Dissertation Type: research

DEPARTMENT OF COMPUTER SCIENCE

Estimating Neural Input Statistics from Whole-Cell Electrophysiology

David Sharp

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Wednesday 27th May, 2020

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

David Sharp, Wednesday 27th May, 2020

Contents

Abstract

This project explores how the population statistics of excitatory and inhibitory inputs to a neuron can be inferred via numerical optimisation of the power spectrum of a recording of membrane potential from that neuron.

A simulation was developed to investigate if a commonly used numerical optimisation algorithm suited the task well and then the algorithm was tested on real-world data from *in-vivo* recordings of mice hippocampi to examine how well the algorithm performed on real data compared to synthetic recordings from the simulation.

Work was split over a number of key areas from November 2019 to May 2020 with approximate spread of workload:

- Researching related work and examining Puggioni's analytical model[16] : 5% time
- Exploration of data obtained from Dr. Jon Palacios including importing data from recording format : 5% time
- Development of Leaky Integrate and Fire simulation : 35% time
- Integrating *pycma* with real-world to optimise simulation to fit on power spectra and other statistics : 30% time
- Exploration of basic optimiser results to obtain more nuanced visualisations/measures : 25%

The key deliverables of this project are an evaluation of an numerical optimisation algorithm used to adjust parameters of a neural simulation to fit real-world patch-clamp recordings, and an initial version of the code required to carry out this method on real-world recordings. This initial version is available at `https://github.com/dsPolar/poseidon`.

Supporting Technologies

Python3 was the language used for this project along with some of its common modules,

1. *numpy* for array management and random variables
2. *scipy* for signal processing
3. *multiprocessing* and *itertools* were used for code parallelisation

A couple more niche modules were also used,

1. *neo* for importing the neuron recordings [4]
2. *pycma* for the implementation of the CMA-ES algorithm [3]

Notation and Acronyms

CMA-ES : Covariance Matrix Adaptation - Evolution Strategy
LINF : Leaky Integrate and Fire (model)

in-vivo : Recording from a animal while it is alive
in-vitro : Recording from brain slices

λ_x : The firing rate of input neurons of type x, excitatory or inhibitory
μ_x : The mean synaptic strength of input neurons of type x, alternately mean conductance
σ_x : The standard deviation of synaptic strength
τ_x : The conductance decay time constant for neurons of type x
$\mathcal{F}_y(\omega)$: The Fourier transform of a signal $y(t)$
$\mathcal{P}_y(\omega)$: The Power Spectrum or Power Spectral Density of a signal $y(t)$

Acknowledgements

I would like to take this opportunity to thank Cian for being an excellent supervisor, both the in-person and online meetings have never failed to motivate me to solve the next problem.

I would also like to thank Jon for generously providing the recordings used in this project, the results section would likely be a lot emptier without them.

Chapter 1

Motivation

1.1 Objectives

Neuroscientists are trying to understand the way neural circuits, populations of connected neurons, represent and process information. A common way to measure electrical activity from a neuron is patch-clamping which yields a trace of the membrane potential of the cell, also called whole-cell electrophysiology. This method only gives a single time series from one neuron when neuroscientists would like to know about the activity of the whole circuit.

Inference of behaviour and statistics of this circuitry is a challenging problem as typically a neuron might receive input from anywhere from 10^2 to 10^4 other neurons depending on cell type. While the firing rate of an individual input might only be on the scale of $10^-1 - 10^0$Hz, the sheer number of inputs makes it difficult to separate input events within a patch-clamp recording. There exists no good way currently to infer statistics of the input population given a single patch-clamp recording of membrane potential.

The objective of this project is to assess the efficacy of using an optimisation algorithm to try and infer the neural properties (e.g firing rate or synaptic strength) of the inputs to a neuron that has been recorded from experimentally via patch-clamping. In order to achieve this objective, sub-goals were created:

1. Develop a simulation that modelled a neuron and its input synapses.
2. Attempt to recover known simulation parameters from the power spectra and other statistics of the simulation trace, using an optimisation algorithm.
3. Set up the optimisation algorithm to alter the simulation parameters in order to try and match the power spectra and other statistics from a real-world trace and thus retrieve the real-world input parameters.

The recordings used in this project were kindly given by Dr. Jon Palacios of the School of Pharmacology, Physiology and Neuroscience, of the University of Bristol. Dr. Palacios has provided several recordings of membrane potential from pyramidal cells in the hippocampus of a live mouse, and was going to later make recordings of mice while their levels of the neurotransmitter *Acetylcholine* had been modified and it would be of great benefit to his project to be able to see not only how the recordings changed but also whether specific properties of the input were more greatly affected than others. The project aims to use these recordings as typical real-world data to develop a method that works generally, rather than as a specific problem domain to be solved.

If this project succeeds in its aims, and that technology can be further built upon, then experimental neuroscientists will gain a new tool to more effectively let them extract information about brain function from recordings of electrical activity. This would allow us to gain greater insight into how the brain processes information within circuits of neurons and further develop theories of brain function.

The key deliverables of this project are an evaluation of an numerical optimisation algorithm used to adjust parameters of a neural simulation to fit real-world patch-clamp recordings, and an initial version of the code required to carry out this method on real-world recordings. This initial version is available at https://github.com/dsPolar/poseidon.

Currently there are several ways to try and measure neural activity, for example recordings of electrical activity or imaging methods like calcium imaging. Electrical recordings have some key advantages over other existing methods, specifically for the aims of this project, precise timescale and high signal to noise ratio. These recordings can be made at sampling rates of tens of thousands of Hertz compared to the much lower sampling frequencies found in imaging techniques of around 3Hz[17]. However a drawback of these electrical recordings is that - especially *in-vivo* - it is very difficult to record from multiple parts of a neuron and difficult to record from the inputs to a specific neuron since the axon is much longer than the soma so inputs may be very sparse in space.

1.2 Related Work

This problem of inferring input statistics is inherently difficult and has not been attempted on the same scale in the past. We only have a single recording to be able to try and estimate statistics of what could be up to tens of thousands of inputs to some neuron types (like the pyramidal cells in this project), so it is not clear how much can even be inferred from limited data.

The attempts of Puggioni in 2015 [16] to carry out a similar task of input statistics inference used cerebellar granule cells which have a smaller soma and have less dendrites, so have less inputs and suffer less attenuation from travel distance in action potentials before they reach the soma [12][6]. Puggioni used an analytical model meaning that the underlying neural model cannot be easily adjusted to fit other neuron types or input quantities without adjustment of the analytical model. The simulation model used in this project could instead be freely adjusted without significant changes to the optimisation approach used.

More broadly the task of parameter inference and model optimisation has been pursued extensively, the key difference being the parameters inferred are of the cell being recorded from not the parameters of its inputs. Tabak et al. in 2000 [18] used the Nelder-Mead Simplex algorithm to estimate parameters for a Hodgkin-Huxley-type model, a model of a neuron with ion gates and some input current. Vavoulis et al. in 2012 [22] with a similar Hodgkin-Huxley-type model used a variation of Covariance Matrix Adaptation for parameter with a greater number of parameters than Tabak et al. The approaches used in these works are insufficient to be applied to the harder problem of inferring input population statistics due to the neural model used.

A review of the different commonly used families of algorithms for neuron model optimisation was written by Van Geit et al. in 2008 [21] and gives a good, broad view of the general behaviour of the tools.

1.3 Skillset

Overall this project likely fits more cleanly within the remit of the field of data science, and that is to a degree how the project was approached; with data exploration and model development. But also it clearly requires knowledge of the data itself i.e neuroscience to be able to effectively select a model and understand which optimisation techniques will best suit the problem space.

Chapter 2

Technical Background

2.1 Neuroscience Background

2.1.1 Neurons

Neurons are a type of cell present in most animals that use electrical and chemical signals to communicate with other cells [13]. Neurons have a soma (the central part of the cell) which is about 20-30μm across, a long axon, and thin dendrites about 1-4μm in length that use synapses to communicate with neurons and other cell types. The entire cell has a membrane surrounding it that allows the neuron to act as a capacitor, storing charge and releasing it in a single burst called an action potential (alternatively spike or nerve impulse) once the potential of the cell passes the firing threshold. This action potential travels along the axon to the synapses. The difference in electrical potential across the cell membrane is called the membrane potential, typically valuing around negative 40-80mV but can reach higher during an action potential. The specific type of neuron being modelled in this project is a pyramidal cell within hippocampus CA1, the first subarea within the hippocampus. Measurements given are based on this cell type. Principles of Neural Science [13] gives a strong introduction to any additional neuroscience not covered in this chapter if that is required.

Synapses

Synapses are bulb-like protrusions from dendrites that use chemical neurotransmitters to intercommunicate and pass on the action potential. Two synapses from different neurons are separated by a small gap of about 20-40nm called the synaptic cleft. When the presynapse receives an action potential it releases small stores of neurotransmitter called vesicles, the vesicles travel towards the cleft and releases the neurotransmitter. The neurotransmitter diffuses across the cleft and is picked up by receptors in the postsynapse that triggers an action potential in the postsynaptic cell, that then propagates to the soma.

Refractory Period

Neurons are only able to generate action potentials so frequently, after every action potential parts of the cell need to ‘reset‘ in order to be able to generate a new action potential. This time is called the refractory period and is typically around 1-5ms long. Part of this refractory period is due to needing to adjust ion concentrations in the cell, which carry the charge and enable the rapid increase in membrane potential, another part of the period is due to needing to replenish the vesicles in the synapse with neurotransmitter.

There are two parts to the refractory period, absolute and relative. The absolute refractory period is shorter and is when the cell will not let in any more sodium (Na^+) ions so it is impossible for another action potential to be triggered. The relative refractory period is longer and is when the cell is still resetting the ion concentrations so it is far more difficult to trigger a new action potential, the effective firing threshold is higher than normal. The cell can still receive incoming action potentials from other neurons during this period, it is just more unlikely that they will trigger new action potentials.

Excitation and Inhibition

While there are many ways to categorise neurons, for the purposes of this thesis it suits to separate them by their effect on other neurons. Excitatory neurons serve to encourage an increase in membrane potential, and thus encourage generation of action potentials. Inhibitory neurons on the other hand discourage the generation of action potentials, reducing overall activity. Typical neuron behaviour relies upon a balance of excitation and inhibition; over-excitation may have a domino effect in a neural circuit, while over-inhibition may result in the neuron never generating action potentials.

2.1.2 Recordings

The neural recordings used in this thesis from Jon Palacios were recorded *in vivo*, in the animal (a mouse) while it is still alive and active. This involves placing an electrode into the animal's brain so that we can record the membrane potential of a single neuron over time. While this allows experimenters to record neural activity when the brain is fully connected up, the process of inserting the electrode into the cell combined with the fact that the mouse is able to move around which causes small amounts of brain movement means that recordings cannot be as long as *in vitro* recordings where slices are used. Due to the comparative sparsity of inputs to a neuron compared to the density of neurons in an area it is difficult to try and record the inputs to a specific neuron.

There is the potential for noise to be introduced into these recordings, both from magnetic and high frequency electrical sources. Most noise that could be introduced to the recordings is likely to have frequencies quite a bit higher than the expected neural input firing rates.

2.2 Power Spectra

The power spectrum of a time series describes the power of component frequencies of the signal and has units $\frac{V^2}{Hz}$.

For a signal with known function $y(t)$, the power spectrum is computed as

$$\mathcal{P}_y(\omega) = 2\mathcal{F}(y(\omega))^*\mathcal{F}(y(\omega)) \tag{2.1}$$

where $\mathcal{F}(y(\omega))$ is the Fourier transform of the signal and * denotes the complex conjugate [16].

Alternatively the auto-correlation function $R(\tau)$ can be used where

$$R_y(\tau) = E[y(t)y(t+\tau)]\mathcal{P}_y(\omega) = \int_{-\infty}^{\infty} R_x(\tau)e^{-2\pi\omega t}dtR_y(\tau) = \int_{-\infty}^{\infty} \mathcal{P}_y(\omega)e^{-2\pi\omega\tau}d\omega \tag{2.2}$$

if we set $\tau = 0$ in (2.2) then

$$R_y(0) = E[y(t)y(t)] = \int_{-\infty}^{\infty} \mathcal{P}_y(\omega)d\omega \tag{2.3}$$

(2.3) shows that if we assume the underlying signal has zero mean then the integral of the power spectrum yields the variance in the signal [8].

More simply the power spectrum also contains information about the strength of frequencies present in the signal which will allow detection of received input frequencies in the recordings.

2.3 CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is the optimisation algorithm used in this thesis and here will be offered a brief overview of how it operates. Why this optimisation algorithm has been chosen in particular for this task will be discussed later.

2.3.1 Overview

CMA-ES is a stochastic derivative-free optimisation algorithm for non-linear or non-convex spaces [11]. It broadly functions by getting a number of candidate parameter fits by sampling from a multivariate Gaussian, which it then evaluates the error for allowing the algorithm to estimate the shape of the error plane surrounding the candidates. It then calculates an evolution path, the path in which a lower error space lies, and updates the parameters of the Gaussian according to the vector of this path. 2.1 shows the

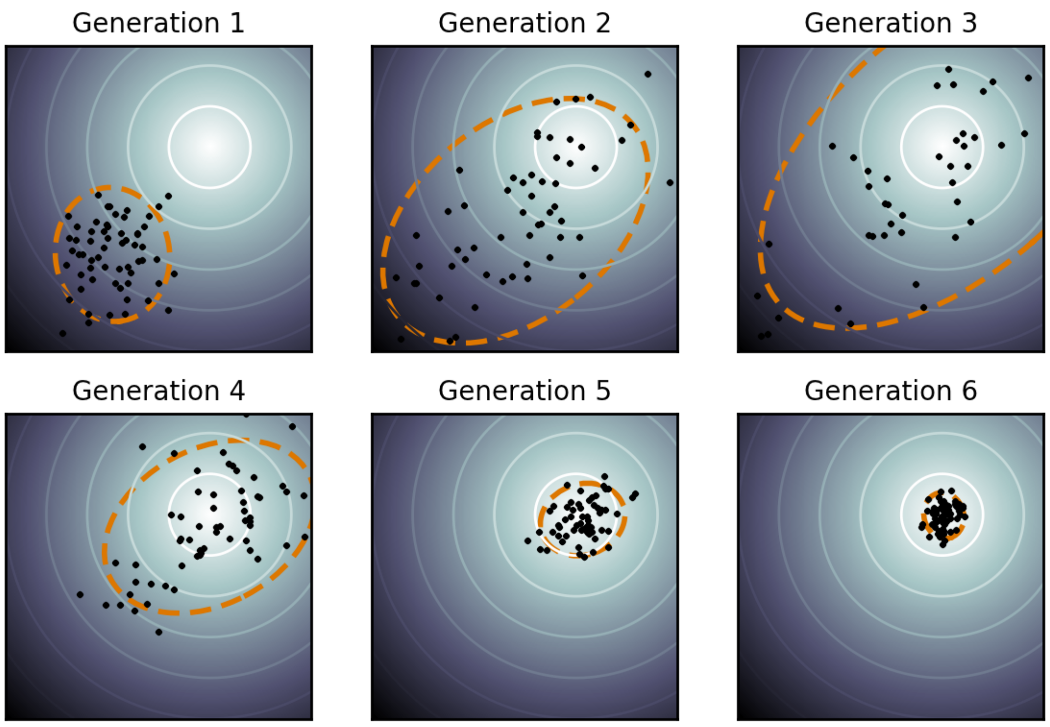

Figure 2.1: Approximation of CMA-ES approach in two dimensions. Sourced from Wikipedia

approach for a simple two-dimensional error space, where the dots represent the candidate parameter fits and the ellipse is the underlying Gaussian. This example uses a significantly higher than required number of sampled candidates. (2.4) shows the how the population size, p (alternatively λ in other texts), where $\mathcal{D}$ is the dimensionality of the error space. So for a two-dimensional error space, six candidates would be sampled per iteration. A tutorial [10] by Hansen gives a description of the intricacies of the algorithm that cannot be explored here.

$$p = \lfloor 4 + 3 \log \mathcal{D} \rfloor \tag{2.4}$$

Chapter 3

Project Execution

3.1 Method

The work in this project followed this general form. This chapter will elaborate the steps taken

1. Analysis of real-world traces
2. Simulation Development
3. Generating power spectra from simulation
4. Fitting to synthetic trace (known parameters)
5. Fitting to real-world trace (unknown parameters)

The simulation (and project in general) was coded in *Python3*, with the use of *git* and *Github* for version control. The implementation of the CMA-ES optimisation algorithm was *pycma*[3].

3.1.1 Equation Approach

Originally we hoped to use the work of Puggioni[16] to be able to optimise the parameters with his equation for the power spectra given the input parameters (3.1), however it seemed to be a poor fit for the data in this case and consistently underestimated the magnitude of the spectrum. Because of this, it was decided to instead evaluate a given parameter setting by running the simulation and then computing a power spectrum from it, rather than simply trying to compute the power spectrum from the parameter values via an analytical model.

$$\mathcal{P}_V(f) = \left(\frac{R_{eff}^2}{1+(2\pi f\tau_{eff})^2}\right)\left(\frac{2N_e\nu_e\tau_e^2(\mu_e^2+\sigma_e^2)(E_e-\overline{V})^2}{1+(2\pi f\tau_e)^2} + \frac{2N_i\nu_i\tau_i^2(\mu_i^2+\sigma_i^2)(E_i-\overline{V})^2}{1+(2\pi f\tau_i)^2}\right) \tag{3.1}$$

3.2 Initial Data Analysis

While there were ten total traces in use as part of this project, a few are included here to give an idea of what these traces look like since they fall into broad categories. Figure 3.1 shows a trace with a relatively small variance in membrane potential with a mean around -60mV, figure 3.2 has distinct periods of different mean membrane potentials and thus a higher overall variance in membrane potential. These two traces represent a split in the ten traces, with seven being more like figure 3.1 and three like figure 3.2 with a high variance in membrane potential. While methods to try and effectively model the different periods in figure 3.2 were considered - and shall be briefly discussed - the project largely ignores these distinct steps in membrane potential. These were recorded at 20000Hz sampling rate and so comprise a vast amount of data points.

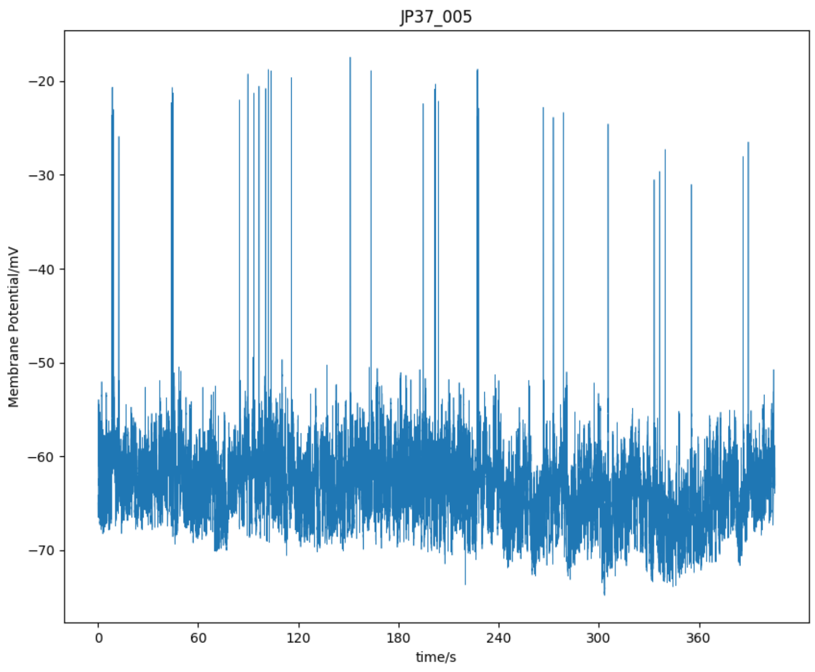

Figure 3.1: Low variance membrane potential trace

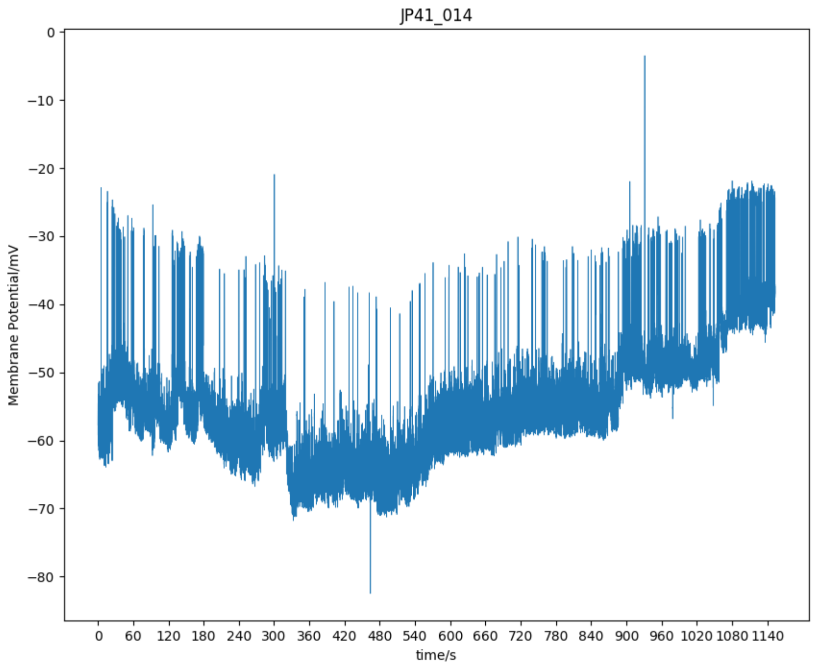

Figure 3.2: High variance membrane potential trace

3.2.1 Importing the data

The data was originally encoded in the *smr* file format used by the software used to record the traces - Spike2 [2]. The Python module *neo* [4] was used to separate the recording channels and then convert the data channel into a *numpy* array. These arrays were then pickled and the rest of the project unpickles the file back into a numpy array when it is required.

3.3 Simulation

The purpose of the simulation was to be able to generate synthetic traces that behaved similarly to the real traces provided by Jon Palacios. Synthetic traces could then be used to test the parameter fits created by the chosen optimisation algorithm as the true parameter values used to generate the trace were completely known.

As we had no data on the stimulus provided that evoked the trace and only had to model the membrane potential of a single neuron with many inputs, the Leaky Integrate and Fire (LINF) model was chosen.

3.3.1 The Leaky Integrate and Fire (LINF) model

The LINF model is relatively simple and has two key equations, a membrane potential update equation and a conductance update equation. These equations are given below.

$$C_m \tau_m \frac{dV}{dt} = g_l(E_l - V) + g_e(E_e - V) + g_i(E_i - V) \tag{3.2}$$

$$\tau_e \frac{dg_e}{dt} = -g_e \qquad\qquad \tau_i \frac{dg_i}{dt} = -g_i \tag{3.3}$$

The prototypical analogy for the leaky integrate and fire model is that of a bucket holding water, but with multiple holes. As the water level gets further from the height of the whole the leak rate increases. The first term in (3.2) is the leak term where g_l is a fixed value for the leak conductance and tries to pull the membrane potential V towards the leak resting potential E_l, typically around -90mV. The second and third terms refer to the excitatory and inhibitory inputs respectively, with variable conductances. The excitatory resting potential is typically around 0mV while the inhibitory resting potential sits around -62mV - which implies that the inhibitory term is positive if membrane potential is under -62mV. This whole potential update equation is controlled by the time scale of τ_m and capacitance of the membrane, C_m.

The conductance update equations (3.3) cause both the excitatory and inhibitory conductances (increased by incoming action potentials from the input synapses) to exponentially decay towards 0 in the absence of input at a rate dependent on the time constant. If we wanted a more realistic model we might also incorporate a rise time constant, where it takes some time for an input action potential to have full impact on the conductance, however typically the rise constant is a lot smaller than the decay constant so for this model we simply assume that the rise constant is 0.

3.3.2 Firing

Strictly speaking we are using the LINF model just as a leaky integrator, i.e a circuit with capacitance and resistance. The variation in use here makes no attempt to model the action potentials generated by the cell we are recording from, this is partially because the circuit in our model - of many inputs to one neuron that we record - has no feedback so the spikes have no impact upon the inputs that we are trying to fit the parameters of. Further even a visual examination of 3.1 or 3.2 shows that the firing rate of the pyramidal neuron we are attempting to model is quite low, perhaps around one every couple of seconds in the more active traces, so should have limited impact on the relevant areas of the power spectra of the traces for fitting purposes.

3.3.3 Synaptic Strength

A synaptic connection typically has some strength, or weight, that determines how much impact a received action potential has upon the membrane potential of the neuron. There are many different biological factors that lead to differences in synaptic strength between connections and to model these differences the simulation samples a log-normal distribution. This leads to a higher proportion of lower values,

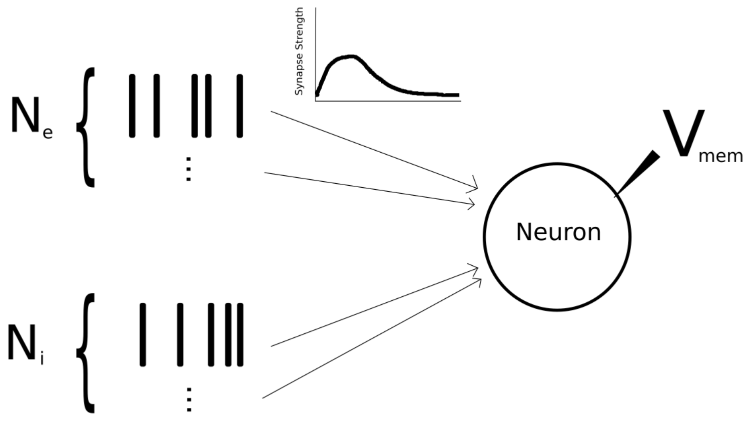

Figure 3.3: Diagram of simulation structure

modelling many synapses being at the end of the axon with action potentials being attenuated by the distance travelled to the soma, and a lower proportion of high values for synapses close to the soma.

3.3.4 Input

In order to model the input synapses to the neuron, independent homogeneous Poisson processes were used. The structure of the simulation is summarised in figure 3.3 with excitatory and inhibitory input spike trains causing increases in conductances respectively in the neuron with a log-normal synaptic strength distribution. The simulation uses a population of 1000 inputs with each synapse selecting a spike delay via the formula for sampling an exponential distribution - as intervals between events in a Poisson process are exponential:

$$t_k = t + \frac{-\ln U}{\lambda_x} \tag{3.4}$$

Where t_k is the time of the next spike event, t is the current time, λ_x is the excitatory/inhibitory rate parameter, and U is sampled uniformly from the range [0..1].

The use of independent homogeneous Poisson processes requires two assumptions to be made [7]: that uncorrelated input can serve effectively, and that the mean firing rate of the inputs does not change over time.

The number of inputs is on the low end for an accurate model of a pyramidal cell, which typically receive inputs in the scale of 10^4 [14]. This is a tradeoff for the model and should not make a significant difference for statistics inference since we can only infer population averages rather than the statistics of a specific input, in this way each input in the simulation could be treated as a combination of 10 inputs. Indeed other attempts [16] at inferring input statistics have used a model with a two synapses, one excitatory and one inhibitory.

3.3.5 Justification of Model Choice

The LINF model is commonly used both historically and in modern neuroscience [7] and has been shown to effectively model many different neuron types [19]. The task at hand would likely not benefit from a more biologically realistic model like the Hodgkin-Huxley models used in single neuron model parameter inference [18][22], especially since the focus is on the inputs and connections to the neuron. Additionally a simpler model will result in increased error evaluation speed which will be of significant benefit when fitting parameters.

3.3.6 Simulation Parameters

Simulation parameters for testing were selected from *NeuroElectro* [20][5] filtered to select from papers on Hippocampus CA1 pyramidal cells (target recording cells for the real-world data). In full : $E_e = 0mV$, $E_i = -62mV$, $E_l = -90mV$, $\tau_e = 3ms$, $\tau_i = 10ms$, $\tau_m = 25ms$, $\lambda_e = 80Hz$, $\lambda_i = 20Hz$, $\mu_e = \mu_i = 0.00075\mu S$, $\sigma_e = \sigma_i = 0.00075\mu S$, $C_m = 0.09nF$. The simulation had 800 excitatory inputs

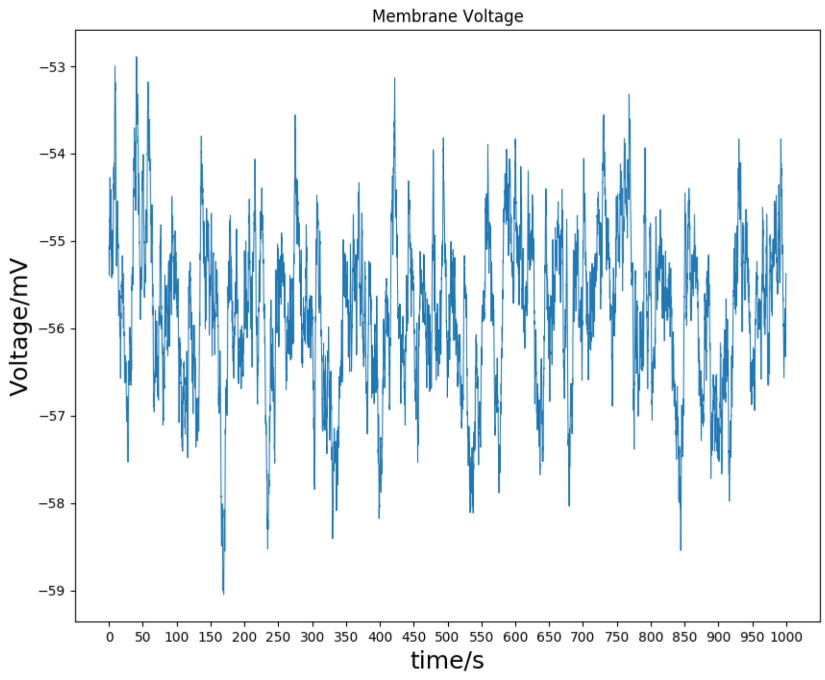

Figure 3.4: 1 Second Simulation Trace

and 200 inhibitory inputs which each were assigned a synaptic strength from a log-normal distribution, with parameters (μ_x, σ_x), at initialisation and this strength did not change over the simulation. It is important to note that the values of λ_x refer to the firing rates of the input neurons individually rather than the frequency of received action potentials at the recorded cell's soma - which is on average $N_x\lambda_x$ for excitatory/inhibitory inputs.

3.3.7 Sample Simulation Traces

Figure 3.4 shows a trace from a 1 second simulation (with timestep of 0.05ms to achieve 20000Hz sampling rate), while figure 3.5 shows a 30 second trace. While the mean membrane potential does not match that of the real-world data that is not a problem as the mean membrane potential can be altered by the parameters we are trying to fit, especially the ratio of the two firing rates, and because the simulation does not need to fit the data perfectly, it exists as a model to explore how well the optimiser works since we know all the parameters of the simulation. If the simulation trace had had a completely different shape then a different model might have had to be used.

3.4 Power Spectra

Initial spectra from the simulation were estimated via Welch's method [23][15] with constant width frequency windows between 0 and 10000Hz. The Periodogram estimation method was tested but did not perform as consistently on the real data, perhaps due to noise in the signal. The equation described in [16] (3.1) was used to see how well it compared to the estimation methods but as previously mentioned it consistently underestimated the magnitude and did not match the general curve of the power spectra, as such Welch's method was used going forward. 3.6 and 3.7 show the power spectra of the real-world traces in 3.1 and 3.2 respectively. 3.8 shows the power spectrum of a 1 second synthetic trace and 3.9 shows the power spectrum of a 30 second synthetic trace with the same parameters.

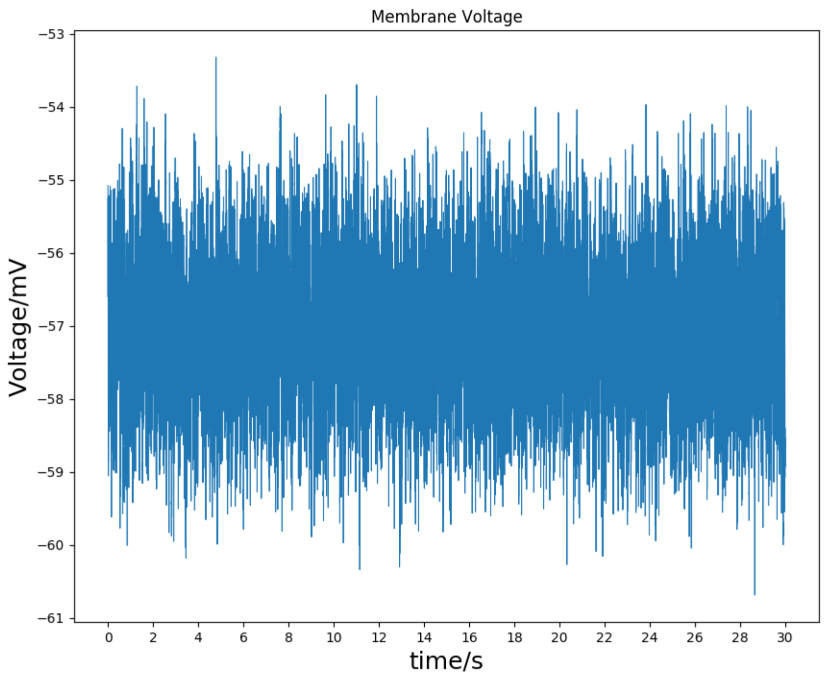

Figure 3.5: 30 Second Simulation Trace

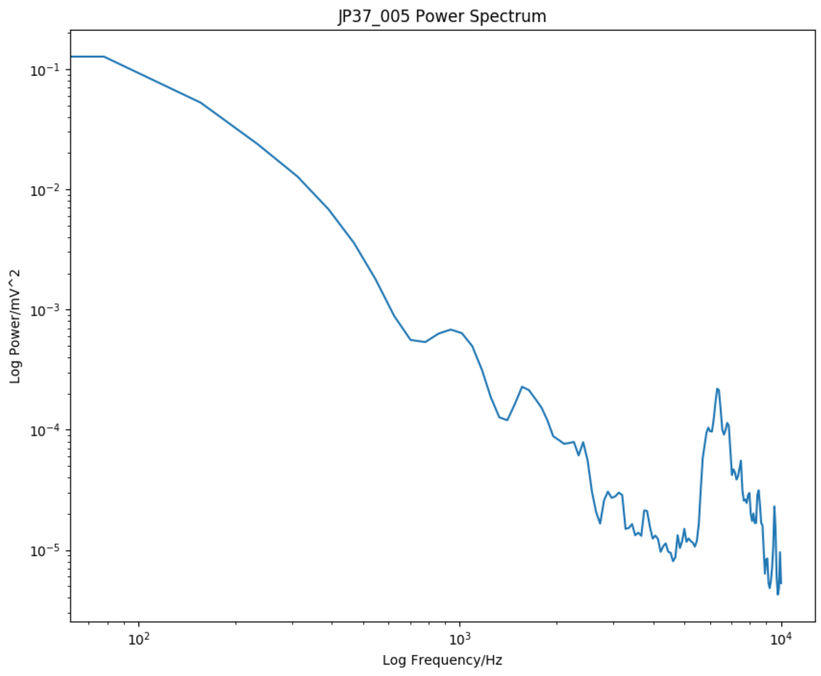

Figure 3.6: Power Spectrum of Figure 3.1

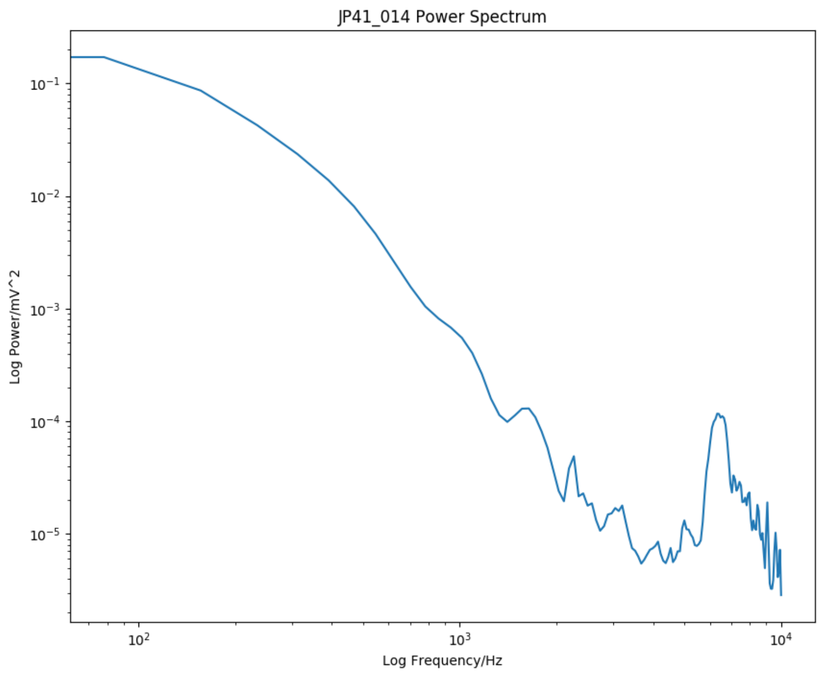

Figure 3.7: Power Spectrum of Figure 3.2

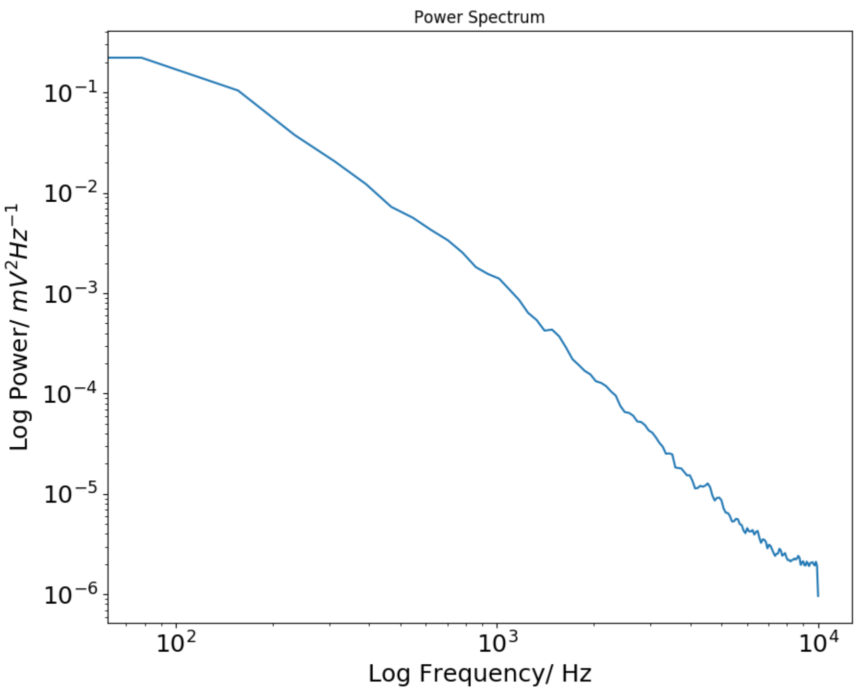

Figure 3.8: 1 Second Simulation Trace Power Spectrum

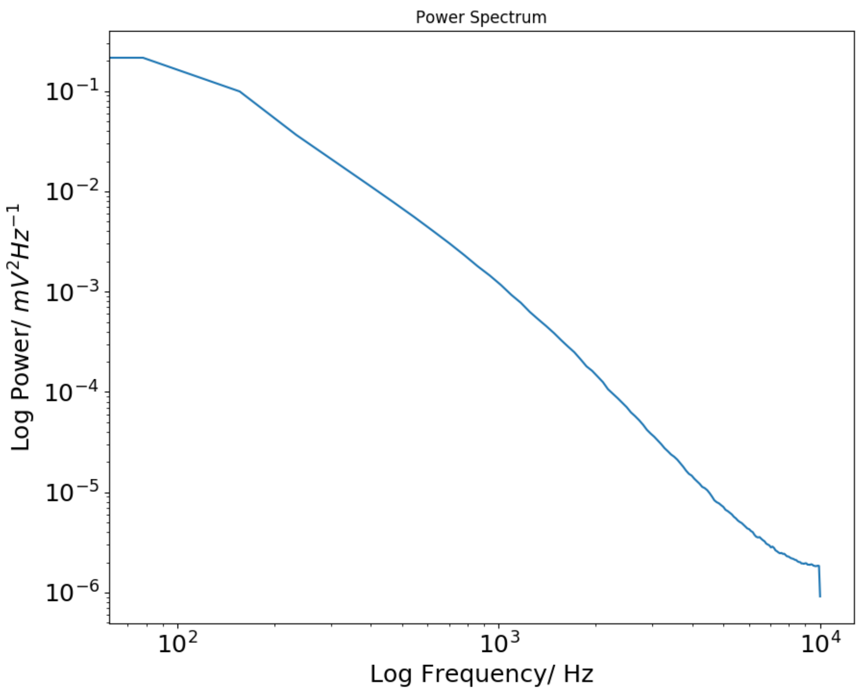

Figure 3.9: 30 Second Simulation Trace Power Spectrum

3.5 Optimisation Algorithm

3.5.1 Algorithm Choice

The problem space had a few features that would direct the choice of algorithm used to optimise the simulation fits to the target trace.

- Black box evaluation of error
- Expected a dimensionality in error space from 2 to 8 depending on how many parameters fitted
- Numerical, continuous optimisation
- The shape of the error space is unknown

While the original idea as part of this project was to use the equation for the power spectrum of the single neuron model derived by Puggioni[16], when this ended up not working the error evaluation had to be treated as a black box since it was dependent on a simulation run. This meant that the algorithm used would have to either be derivative-free or estimate the derivative as part of its operation. The algorithm would have to be able to cope with high dimensionality at the top end of parameters fitted, and while the highest simultaneously fitted in this project was eight parameters more variables used in the model could have been fitted - though the error function would likely have to be altered to better capture the changes due to altering those variables. While an estimate of the error space is shown in figure 4.12, it was infeasible to calculate the landscape of the error plane for any higher dimensionality than the two parameter model and was only calculated for a single recording.

Broadly the choice lay between two algorithms commonly used in this area of problems, the Nelder-Mead simplex method and the Covariance Matrix Adaptation Evolution Strategy. Ultimately the choice was made to use CMA-ES over Nelder-Mead since it scales well with higher dimensions[11] while Nelder-Mead reduces in effectiveness [9].

3.5.2 CMA-ES Implementation

To allow the CMA-ES algorithm to try and find an optimal parameter fitting to most correctly match the underlying neural properties three further requirements had to be met.

- An error equation given a target and test membrane potential trace
- Recode parameters into CMA-ES friendly format
- Model code optimisation to speed up fitting procedure

3.5.3 Error Equation

A combination of error terms (3.5) was used in the final implementation, these were simply summed to give a comprehensive error value. Error terms were selected in order to incentivise CMA-ES to accurately fit all parameters.

$$E_{total} = E_{\mathcal{P}} + E_{\overline{V}} + E_{\sigma_V} \tag{3.5}$$

To correctly fit the input firing rates (both excitatory and inhibitory) error between the target and test power spectra was used. As shown in (3.6) the mean squared error is used. The squared difference is taken for every frequency ω that the power spectrum is evaluated at. The log of the spectrum value is used as the magnitude of the power varies from around 10^0 down to 10^{-4} so if the log were not taken, differences in the power values for lower frequencies would dominate the error term.

$$E_{\mathcal{P}} = \frac{1}{N}\sum^{\omega}(\log \mathcal{P}_{target}\omega - \log \mathcal{P}_{test}\omega)^2 \tag{3.6}$$

The next error term, shown in (3.7), is the squared difference between the target and test mean membrane potential. This was calculated from the membrane potential trace. This term should not only encourage the correct balance in firing rates of excitation versus inhibition to maintain the target mean membrane potential, but also help tune the action potential amplitude distribution.

$$E_{\overline{V}} = (\overline{V_{target}} - \overline{V_{test}})^2 \tag{3.7}$$

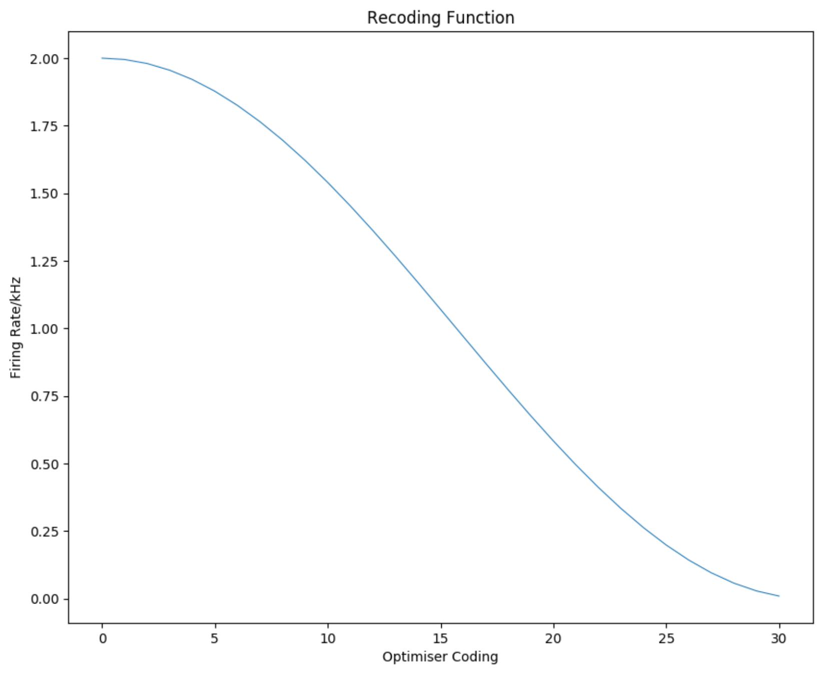

Figure 3.10: Optimiser decoding function

The final error term (3.8) is the squared difference of the target and test standard deviation of the membrane potential. This was calculated from the integral of the power spectrum as shown in (2.3), though the membrane potential trace could also have been used. This helps tune the action potential amplitude distribution, since larger variance in the amplitudes will result in a greater variance in the overall membrane potential trace.

$$E_{\sigma_V} = (\sigma_{V_{target}} - \sigma_{V_{true}})^2 \tag{3.8}$$

3.5.4 Recoding

In order to get the best performance from the CMA-ES algorithm, it is recommended to try and get each parameter to be equally scalable and typically have magnitude of around 10^1[11][10]. To achieve this a wrapper function is established in the code in order to translate between the CMA-ES representation and the real-world representation. Equation (3.9) gives a mapping where for any value of x, $a \leqslant y \leqslant b$. This is especially useful as this means we can force CMA-ES to stay strictly positive within the real-world domain if required. Boundary values were selected for each set of parameters, with no discrimination between excitatory and inhibitory parameters.

Figure 3.10 shows this function for the input range [0..30] and shows the complete output range. As the mapping is periodic, this wave simply repeats ensuring that output stays in the range [0..2]kHz regardless of what values CMA-ES is exploring.

A simpler to interpret version in (3.10) provides the same concept of equal scalability between parameters but does not provide strict positiveness. Experimentation was done to see if there was a significant difference between these two approaches, and this simpler version consistently ran into issues with CMA-ES trying to explore negative values for strictly positive quantities, so the tradeoff was made to have a less clear encoding in exchange for far superior fits.

$$y = a + (b-a)\frac{(1 - \cos(\pi - \frac{x}{10}))}{2} \tag{3.9}$$

$$y = a + (b-a)\frac{x}{10} \tag{3.10}$$

3.5.5 Code Optimisation

During the development of the simulation an effort was made to keep code as interpretable at a glance as possible with a view to handing to code over to experimenters who may have limited experience with coding or Python. This included using a hierarchical object structure to encapsulate separate functions and components of the simulation e.g each Poisson input process was a separate object controlled by a neuron object. However upon reaching the stage where CMA-ES would be running a simulation K times per iteration (around 6-10), with anywhere up to 1000 iterations per overall fit, this encapsulation was abandoned to reduce simulation initialisation time. Instead of constructing N Poisson process objects, two N element arrays were used - one tracking synaptic strength, one tracking next action potential time.

In addition to serial code optimisations, the inbuilt *multiprocessing* module was used so that a pool of K threads would run the simulation in parallel for each CMA-ES iteration.

Optimisation runs were split between running on BlueCrystal Phase 3 (specifications found here[1]) and a desktop computer with an AMD Ryzen 5 6-core CPU. Since CMA-ES iterations are not independent, code can only be parallelised onto at most as many threads as candidate evaluations per iteration (default value in *pycma* given by (2.4)) with one additional thread for managing the update of the distribution. The two and six parameter models were typically run on the desktop computer, while the eight parameter model required BlueCrystal to provide sufficient concurrent threads.

3.5.6 *pycma* Implementation

The *pycma* module has two base classes, *cma* that will just optimise whatever is given to it and *CMAEvolutionStrategy* that is more interactive. The code fragment below gives a good sample invocation and provides a good opportunity to explain how the algorithm functions.

```
import cma as es
from cma.fitness_transformations import EvalParallel
def cma_mp_optimize(args, x0):
    sigma = 2.0
    options = {'verb_filenameprefix':'outcmaes/sixparam/', 'maxiter':1000}
    print("begin␣optimisation")
    evol = es.CMAEvolutionStrategy(x0, sigma, options)
    with EvalParallel(evol.popsize+1) as eval_all:
        while not evol.stop():
            X = evol.ask()
            evol.tell(X, eval_all(ob.sim_eval, X, args=args))
            evol.disp()
            evol.logger.add()
        evol.result_pretty()
    evol.plot()
    return evol.result
```

The function takes two input lists, *x0* is a *numpy* array of length equal to the number of parameters to be optimised and is the initial guess as to where the minima lies, and *args* is just a tuple of arguments required to run an error evaluation, for example the target power spectra. Then a dictionary of options is created, this allows the hyperparameters to be altered like in this case the number of maximum iterations before the strategy stops. It should be noted that *pycma* supports a wide variety of termination conditions with almost every successful run in this project finishing due to a stagnant iterations parameter and the cap of 1000 max iterations was never reached. We also set an initial value of sigma for the sampling distribution. An opportunity exists in tuning this distribution via *x0* and σ over multiple runs in order to sacrifice wide consideration of the error space, if the approximate location of the minima is known, in exchange for fewer iterations required to hit the optimum.

EvalParallel sets up a *multiprocessing* pool so that when we *ask* for candidate solutions we can then evaluate all concurrently before *tell*ing the strategy about the error values of each so it can update the sampling distribution.

3.6 Fitting

3.6.1 Fitting Synthetic Trace

To gauge how well CMA-ES could handle the dimensionality of the problem and whether it could get accurate fits, it was first run on membrane potential traces generated by the simulation. These traces were 1s in total length with a sampling rate of 20000Hz.

3.6.2 Fitting Real-World Trace

To go from fitting a synthetic trace to fitting a prerecorded trace a decision had to be made about how much of that trace to use. Since all of the error terms in (3.5) are calculated based on the estimated power spectrum rather than the membrane potential trace, the difference in recording time could be ignored since the power spectrum should theoretically be invariant as it exists in the frequency domain.

The observation was made that the majority of anomalous behaviour occurred near the end of recording, likely due to the recorded cell dying, as such the trace was sliced to remove the first 5% and the final %20. Even in the case of the shortest trace this still left around 135s of recording, so this should not greatly impact the fit due to lack of data.

3.6.3 Parameter Models

In this project, three selections of parameters were fitted onto the real-world data. These consisted of two, six, and eight parameters and shall be referred to by the number of parameters they fitted.

1. The simplest model had two parameters and simply aimed to fit λ_e and λ_i, the excitatory and inhibitory firing rates.
2. The six parameter model builds upon the two parameter model and adds (μ_x, σ_x), the parameters of the synaptic strength distribution, for both excitatory and inhibitory inputs.
3. The final model uses all parameters from the six parameter model with the addition of τ_e and τ_i, the decay time constants for the conductance of the excitatory and inhibitory inputs.

As each model has a higher number of parameters they also each have a higher dimensionality in the error space, making it harder to navigate to try and find minima. Because of this, the higher parameter count indicates a tradeoff between the number of obtained results and the computational complexity of fitting in a higher parameter space. The ability of CMA-ES to generalise well to higher dimensions was key in allowing the higher parameter models to find fits in reasonable time.

Chapter 4

Results and Discussion

4.1 Results

4.1.1 shows the distribution of final error values across all parameter models for synthetic and real traces with values to three significant figures. As can be seen in 4.13, typically this distribution had skew towards a couple of much higher values. New target synthetic traces were generated for each optimisation run.

The simulation was able to be far better fitted than the real-world traces could be, potentially this was due to noise in the real-world traces or perhaps the model needs to be adjusted to better fit the data seen in the real-world traces. Fewer tests were run for higher parameter number models as although the run-time per iteration did not change greatly, the number of iterations before stagnation was reached increased causing the optimisation to take longer.

Values for mean error and standard deviation of error were calculated first for all recordings equally, those recordings with a mean error of higher than the mean plus three standard deviations were then discarded as insufficiently fitted. Values in 4.1.1 are those after the far outliers had been removed. Taking the error values for these anomalous fits separately into the three components mentioned in (3.5), anywhere from 95-99% of the total error came from the E_{σ_V} term. One of the three recordings removed from results for high error is shown in 4.1, with the final fitted power spectrum shown in figure 4.2. This recording had a mean error value for the two parameter model of 194 across four trials. Compared to the fitted power spectra shown in 4.16, 4.21, and 4.28 we can see that this spectrum underfits the low frequency band dramatically. This seems to be caused by the original high estimates of the power of the low frequency bands, likely due to the comparatively high frequency of action potentials in the trace for this recording.

Table 4.1.1 gives the distribution of each parameter in each model for the trace shown in figure 3.1. Values in brackets being those used when not fitting those specific parameters.

While values are not consistent between the different parameter models, they are of comparable magnitude. Interestingly in the six and eight parameter models, σ_e is up against the bounds imposed by the recoding function. It is possible that this is because the distribution for the excitatory synaptic strength should be higher, but it could also be that since some parameters have overlapping effect on the power spectrum that these effects cannot be separated and that the synaptic strength is being increased in place of the excitatory firing rate or conductance decay time constant.

Unfortunately due to the nature of the problem we are attempting to solve, we do not have the true parameters of the inputs in this recording and so can only say that optimised parameters are consistent, not that they are necessarily correct - though they are at least plausible since they lie within the bounds obtained from NeuroElectro[5] for the cell type.

4.1.1 Variance in Error Evaluation

Since both the model and CMA-ES have some level of inherent variance due to their stochastic nature - Poisson processes in the model and candidate sampling in CMA-ES - when a trace is run through CMA-ES multiple times it will give a range of final error values regardless of whether it is finding the same global minimum each time. The variance due to CMA-ES was examined by running the same trace multiple times and examining the spread of results.

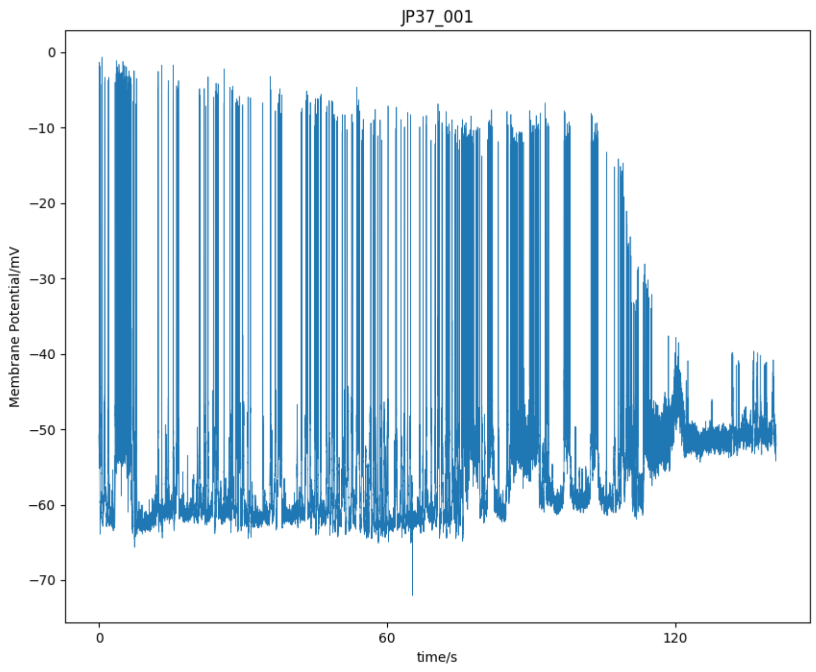

Figure 4.1: Trace removed from results for anomalously high error evaluation

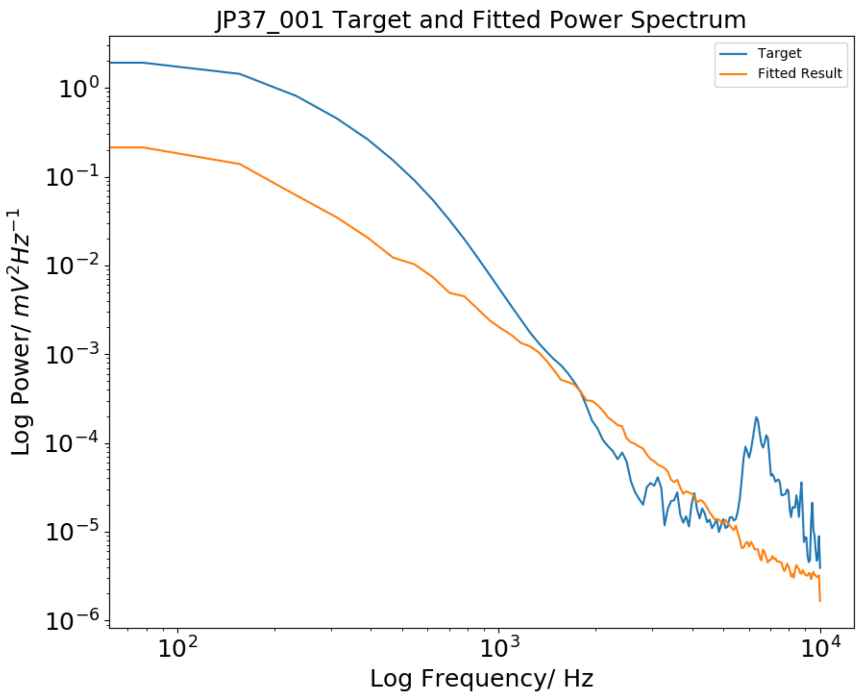

Figure 4.2: Fitted power spectrum and target power spectrum for trace shown in 4.1

	Simulation Results			Real Traces Results		
Number of Parameters	Tests run	Mean Error	Standard Deviation	Tests run	Mean Error	Standard Deviation
Two	38	0.0574	0.0881	86	7.90	8.58
Six	23	0.0208	0.0557	72	3.34	1.56
Eight				30	3.20	2.33

Table 4.1: Distribution of final error for synthetic and all real traces

	Two Parameter Model		Six Parameter Model		Eight Parameter Model	
Parameter Name	Mean Value	Standard Deviation	Mean Value	Standard Deviation	Mean Value	Standard Deviation
λ_e (kHz)	0.0509	0.00370	0.0288	0.00329	0.0933	0.0228
λ_i (kHz)	0.362	0.300	0.766	0.510	0.849	0.782
μ_e (μS)	(0.00075)		0.00113	0.000226	0.00115	0.000245
μ_i (μS)	(0.00075)		0.000726	0.000315	0.000673	0.000330
σ_e (μS)	(0.00075)		0.00124	0.00000687	0.00122	0.0000795
σ_i (μS)	(0.00075)		0.000862	0.000381	0.000768	0.000333
τ_e (ms)	(3)		(3)		1.75	0.719
τ_i (ms)	(10)		(10)		7.32	5.94

Table 4.2: table

Distribution of final parameter results for real-world trace $JP37_005$

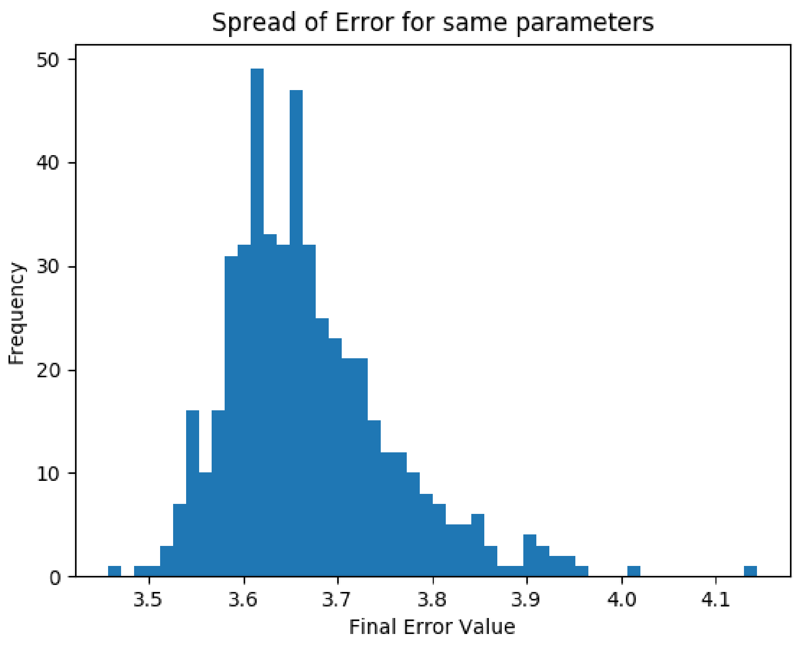

Figure 4.3: Histogram of error evaluation for the same input parameters

The variance due to using a stochastic model was examined by running CMA-ES on a trace, taking the final optimised firing rates and then performing 500 additional evaluations of the error at those parameters. The distribution of these evaluations is shown in figure 4.3, the best error given by the optimiser was 3.42 (3s.f) and single occurrences of that amount of error can be seen in the histogram, which has a mean error value of 3.67 and a standard deviation of 0.0887. Clearly it is important that regardless of optimisation algorithm choice, many evaluations would have to be performed to locate a parameter setting that could most consistently produce low error values.

4.1.2 Simulation Data

4.1.3 Parameter Effects

Figure 3.4 shows a 1 second synthetic trace for the parameter setting listed in section 3.3.6. In order to see the approximate effects on the trace, simulations were carried out with the individual parameters manipulated in optimisation doubled. Figure 4.4 shows the effect of doubling the excitatory firing rate, the mean membrane potential is increased and has a reduced range of approximately $3mV$ compared to figure 3.4. Figure 4.5 conversely shows the effect of doubling the inhibitory firing rate, the mean membrane potential is slightly reduced but the spread is comparable. Figure 4.6 is the result of doubling both parameters in the synaptic strength distribution for excitatory inputs, while figure 4.7 is the same for inhibitory inputs. It can be easily seen that these are similar to effects of doubling the excitatory and inhibitory firing rates respectively, indicating that results for optimising both firing rate and synaptic strength simultaneously may be non-unique as both provide similar effect.

Figures 4.8 and 4.9 show the effect of doubling the conductance decay time constants for excitatory and inhibitory inputs respectively, with 4.8 showing a greater effect on mean membrane potential than increasing firing rate or synaptic strength distribution and 4.9 showing a minor reduction in mean membrane potential compared to the baseline setting in 3.4.

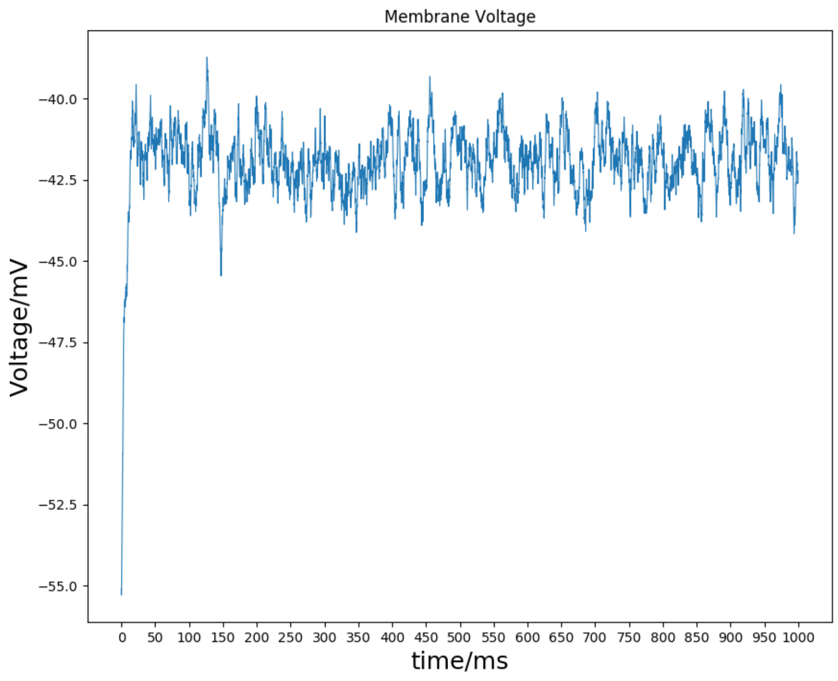

Figure 4.4: Synthetic trace for doubled λ_e

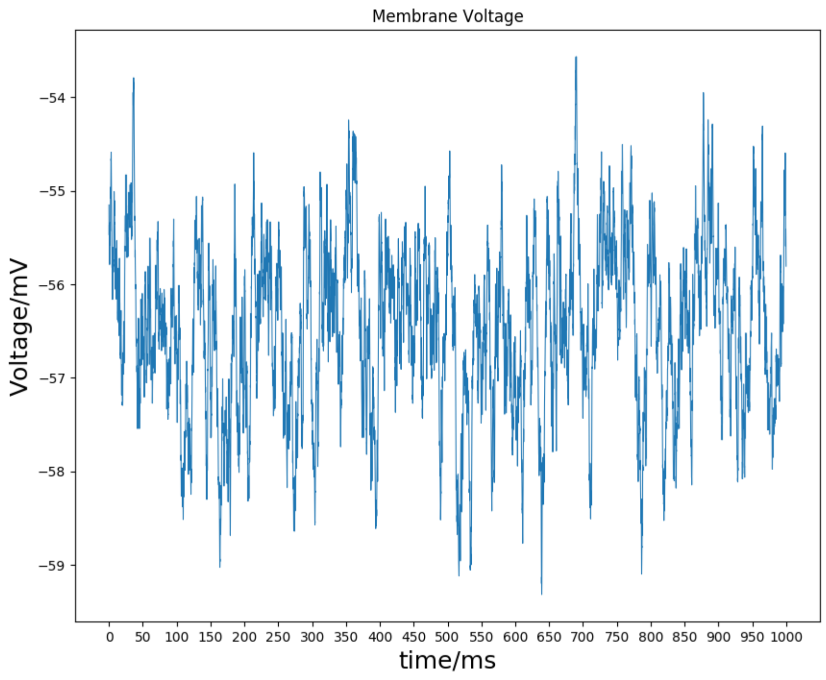

Figure 4.5: Synthetic trace for doubled λ_i

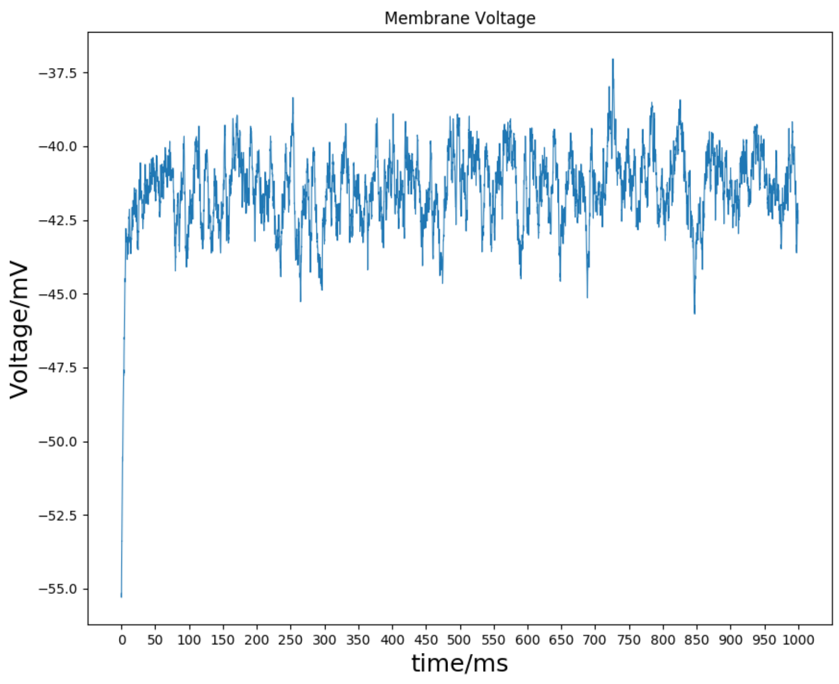

Figure 4.6: Synthetic trace for doubled (μ_e, σ_e)

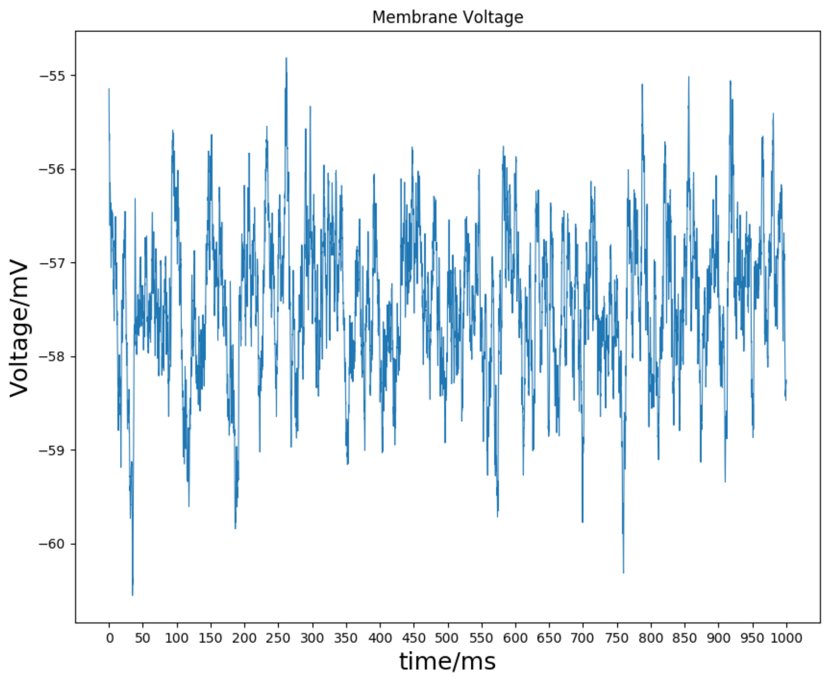

Figure 4.7: Synthetic trace for doubled (μ_i, σ_i)

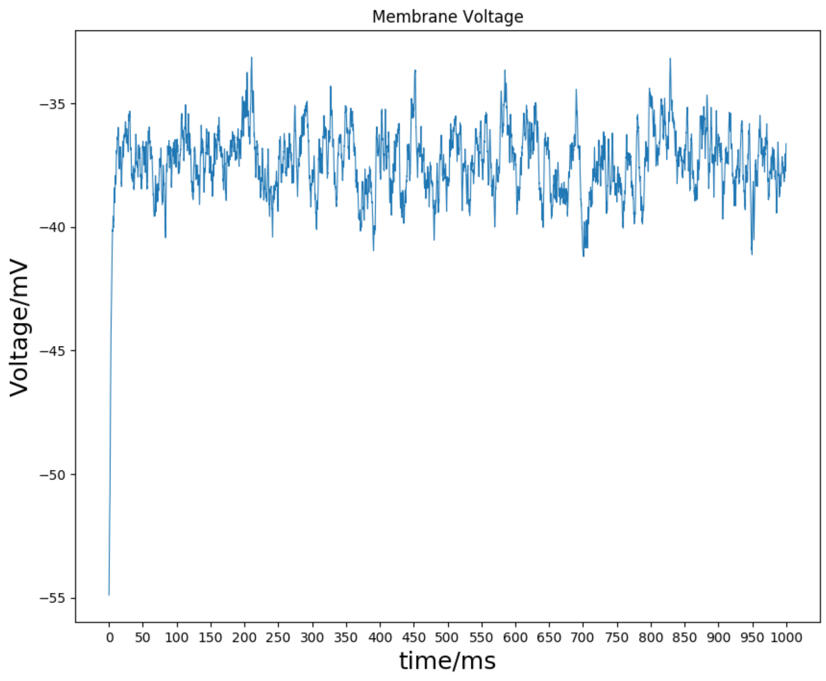

Figure 4.8: Synthetic trace for doubled τ_e

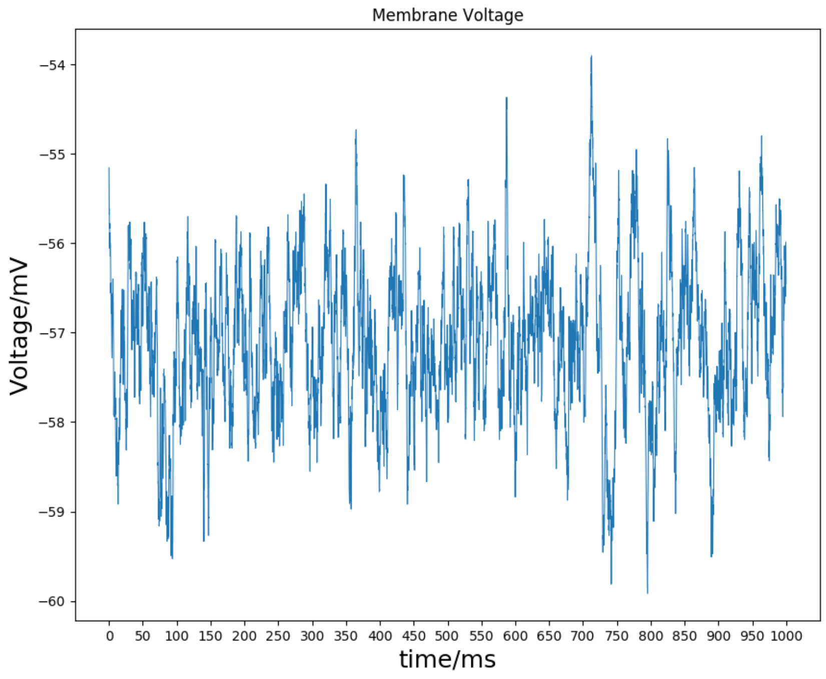

Figure 4.9: Synthetic trace for doubled τ_i

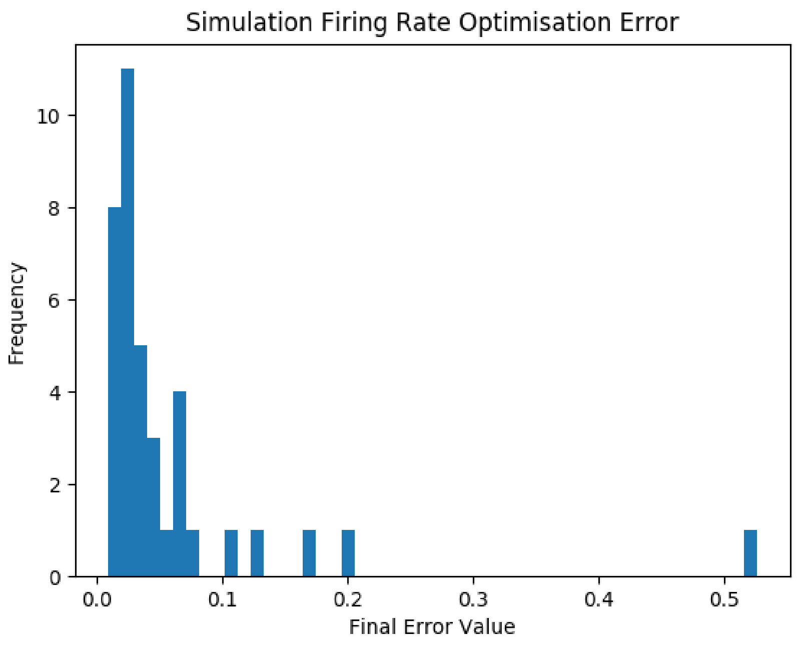

Figure 4.10: Histogram of final error for two parameter fit on synthetic trace

Two Parameter Model

4.10 shows the final optimised error value when run on a simulation with parameters described in section 3.3.6, the histogram was generated by separating data into 50 equal width bins. 38 runs were carried out.

Six Parameter Model

4.11 shows the final optimised error value for six parameter model (firing rates and synaptic strength distribution) over 23 runs. Traces were independent between the two parameter tests and the six parameter tests so the coincidence of a single high error outlier is particularly interesting.

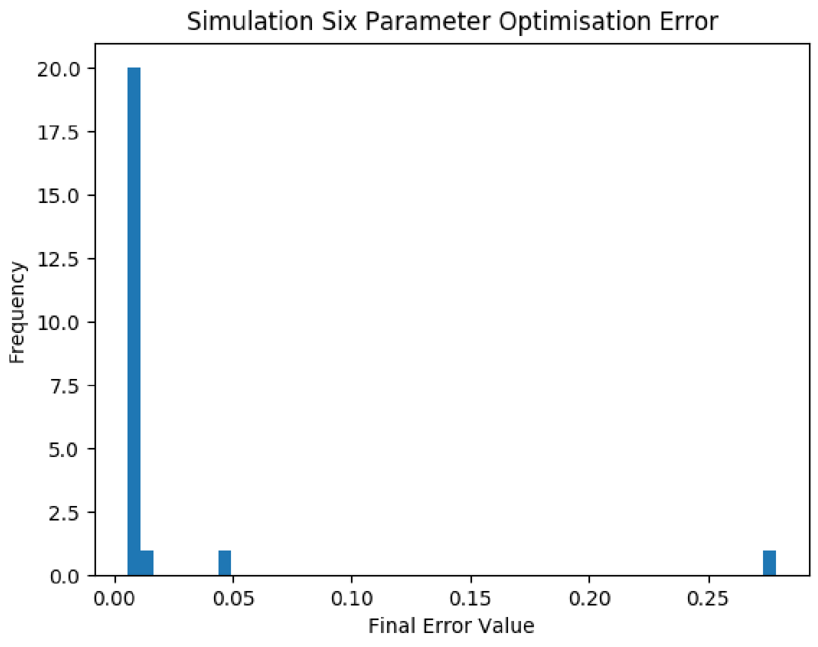

Figure 4.11: Histogram of final error for six parameter fit on synthetic trace

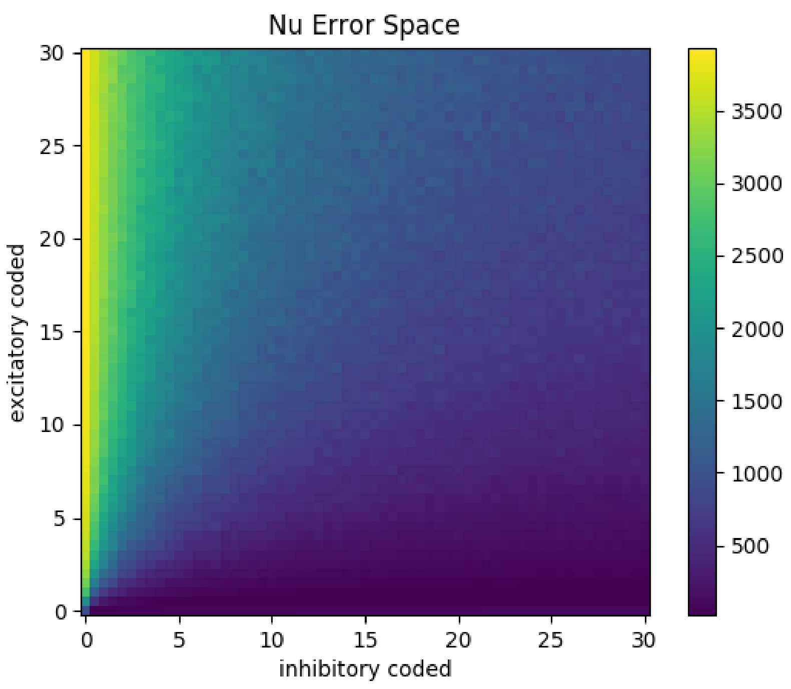

Figure 4.12: Error plane for two parameter fit for trace shown in 3.1

4.1.4 Real-World Data

Two Parameter Model

In order to check that CMA-ES really was optimising to the approximate global minimum, an approximation of the error plane within the constraints was calculated for the two parameter model on the trace shown in figure 3.1. This was done by calculating the error at each parameter setting for paired firing rates in the range [0..30] at steps of 0.5 within the CMA-ES encoding space, recall that the encoding at place here is decoded by the function (3.9) and that this function is periodic. This was only computationally reasonable for the two parameter model.

Figure 4.13 shows the distribution of final error values for those traces not removed for having mean error values three standard deviations away from the mean for all traces. Note that each trace typically had a consistent optimised error value with occasional outlying high values, see figure 4.14 for example.

Not only was the error typically consistent, but the actual parameter values were as well. An additional 40 runs were carried out on the trace shown in 3.1, figure 4.15 shows the final values for the two firing rates in each of the 40 runs. There is clear clustering around the mean $(0.051, 0.36)$ with three outliers. It is important to remember that these are firing rates for the input synapses and are distinct from the frequency of action potentials received by the soma of the recorded cell though related by $N_x\lambda_x$.

Figure 4.16 shows the final optimised power spectrum overlaid onto the original target power spectrum of the trace, it fits closely at lower frequencies but then doesn't manage to accurately fit the higher frequency behaviour. This higher frequency behaviour was expected to be magnetic or electrical noise from the recording setup so this result is not surprising. It is possible that some of the perturbations in the target power spectrum at frequencies of 10^2 or 10^3 are due to correlations in the real-world input that a simulation using uncorrelated Poisson processes as input simply cannot recreate, an adjustment to the simulation could allow these to be fitted.

Figure 4.17 shows the best ever error evaluation found by CMA-ES plotted against iteration count. The initial iteration steps from a randomly selected initial *x0* led to rapidly decreasing error before a plateau with only small improvements over the rest of the runs. Typically the best error evaluation was found at around 200 iterations in the two parameter tests though this is variable due to the stochastic nature of the algorithm.

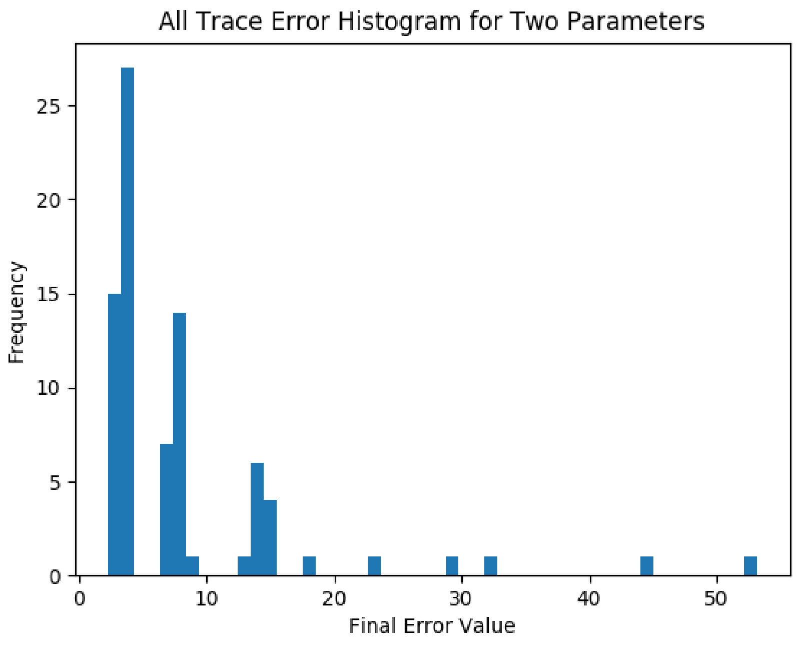

Figure 4.13: Histogram of final error for two parameter fit

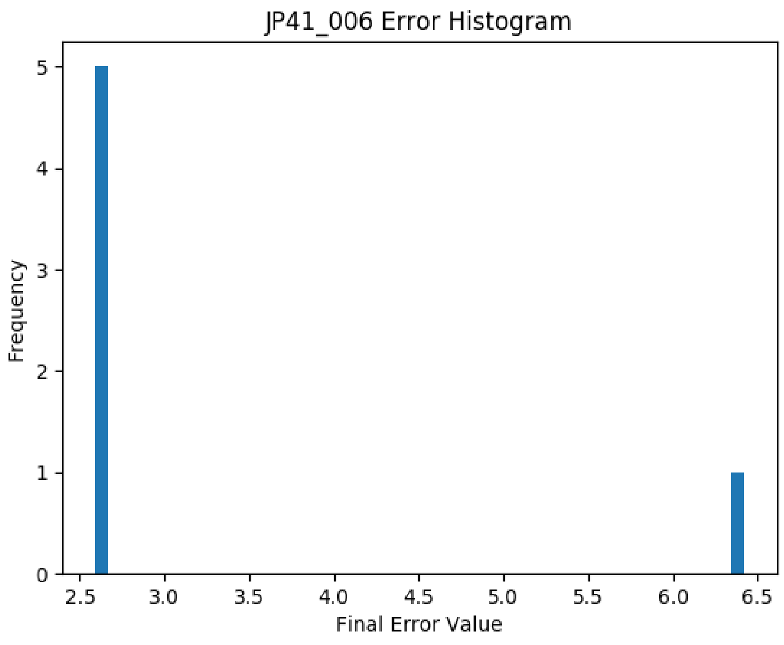

Figure 4.14: Histogram of final error for two parameter fit for individual trace

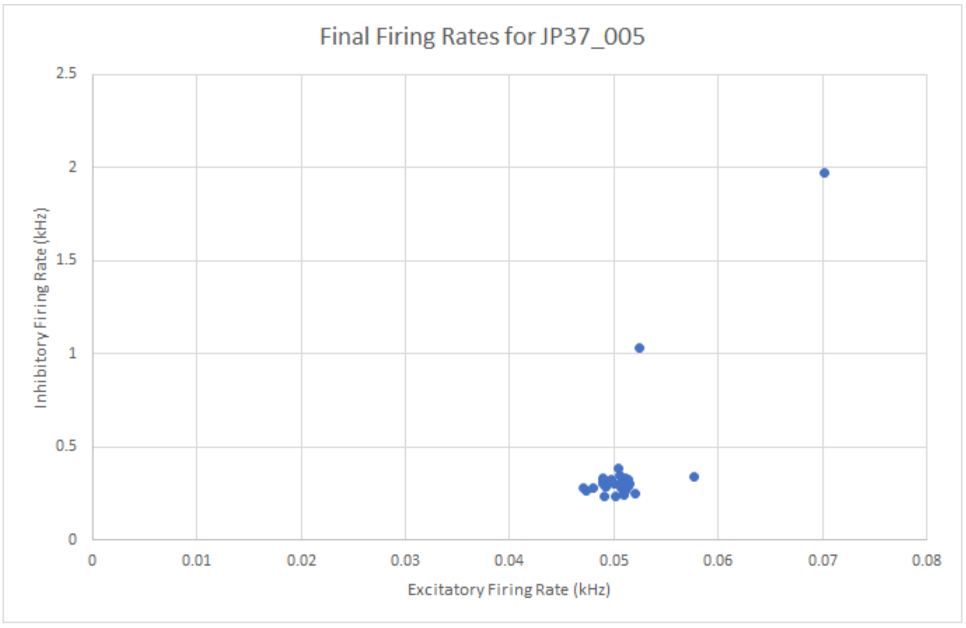

Figure 4.15: Spread of optimised parameter values for individual trace

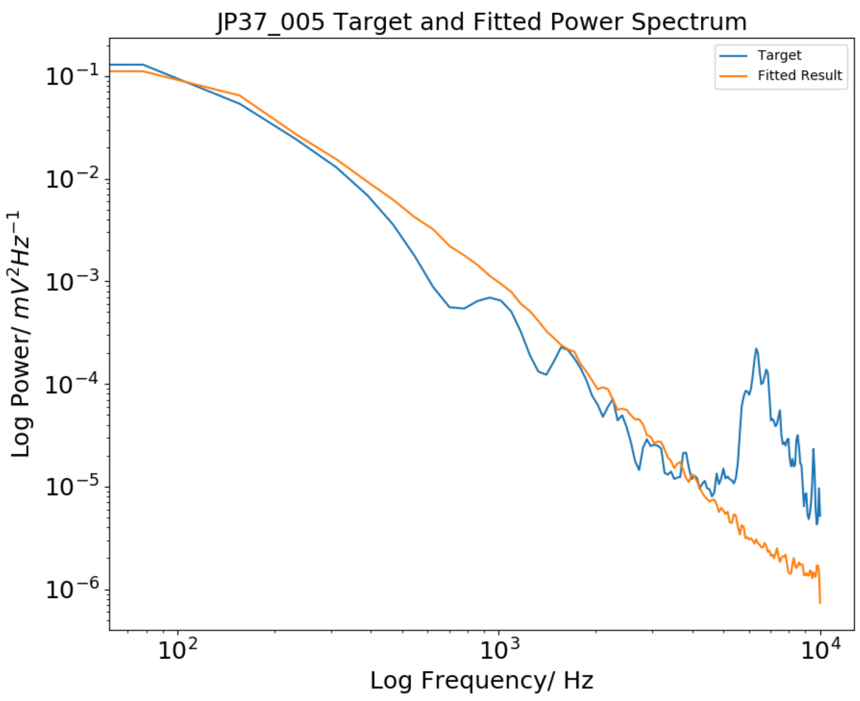

Figure 4.16: Optimised power spectrum with target power spectrum for two parameter model

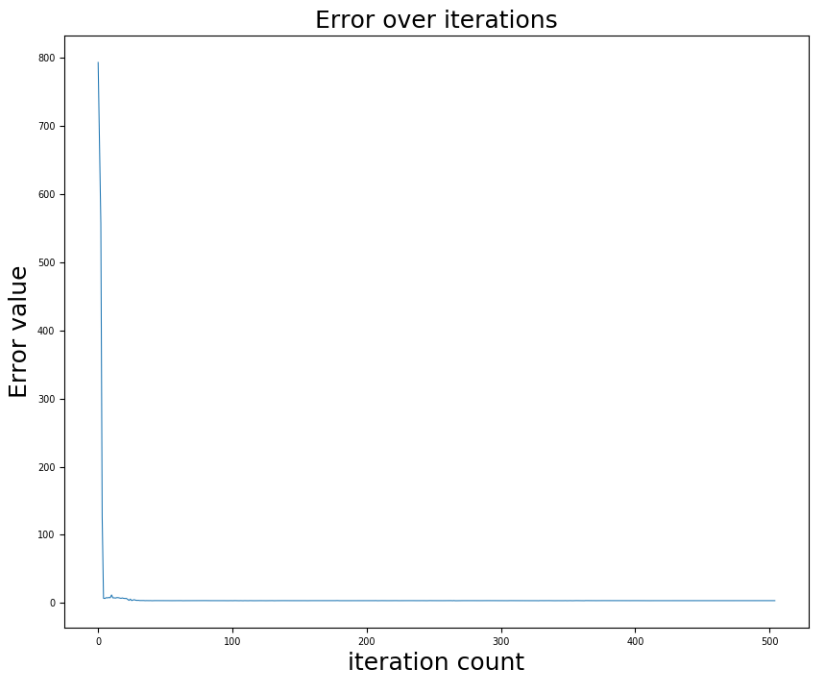

Figure 4.17: Best error against iteration count for two parameter model

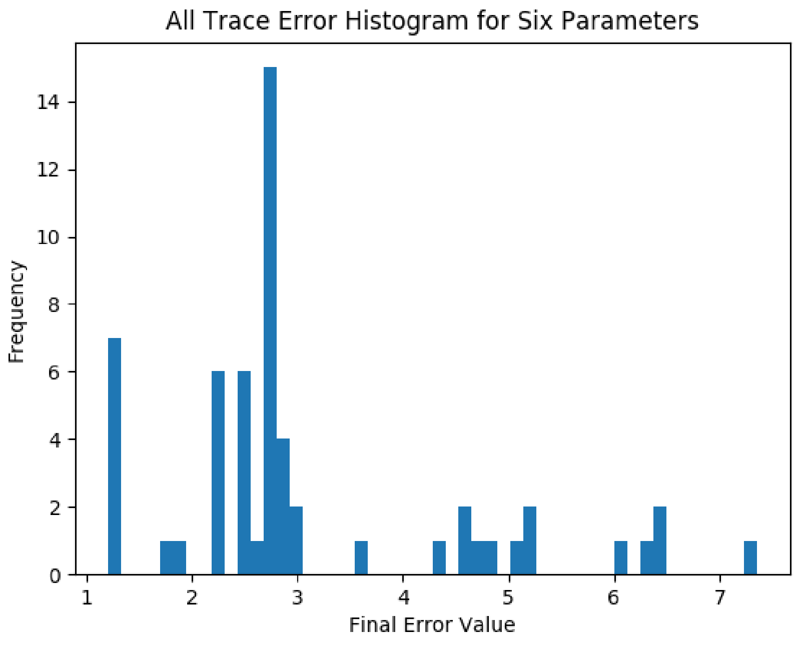

Figure 4.18: Histogram of final error for six parameter fit

Six Parameter Model

Figure 4.18 shows the spread of final error values after those traces with anomalously high means were removed. It can be seen that both the mean error value and the spread of errors is lower than that of the two parameter model, meaning that CMA-ES does indeed cope well with the increased dimensionality. Similarly to the two parameter model, most recordings had a cluster of low error evaluations with a few high evaluations, see for example figure 4.19.

In figure 4.20 clustering of the excitatory/inhibitory firing rates can be seen, while the cluster sits around half the excitatory firing rate seen in 4.15, the results in 4.1.1 show that the excitatory synaptic strength distribution has been amplified so overall it likely has the same result upon the conductance, just with fewer high strength action potentials.

An optimised power spectrum is shown in figure 4.21 and fits slightly worse for low frequency values but slightly better for higher values. This is not necessarily desired behaviour since it fitting the possible noise section better rather than the section we care about. This is partially due to the fact that the power spectrum estimation method used was given a equal width range of frequencies to evaluate at, so when we care about the log of the power spectrum to cancel out magnitude differences there are more points between 10^3 and 10^4 than between 10^2 and 10^3 so the algorithm is more incentivised to fit the higher frequency range better.

Figure 4.22 shows that fitting followed a similar pace to that of the two parameter model, though perhaps due to the increased number of parameters the spread of final iteration counts were more varied with the best ever error evaluation typically found around iteration 300-350.

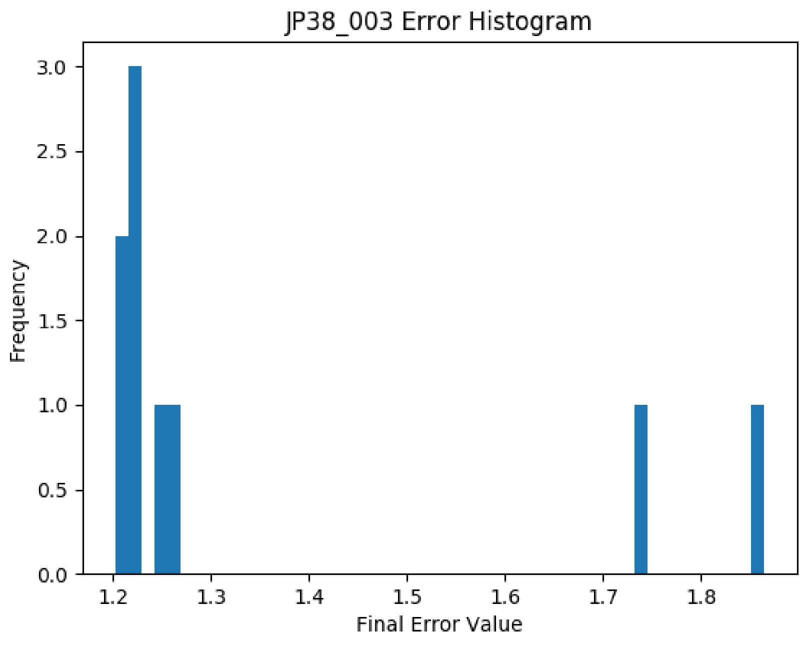

Figure 4.19: Histogram of final error for six parameter fit for individual trace

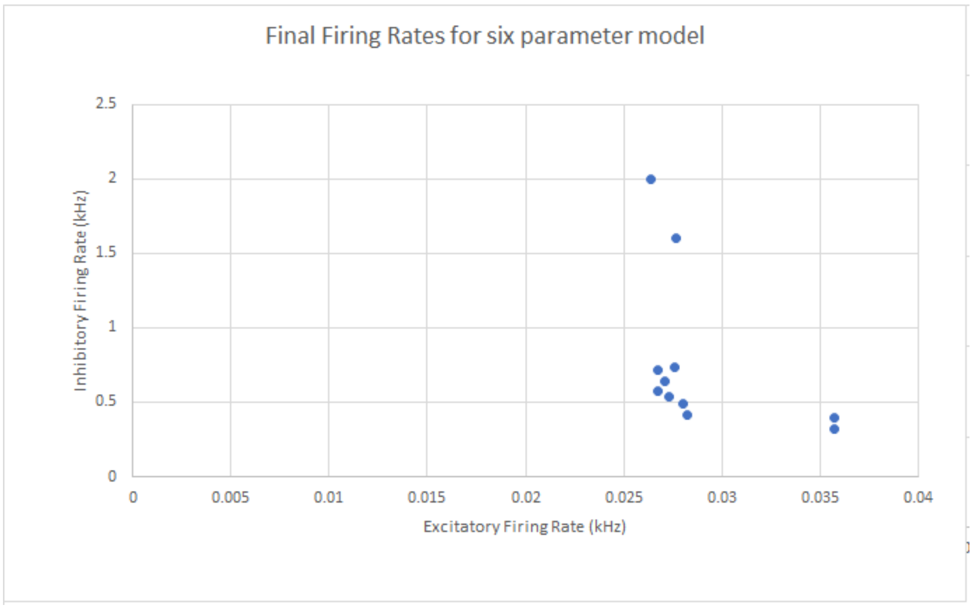

Figure 4.20: Spread of firing rates in six parameter model for individual trace

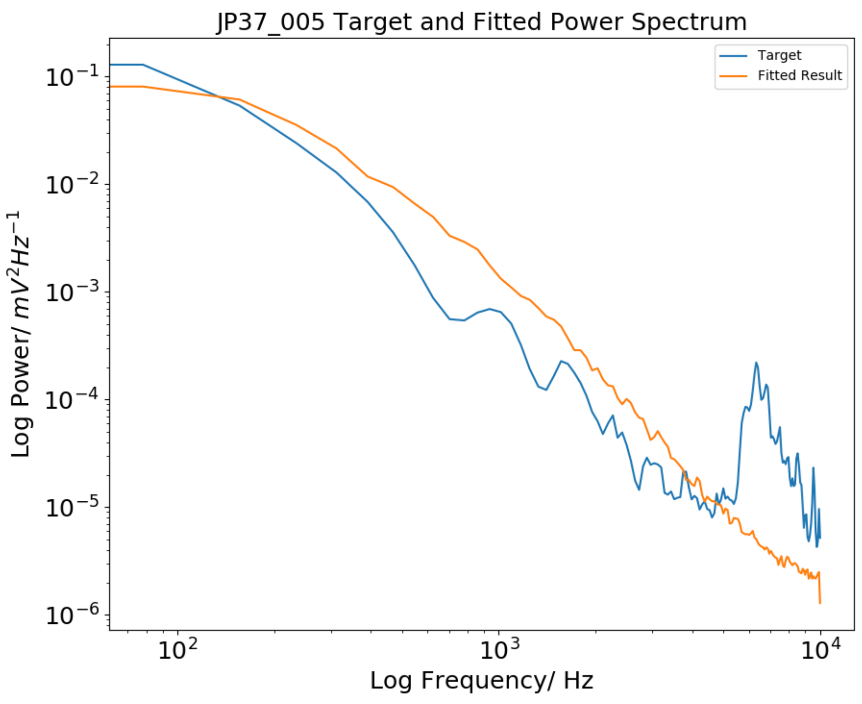

Figure 4.21: Optimised power spectrum with target power spectrum for six parameter model

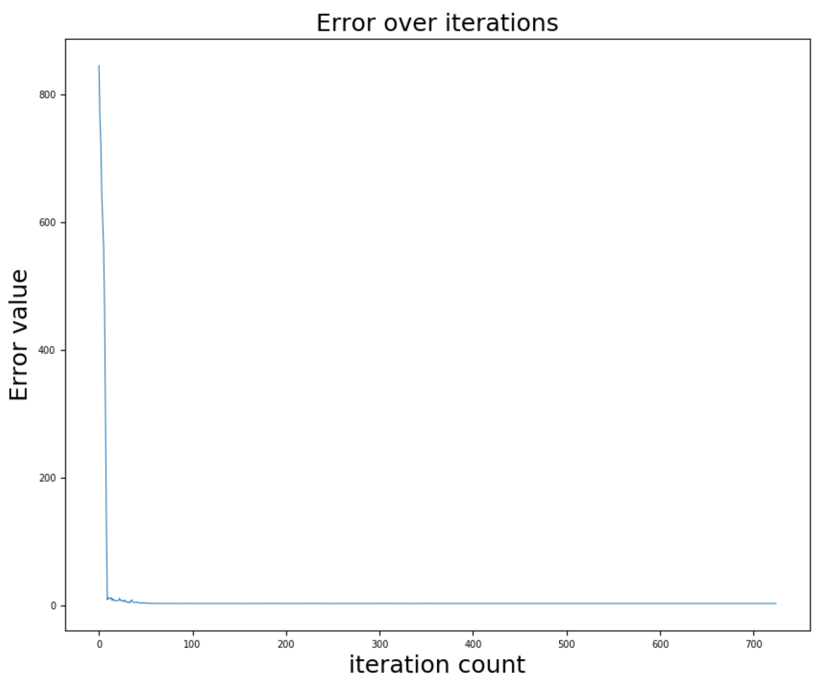

Figure 4.22: Best error against iteration count for six parameter model

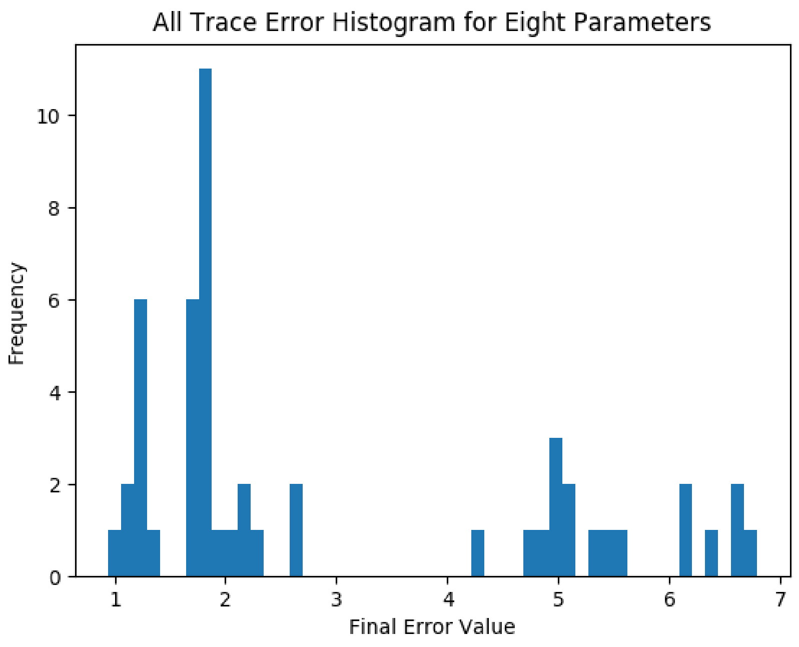

Figure 4.23: Histogram of final error for eight parameter fit

Eight Parameter Model

The distribution of final error values is shown in 4.23. The eight parameter model offers a slight improvement in mean error over the six parameter model it is not as great as the jump between the two and six parameter model. While this histogram does not show them, the recordings removed for the other two models had error minimums across multiple runs that fell within the mean plus three standard deviation cut off unlike for the other two models where the traces were optimised consistently badly. Not as many optimisation runs could be done as desired as the eight parameter model typically required more iterations to find an error minimum and BlueCrystal (where the runs where carried out) performed much slower per iteration, resulting in runs that were much longer overall.

Much like the other two models, traces had a consistent mean minimum error with a handful of outliers, as shown in figure 4.24. Spread of excitatory/inhibitory firing rates was far greater than in the two or six parameter models, as seen in 4.25. As seen in 4.1.1 the firing rates are quite a bit higher than the results for the other models, however with smaller conductance decay constants leading to more brief action potential effect on the conductance of input and comparable synaptic strength distributions to those of the six parameter model this is not an outlandish result.

When the firing rate is plotted against the conductance decay constant for excitatory and inhibitory inputs in figure 4.26, the excitatory rates and time constants are fairly clustered but the inhibitory rates and conductances can strangely be separated into two groups: high firing rate with low time constant, and high time constant with low firing rates. This is strange as these two groups should have completely different impact on the conductance of the inhibitory input, with the former having a large effect and the latter doing comparatively little to the conductance. This could be explained by these two groups having different scales of synaptic strength distribution but figure 4.27 shows that there this is not the case as the positive correlation between mean synaptic strength and conductance decay constant simply does not exist.

The eight parameter model also fitted at a pace similar to the previous two models, as shown in figure 4.29. Typically the eight parameter model runs took more iterations to complete but this was highly variable.

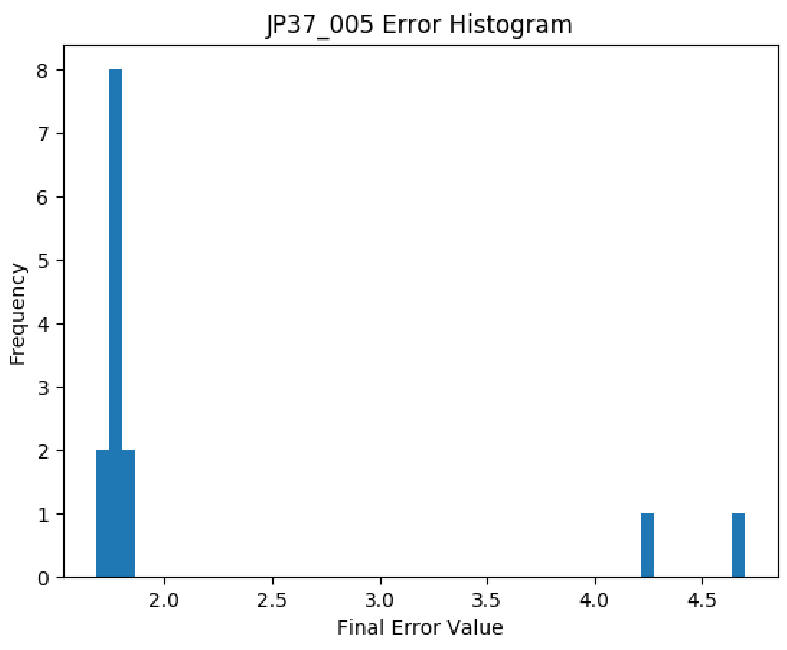

Figure 4.24: Histogram of final error for eight parameter fit for individual trace

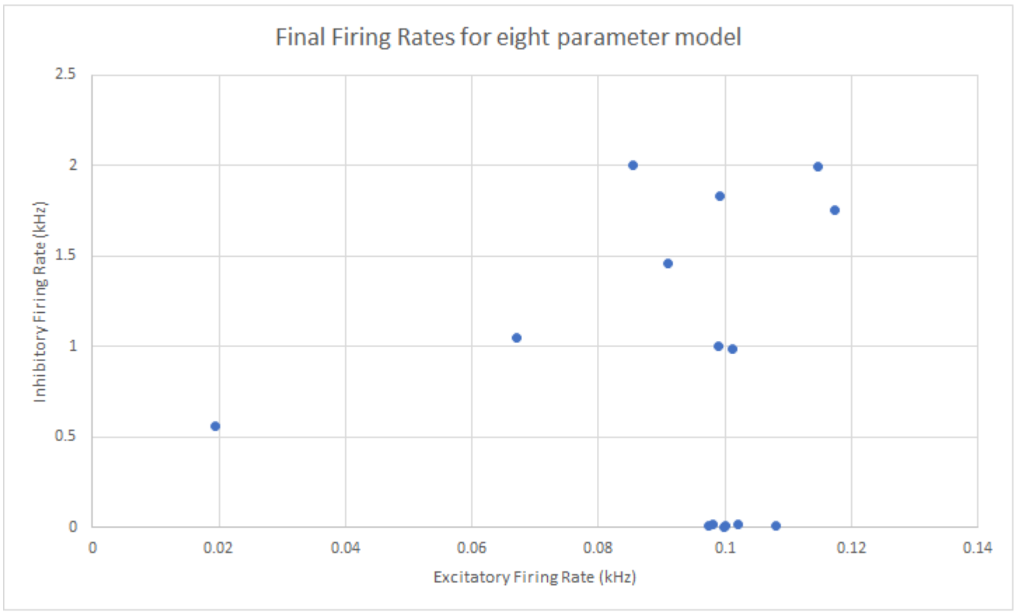

Figure 4.25: Spread of firing results in eight parameter model for individual trace

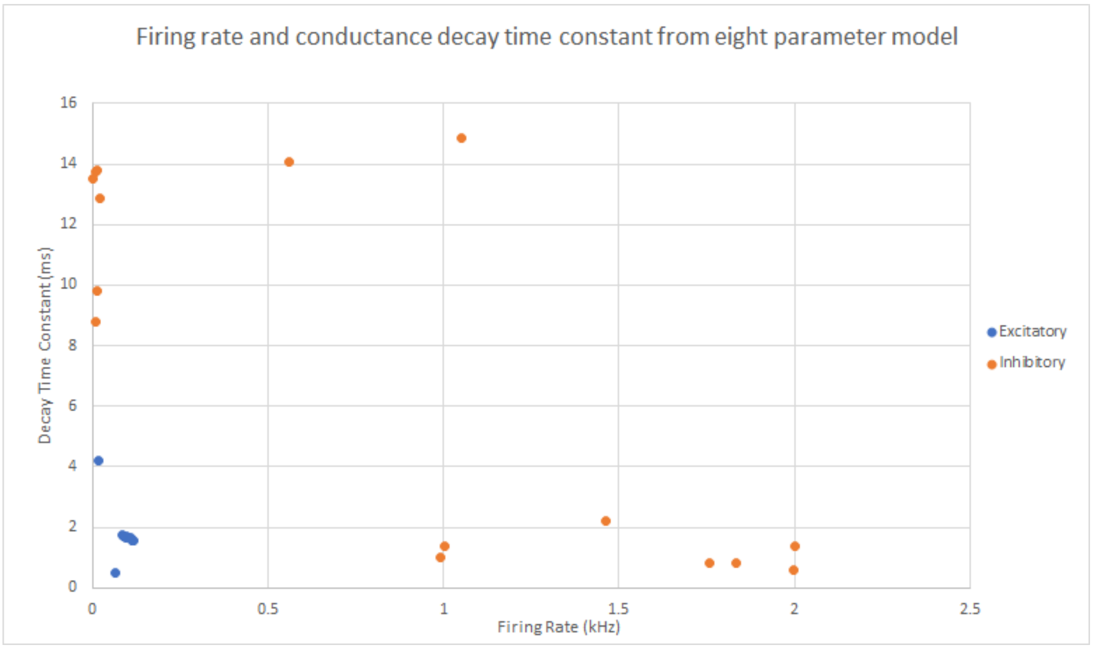

Figure 4.26: Firing rates against conductance decay time constant for eight parameter individual trace

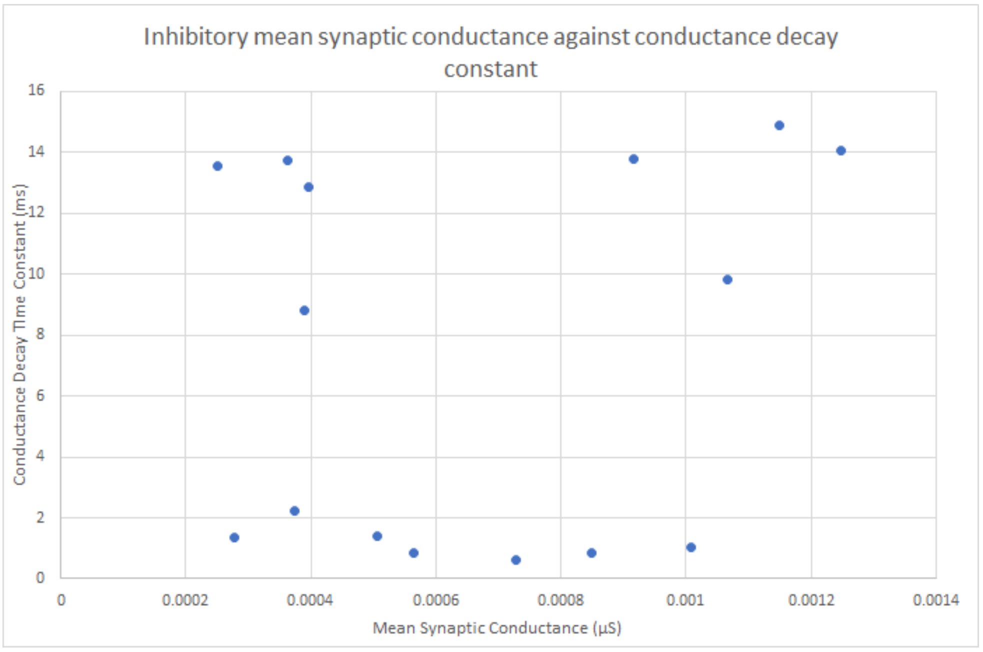

Figure 4.27: Mean synaptic conductance strength μ_i against conductance decay time constant τ_i for inhibitory input

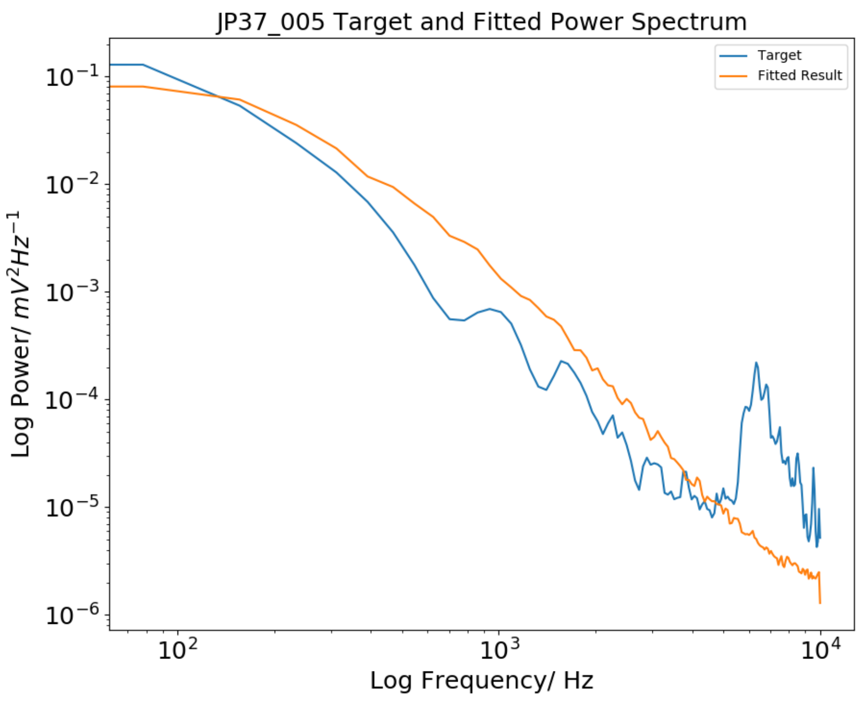

Figure 4.28: Optimised power spectrum with target power spectrum for eight parameter model

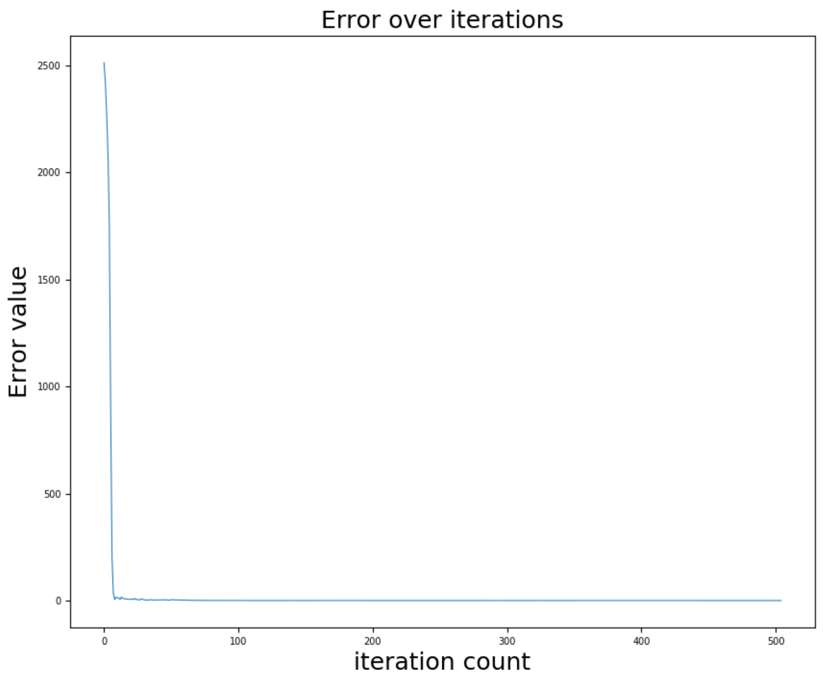

Figure 4.29: Best error against iteration count for eight parameter model

4.2 Discussion

CMA-ES seems to handle the increased dimensionality of the parameter models well, as shown by the significant decrease in error between the two parameter model and higher six and eight parameter models in 4.1.1. The fact that the decrease in error between the two and six parameter models is consistent in scale between synthetic and real-world data suggests that inaccuracies in the model and noise in the signal have consistent effect across at least those parameter models.

I suspect that the six and eight parameter models have inherent issues in that in the simulation model's current form the additional parameters are difficult to separate, there exist multiple distinct parameter settings that yield very similar final error values. The result shown in 4.26 is the key reason for why I believe this to be the case along with the abnormally high standard deviation in the final optimised firing rates for inhibitory inputs shown in 4.1.1. More work would need to be carried out to see if this is a peculiarity of this one recording or a systemic issue. I expect it to be the latter since I cannot see significant difference in the structure of the recording used for 4.1.1 compared to other recordings that were fitted similarly well.

While values for firing rates of inputs may appear somewhat high, this is likely due to the size of our input population being on the low end since the true parameter being estimated is $N_x \lambda x$ with the size of the population being fixed.

The consistency in final values for other parameters within model runs is encouraging but it cannot be conclusively said that this consistency represents the model approaching the true values of the real-world recordings since those values are not known within in this dataset, with solutions discussed below.

Some recordings were not fitted well in terms of error, though improved with higher parameter models. These recordings were those that had a comparatively high number of action potentials in the membrane trace, indicating that perhaps some filtering needed to take place.

4.2.1 Data Improvements

Filtering

Comparing traces that were fit well, 3.1 and 3.2, with traces that were fit poorly, 4.1, a clear difference is the amount of spiking activity. Since the difficulty in fitting came from not being about to fit the standard deviation of the membrane potential well, the best solution would likely be to filter out action potentials from the recording via some form of interpolation. This was not done initially as a majority of the recordings had few action potentials and the downsides of possibly filtering baseline activity out outweighed the possible benefits to a minority of recordings.

It should also be considered to either try and filter noise out of the original recording if possible, or alternatively the power spectra could be trimmed to remove the high frequency noise likely from magnetic or electronic sources in the setup.

Target Dataset

The traces given as part of this project by Dr. Palacios allowed an examination of how consistent CMA-ES could be in finding parameter fits on real data rather than simply fitting a simulation on a trace generated synthetically by that same simulation. If a dataset could have been obtained where the parameters we are trying to fit have already been determined then results could also be compared on how accurate they are to the true underlying value rather than simply checking consistency. If more time had been available I would have hoped to locate such a dataset if one exists. Since the parameters exist as part of a system, if even one or two parameters could be experimentally determined then that could allow the separation of the multiple solution clusters like those seen in 4.27.

4.2.2 Model Improvements

Correlated Inputs

While independent Poisson processes represent a commonly used abstraction within neuroscience it would be interesting to see how much of an improvement could be made to fitting the power spectra by introducing a variable firing rate common between inputs. This could be achieved by scaling the existing firing rate parameters by some periodic function to introduce correlation into the spike trains used as input. This is likely the highest payoff improvement that could be made initially since the majority of the error in final evaluations has been in the standard deviation of the membrane potential, which is consistently

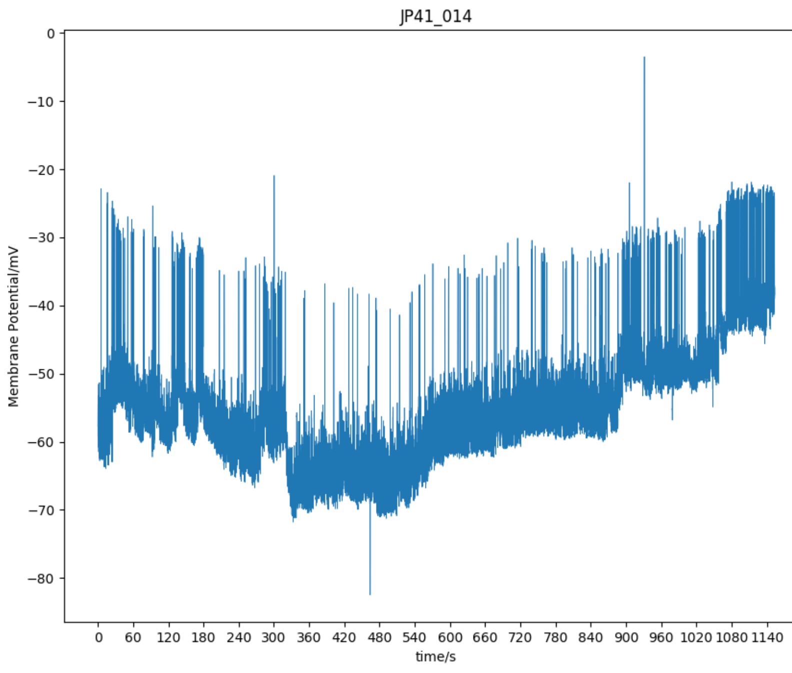

Figure 4.30: High variance membrane potential trace

underestimated. Increasing the correlation of the inputs would increase the standard deviation in the trace and potentially allow equivalent error values to that of the synthetic trace, though some would likely remain due to noise in the recordings.

State Based Model

In order to better generate traces like that in 3.2 some form of state based model could be used. Instead of selecting smaller traces to fit to to handle the jumps between mean membrane potential, modelling them could be attempted using some form of Hidden Markov Model so that some distinct set of states exist perhaps with a few parameters altered while in that state. So for example to model a period of higher membrane potential, a state could have a higher value of excitatory input firing rate. The frequency of these types of changes within the data used in this project however are such that it would be highly unlikely for such a state change to occur during the 1 second trace generated for evaluation.

4.2.3 Fitting Improvements

Power Spectrum Error

As shown by the plots of the overlap of the optimised power spectra for all three models, the estimation method for obtaining the power spectra in the first place likely needs to not use a uniform spread of estimation points. Since we care more about the lower end of the frequency spectrum as much or more than the high end, there should be a higher density of estimation points in that region. Alternatively the way we calculate error between the two spectra could be weighted to make deviations at low frequency more expensive in the error function.

Chapter 5

Conclusion

5.1 Summary of Project

In this project, a conductance based leaky integrate and fire model was built to generate synthetic recordings of membrane potential in a neuron with two input populations - excitatory and inhibitory synapses - and power spectra were estimated from this model via Welch's method that matched in aggregate the approximate shape and magnitude of power spectra from real world recordings.

The CMA-ES optimisation algorithm was tested on fitting three different parameter models to power spectra from real world recordings, fitting most of those recordings well in terms of error evaluations. Those recordings that did not seem to be well fitted had much higher occurrences of action potentials in the recorded neuron, possibly indicating that for inference of input population parameters in a general case, spike detection and filtering would need to occur before fitting.

Parameter fittings were also fairly consistent on a per trace basis except for the inhibitory firing rate λ_i and the inhibitory conductance decay time constant τ_i in the eight parameter model, with the cause being unclear though it could perhaps have been a unique result for a single trace.

The six parameter model that included firing rates and synaptic strength distribution parameters for excitatory and inhibitory inputs performed significantly better than the two parameter model with just firing rates being fitted. The eight parameter model that included the conductance decay time constants performed comparably to the six parameter model. This shows that the CMA-ES algorithm manages the increased dimensionality of the larger models well, but that the eight parameter model may introduce non-unique results due to parameter effect overlap that negatively impacts fitting.

An initial code release exists at `https://github.com/dsPolar/poseidon` though this will be updated to make it more intuitive for those with limited programming experience or are just new to *Python*.

5.2 Future Actions

As mentioned, the eight parameter model yielded unintuitive results with two broad clusters of parameter values seen in figure 4.26, where one cluster should have a significant impact upon the inhibitory conductance while the other should have comparably low impact. Further exploration is needed to determine what the cause of this is, perhaps the values obtained from the eight parameter model could be fed-forward into a two parameter model optimisation to see if it obtains similar results.

It was noticed that fitting to synthetic traces yielded error of less than 0.1 with a random spread of final error in terms of the three components, while the real-world traces could not approach this magnitude of error due to comparatively high error in the standard deviation of the membrane potential. Expanding the simulation to use dependent Poisson processes by varying the firing rate with time with a periodic function and including some parameter of this function in the model to be optimised could allow smaller error evaluations.

Bibliography

[1] BlueCrystal Phase 3 - ACRC, University of Bristol.

[2] CED Spike2: System introduction.

[3] cma · PyPI.

[4] Neo - NeuralEnsemble.

[5] NeuroElectro :: Neuron Index.

[6] David G. Amaral, Helen E. Scharfman, and Pierre Lavenex. The dentate gyrus: fundamental neuroanatomical organization (dentate gyrus for dummies), jan 2007.

[7] Peter Dayan, L. F. Abbott, Wesley J. Wildman, Richard Sosis, and Patrick McNamara. Computational and Mathematical Modeling of Neural Systems. *The MIT Press*, 4(3):181–182, 2005.

[8] Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985.

[9] Lixing Han and Michael Neumann. Effect of dimensionality on the Nelder-Mead simplex method. *Optimization Methods and Software*, 21(1):1–16, feb 2006.

[10] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. apr 2016.

[11] Nikolaus Hansen and Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. *STUDFUZZ*, 192:75—-102, 2006.

[12] Marta Jelitai, Paolo Puggioni, Taro Ishikawa, Arianna Rinaldi, and Ian Duguid. Dendritic excitation-inhibition balance shapes cerebellar output during motor behaviour. *Nature Communications*, 7(1):1–13, dec 2016.

[13] Hudspeth A. J. Kandel Eric R., Schwartz James H., Jessell Thomas M., Siegelbaum Steven A. *PRINCIPLES OF NEURAL SCIENCE Fifth Edition*, volume 53. Fifth edit edition, 2013.

[14] M. Megías, Zs Emri, T. F. Freund, and A. I. Gulyás. Total number and distribution of inhibitory and excitatory synapses on hippocampal CA1 pyramidal cells. *Neuroscience*, 102(3):527–540, feb 2001.

[15] Jr. Otis M. Solomon. PSD Computations Using Welch's Method. Technical report, 1991.

[16] Paolo Puggioni. *Input-Output Transformations in the Awake Mouse Brain Using Whole-Cell Recordings and Probabilistic Analysis.* PhD thesis, University of Edinburgh, 2015.

[17] James T. Russell. Imaging calcium signals in vivo: A powerful tool in physiology and pharmacology, aug 2011.

[18] Joël Tabak, C. Richard Murphey, and L. E. Moore. Parameter estimation methods for single neuron models. *Journal of Computational Neuroscience*, 9(3):215–236, 2000.

[19] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, Christof Koch, and Stefan Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature Communications*, 9(1):1–15, dec 2018.

[20] Shreejoy J. Tripathy, Judith Savitskaya, Shawn D. Burton, Nathaniel N. Urban, and Richard C. Gerkin. NeuroElectro: A window to the world's neuron electrophysiology data. *Frontiers in Neuroinformatics*, 8(APR), apr 2014.

[21] W. Van Geit, E. De Schutter, and P. Achard. Automated neuron model optimization techniques: A review, nov 2008.

[22] Dimitrios V. Vavoulis, Volko A. Straub, John A.D. Aston, and Jianfeng Feng. A self-organizing state-space-model approach for parameter estimation in Hodgkin-Huxley-type models of single neurons. *PLoS Computational Biology*, 8(3), mar 2012.

[23] Peter D. Welch. The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.