification on the pre-post synaptic delay varies greatly between different preparations [10]. Statistical tools capable of inferring the STDP rule from spike train recordings from limited

Preprint. Under review.

data would allow neuroscientists to systematically probe plasticity in the brain, study metaplasticity (effect of neuromodulators and homeostatic plasticity), investigate the effects of disease on synaptic plasticity, and potentially inform brain-machine interface based paradigms for neural rehabilitation [11].

Recently, a Bayesian statistical inference framework [12] has been proposed to infer the STDP learning rule from spike train data. In this framework the generative model consists of a dynamical system obeying the STDP learning rule for the synaptic weights embedded in a generalised linear model (GLM) [13] for the postsynaptic neuron activity. Full Bayesian learning is approximated using particle filtering methods to integrate out the latent synaptic weight trajectory. The authors [12] show anecdotal good performances of the algorithm on synthetic data generated from the biophysical simulator NEURON.

The first contribution of this paper consists of illustrating that the aforementioned method [12] requires long recordings to achieve sufficient confidence on the estimated STDP parameters, when applied to extracellular-juxtacellular recordings from the hippocampus in the absence of external stimulation. Notably, on synthetic data we show that efficiency can be substantially improved by stimulating the presynaptic neuron, providing stable reconstruction of the learning rule. Electrophysiological stimulation has played a long-standing role in studying synaptic plasticity, starting from the seminal work of Bliss and Lømo [14], in which high-frequency stimulation (frequency: 100Hz, duration: 1-5s) of afferent fibers was employed to induce long-term potentiation of the synapses. Over the years, many diverse stimulation protocols have been developed that appear efficient at eliciting plasticity and that are more similar to the physiological features of neural activity than the classical approach [15]. Yet, whether the existing protocols are relevant for inferring the learning rule is an open question [16].

The second and main contribution of this paper consists of a Bayesian active learning procedure, providing a formal solution to the optimisation problem: what is the stimulus that ensures optimal reconstruction of the learning rule? Building on the inference framework proposed in [12], we propose a stimulation protocol which maximises the mutual information between the data to be collected in the next experimental trial and the learning rule parameters, given the accumulated knowledge about the system. In section 4.3, we demonstrate that such an adaptive procedure significantly improves the reconstruction of the STDP parameters over standard stimulation protocols on synthetic data.

1.1 Related work

Although closed-loop electrophysiological stimulation is an emerging field in neuroscience [17], to our knowledge, no Bayesian active learning protocol to estimate the STDP rule exists so far. Bayesian adaptive learning algorithms have been designed for addressing the neural coding problem by actively selecting the stimulus to best characterise tuning curves [18]. Within the synaptic plasticity community few efforts have been made to devise stimulation paradigms that induce plasticity, rather than trying to infer the learning rule [19].

2 The model

Inspired by the work of Linderman, et al. [12] we choose a GLM as a neural spiking model, with non-stationary synaptic weights obeying the classical STDP learning rule. The corresponding recurrent probabilistic graphical model for two neurons and one synapse is drawn in Figure 1C. Such a generative model is then used to infer the learning rule as explained in section 3.

2.1 GLM spiking framework

We bin our time domain into T bins denoted by indices $t \in \{1, ..., T\}$. Furthermore, we define the spike trains $s_i^{1:T}$ for neuron i. The trajectory for the non-stationary directed synaptic connection between neurons i and j is defined as $w_{ij}^{1:T}$. Spiking events are modelled with a Bernoulli GLM, hence $s_i^t \in \{0, 1\}$, where $s_i^t = 1$ denotes a spike of neuron i in time bin t. This leads to the full spiking model, where the conditional spiking probability for neuron n in time bin $t + 1$, λ_n^{t+1} is

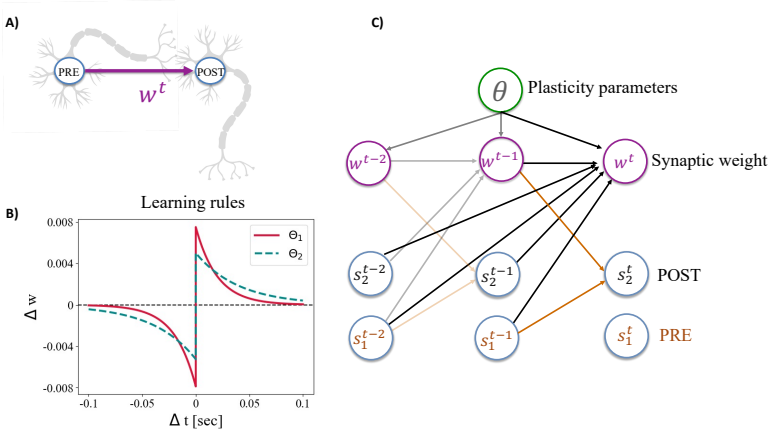

Figure 1: Spiking and STDP model. A): Monodirectionally connected neuron pair with non-stationary synaptic weight w^t. B): STDP learning rules with paramaters $\theta_1 = \{0.0075, 0.0079, 0.02, 0.02\}$ and $\theta_2 = \{0.0050, 0.0053, 0.04, 0.04\}$. C): Probabilistic graphical representation of the full Bayesian hierarchical model.

defined as

$$\lambda_n^{t+1} = g\Big(b_n^{t+1} + \sum_{m=1, m\neq n}^{M} w_{mn}^t s_m^t\Big) \tag{1}$$

for a network of M neurons. The time-dependent external field b_n^t is the sum of a stationary parameter b_n setting the baseline firing rate of the neuron and a non-stationary one, which represents the external stimulation. In this work, we will study a simplified network only consisting of two neurons and a monodirectional synapse as in Figure 1A).

2.2 STDP model

The non-stationary synaptic connectivity w is assumed to follow a Markov process, dependent on the observed interspike intervals and a set of parameters, $\theta = \{A_+, A_-, \tau_+, \tau_-\}$. The weight connectivity evolves according to the density

$$w^t \sim \mathcal{N}\Big(w^{t-1} + l(s_1^{1:t-1}, s_2^{1:t-1}, \theta), \sigma\Big). \tag{2}$$

where l is the classical additive STDP learning rule, depicted in Figure 1B

$$\begin{aligned} l(s_1^{1:t}, s_2^{1:t}, \theta) &= l_+(s_1^{1:t}, s_2^{1:t}, A_+, \tau_+) - l_-(s_1^{1:t}, s_2^{1:t}, A_-, \tau_-), \\ l_+(s_1^{1:t}, s_2^{1:t}, A_+, \tau_+) &= s_2^t \sum_{t'=\gamma_t}^{t} s_1^{t'} A_+ \exp\Big(\frac{t'-t}{\tau_+}\Big), \\ l_-(s_1^{1:t}, s_2^{1:t}, A_-, \tau_-) &= s_1^t \sum_{t'=\gamma_t}^{t} s_2^{t'} A_- \exp\Big(\frac{t'-t}{\tau_-}\Big). \end{aligned} \tag{3}$$

Here γ_t denotes the first time bin of previous history to be considered. For computational reasons, we chose to set $\gamma_t = t - 10\tau_+$, which is more biologically plausible than having the synapse tracking a longer spike history and has a negligible impact on the weight trajectory.

3 Inference

For studying synaptic plasticity using the model introduced in section 2, our goal is to recover the underlying STDP parameters θ from the observed spike train data through the latent process for the synaptic weight. The inference framework heavily inspired by [12] and described in section 3.1, gets tested on real and synthetic data, as explained in section 3.2.

3.1 Method

In order to infer the plasticity parameters θ, we combine a Metropolis Hastings algorithm with particle filtering [20] to obtain posterior distributions and estimators in Bayesian fashion. Particle filtering targets the posterior of the weight trajectories $P(w^{1:T}|s_1^{1:T}, s_2^{1:T}, \theta)$ at fixed learning rule parameters. Particle filtering is suitable here due to the time-recursive properties of the posterior. Averaging over the sampled weight trajectories then provides an estimate of the likelihood function $P(s_1^{1:T}, s_2^{1:T}|\theta)$, which is employed by the Metropolis Hastings algorithm to approximate the desired posterior $P(\theta|s_1^{1:T}, s_2^{1:T})$.

The choice of proposal distribution is important for the performance of Metropolis Hastings, for which we are using Gamma distributions centered at the current θ sample, ensuring that the parameter space for θ consists of only non-negative values. For the proposal distribution, we also employ an adaptive variance approach, as discussed in [21]. Details regarding the inference algorithm, data simulation as well as specific choices we make in our experiments are presented in the Supplementary Material.

Stationary parameters

In a fully Bayesian approach, one would average over the stationary GLM parameters b_2 and $w^{t=1}$ to estimate the likelihood function $P(s_1^{1:T}, s_2^{1:T}|\theta)$. We experience that a hybrid approach, where the stationary parameters are estimated first and then kept fixed during the inference of the posterior $P(\theta|s_1^{1:T}, s_2^{1:T})$, as described in the previous section, does not significantly affect the results (not shown here). In our inferences b_2 and $w^{t=1}$ are thus learned by maximum likelihood employing the Fisher Scoring Algorithm on a subset of the data from the start of the experiment/simulations. Only for this purpose we assume w to be stationary, so no learning.

3.2 Results on real and simulated data

We tested the proposed inference framework on electrophysiological data from CA1 [22] during stimulation and no stimulation periods. Some examples of posteriors for the learning rule parameter A_+ are displayed in Figure 2. From Figure 2A)-D) it can be appreciated that parameter estimation in the presence of stimulation is both more stable across different datasets and posteriors are more peaked, which speaks of our confidence in the estimates. Performances are degraded in the 20s datasets when compared to the 100s datasets, and yet the variability between experiments with stimulation is lower, see Figure 2E)-H).

These results suggested that inference of the learning rule parameters might benefit from the presence of an external stimulus driving the activity of the presynaptic neuron. We therefore turned to synthetic data to validate this hypothesis and compare inferences in the absence of stimulus, with stimulation protocols in which the presynaptic neuron is triggered at a fixed constant frequency. Figure 3A) shows that stimulation can significantly improve data quality, with the 100Hz protocol systematically outperforming the "no stimulation" and the 250Hz ones. On the contrary, the 250Hz protocol is not always superior to the "no stimulation" one, indicating that the stimulation frequency might require fine tuning. The optimal stimulation frequency indeed depends on a number of factors like synaptic strength and ground-truth learning rule parameters (see Supplementary Material). We therefore moved on to designing a procedure that selects the optimal stimulation frequency in the course of the experiment, as explained in section 4.

4 Bayesian active learning

The main goal of this work is to develop optimal stimuli paradigms in order to achieve satisfactory inference with minimal data required. Such a paradigm will have to exploit the increase in data

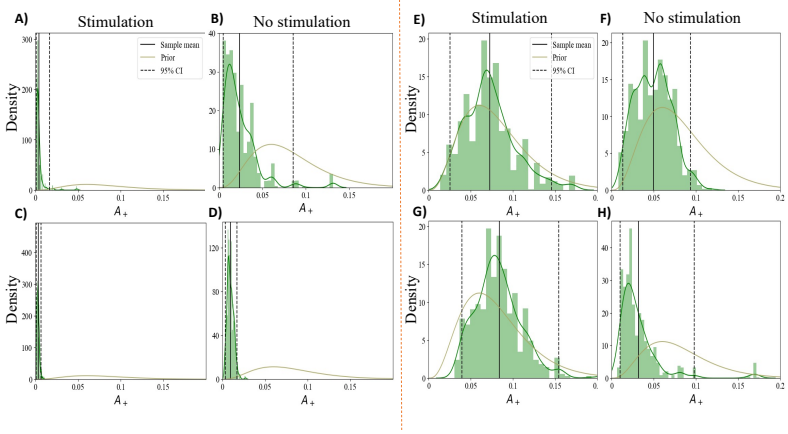

Figure 2: Sample posteriors of STDP parameter A_+ from real data. A)-D): Inference on datasets of 100 seconds. E)-H): Inference on datasets of 20 seconds. A),C),E),G): datasets with repeated 50ms juxtacellular stimulation intervals. B),D),F),H): datasets with no stimulation. Dashed lines delimit 95% credible intervals. Solid green lines were obtained by Gaussian kernel density estimation on the sample posteriors.

quality given by stimulation, by adaptively choosing the stimulus frequency in the course of the experiment. The Bayesian optimal design of experiments framework [23] lends itself for such an active learning approach since it optimises the design, as data gets accumulated by updating our beliefs on the system of interest. This section introduces the theory and methods for the Bayesian active learning procedure and then presents validation on synthetic data.

4.1 Utility function for optimisation

In the Bayesian optimal design framework, given some sample to be yet collected, $Y \in \mathcal{Y}$ and model parameters $\theta \in \mathcal{S}$, one seeks for the design $X \in \mathcal{X}$ which maximises the expectancy of some appropriate utility function, $u(Y, \theta, X)$

$$X^* = \underset{X \in \mathcal{X}}{\operatorname{argmax}} \, E_{Y|X}[u(Y, \theta, X)]. \tag{4}$$

To infer the learning rule parameters θ, our choice for the utility function is the mutual information between the parameters and the data. Furthermore, we adopt an online approach for the experimental design, which under some conditions is guaranteed to improve over random iid selection of the stimulus [24]. That is, as depicted in Figure 3C), we are alternatingly simulating a subset of data and optimising the experiment for the next subset. These subsets are referred to as trials. The time domain is being split into smaller equal sized trials, and between every trial we aim to optimise the design X for the next trial subject to our chosen utility function. By design, for our experiment we specifically mean applying a fixed frequency stimulation on the presynaptic neuron during a whole trial.

Let $\mathcal{D}^n = \{s_1^n, s_2^n\}$ be the data to be collected in trial n, being the observed spike trains. Suppose X^n is an experimental design for trial n, then we can express our utility function as

$$I(\theta, \mathcal{D}^n | X^n, \mathcal{D}^{n-1}) = \sum_{\mathcal{D}^n} \int P(\theta, \mathcal{D}^n | X^n, \mathcal{D}^{n-1}) \ln\Big(\frac{P(\theta, \mathcal{D}^n | X^n, \mathcal{D}^{n-1})}{P(\theta | X^n, \mathcal{D}^{n-1}) P(\mathcal{D}^n | X^n, \mathcal{D}^{n-1})} \Big) d\theta. \tag{5}$$

Notice that the mutual information in Eq. (5) can be rewritten as

$$I(\theta, \mathcal{D}^n | X^n, \mathcal{D}^{n-1}) = -\sum_{D^n} P(D^n | X^n, D^{n-1}) \cdot H(\theta | \mathcal{D}^n, X^n, \mathcal{D}^{n-1}) \quad + \quad \text{const}, \tag{6}$$

where the last term is independent of the experimental design X^n.

The expected utility function is therefore maximised when the posterior entropy of θ is minimised with respect to the data $\mathcal{D}^n$. The entropy of the posterior does indeed well correlate with the accuracy of the estimation even for small data sizes, as illustrated in Figure 3B.

4.2 Estimating the objective function

As explained in section 3, our inference procedure provides us with an empirical posterior for θ which can be approximated by a multivariate Gaussian

$$P(\theta|\mathcal{D}^n, X^n, \mathcal{D}^{n-1}) \approx \mathcal{N}(\boldsymbol{\mu}, \Sigma), \tag{7}$$

where $\boldsymbol{\mu}$ and Σ are the mean and the covariance matrix of the MCMC sample. This approximation in turn yields an estimate for the desired entropy $H(\theta|\mathcal{D}^n, X^n, \mathcal{D}^{n-1})$ in Eq. (6).

Furthermore, by drawing N samples in the spike train space according to $P(D^n|X^n, D^{n-1})$, our utility function can be approximated by importance sampling

$$I(\theta, \mathcal{D}^n|X^n, \mathcal{D}^{n-1}) \approx -\frac{1}{N}\sum_{j=1}^{N} H(\theta|\mathcal{D}^n_{(j)}, X^n, \mathcal{D}^{n-1}), \tag{8}$$

where j labels the realisation from the spike train space.

To be able to search for an optimal design in the frequency space $\mathcal{X}$, but still keep the computational cost at a reasonable level, we employ a grid search in the frequency domain. We assume, that the provided stimulation is strong enough to deterministically trigger the presynaptic neuron. After every trial, the stimulation frequency which maximises the utility function (8), will be the chosen stimulus for generating the next chunk of data.

4.3 Evaluation on simulated data

We tested the performances of the Bayesian active learning procedure in comparison to other standard stimulation protocols on simulated data generated using the spiking and learning rule models, described in section 2. We devised the experiment illustrated in Figure 3C), as composed of 5s trials after which the spike data accumulated up to that time point is used to infer the posterior over the learning rule parameters for each of the stimulation protocols. The mean of such posterior is then used as an estimator for the underlying STDP parameters, while performance is evaluated as the euclidean distance between the generative and estimated learning rule curves, see Figure 1B). After each trial, the posterior of the learning rule parameters is also employed by the Bayesian active learning procedure to optimise the stimulation frequency in the upcoming trial. Notice that Dale's law is enforced to the weight trajectories preventing them from changing sign.

Results of experiments with optimisation and random frequency selection in the frequency range $\{20, 50, 100, 250\}$Hz are displayed in Figure 4A)-C), while Figure 4D)-F) shows results for the frequency range $\{10, 20, 50, 100\}$Hz. Interestingly the Bayesian active learning procedure leads to depression of the synaptic weight in the frequency range $\{20, 50, 100, 250\}$Hz, Figure 4A). Such a strategy reflects our specific choice for the generative learning rule parameters (see Supplementary Material), where depression is favoured over potentiation. Specifically, we hypothesise that the optimal design chooses to probe pre-post synaptic delays for which the learning rule is the steepest.

Performances of the different protocols are visualised in Figure 4C) and 4F). As expected from the inference results on real and simulated data, Figures 2-3 all stimulation protocols outperform the "no stimulation" one, with the Bayesian active learning improving over the latter of one order of magnitude in the course of the experiment. The random frequency achieves good performances, comparable with the Bayesian active learning of the protocol only in the late phases of the experiment. This can be explained by looking at the frequencies chosen by the Bayesian active learning stimulation, Figure 4B),E). If at the start of the experiment, the optimal design privileges the highest frequency, towards the end of the experiment the distribution of chosen frequencies is more uniform. Remarkably this trend seems to be independent of the generative synaptic weight trajectory, compare Figure 4A),D).

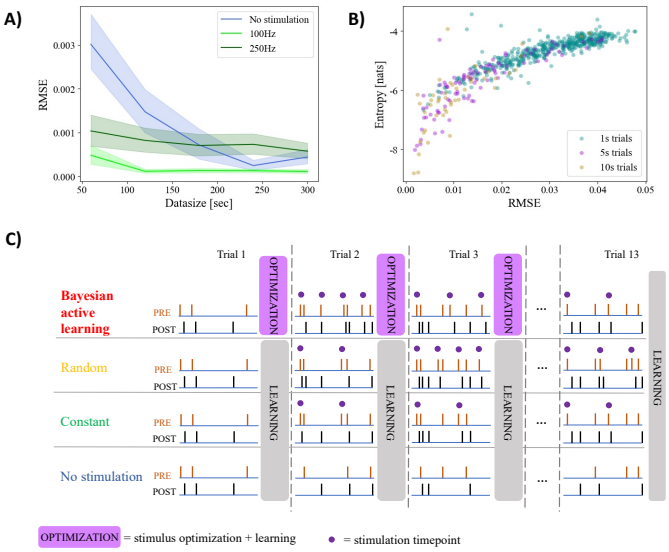

Figure 3: Stimulation on synthetic data. A): RMSE of the STDP learning rule on synthetic data as a function of the datasize for different stimulation protocols. Solid lines correspond to RMSE means across 20 datasets, while shaded regions to 95% confidence intervals. B): Entropy of the posterior in the STDP parameters space vs RMSE of the estimated learning rule for different trial durations. C): Experimental setting and inference for the considered stimulation protocols. The Bayesian active learning procedure optimises the stimulus for the next trial based on the inference of the STDP parameters from the history of spike trains.

5 Limitations

Our framework adopts phenomenological models for the learning rule and spiking process. The latter assumes a fixed time kernel for the integration of the presynaptic input. The model does not account for burstiness or fatigue, while the refractory period is enforced through the time bin size choice. Extending our model in these directions is straightforward, however, accurate inference of these additional parameters in practice might require longer recordings or parameter sharing across neurons.

A more principled approach to model misspecification within our framework would be to place a prior distribution over hyperparameters so that the objective function incorporates model uncertainty [25].

A potential limitation of the protocol that we have introduced is that it is "greedy": it chooses the stimulus that maximises the objective function at each trial. The greedy approach may be sub-optimal compared to strategies that select stimuli based on the objective function over some finite number of trials in the future. For example, in our experiments, we observe that when the Bayesian active learning procedure selects the 250Hz stimulation, best performances are achieved in the short run, while accuracy saturates in the long run due to depression of the synaptic weight, see Figure 4. Unfortunately computational complexity of the optimisation over multiple trials into the future makes this a challenging problem to undertake. On the bright side, a technical result [24] shows that greedy Bayesian active learning methods are still provably better than standard methods under certain consistency conditions.

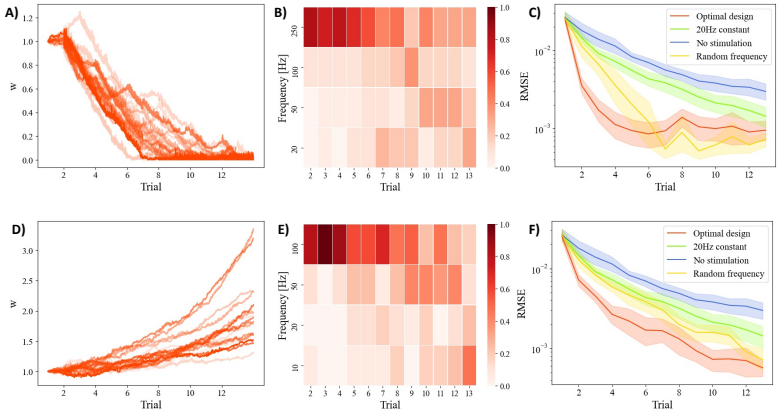

Figure 4: Bayesian active learning performances. A)-C): Stimulus optimisation in the frequency interval $[20Hz, 250Hz]$. D)-F): Stimulus optimisation in the frequency interval $[10Hz, 100Hz]$. A),D): Synaptic weight trajectories induced by the Bayesian active learning protocol. Higher transparency reflects lower RMSE of the learning rule after trial 13. B),E): Frequency heat maps showing the fraction of experiments in which the different frequencies were chosen at each trial. C),F): RMSE of the learning rule versus the number of trials into the experiment for different stimulation protocols. Solid lines indicate RMSE averages over 20 independent experiments, while the shaded regions correspond to 95% confidence intervals.

Another limitation of the method is the computational complexity of the MCMC inference procedure. At every optimisation step (after each trial in our experiment) for every frequency considered, one has run the inference algorithm explained in section 3 N times (see eq.(6)). Our simulations were performed on a CPU, and there we estimate the time consumption for one optimisation step of around 1,5 hours per frequency when $N = 15$. Note that the algorithm is parallelisable over the N samples per frequency, parallelisation which would shorten the required running time significantly.

6 Discussion

Motivated by the technological advancements paving the way for closed-loop neuroscience, we have devised a Bayesian active learning algorithm to optimally infer the STDP learning rule, building on the inference framework introduced in [12]. Our initial results suggest that our paradigm boosts the efficiency of the inference and outperforms standard stimulation protocols. This work also indicates that the randomised stimulation protocol, despite being sub-optimal, improves over constant frequency stimulation on synthetic data, a result which may inform experimental settings. Beyond the proposed and considered protocols in this paper, the Bayesian inference approach we developed from [12] is suitable for enforcing a stopping criterion to the experiment based on the entropy of the posterior. The latter, as we have shown in Figure 3B), indeed correlates well with the RMSE of the learning rule parameters.

In the future, in addition to the model augmentation mentioned in section 5, we plan to improve the proposed optimisation protocol by increasing its computational efficiency to make it suited for inference on populations of neurons. For this purpose, resource optimisation can be achieved by Bayesian active learning procedures which select the neuron pair to be stimulated [26], while keeping the computational complexity under control via, for example, variational Bayes approximations.

Another avenue for research would be to try to make the algorithm less “greedy”. A straightforward attempt in this direction, inspired by the results plotted in Figure 4, would imply a small addition to

the proposed objective function (mutual information), namely a regulariser penalising stimulation frequencies which in the long run lead to small synaptic weights.

Adoption of our stimulation paradigm by experimental labs will help towards being able to systematically probe the learning rule across brain regions and neuron types. It is known that there are a number of factors that influence STDP induction, such as neuromodulation [25] and homeostasis [26], however, including these factors into a model would make inference difficult or impossible. This framework should allow to identify the learning rule at shorter time scales than those at which metaplasticity operates and further enable the study of the influence of these factors. Indirectly this framework could also inform bio-inspired neural networks [27] through discoveries on the functional organisation of STDP in the brain and its role in computation, prompting architecture and learning in this relatively novel paradigm in neural computation.

The paradigm for resource optimisation that we propose in this paper conforms well to the ethical guidelines of NeuroIPS. It contributes to reducing the environmental impact of basic research on synaptic plasticity by limiting computational resources and leveraging high density recordings, as opposed to single cell intracellular ones.

Acknowledgments and Disclosure of Funding

Acknowledgments

The authors are thankful to Astrid Langsrud for inspiring this work through her master thesis. We are also grateful to the Python Software Foundation for developing and maintaining the language we adopted for this project. High Performance Computing resources were provided by NTNU, IDUN cluster. [28].

References

[1] Whitlock JR, Heynen AJ, Shuler MG, Bear MF. Learning induces long-term potentiation in the hippocampus. science. 2006;313(5790):1093–1097.

[2] Pastalkova E, Serrano P, Pinkhasova D, Wallace E, Fenton AA, Sacktor TC. Storage of spatial information by the maintenance mechanism of LTP. science. 2006;313(5790):1141–1144.

[3] Hebb DO. The organization of behavior; a neuropsycholocigal theory. A Wiley Book in Clinical Psychology. 1949;62:78.

[4] Malenka RC, Bear MF. LTP and LTD: an embarrassment of riches. Neuron. 2004;44(1):5–21.

[5] Markram H, Lübke J, Frotscher M, Sakmann B. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science. 1997;275(5297):213–215.

[6] Bi Gq, Poo Mm. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of neuroscience. 1998;18(24):10464–10472.

[7] Caporale N, Dan Y. Spike timing–dependent plasticity: a Hebbian learning rule. Annu Rev Neurosci. 2008;31:25–46.

[8] Froemke RC, Dan Y. Spike-timing-dependent synaptic modification induced by natural spike trains. Nature. 2002;416(6879):433–438.

[9] Mansvelder HD, Verhoog MB, Goriounova NA. Synaptic plasticity in human cortical circuits: cellular mechanisms of learning and memory in the human brain? Current opinion in neurobiology. 2019;54:186–193.

[10] Bell CC, Han VZ, Sugawara Y, Grant K. Synaptic plasticity in a cerebellum-like structure depends on temporal order. Nature. 1997;387(6630):278–281.

[11] Oweiss KG, Badreldin IS. Neuroplasticity subserving the operation of brain–machine interfaces. Neurobiology of disease. 2015;83:161–171.

[12] Linderman S, Stock CH, Adams RP. A framework for studying synaptic plasticity with neural spike train data. Advances in Neural Information Processing Systems. 2014;27:2330–2338.

[13] Nelder JA, Wedderburn RW. Generalized linear models. Journal of the Royal Statistical Society: Series A (General). 1972;135(3):370–384.

[14] Bliss TV, Lømo T. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. The Journal of physiology. 1973;232(2):331–356.

[15] Morgan S, Teyler T. Electrical stimuli patterned after the theta-rhythm induce multiple forms of LTP. Journal of Neurophysiology. 2001;86(3):1289–1296.

[16] Albensi BC, Oliver DR, Toupin J, Odero G. Electrical stimulation protocols for hippocampal synaptic plasticity and neuronal hyper-excitability: are they effective or relevant? Experimental neurology. 2007;204(1):1–13.

[17] Tanskanen JM, Ahtiainen A, Hyttinen JA. Toward Closed-Loop Electrical Stimulation of Neuronal Systems: A Review. Bioelectricity. 2020;2(4):328–347.

[18] Park M, Pillow J. Bayesian active learning with localized priors for fast receptive field characterization. Advances in neural information processing systems. 2012;25:2348–2356.

[19] Franke F, Jäckel D, Dragas J, Müller J, Radivojevic M, Bakkum D, et al. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. Frontiers in neural circuits. 2012;6:105.

[20] Andrieu C, Doucet A, Holenstein R. Particle markov chain monte carlo methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2010;72(3):269–342.

[21] Haario H, Saksman E, Tamminen J. Adaptive proposal distribution for random walk Metropolis algorithm. Computational Statistics. 1999;14(3):375–395.

[22] English DF, McKenzie S, Evans T, Kim K, Yoon E, Buzsáki G. Pyramidal cell-interneuron circuit architecture and dynamics in hippocampal networks. Neuron. 2017;96(2):505–520.

[23] Lindley DV. On a measure of the information provided by an experiment. The Annals of Mathematical Statistics. 1956:986–1005.

[24] Paninski L. Asymptotic theory of information-theoretic experimental design. Neural Computation. 2005;17(7):1480–1507.

[25] Roy N, McCallum A. Toward optimal active learning through monte carlo estimation of error reduction. ICML, Williamstown. 2001:441–448.

[26] Shababo B, Paige B, Pakman A, Paninski L. Bayesian Inference and Online Experimental Design for Mapping Neural Microcircuits. In: NIPS. vol. 26; 2013. p. 1304–1312.

[27] Vigneron A, Martinet J. A critical survey of STDP in Spiking Neural Networks for Pattern Recognition. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE; 2020. p. 1–9.

[28] Själander M, Jahre M, Tufte G, Reissmann N. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. arXiv:191205848 [cs]. 2019 Dec.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We comment on it in section 6
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplementary Material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figures2,3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Acknowledgments section
   (b) Did you mention the license of the assets? [No] The assets are free license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data owner is one of the authors of the paper.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The data is not human and has been collected and handled according to the relevant ethics code.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

NTNU

Norwegian University of
Science and Technology