# Learning systems biology models from data

*Edon Aliko*

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2021

# Abstract

Ordinary differential equations are often used in systems biology models to estimate the kinetic rate parameters which shape the dynamical systems of the model. One such experiment where this method is used is proposed by Faas et al. [6], who model the interaction between Calcium and the Calcium-binding protein Calmodulin which occurs during synaptic plasticity. Due to the non-linearity of ODE's and experimental conditions which lead to uncertainty, these estimates can not often be trusted, as they only provide a point estimate of the model parameters. In this paper, we propose a way of obtaining the posterior distributions of the kinetic rate constants in Faas et al.'s experiment by using the Metropolis-Hastings Markov Chain Monte Carlo method. We analyze different techniques to improve the estimates as well as to try and optimize the convergence of the chains. In conclusion, we found that our parameter estimates resulted in superior model outputs compared to the model outputs obtained through the Faas et al. estimates. Furthermore, we were able to find the joint and single distributions of all the parameters, which allows our method to be generalized and be used in various ODE systems biology models.

## Acknowledgements

I’d like to thank my two supervisors during the course of my thesis, Dr. David Sterratt and Dr. Melanie Stefan. It has been a pleasure working with them and learning from them. They have been insightful, understanding and encouraging all throughout the year, and despite this year’s challenges, working with them has been very enjoyable and easy.

# Table of Contents

# Chapter 1

# Introduction

## 1.1 Motivations

A special challenge in systems biology is that parameters of interest are often unknown and need to be approximated through data. Frequently, Ordinary Differential Equations (ODE) have been used to model dynamical systems, yet due to the nature of systems biology, experimental noise and non-linearity of ODE systems pose a challenge when trying to estimate these parameters [19]. Optimization methods are often applied in systems biology [8][13], however due to the aforementioned challenges, these methods will only result in parameter estimations corresponding to some local maxima of their likelihood, thus not providing posterior and joint distributions over parameter values [19]. To overcome this issue, an alternative to optimization techniques are Markov Chain Monte Carlo (MCMC) methods, which use Bayesian inference to infer posterior distributions for the parameters of interest.

Faas et al. designed a model describing the chemical reactions which occur during synaptic plasticity. Synaptic plasticity is a process involved in learning and memory in the brain, and certain neuropsychiatric disorders can be attributed to faults in this process. Synaptic plasticity is often initiated by changes in intracellular Calcium ($Ca^{2+}$) [21][16] and its interactions with Calmodulin (CaM)[20]. Faas et al.'s model describes the dynamics of the binding between $Ca^{2+}$ and CaM, which they used to find estimates of the kinetic rate parameters through the use of ODEs. Despite the fact that they have estimates for the parameters of interest, their method to infer those parameters is complex and tailored to their specific model and experiment, as well as to their set of ODEs. In this paper, we explore the use of MCMC methods to obtain posterior distributions of the kinetic rate constants and find the join distributions between parameters, thus validating a method which can be generalized for various experiments in systems biology and overcoming the challenges which arise when using classical optimization methods.

## 1.2 Objectives

The primary objective of this paper is to estimate the posterior probability distributions of the various parameters used by Faas et al. [6] to model the dynamics and interaction between $Ca^{2+}$ and CaM involved during the process of synaptic plasticity. The data that will be used in the project is the experimental data from the Faas et al. experiment, and we hope to find estimates of the parameters which align and confirm their findings and provide good model outputs. Despite the use of this dataset, we expect that this method can be easily extended to be used on various systems biology models in finding kinetic rate parameters.

As described in the previous section, systems biology models suffer from the challenges attributed to the non-linearity of ODEs and experimental noise. To overcome this, we will use the Metropolis-Hastings Markov Chain Monte Carlo method implemented in the `emcee` python library [3]. Previous work carried out by Candel [2] attempted to solve these challenges, however they encountered challenges regarding the convergence of the MCMC chain, thus an objective for this paper is to better understand the viability of MCMC as a technique for parameter estimation given our data, and we will then compare our results to the ones found by Faas et al. We will analyze the MCMC results by examining the one and two dimensional projections of the posterior probability distributions of each parameter, and then judge the model output achieved when using those parameters. Furthermore, we aim to have a method which allows the MCMC chains to converge, thus allowing us to trust our results.

## 1.3 Summary of results

Throughout this project, we determined that MCMC is a viable technique to use on the data provided, however, it is necessary to somehow include nuisance parameters which introduce uncertainty in the model outputs. Firstly, we ran MCMC on a synthetic dataset and found that the joint distributions produced followed a predictable Gaussian curve, which is what is to be expected for the parameters since we assumed a Gaussian noise model for the likelihood function. We then proceeded to run 5000 iterations of MCMC using the interpolarized likelihood function, which allowed us to more evenly distribute the data points, without the inclusion of nuisance parameters. The results from this experiment aligned with the hypothesis that nuisance parameters have a large effect on the model output as seen in Section 4.3, where we saw that when including randomly generated nuisance parameters in our synthetic data creation, the posterior distributions were no longer Gaussian. Lastly, we included nuisance parameters in our likelihood computation and found that this method successfully removed the error caused by the uncertainty in the nuisance parameter values. Although by our analysis, after 5000 iterations the chain had not converged, the results from this experiment give us estimates of the posterior distribution of the parameters, and the maximum a posteriori estimates are similar to the ones found by Faas et al. in their experiment.

## 1.4 Structure of the report

Chapter 2 provides an insight into synaptic plasticity and more details on the Faas et al. experiment. We also explore the theory behind MCMC and discuss the previous work carried out on this topic[2]. We will identify the use for using MCMC methods and what value they can being in systems biology model. Chapter 3 outlines the methods for the various experiments which have been carried out throughout the project. It includes looking at how well MCMC performs against a synthetic dataset, performing MCMC without using nuisance parameters, using interpolation to evenly distribute the data and marginalizing over the nuisance parameters. Chapter 4 later reports the results achieved from running the experiments from Chapter 3, where we analyze the convergence of the MCMC chains and the parameter estimates. Finally, in Chapter 5 we summarize the project findings and techniques and propose future work which can be carried out to further improve the results achieved using MCMC. Lastly, we found the mean squared error values between the model output and experimental data achieved through our estimates using MCMC were much lower than the MSE values obtained by using the Faas et al. parameter estimates.

# Chapter 2

# Background

## 2.1 Synaptic Plasticity

The ability of the brain to change and adapt to new information is referred to as plasticity. The phenomenon by which the neural activity generated by experiences modifies brain functions of synaptic transmission is called synaptic plasticity. Synaptic plasticity refers to the activity-dependent modification of the strength or efficacy of synaptic transmission at preexisting synapses, which are connections between neurons, and it is believed to play a central role in the brain when incorporating transient experiences into persistent memory traces. Furthermore, it is thought to also contribute to neuropsychiatric disorders when it malfunctions [14].

Due to the variety of functions associated with synaptic plasticity, many forms and mechanisms have been described. Synaptic transmission can be either enhanced or depressed by activity with varying temporal domains. Most of the research in this domain explores long-term potentiation (LTP) and depression (LTD) [14]. In both cases, synaptic transmission is dependent on changes in intracellular Calcium ($Ca^{2+}$) [21][16], and its interaction between $Ca^{2+}$ binding proteins such as Calmodulin (CaM) to regulate synaptic transmission[16][20].

## 2.2 Calmodulin model by Faas et al. Model

### 2.2.1 Experiment

Previous work has been carried out to understand the interaction between Calmodulin as a direct detector of $Ca^{2+}$ signals, and in particular, Faas et al. [6] found that CaM directly intercepts incoming unbound $Ca^{2+}$ and strongly contributes to fast $Ca^{2+}$ buffering, therefore making CaM and efficient transducer. To achieve these results, they tracked the concentration of unbound $Ca^{2+}$ as it reacted with CaM in a reaction vessel using a $Ca^{2+}$ sensitive fluorescent dye. [6]

There are two $Ca^{2+}$-binding sites at both termini of CaM (N- and C-lobes). Both these sites have distinct $Ca^{2+}$-binding properties, and the goal of the experiment was to

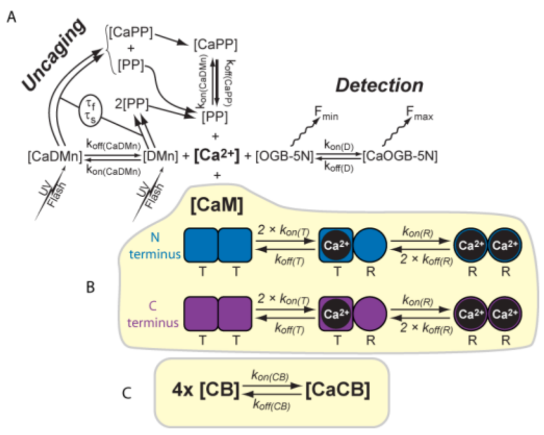


Figure 2.1: Schematics of the simulated reactions by Faas et al. [6]

find the on-rate ($k_{on}$) constants and off-rate ($k_{off}$) kinetic constants, describing different binding dynamics [6]. To find the rate at which $Ca^{2+}$ binds to CaM, they tracked the fall in unbound $Ca^{2+}$ ($[Ca^{2+}]_{free}$ ) after a rapid ($<100$ µs) change in the total $Ca^{2+}$ produced by flash photolysis of DM-nitrophen (DMn) by using a $Ca^{2+}$ sensitive fulorescent dye called OBG-5N. Figure 2.1 shows the schematics of the simulated reactions as described and modelled by Faas et al.

The results show that CaM is activated upon a rise in cellular $Ca^{2+}$ producing 10-100 times more activated CaM than previously thought. In addition, there is a difference in $Ca^{2+}$ binding between each of the two lobes, with the N-lobe being the first to bind $Ca^{2+}$.

### 2.2.2 Model

CaM has two $Ca^{2+}$-binding sites at both its N- and C-lobes, both having distinct properties and show cooperativity. Faas et al. quantified the binding kinetics by fitting the data with a two-step binding model of cooperative binding to each lobe.

$$N_TN_T + Ca^{2+} \underset{k_{off(T),N}}{\overset{2\cdot k_{on(T),N}}{\rightleftharpoons}} CaN_TN_R \tag{2.1}$$

$$CaN_TN_R + Ca^{2+} \underset{2\cdot k_{off(R),N}}{\overset{k_{on(R),N}}{\rightleftharpoons}} CaN_RCaN_R \tag{2.2}$$

$$C_TC_T + Ca^{2+} \underset{k_{off(T),C}}{\overset{2\cdot k_{on(T),C}}{\rightleftharpoons}} CaC_TC_R \tag{2.3}$$

$$CaC_TC_R + Ca^{2+} \underset{2\cdot k_{off(R),C}}{\overset{k_{on(R),C}}{\rightleftharpoons}} CaC_RCaC_R \tag{2.4}$$

where $N_x$ and $C_x$ represent binding sites on the N and C lobes, respectively. After the first binding, the state changes from T (nothing is bound to the lobe) to R ($Ca^{2+}$

is bound to lobe), giving rise to cooperativity. The $k_{\text{x}}$ values represent the kinetic constants which describe binding dynamics, values which this paper tried to infer.

The model fits reliably described the data and were consistent with previous work, showing that the C-lobe bound Ca$^{2+}$ with higher affinity than the N-lobe. To determine the kinetic parameters from the recordings, Faas et al. used an Ordinary Differential Equation (ODE) solver to simulate a time course of the system, allowing them to achieve an approximate numerical solution for the kinetic parameters by testing different combination of parameters, starting with a broad range and then narrowing it down, and comparing them to a randomly selected subset of traces. The ODEs for the N lobe are found below, and since the dynamical equations for the two lobes are identical, the same ODEs are valid for the C lobe.

$$\begin{aligned}
\frac{d[N_TN_T]}{dt} &= -2k_{on(T),N}[N_TN_T][Ca^{2+}] + k_{off(T),N}[CaN_TN_R] \\
\frac{d[CaN_TN_R]}{dt} &= 2k_{on(T),N}[N_TN_T][Ca^{2+}] - k_{off(T),N}[CaN_TN_R] \\
&\quad -k_{on(R),N}[CaN_TN_R][Ca^{2+}] + 2k_{off(R),N}[CaN_RCaN_R] \\
\frac{CaN_RCaN_R}{dt} &= k_{on(R),N}[CaN_TN_R][Ca^{2+}] - 2k_{off(R),N}[CaN_RN_R] \\
\frac{Ca^{2+}}{dt} &= -2k_{on(T),N}[N_TN_T][Ca^{2+}] + k_{off(T),N}[CaN_TN_R] \\
&\quad -k_{on(R),N}[CaN_TN_R][Ca^{2+}] + 2k_{off(R),N}[CaN_RCaN_R]
\end{aligned} \tag{2.5}$$

To see that the reactions can be implemented as ODEs and that the set of ODEs work, Figure 2.2[6] shows Ca$^{2+}$ buffering by CaM, where the individual points are the experimental values and the solid black lines are the fit achieved through a mathematical model using the kinetics parameters found through the use of the set of ODEs from equation 2.5.

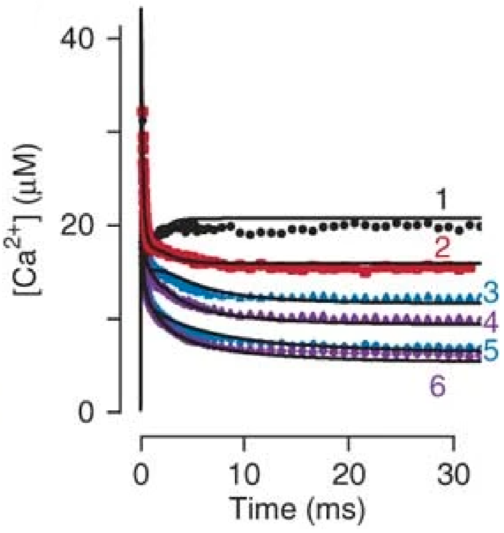


Figure 2.2: Traces from Faas et al. experiment [6]

Faas et al. carried out 94 experiments which they used to fit the kinetic parameters and find values for the $k_{\text{on}}$ and $k_{\text{off}}$ rates, where each experiment has 259 entries corresponding to a specific time point. The experimental data is split up into 7 batches, each one containing a different number of experiments (between 12 and 14), where experiments in the same batch are characterized by the fact that the concentration of the reagent is the same, but the length of the length of the flash differed. The issue with this approach is that there is no distribution of possible parameters, but rather just a numerical solution with error bars. Furthermore, this method is very specific to their model and dataset, meaning that it cannot be generalized to find parameters of different models. [6]

## 2.3 Sloppiness property in systems biology

Other models have been researched which explored the interactions between $Ca^{2+}$ and CaM. One of these is the Pepke et al. model [17], which is similar to the one created by Faas et al [6]. They used gradient descent on 3 experimental data sets to find point estimates kinetic rates such as the ones in the Faas et al. experiment. However, the values they found for parameters in common with Faas et al.'s experiment, such as the dissociation constants, differ widely from the ones found by Faas et al., suggesting that either the method used for parameter estimation, dataset or model influence the outcomes. Previous experiments also show that there is a difference in kinetic parameter values compared to the values reported by Faas et al [10][11]. Furthermore, work carried out by Judith Borowski [1] found that the model proposed by Faas et al. suffered from this very property, thus further reinforcing the need for a more robust and generalized model which can be used for various experimental data.

It is possible that the difference in parameter values suggests that the model, inference method and data used affect the outcomes. Furthermore, the inference methods used in these experiments, such as the one by Faas et al. are not easily applicable to new data and thus not easily generalized. Thus it would be useful to have a generalized inference method that can be used to infer the parameters of any systems biological model, allowing for comparisons between different models.

Another factor that would lead to different parameter values is the sloppiness property, which is present in most systems biology models. Sloppiness refers to the property where model behaviour is insensitive to changes except along a few combinations of parameters, with an large sloppy neutral subspace [4], leading to multiple value combinations having equally good fits to the experimental data [7]. Due to this property and the fact that similar experiments can result in different outcomes, it is desirable to find the full posterior distribution of the parameter space, as opposed to point estimates. This will return the parameter values and the confidence associated with such parameters, as well as showing dependence between parameters.

## 2.4 Markov Chain Monte Carlo

The desired outcome which builds on Faas et al.'s experiment would be to find the posterior distribution of the parameters given the data. A popular technique for sampling from a high-dimensional distribution is the use of Markov Chain Monte Carlo methods (MCMC). Using MCMC methods allows us to generate samples from a given probability distribution and estimate expectations of functions under this distribution [12]. To sample from the unnormalised probability distribution, we can evaluate a function $P^*(\vec{\theta})$ such that

$$P(\vec{\theta}) = \frac{P^*(\vec{\theta})}{Z} \tag{2.6}$$

where $\vec{\theta}$ is the vector containing the parameters of interest and Z is the usually unknown normalizing constant defined as

$$Z = \int d^N\vec{\theta} P^*(\vec{\theta}) \tag{2.7}$$

Where $N$ is the dimension of the parameter space. The issue is that even if Z were known, we would have sample from everywhere on the $\vec{\theta}$-space, where $P(\vec{\theta})$ is large. To avoid having to sample from $\vec{\theta}$ uniformly, we can use different MCMC methods. In the sampling methods, we assume we have a simpler density $Q(\vec{\theta})$ from which we can generate samples from which we can evaluate $Q^*(\vec{\theta})$ (where $Q(\vec{\theta}) = \frac{Q^*(\vec{\theta})}{Z_Q}$) such that in the limit of a large number of samples, the samples will be from $P(\vec{\theta})$ [12]. The most popular method is the Metropolis-Hasting algorithm, in which we sample from a from a proposal density Q which depends on the current state $\vec{\theta}^{(t)}$. The proposal density $Q(\vec{\theta}'; \vec{\theta}^{(t)})$ can be any fixed density from which we can draw samples, where $\vec{\theta}'$ is a tentative new state generated from it. To decide whether to accept the new state, we compute:

$$a = \frac{P^*(\vec{\theta}')}{P^*(\vec{\theta}^{(t)})} \frac{Q(\vec{\theta}^{(t)}; \vec{\theta}')}{Q(\vec{\theta}'; \vec{\theta}^{(t)})} \tag{2.8}$$

If $a \geq 1$ then the new state is accepted, otherwise the new state is accepted with probability $a$. If the step is accepted, we set $\vec{\theta}^{(t+1)} = \vec{\theta}'$, otherwise if it is rejected, we set $\vec{\theta}^{(t+1)} = \vec{\theta}^{(t)}$. Other MCMC algorithms exist, each of them having their own advantages and disadvantages, but they use similar acceptance functions.

### 2.4.1 `emcee`

Previous work has been carried by students regarding finding the posterior distribution of the parameters used in the Faas et al. model, such as Carine Candel's MSc dissertation [2]. Their work included using a Python MCMC package called `emcee` to solve this problem. The method used involves an affine invariant "stretch move" proposed by Goodman and Weare (2010) based on the Metropolis-Hasting algorithm. The affine invariance is adequate for the use in systems biology as it works well in the elongated spaces often seen in biological models. Furthermore, The Metropolis-Hasting algorithm uses $\frac{N(N+1)}{2}$ tuning parameters due to the fact that each term in the covariance matrix of this proposal distribution is unspecified [3], where N is the dimension of the parameter space. However, `emcee` reduces the parameters needing tuning down to one, and all that is needed to sample from the parameter posterior is a function proportional to the log posterior over the parameters. Since the log posterior itself is unknown, Candel [2] used the following log likelihood function to estimate the true log posterior distribution.:

$$\log p(\vec{\theta},\sigma|\vec{X},\vec{\varepsilon}) = \log p(\vec{X}|\vec{\theta},\vec{\varepsilon}) + \log p(\vec{\theta}) + \log p(\sigma) \tag{2.9}$$

where $\vec{\theta}$ represents a vector of the parameters we are interested in, $\sigma$ represents the standard deviation used in the likelihood, $\vec{X}$ represents the data and $\vec{\varepsilon}$ represents the nuisance parameter values modelling the uncertainty in the flash photolysis process. A nuisance parameter is one that is required to model the process that generates the data, but otherwise, its distribution is of little interest. For the purpose of this paper and the methods used, we assume a Gaussian noise model, thus in the equation above, Gaussian noise was added to the outcome of the simulator.

Candel then proceeded to include nuisance parameters to represent the inaccuracies in the uncaging experiments for every experiment. Due to the fact that Faas et al. ran 94 experiments, there are a total of 94 nuisance parameters that we would need to sample over in order to find their posterior distribution. Since our parameter space already consisted of 10 parameters, with the additional 94 nuisance parameters, this would increase the dimensionality from 10 to 104, thus slowing down the rate of convergence of the chain.

To circumvent this, it is possible to marginalize over nuisance parameters, where marginalization is the process of integrating over possible values of a parameter and thus eliminating the effect of its value's uncertainty on the data.

Candel attempted to marginalize over the nuisance parameters, and thus were able to add the nuisance parameters to the MCMC chain without increasing dimensionality thus without a significant increase in computational costs. Although this method seemed promising, the chain obtained when not using nuisance parameters had not converged after 20,000 iterations, thus they were not able to find the posterior distribution of the parameters. Although the experiments ran without the nuisance parameters, the results demonstrated the need to use them in the final MCMC. Furthermore, Candel's parameter estimates did not create Gaussian posterior distributions, and the joint distributions between various parameters did not provide any useful relationships, thus their method was unsuccessfull.

# Chapter 3

# Methods

## 3.1 MCMC Model

### 3.1.1 Model

To simulate the model presented by Faas et al., I used the ODE solver taken from the MSc thesis of Judith Borowski [1] to solve the dynamical equations described in equations 2.1-2.4. The initial input parameters form vector $\vec{\theta}$, which holds the 4 $\log k_{on}$ rates, the 4 $\log K_D$ rates, $m_\alpha$ and $\alpha_0$. These parameters along with the conditional parameters for each of the 94 experiments are passed to the model which then returns the modelled f-ratio data, which is the current modelled fluorescence over the starting fluorescence. Since the conditional nuisance parameters for the 94 experiments are only used to define the experiment and we aren't concerned about their distribution, we are only interested in inferring the values of $\vec{\theta}$. The amount of $Ca^{2+}$ that is uncaged is modelled as:

$$\max((1+\varepsilon_c)(m_\alpha P_c+\alpha_0),0) \tag{3.1}$$

where $\varepsilon_c$ represents the conditional parameters for experiment c, $P_c$ represents Pockels Cell Delay (length of the UV-flash) and $m_\alpha$ and $\alpha_0$ model the intercept and slope of the process.

Equation 2.9 provided earlier outlines the equation of the posterior distribution of the parameters which was passed to the `emcee` package. This is made up of the likelihood and the prior of the parameters. The following subsections describe how the prior and likelihood were achieved in order to compute the posterior.

#### 3.1.1.1 Prior

The prior over the parameters $\vec{\theta}$ can be determined from the original results from the Faas et al. paper with the appropriate ranges by looking at their confidence intervals. From their paper, the values for the $\log k_{on}$ rates were chosen from a uniform prior from 5 to 12, while for the $\log K_D$ rates, the values were chosen from a uniform prior between -8 and -2. The priors for $\alpha_0$ and $m_\alpha$ were defined by the following priors respectively:

$\mathcal{N}(-0.39, (-0.39 \times 2)^2)$ and $\mathcal{N}(0.0011, (0.0011 \times 2)^2)$, where again these were taken from the Faas et al. paper. Finally, all the nuisance parameters representing each experiment were defined by the Gaussian distribution $\mathcal{N}(0, 0.1^2)$. Since the `emcee` package requires log probabilities, we made the simplifying assumption that all the parameters are independent and thus their log probabilities were added to return the log prior.

#### 3.1.1.2 Likelihood

Since we are assuming a Gaussian noise model for the data, we artificially added Gaussian noise to the outcome of the model. This noise represents some of the measurement noise that would be present in the original experiment. The likelihood can be computed as follows.

Denote the f-ratio of the experimental data as $\vec{X_c}$ and the f-ratio outcome of a simulation from using the model as $\vec{x_c}$. $\vec{x_c}$ can be calculated directly from the model by providing the input parameters $\vec{\theta}$ and the experiment specific parameters $\varepsilon_c$. The likelihood of a single data point at time $t$ is as follows:

$$p(X_c^{(t)}|\vec{\theta}, \varepsilon_c, \sigma) = \mathcal{N}(X_c^{(t)}; x_c^{(t)}, \sigma^2) \tag{3.2}$$

Where the notation $\vec{x_c}^{(t)}$ represents the solution to the set of ODEs presented earlier parameterised by $\vec{\theta}$. By making the simplifying assumptions that different time points are independent, and that different experiments are also independent, we can calculate the likelihood through:

$$p(\vec{X_c}|\vec{\theta}, \varepsilon_c, \sigma) = \prod_t p(X_c^{(t)}|\vec{\theta}, \varepsilon_c, \sigma) \tag{3.3}$$

$$p(\vec{X}|\vec{\theta}, \varepsilon, \sigma) = \prod_c p(\vec{X_c}|\vec{\theta}, \varepsilon_c, \sigma) \tag{3.4}$$

Again, since the `emcee` package uses log probabilities, the log likelihood was computed with the following formula:

$$\begin{aligned} \log(p(\vec{X}|\vec{\theta}, \vec{\varepsilon}, \sigma)) &= \sum_c \sum_t \log(p(X_c^{(t)}|\vec{\theta}, \varepsilon_c, \sigma)) \\ &= \sum_c \sum_t \frac{x_c^{(t)} - X_c^{(t)}}{2\sigma^2} - \frac{\log(2\pi\sigma^2)}{2} \end{aligned} \tag{3.5}$$

### 3.1.2 Running emcee

The `emcee` has an object called `EnsambleSampler` which is used to run the MCMC. One of the parameters that is required for this class is a $\theta$ matrix which holds initial values of the parameters we want to find the posterior of. We initialized 20 walkers to run MCMC, where a walker is a member of the ensamble which can be thought of as a Metropolis-Hastings chain where the proposal distribution for a given walker depends on the positions of all the other walkers in the ensemble. We used 20 walkers since this is twice the number of parameters we are estimating, as recommended [3]. Since we are aiming to sample over 10 parameters, $\vec{\theta}$ will therefore be a $20 \times 10$ matrix, where each row is initialized to a Gaussian centered around the original parameter values found by Faas et al. with a standard deviation of $1 \times 10^{-4}$. Although the initial values aren't very important, this can reduce the time it takes for the walkers to branch out and thus to converge. Lastly, for all the experiments outlined later, the standard deviation was kept constant at 1.5.

## 3.2 Experiments

### 3.2.1 Using fixed values

One of the drawbacks of `emcee` is the fact that it does not scale well in high dimensions. Even when using the full 10 parameter space from the Faas et al. experiment, running MCMC until convergence takes a considerable amount of time due to the fact that each iteration takes an average of 7.1s. To combat this, we decided to fix some parameter values and only run MCMC on the 4 remaining parameters. Firstly, we fixed the 4 forward rate constants and the parameters $\alpha_0$ and $m_\alpha$, which link the fraction of $Ca^{2+}$ uncaged to the Pockels Cell Delay, to be the same as from the Faas et al. experiment, and only found the distributions for the dissociation constants. Here we refer to a dissociation constant as one which describes the affinity between a protein and a ligand, which in this case is between $Ca^{2+}$ and CaM. The justification for this is that by finding the dissociation constants of the experimental data once it has reached the equilibrium state, we can then compute the forward rate constants using these values, thus allowing them to as constant when running MCMC. However, we decided to also include the forward rate constants when running MCMC while still keeping the last two parameters fixed. This still allowed for a performance improvement when running MCMC until convergence, decreasing the time per iteration to 5.5s.

### 3.2.2 Synthetic Data

To understand whether MCMC would produce adequate parameter posterior distributions given the model, we ran `emcee` on a synthetic dataset produced by running one simulation of the model with the Faas et al. parameter values as inputs. The simulation returns F-ratio values as described earlier, and we use these instead of the experimental data in our likelihood function.

The resulting corner plot after 2000 iterations is shown in Figure 4.2, and it shows that the model output data can be sampled using MCMC, providing strong parameter posterior distributions as well as covariances between parameters. The results can be seen in Chapter 4, where we ran MCMC using 8 parameters on the synthetic dataset.

### 3.2.3 Interpolation

The experimental data provided is a matrix of 94 rows (one row per experiment) and 259 columns (one column per time point). However, when analyzing the data, it is possible to see that the time points are not evenly distributed, with over 90% of readings occurring in the first 5ms of the experiment. This will have an effect on how well the parameters will fit the model and how they compare to the experimental data. An attempt to overcome this is to use linear interpolation to evenly distribute the data and then use it to carry out MCMC. Figure 3.1 shows how the data is distributed originally and how interpolation of the experimental data distributes the points more evenly across the time course, thus smoothing out the initial points and giving an even weight to the later points when calculating the likelihood of the model data. Without this, the later points of the model data would not contribute much to the computation of the likelihood value, and thus they could contribute to a worse model fit.

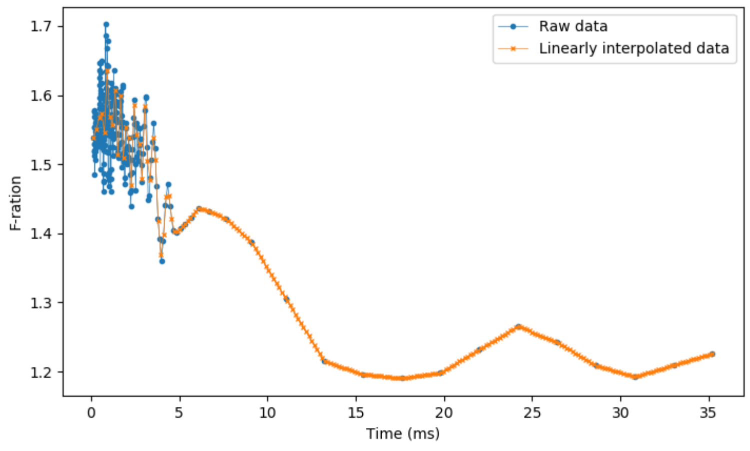


Figure 3.1: F-ratio during timepoints for raw experimental data and linearly interpolated data

### 3.2.4 Marginalization

One of the disadvantages of MCMC is that it performs poorly at high dimensions. This poses a challenge when trying to accurately model the experiment at hand, where we have 94 nuisance parameters which represent the conditions of each experiment. If these conditions aren't included, we would have a poor posterior distribution of the parameters and thus a poor model fit. However, simply adding and sampling over 94 parameters would increase the time taken for the MCMC to converge exponentially, and furthermore, learning about their distribution isn't important since they only shape the experimental conditions. To overcome this, we can marginalize over the nuisance parameters and thus indirectly include their distribution when computing the likelihood, and thus in turn removing the uncertainty added by these parameters in the simulations.

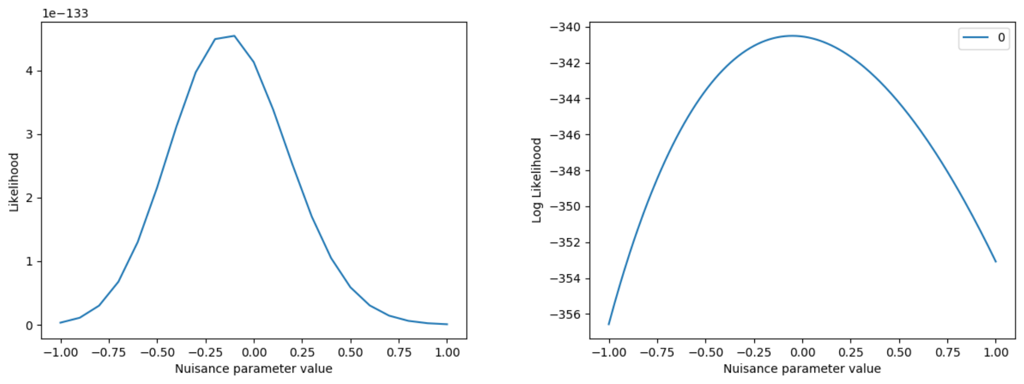
The posterior distribution expressed in equation 1.8 when marginalizing out the nuisance parameters can be formulated as:

$$\begin{aligned} p(\vec{\theta},\sigma|\vec{X}) &\propto p(\vec{\theta})p(\sigma)p(\vec{X}|\vec{\theta},\sigma) \\ &= p(\vec{\theta})p(\sigma)\int\cdots\int p(\vec{X}|\vec{\theta},\vec{\varepsilon},\sigma)p(\vec{\varepsilon})\,d\vec{\varepsilon} \\ &= p(\vec{\theta})p(\sigma)\prod_c\int p(\vec{X}_c|\vec{\theta},\varepsilon_c,\sigma)\,d\varepsilon_c \end{aligned} \tag{3.6}$$

The marginalized log posterior is thus the sum of the log prior over theta and sigma plus the sum of the integral under the log likelihood curves for each experiment.

$$\log(p(\vec{\theta},\sigma|\vec{X})) = \log(p(\vec{\theta})) + \log(p(\sigma)) + \sum_c \log\int p(\vec{X}_c|\vec{\theta},\varepsilon_c,\sigma)\,d\varepsilon_c + C \tag{3.7}$$

The integral in Equation 3.7 represents the area under the curve of the likelihood over different epsilon values, which we assume to form a Gaussian distribution. We can verify that the shape formed by plotting the likelihood against values of epsilon ranging from -1 to 1 is indeed Gaussian as seen in Figure 3.2a when using the experimental data from the first Faas et al. experiment.



(a) Likelihood vs nuisance parameter values (b) Log Likelihood vs nuisance parameter values

Figure 3.2: Likelihood and log likelihood for different nuisance parameter values for one experiment

One way in which we can compute this area is through numerical integration which can be computationally expensive, since we would need to calculate the likelihood for many different epsilons. Another approach which is sufficiently accurate for the purpose of marginalizing over the nuisance parameters is to approximate the area under the conditional likelihood. This approximation can be done by inferring the parameters of the Gaussian indirectly. Finding the mean and standard deviation of a Gaussian can easily be done when we have many points, however, we want to minimize the number of likelihood computations done to lower the time taken per iteration, since for each computation, we run a simulation of the model. Thus finding the Gaussian parameters directly from data points is not feasible. As an alternative, we can instead plot the log likelihood against different nuisance parameter values, which forms a quadratic curve as seen in Figure 3.2a. This is the technique which is used to to approximate the area of the conditional likelihood as outlined below.

1. Firstly, we find the nuisance parameter value which minimises the negative log likelihood, which is the equivalent to maximising the log likelihood. This gives us the peak of the quadratic curve which we can use to determine the function of the quadratic curve

2. We then calculate the log likelihood for points on either side of the optimal nuisance parameter value, these points being equidistant from the maximum and from each other and whose resulting likelihood falls within a certain range. It is important that the likelihood of the calculated points remains within a certain range due to the fact that the log likelihood values can approach negative infinity for certain experiments, and when this happens, the curves are no longer quadratic. Thus we limit this situation by only calculating likelihood for points near the maximum which do not fall below a log likelihood threshold set by:

$$\log p(\vec{X}_c|\vec{\theta},\varepsilon_c,\sigma)_{\max} - \log 0.01 p(\vec{X}_c|\vec{\theta},\varepsilon_c,\sigma)_{\max}$$

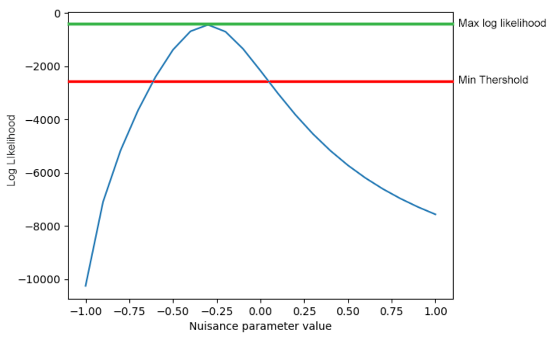


Figure 3.3: Log likelihood over nuisance parameter values for an experiment with an irregular shape

Figure 3.3 shows the log likelihood values for different epsilons for one specific experiment where certain extreme values of epsilon can lead to irregularities in the shape of the log likelihood curve. In situations like these, it is useful to limit the range of the log likelihood which is considered in order to keep the curve quadratic. For this specific experiment and values of $\vec{\theta}$, the highest log likelihood was $-434.8$, thus the minimum log likelihood threshold becomes $-434.8 - \log 0.001 \times -434.8 = -2437.13$, under which no more points will be used to calculate the likelihood.

3. We can then fit a quadratic curve through these points by using a `NumPy`'s (a `Python` library) `polyfit` function to find its coefficients.

4. We can now compute the integral of exponential of this curve to integrate over the nuisance parameters.

   Using the coefficients found in step 3, we now have a function $ax^2 + bx + c$ which we use to compute the integral over the nuisance parameter values within the range of -1 and 1.

$$\int_{-1}^{1} e^{ax^2+bx+c}\, d\varepsilon = -\frac{\sqrt{\pi}(erf(\frac{b+2a}{2\sqrt{-a}}) - erf(\frac{b-2a}{2\sqrt{-a}}))e^{c-\frac{b^2}{4a}}}{2\sqrt{-a}} \tag{3.8}$$

Where *erf* is the error function. The result of this integral provides an approximation of the integral from equation 1.17.

To implement this change in the MCMC code, on each iteration of the chain, we minimize the negative log likelihood with respect to the nuisance parameter for the specific experiment, giving us the value which maximizes the log likelihood for a given theta and for a given experiment. After doing this, we compute points on either side of the maximum and we fit a quadratic curve through those points to obtain the values for the constants of the quadratic terms. Finally, we compute the integral using equation 3.8, and use this as the value of the log likelihood which is used by MCMC.

### 3.2.5 Alternative to marginalization

Although marginalizing over the nuisance parameters propagates the effects of uncertainty about their value into the final result [3], the process defined above is computationally expensive. Specifically, steps 1 and 2 described above require repeated calculation of the likelihood function. In step 1, minimizing the negative log likelihood function requires several calls, which in turn needs to compute the model output given specific parameter values and a nuisance parameter value. Although the number of calls can be limited, an average of 10 calls is needed to accurately find the maxima of the quadratic curve. Secondly, in step 2 we need to make additional function calls in order to determine the quadratic coefficients. Due to the fact that we are using 20 walkers, and each of them needs to marginalize over epsilon for 94 experiments, when combined, one `emcee` iteration required on average 220s to complete. As an alternative, we simplified the problem and instead only found the value of epsilon which led to the highest log likelihood for every theta and every experiment, and then compared the associated model output to the experimental data. This reduced the time taken per iteration to 45.6s per iteration, a considerable improvement from fully marginalizing. This method works under the assumption that the variance of the likelihood curves are equal.

## 3.3 Evaluation of convergence

When using MCMC techniques, before we are able to make any conclusions about the posterior distributions created from the MCMC chain, we need to ensure that the chain has converged. One popular method for quantifying MCMC convergence proposed by Gelman and Rubin [5] which analyzes the estimated Markov between-chain and within-chain variances for each model parameter with the assumption that the chains are independent. However, the chains within the `emcee` ensemble are not independent [3], thus this method cannot be used in this situation. Alternatively, Goodman & Weare [9] recommend using a measure called the integrated autocorrelation time to analyze whether a chain has reached convergence and quantify the effects of sampling error on the results. Due to the fact that samples from our chain are not independent, we must estimate the effective number of independent samples [3]. Thus to quantify the robustness of our results from MCMC, we can use the integrated autocorrelation time, since it directly quantifies the Monte Carlo error (and hence the efficiency of the sampler) on any integrals computed using the MCMC results [9]. As explained by Sokal [18], the approximated integrated autocorrelation time is defined as

$$\hat{\tau}_f(M) = 1 + 2\sum_{\tau=1}^{M} \hat{\rho}_f(\tau) \tag{3.9}$$

where $\hat{\rho}_f(\tau)$ is the approximated normalized autocorrelation function of the stochastic process that generated the chain for $f$, and $M$ is a number such that $M << N$, where $N$ is the total number of samples in the chain. Although Sokal says that the procedure works best for chains longer than $1000\tau_f$, thanks to the use of parallel chains in `emcee`, chains longer than $50\tau_f$ are often sufficient [3]. With this in mind, we will analyze whether the chain has converged by looking at the plot of the average autocorrelation time of all the parameters over the number of MCMC steps, knowing that it converges when this curve approaches an asymptote. The `emcee` library has an built-in function to determine the autocorrelation time, which we will use to analyze our chain's convergence.

# Chapter 4

# Results

## 4.1 Synthetic Data and Fixed Parameters

As a first experiment, we wanted to understand whether MCMC was a viable method for the task at hand, and to test this out, we used a synthetic dataset which was built by running one simulation using the original parameters found by Faas et al. and running `emcee` on this data. The log likelihood function we used for this experiment did not contain any interpolation or marginalization of nuisance parameters, and the synthetic data was obtained by setting all the nuisance parameters to 0, thus effectively keeping the conditions in all the experiments the same. We also simultaneously tested how using fixed values for certain parameters would affect the curves and parameter values obtained. We initially started by fixing the 4 forward rate constants, $\alpha$ and $m_\alpha$, thus reducing the parameter space of MCMC down to the 4 dissociation constants and producing the corner plot seen in Figure 4.1.

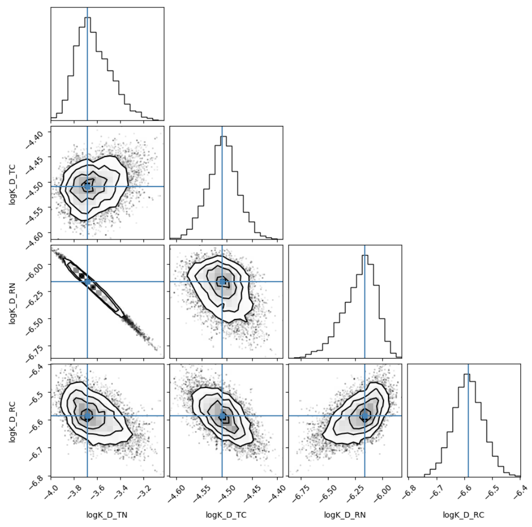


Figure 4.1: Corner plot of 4 dissociation constant parameter distributions on synthetic data

Corner plots show the one and two dimensional projections of the posterior probability distributions of our parameters, thus also showing the covariance between them. Later, we also included the forward rate constants in MCMC, thus bringing the number of parameters up to 8. By including these parameters, we can see the posterior distributions of the forward rate constants and the covariance between different parameters, instead of numerical approximations. From 4.2 we can see the corner plot produced by running `emcee` using 8 parameters on the synthetic data, while still keeping $\alpha$ and $m_\alpha$ as fixed parameters.

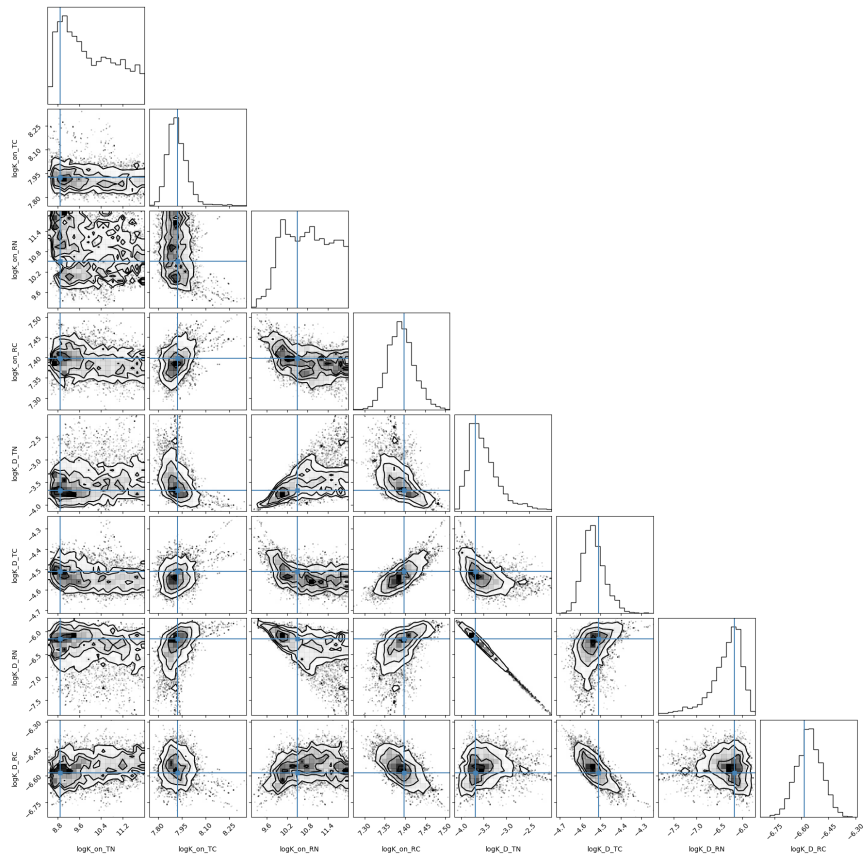


Figure 4.2: Corner plot of parameter distributions on synthetic data with 8 parameters

As we can see from the two figures above, the covariances between the dissociation constants share similar shapes, however in 4.2 we now additionally have the posterior distributions and covariances of the forward rate constants. Examining Figure 4.1, we can notice Gaussian posterior distributions with clear peaks as well as their join distributions. Similarly, The posterior distributions of the 4 dissociation constants achieved when expanding the parameter space to 8 parameters are similar to the ones achieved when fixing the 4 forward rate constants. In both plots, the blue points represent the original parameter values which were used at the beginning of the MCMC process. Naturally, when running MCMC using a synthetic dataset created by running a simulation of the model on some initial parameter values, these values will yield the highest likelihood values. Therefore it is understandable that in the plots above, the blue points coincide with the center of each joint distribution and with the peak of each individual posterior distribution since they give the highest likelihood. This however isn't the case when using experimental data as we will see later. Nonetheless, we hope that the corner plot achieved on the synthetic data can provide us with an estimate of the shapes of the projections of the posterior probability distributions.

## 4.2 Interpolation

To ensure that interpolarizing the data does not negatively impact the shape of the distributions, we ran MCMC using the modified log likelihood function with the interpolarized data, expecting to achieve corner plots similar to the figures above, yet with possibly different maximum a posteriori values. This was in fact the case, and as we can see from Figure 4.3, the shapes of the covariances and posteriors are similar to the ones observed when not using interpolation, and once again, the blue points corresponding to the original values yield the highest probabilities.

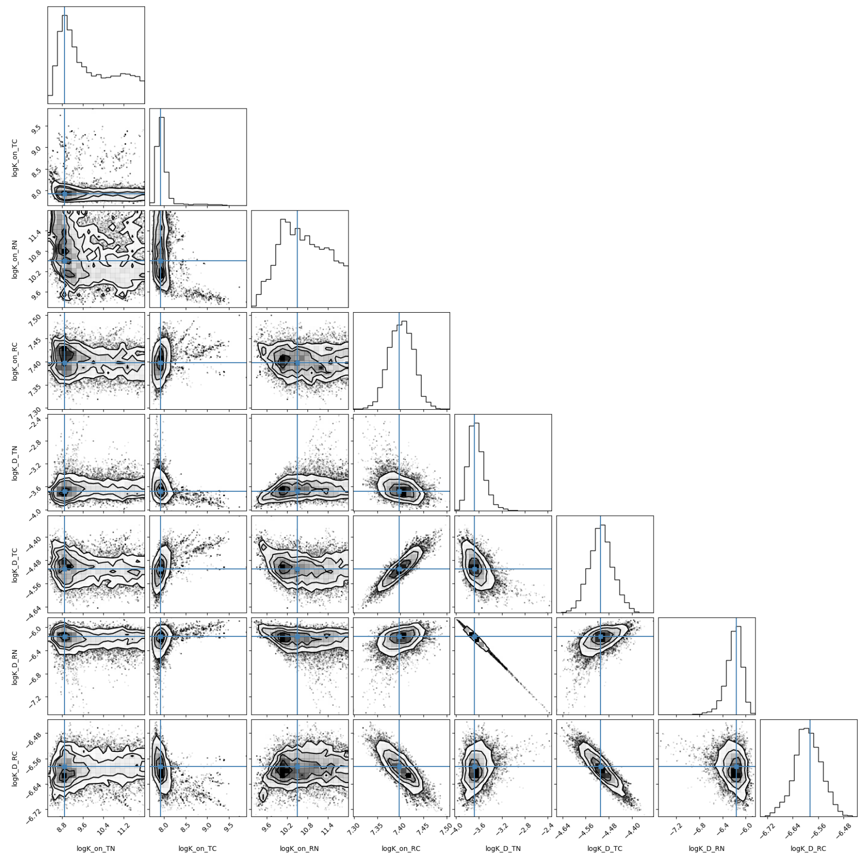


Figure 4.3: Corner plot of parameter distributions on synthetic data with 8 parameters and interpolation

## 4.3 Synthetic Data with Nuisance Parameters

The nuisance parameters represent the difference in conditions between experiments, thus since 94 experiments were originally carried out by Faas et al., we would need to find the distribution of 94 different nuisance parameters. To combat this, we propose a method to marginalize over the nuisance parameters. However, before trying this method, we wanted to understand the effect that different nuisance parameter values have the data. Until this point, we have dealt with synthetic data produced by running a simulation of the biological model once and setting nuisance parameter values to 0. To understand the effect of the nuisance parameters, we then used a synthetic dataset produced by setting the nuisance parameter values randomly within a range of -0.5 to 0.5. This introduces noise to the data, which more accurately represents the experimental data from the Faas et al. experiments. Figure 4.4 shows the corner plot achieved by using the same interpolarized log likelihood function as above on the new synthetic data.

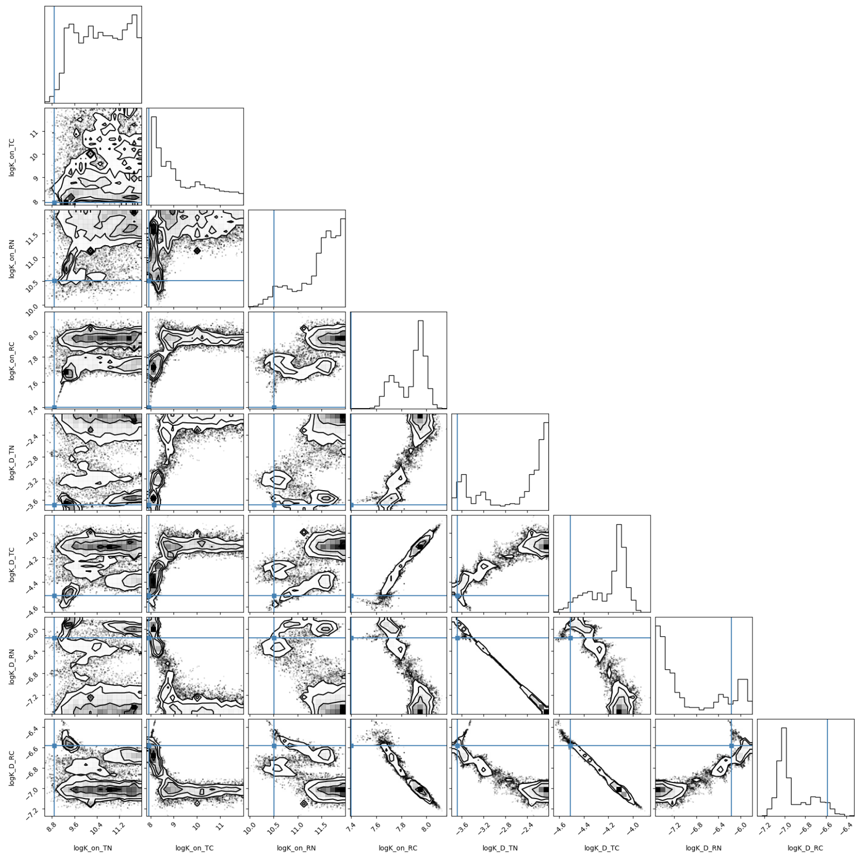


Figure 4.4: Corner plot of parameter distributions on synthetic data with noise with 8 parameters and interpolation

As we can see from Figure 4.4, the projections of the posterior distributions are less clearly defined than the ones in Figure 4.3. Furthermore, the posterior distributions now have multiple peaks, implying the possibility of bi-modal distributions or perhaps indicating that MCMC has not yet converged. Thus we now know that introducing noise in the form of adding nuisance parameters to our simulations alters the data in a way that makes it more challenging for MCMC to converge and provide a Gaussian posterior distribution for the various parameters. Although the join distributions between certain parameters, such as $\log K_{D(T),N}$ and $\log K_{D(R),N}$, have the same direction and shape as in Figure 4.3, the majority of the posterior and joint distributions are less clearly defined. Lastly, we can see that the blue points corresponding to the original parameter values used to generate the synthetic data no longer correspond to the points with the highest posterior probabilities. The results from this experiment suggest that only using 8 parameters without modelling the uncertainty introduced by the nuisance parameters is adequate on data which is consistent between experiments, however due to the fact that the nuisance parameters have a real impact on the model output, they need to be included in the likelihood computation.

## 4.4 MCMC without Nuisance Parameters

As we saw from section 3.3, running MCMC on a synthetic dataset with added noise results in worse posterior distributions for each parameter, so we expect this to be the case when MCMC is ran on the experimental data without accounting for the nuisance parameters. To compare how well our marginalized model works however, we tested using our interpolarized likelihood function with MCMC on the experimental data with all the nuisance parameters set to 0. Figure 4.5 shows the corner plot achieved after 5000 iterations of MCMC without altering nuisance parameters.

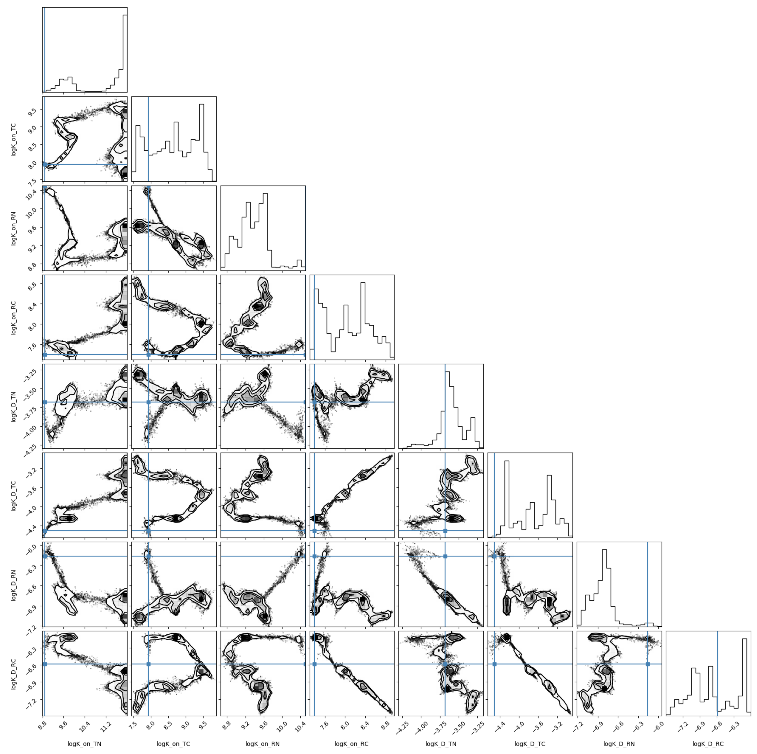
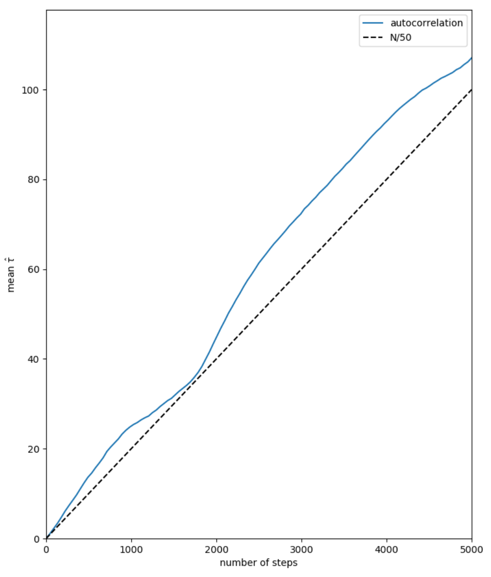


Figure 4.5: Corner plot of parameter distributions on experimental data with 8 parameters and interpolation

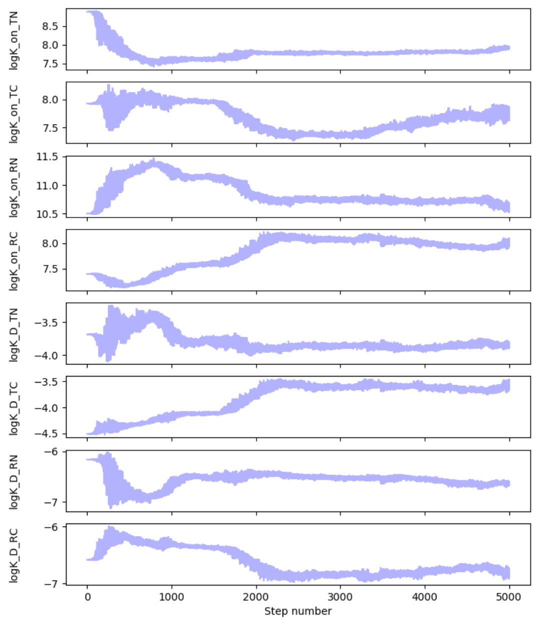
Examining this plot, we can see that most parameters don't have one clearly defined posterior distribution peak, similarly to Figure 4.4. Furthermore, none of the posterior distributions follow a Gaussian curve, and the majority of the joint distributions don't represent a real relationship between parameters. This could be attributed to the fact that the chain has not converged yet, however, due to the fact that we saw the same effect take place in section 3.3, these results can likely be attributed to the fact that the nuisance parameters have not been included.

### 4.4.1 Convergence

As mentioned in section 2.3, one way to see if our chain has reached convergence is to analyze the average autocorrelation time of our parameters. Figure 4.6a shows the average autocorrelation time of all the parameters after running 5000 iterations of the interpolarized likelihood function on the experimental data. From this, it is apparent that the chain has not yet converged due to the fact that the autocorrelation time has not reached an asymptote and the number of iterations is less than the 50 times the autocorrelation time as suggested by the documentation of `emcee`[3]. For the chain to have converged, the autocorrelation curve should have reached an asymptote and the $N > 50\tau$ dashed line should've intercepted it.



(a) Average autocorrelation time when running MCMC without nuisance parameters and with interpolation

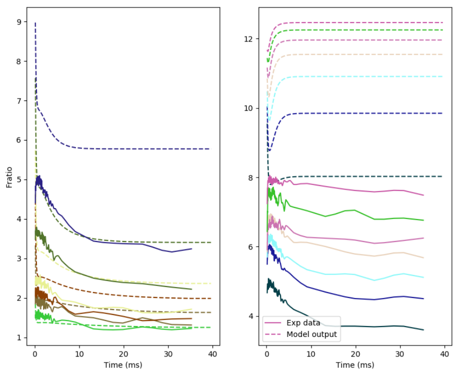


(b) Plot of parameter values during each MCMC iteration

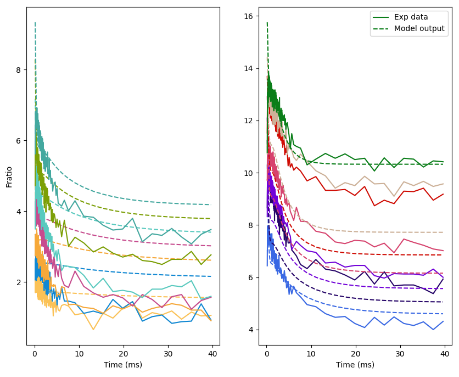
Another way in which we can verify whether the chain has converged is to look at Figure 4.6b, where we can see the raw parameter values throughout each iteration of MCMC. The traces suggest that the full space of possible values has not been explored due to the very narrow parameter values. Furthermore, the traces indicate that there there are local optima for each parameter value due to the fact that the parameter values remain in the same range of values for many iterations. This would explain the shape of the posterior distributions seen in Figure 4.5 and the existence of multiple peaks. One way to remedy this would be to increase the standard deviation passed to the likelihood function, which would allow the parameters to explore the space further.

### 4.4.2 Model Fits

To further analyze the results produced by running MCMC without using nuisance parameters, we can plot the f-ratio over time of the experimental data and the model output obtained using the Maximum a Posteriori (MAP) parameter values. Figure 4.7 can be used to form a comparison between the experimental results (solid lines) and the model output (dotted lines). As explained in the background chapter, the experimental data was separated into 7 different batches, thus we plotted experiments from within batches in the same figure. The plots of the other batches can be found in the Appendix.



(a) Batch 0



(b) Batch 4

Figure 4.7: Data and model outcomes using MAP values and without nuisance parameters

Examining Figure 4.7, it can be seen that model outputs for certain experiments is closer to the experimental data than for other experiments. This could be attributed to the fact that we haven't included nuisance parameters in our model inputs or due to the fact that, as seen earlier, the chain for this experiment had not yet converged.

## 4.5 MCMC with nuisance parameters

Similarly to Section 3.4, we will be analyzing the results achieved by running MCMC with the inclusion of nuisance parameters. In Chapter 2, I proposed marginalizing over the nuisance parameters for each iteration of MCMC, thus removing the uncertainty caused by the emission of the nuisance parameters in the calculation. However, when running MCMC on 32 cores, each iteration of MCMC took an average of 220s. It was thus unfeasible to use this as an approach to obtain the distributions of the parameters. As an alternative, on each MCMC iteration, I calculated the maxima of the log likelihood over epsilon curve and used this to compute the model output, under the assumption that the variance of the likelihood curves for all experiments are equal.

Figure 4.8 shows the corner plot achieved by including the nuisance parameters in the method described above for 5000 iterations.

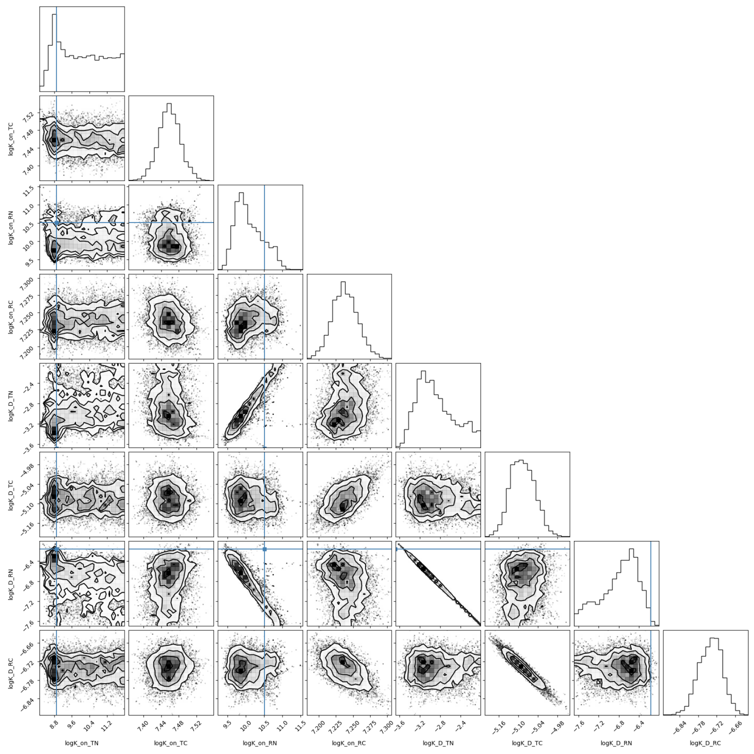


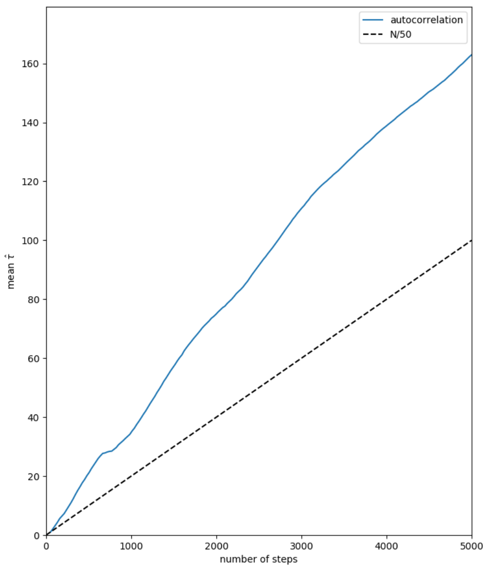
Figure 4.8: Corner plot of parameter distributions on experimental data with 8 parameters, interpolation and nuisance parameters

As we can see from Figure 4.8, the parameter posterior distributions now have the expected Gaussian shape, similarly to the corner plot 4.3 of the parameters sampled using the synthetic data. Furthermore, we can see that the joint distributions of the parameters also follow a traditional Gaussian contour shape, and once again, the joint distributions between the parameters have similar shapes to the ones obtained in Figure 4.3. One commonality that can be observed from Figures 4.3, 4.4 and 4.8 is that the joint distribution between the $\log K_{on(T),N}$ parameter and the rest of the parameters seems to not be greatly affected by the its value. This can be explained by the fact the parameter, which is a forward rate constant, is rate limiting, meaning that it is the slowest step of a chemical reaction [15]. Due to this, any values of this constant which occur after a certain time period will not have an effect on other parameters. This phenomenon explains the invariant joint distribution produced between the $\log K_{on(T),N}$ parameter and the other parameters.

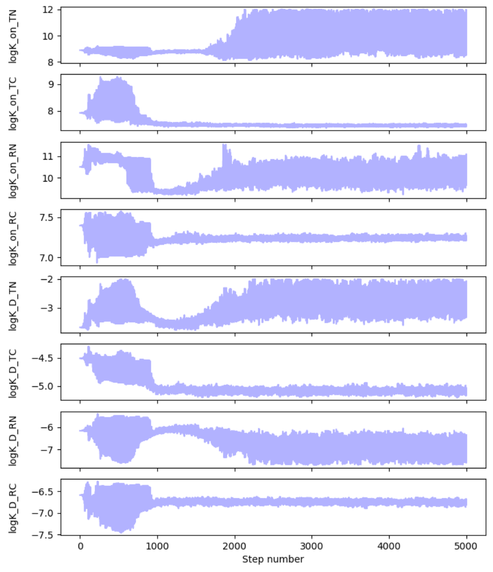
### 4.5.1 Convergence

Once again, we can analyze the convergence of the MCMC chain by examining the autocorrelation time plot throughout the various steps of MCMC. From Figure 4.9a we can see that once again, the average autocorrelation time from the parameters after

5000 iterations is still higher than 50 times the theoretical autocorrelation time shown by the dotted line. Differently from Figure 4.9b, Figure 4.9b shows that certain parameters seem to have already converged to a specific value after having explored a larger range of parameter values, such as the $\log K_{on(T),C}$ parameter. Contrarily, parameters such as $\log K_{on(T),N}$ seem to be stuck in a local maxima for some time before exploring a larger parameter space. One possible solution to this issue, which was also encountered in Figure 4.6b would be to include the standard deviation which is passed to the likelihood function as one of the parameters which we sample over. This would allow for a wider range of parameter values, and it would alter the curvature of the quadratic curves described in Section 2.2.4, which would potentially lead to faster convergence and potentially better model fits.



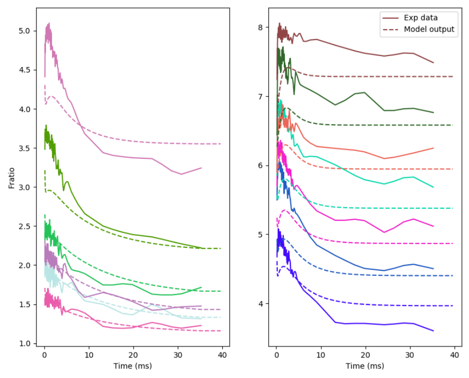
(a) Average autocorrelation time when running MCMC with nuisance parameters and with interpolation



(b) Plot of parameter values during each MCMC iteration

## 4.5.2 Model Fits

Once again, we can examine the model outputs obtained from the MAP parameter values and using nuisance parameters and how they compare to the experimental data by plotting the f-ratio over time. From Figure 4.10 it is clear to see that the model outputs more closely resemble the experimental data when using the MAP parameter values and nuisance parameters. This can especially be noticed in the experiments belonging to batch 0, where previously, the model outputs were much higher than the experimental data. Furthermore, the fit during the initial stages of the reaction seem to be more accurately fit when using the nuisance parameters, since the shape of the curves is similar to those of the experimental data.


(a) Batch 0

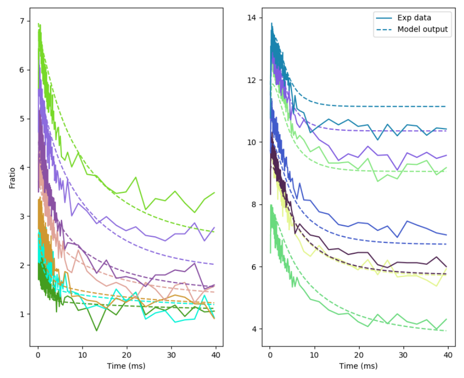

(b) Batch 4

Figure 4.10: Data and model outcomes using MAP values and with nuisance parameters

## 4.6 Summary of results

To summarize the results, Table 4.1 shows the parameter values found by Faas et al. in their experiment, as well as their MAP estimates from out two methods, one where we do not include nuisance parameters and one where we do.

| Parameter | Unit | Faas et al. estimates | MAP estimates | |
|---|---|---|---|---|
| | | | Without nuisance parameters | With nuisance parameters |
| $\log k_{on(T),N}$ | logM | 8.89 | 7.78 | 9.09 |
| $\log k_{on(T),C}$ | logM | 7.92 | 7.66 | 7.47 |
| $\log k_{on(R),N}$ | logM | 10.50 | 10.76 | 9.98 |
| $\log k_{on(R),C}$ | logM | 7.40 | 7.94 | 7.24 |
| $\log K_{D(T),N}$ | logM | -3.68 | -3.85 | -3.06 |
| $\log K_{D(T),C}$ | logM | -4.51 | -3.65 | -5.07 |
| $\log K_{D(R),N}$ | logM | -6.16 | 6.53 | -6.60 |
| $\log K_{D(R),C}$ | logM | -6.59 | -6.74 | -6.74 |
| $\log m_\alpha$ | logM | 1.10E-03 | 1.10E-03 | 1.10E-03 |
| $\log \alpha_0$ | logM | -3.90E-01 | -3.90E-01 | -3.90E-01 |

Table 4.1: MAP estimates from MCMC and the parameter values found by Faas et al.

As we can see from Table 4.1, despite the use of nuisance parameters, the MAP estimates vary from the Faas et al., yet this is to be expected. Due to the fact that we are making certain assumptions throughout the likelihood calculation process, such as assuming independence between time points and between experiments, the MAP estimates are expected to differ from the Faas et al. estimates. Nonetheless, using nuisance parameters produces estimates which are closer to the Faas et al. approximations, which explains the superior model output seen in Figure 4.9b. To more objectively quantify the performance of our methods and the parameter estimates, we analyzed the average mean squared error (MSE) between the model output and the experimental data for each method. Table 4.2 shows the MSE values achieved by using the Faas et al. values and then using the MAP estimates from Table 4.1.

| | | Batch | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| MSE | Faas et al. | 13.08 | 1.42 | 3.35 | 2.07 | 1.14 | 5.45 | 19.75 |
| | Without nuisance params | 9.97 | 2.04 | 3.49 | 3.24 | 1.46 | 4.63 | 19.23 |
| | With nuisance params | 0.38 | 0.11 | 1.06 | 0.38 | 0.24 | 0.50 | 1.53 |

Table 4.2: MSE values of model output vs experimental data from using different methods

As we can clearly see from the results, the method which includes the nuisance parameters performs much better than when only using the Faas et al. values or when not including the nuisance parameters in the likelihood computation. In every batch, the parameter estimates achieved when using nuisance parameters outperforms the model fits generated using the Faas et al. parameters. The results from this table show that Faas et al. might not have found the best parameter estimates due to the fact that their MSE values are higher than both estimates we achieved using MCMC.

# Chapter 5

# Conclusion and future work

## 5.1 Conclusion

In this paper we explored the use of an MCMC method implemented in `emcee` to obtain the posterior distribution of the kinetic rate parameters used to model the interactions between $Ca^{2+}$ and CaM described by Faas et al. We analyzed whether this method is feasible by running an MCMC chain on synthetic data, and then later explored the effects of the nuisance parameters on the model outputs. Due to the limitations of `emcee`, we attempted marginalizing over the nuisance parameters and then later ran an MCMC chain using the nuisance parameters which resulted in the highest likelihoods. Furthermore, we tried to improve the parameter estimates by interpolating the data and thus evenly distributing the data over the full time course of the experiment.

The primary initial goal of the project was to obtain interpretable posterior distributions by using the `emcee` library and thus show that this method is applicable in systems biology, overcoming the challenges posed in such models. Firstly, we tested using MCMC as a method for parameter estimation on a synthetic dataset achieved by running one simulation of the model and using this data in the likelihood function. From this we saw that MCMC is a viable method to use, since the results from Figure 4.2 showed that the distributions of the parameters was Gaussian. We then interpolated the data during the likelihood calculation to give more weight to the points in the later stages of the experiment, and we saw that this did not negatively impact the shapes of the posterior distributions previously observed. We then tested the effect of the nuisance parameters on the parameter distributions, and noticed that when these are included in the creation of the synthetic data, the distributions are no longer Gaussian, thus proving the requirement for us to include nuisance parameters in MCMC. After running MCMC with and without nuisance parameters, we saw that the results achieved from using them are largely better than than when not using them. This was quantified by the much lower MSE values when using this method. Furthermore, despite the fact that our chains had not yet converged, we found that the parameter estimates from using MCMC gave improved model outputs as opposed to the values found by Faas et al. Our findings reinforce the parameter estimates found by Faas et al. and our method can be generalized and be used in similar model which model the interaction between

$Ca^{2+}$ and CaM, such as by Pepke (2010)[17].

However, as we previously mentioned, the MCMC chains did not converge, thus our estimates aren't to be fully trusted. One issue with the MCMC approach when using nuisance parameters is the large computational cost. We ran `emcee` on 32 cores, and to integrate over the log likelihood curve produced by varying the nuisance parameter value for each experiment, each iteration required an average of 220s to complete. We would need to run MCMC for thousands of iterations in order for it to converge, thus taking days achieve trustworthy parameter estimates. Although we were able to slightly shorten the time taken per iteration by fixing certain parameters to the specific values found by Faas et al., this did not show a significant improvement. Therefore, more work could be carried out to find how to optimize the MCMC convergence and each calculation of the log likelihood. Throughout the experiments, we kept the standard deviation constant so that the evaluations between methods could be trusted, however, another improvement which would be made to our method is to include the standard deviation as a parameter to be sampled over. This would allow for parameters to explore more values in the parameter space and possibly converge quicker, or perhaps give better model fits.

## 5.2 Future work

### 5.2.1 MCMC in high dimensions

One of the issues encountered throughout this project and Candel's work is the problem of convergence of the MCMC chain. In our experiments, after running the chain for 5,000 iterations, it was clear from Figure 4.9a that the chain had not converged. Furthermore, another challenge encountered when trying to marginalize over the parameters is the time taken per iteration. The way we computed the likelihood was to calculate many log likelihood values for varying values of epsilon, which vastly increases the computational costs due to the large number of function calls made to the ODE solver. Although we are not interested in the distribution of the nuisance parameters, we sampling over them would prove to be more efficient if we were able to perform MCMC in high dimensions. To help with this, some methods for MCMC in high dimensions exist, including Hamiltonian Monte Carlo.

Since the `emcee` package uses a version of Metropolis-Hasting algorithm, Hamiltonian Monte Carlo is a suitable method since it is also a Metropolis method applicable to continuous state spaces, which uses gradient information to reduce the random walk behavior of the normal Metropolis method. In the Hamiltonian Monte Carlo method, the state space $\theta$ is augmented by momentum variables $\mathbf{p}$, and there is an alternation of two types of proposal. The first randomizes the momentum variable, leaving the state $\theta$ unchanged, while the second changes both $\theta$ and $\mathbf{p}$ using simulated Hamiltonian dynamics, defined as [12]:

$$H(\vec{\theta}, p) = E(\vec{\theta}) + K(p) \tag{5.1}$$

where $K(p)$ is a kinetic energy function. These two proposals create samples from the joint density:

$$P_H(\vec{\theta}, p) = \frac{1}{Z_H} exp[-H(\vec{\theta}, p)] = \frac{1}{Z_H} exp[-E(\vec{\theta})] exp[-K(p)] \tag{5.2}$$

The marginal distribution of  is the desired distribution $exp[-E(\vec{\theta})]/Z$, and if we disregard the momentum variables, we can obtain a sequence of samples that come from the unknown distribution $P(\vec{\theta})$ [12]. This approach would not only improve the time to converge thanks to the momentum variable, but would also allow for a higher dimensional parameter space. Implementing this method would thus further generalize the use of MCMC methods in systems biology in models with many parameters.

# Bibliography

[1] Judith Borowski. Sloppy parameters in a synaptic model. Master's thesis, School of Informatics, University of Edinburgh, 2017.

[2] Carine Candel. Learning systems biology models from data. Master's thesis, School of Informatics, University of Edinburgh, 2019.

[3] Dustin Lang Jonathan Goodman. Daniel Foreman-Mackey, David W Hogg. emcee: the MCMC hammer. *Publications of the Astronomical Society of the Pacific*, 2013.

[4] Bryan Daniels, YJ Chen, JP Sethna, Ryan Gutenkunst, and Chris Myers. Sloppiness, robustness, and evolvability in systems biology. *Current opinion in biotechnology*, 19, July 2008.

[5] Andrew Gelman and Donald B Rubin. A single series from the gibbs sampler provides a false sense of security. *Bayesian statistics*, 4:625–631, 1992.

[6] John E Lisman Istvan Mody Guido C Faas, Sridhar Raghavachari. Calmodulin as a direct detector of Ca2+ signals. *Nature Neuroscience*, 2011.

[7] Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLOS Computational Biology*, 3(10):1–8, 10 2007.

[8] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. Copasi—a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[9] Jonathan Weare Jonathan Goodman. Ensemble samplers with affine in-variance. *Communications in applied mathematics and computational science*, 2010.

[10] Yoshihisa Kubota, John A Putkey, Harel Z Shouval, and M Neal Waxham. Iq-motif proteins influence intracellular free Ca2+ in hippocampal neurons through their interactions with calmodulin. *Journal of neurophysiology*, 99(1):264–276, 01 2008.

[11] Yoshihisa Kubota, John A Putkey, and M Neal Waxham. Neurogranin controls the spatiotemporal pattern of postsynaptic Ca2+/CaM signaling. *Biophysical journal*, 93(11):3848–3859, 12 2007.

[12] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[13] Thomas Maiwald and Jens Timmer. Dynamical modeling and multi-experiment fitting with potterswheel. *Bioinformatics*, 24(18):2037–2043, 2008.

[14] Ami Citri Robert C Malenka. Synaptic plasticity: Multiple forms, functions, and mechanisms. *Nature*, 2008.

[15] Galaxy Mudda, Pamela Chaha, Florence-Damilola Odufalu, and Filmon Tewolde. 3.2.3: Rate determining step, Sep 2020.

[16] Beat Schwaller Isabel Llano Marco R. Celio Alain Marty Olivier Caillard, Herman Moreno. Role of the calcium-binding protein parvalbumin in short-term synaptic plasticity. *Proceedings of the National Academy of Sciences*, 2000.

[17] Shirley Pepke, Tamara Kinzer-Ursem, Stefan Mihalas, and Mary B. Kennedy. A dynamic model of interactions of Ca2+, calmodulin, and catalytic subunits of Ca2+/calmodulin-dependent protein kinase II. *PLOS Computational Biology*, 6(2):1–15, 02 2010.

[18] A. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms note to the reader. 1996.

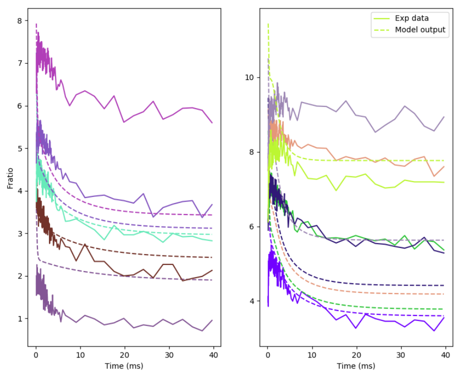
[19] Gloria I. Valderrama-Bahamondez and Holger Frohlich. Mcmc techniques for parameter estimation of ode based models in systems biology. *Frontiers in Applied Mathematics and Statistics*, 5:55, 2019.

[20] Zhengui Xia and Daniel R. Storm. The role of calmodulin as a signal integrator for synaptic plasticity. *Nature Reviews Neuroscience*, 6(4):267–276, 2005.

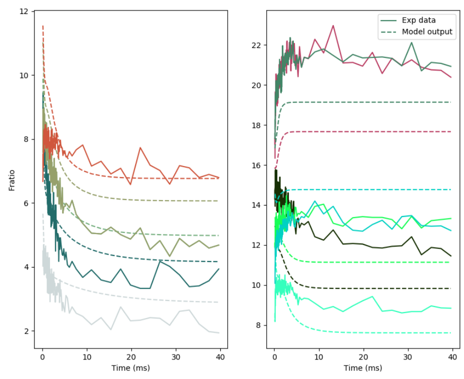
[21] Robert S Zucker. Calcium and activity-dependent synaptic plasticity. *Current Opinion in Neurobiology*, 1999.

# Appendix A

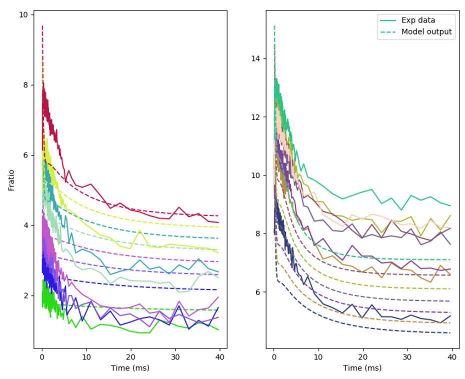
# Experimental data and model outcomes using MAP estimate



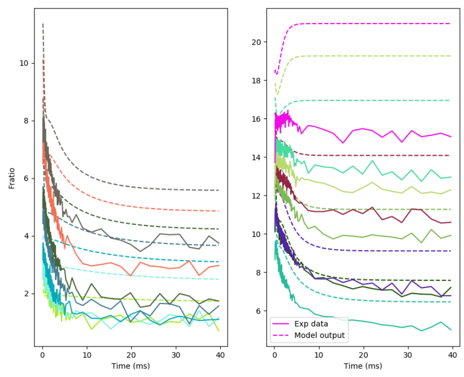
(a) Batch 1



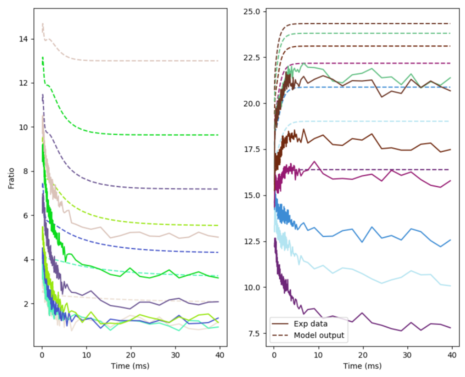
(b) Batch 2



(c) Batch 3

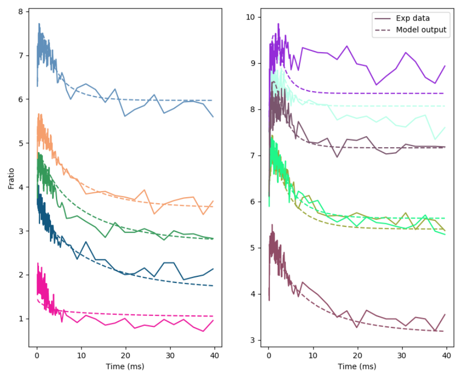


(d) Batch 5

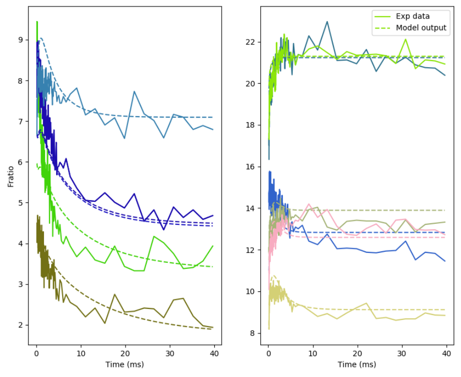


(e) Batch 6

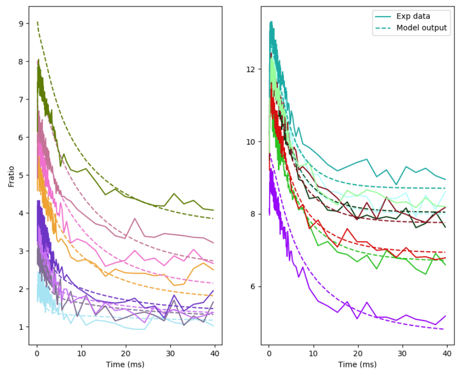
Figure A.1: Data and model outcomes using MAP values and without nuisance parameters



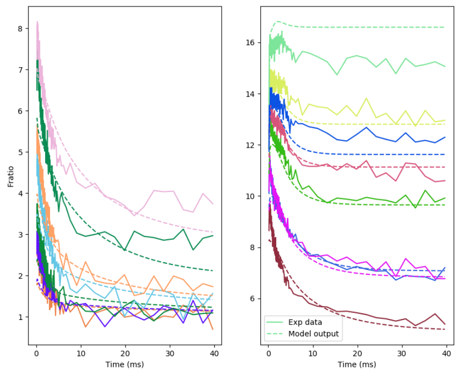
(a) Batch 1



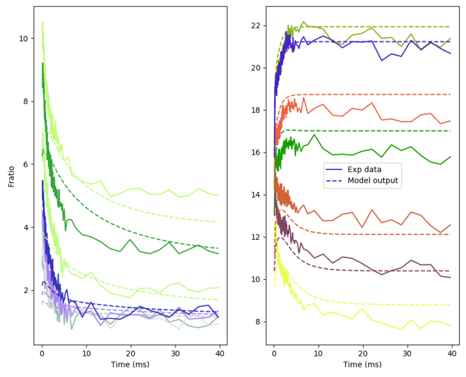
(b) Batch 2



(c) Batch 3

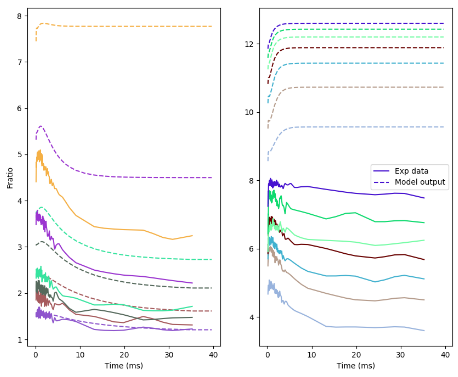


(d) Batch 5

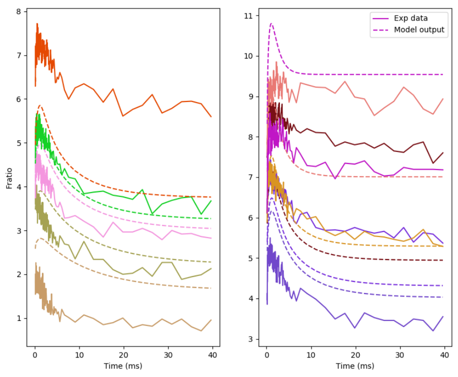


(e) Batch 6

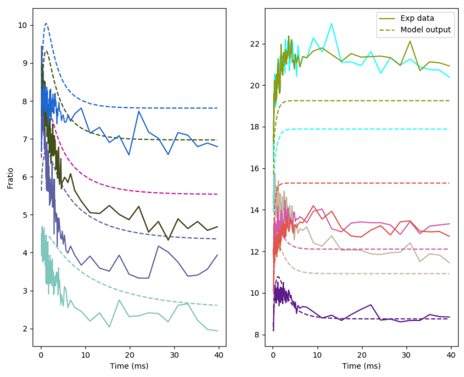
Figure A.2: Data and model outcomes using MAP values and with nuisance parameters



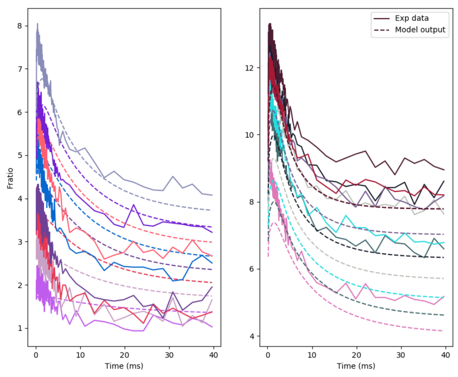
(a) Batch 0



(b) Batch 1

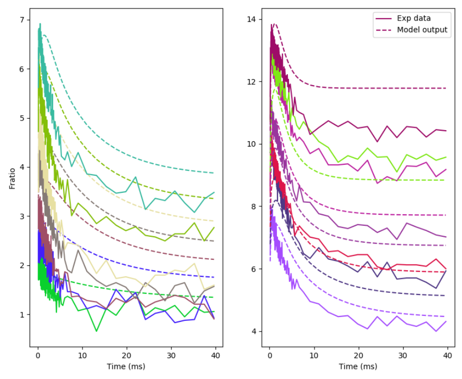


(c) Batch 2

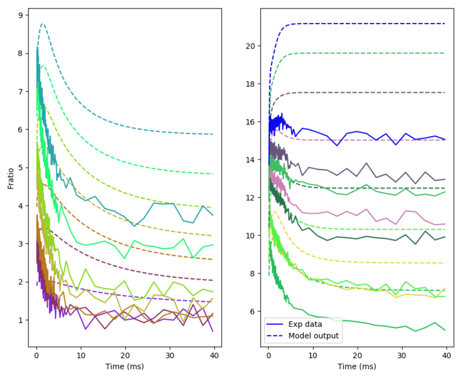


(d) Batch 3

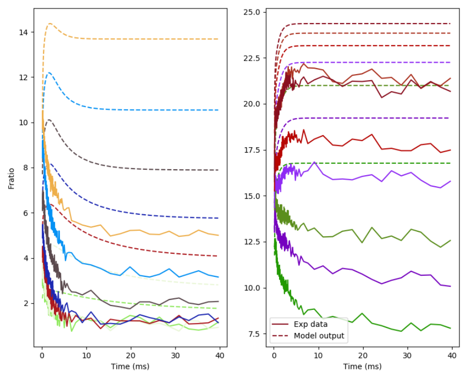
Figure A.3: Data and model outcomes using Faas et al.'s parameter estimates



(e) Batch 4



(f) Batch 5



(g) Batch 6

Figure A.3: Data and model outcomes using Faas et al.'s parameter estimates