Unsupervised methods for large-scale cell-resolution neural data analysis

Gergő Bohner

A dissertation submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Gatsby Computational Neuroscience Unit

Department of Life Sciences

University College London

June 8, 2019

I, Gergő Bohner, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In order to keep up with the volume of data, as well as the complexity of experiments and models in modern neuroscience, we need scalable and principled analytic programmes that take into account the scientific goals and the challenges of biological experiments.

This work focuses on algorithms that tackle problems throughout the whole data analysis process. I first investigate how to best transform two-photon calcium imaging microscopy recordings – sets of contiguous images – into an easier-to-analyse matrix containing time courses of individual neurons. For this I first estimate how the true fluorescence signal gets transformed by tissue artefacts and the microscope setup, by learning the parameters of a realistic physical model from recorded data. Next, I describe how individual neural cell bodies may be segmented from the images, based on a cost function tailored to neural characteristics. Finally, I describe an interpretable non-linear dynamical model of neural population activity, which provides immediate scientific insight into complex system behaviour, and may spawn a new way of investigating stochastic non-linear dynamical systems.

I hope the algorithms described here will not only be integrated into analytic pipelines of neural recordings, but also point out that algorithmic design should be informed by communication with the broader community, understanding and tackling the challenges inherent in experimental biological science.

Acknowledgements

Acknowledge all the things!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Algorithms for neuroscience

.

Discuss why we need algorithms more and more, but why do we need to be careful about developing algorithms for other people (who might not completely understand how they work, just apply them).

1.1.1 The benefits and challenges of unsupervised algorithms

Human biases, incorporating structured knowledge (priors), or unstructured knowledge (human markers, supervision signal), issues with evaluation

1.1.2 Algorithms from raw data to model fitting

Understanding the all modification of raw signal has to be properly understood AND reported.

Physics level characterisation vs black box characterisation of raw data transformation. Unsupervised algorithms can provide a good inbetween, via informed parametric forms, and priors on the parameters, but MAP / ML estimates based on the observed data (and available measurements, put in as strong priors).

Chapter 2

Spatial background equalisation of two-photon calcium imaging videos of neural tissue

In this chapter I examine the popular two-photon video-rate scanning calcium microscopy, and construct an in silico processing algorithm, that aims to recover the signal we were setting out to measure in the first place - the changes in calcium ion concentration, which in turn is thought of as a proxy for the electrical activity of neurons. Unfortunately the recovery of the absolute signal is impossible given the usual experimental setup, but what we may achieve is that we can estimate the true optical signal captured from the biological sample, and then use this signal to infer the calcium level from a fluorescent signal that was likely distorted by uneven illumination, varying expression levels of a fluorescent calcium indicator, and other factors. Ultimately we aim to infer a signal that only depends on actual changes of calcium concentration, free of all spatial distortions caused by experimental method. The current need for such an algorithm will hopefully become clear enough through the detailed description of the issue, as well as the *illuminating* example images, and although at the end of the chapter I discuss experimental modifications that would reduce the future usefulness of current work, this should still benefit all the vast collections of past data.

I'll guide the reader through the thought process in the following steps. 1. I describe the distorted raw data and discuss the possible causes of how the underlying signal we wished to observe got transformed into the data through the experimental procedures. 2. Next, I build a flexible probabilistic inference framework that is capable of both including common parametric models of the experimental procedure, and describing previously unknown or unmodeled distortions supported by the data. 3. I test the framework on openly available experimental data. 4. Finally, based on this study of the system, I point out small, easy to implement changes in the experimental procedures that would lead to data more faithfully describing the signal we intended to measure in the first place.

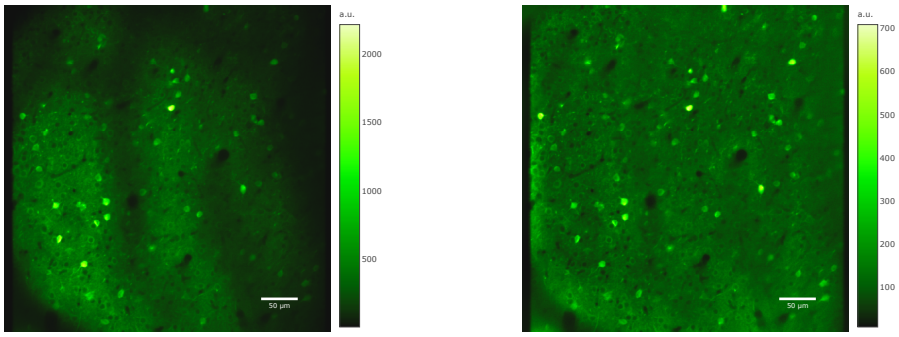


Figure 2.1: Original (left) and corrected (right) mean images

2.1 Calcium imaging - from signal to data

Before discussing the how, it is always important to make the *what* very clear. The reason for the widespread use of two-photon video-rate scanning calcium imaging microscopy - which I will often call calcium imaging in the rest of this chapter - is that it provides an estimate of neuronal activity at the single neuron level, in vivo, in alive, and often awake animals. This activity is measured via the increase of Ca^{2+} ion concentration that stereotypically follows electrical spikes in neuronal somas. The animals' cells are reprogrammed (either genetically or by an injected viral phage) to produce a protein that acts as fluorescent calcium sensor. Fluorescence is the ability of the protein to absorb photons at a certain wavelength, enter a higher energy state and later emit a photon of a different wavelength; and calcium sensing means that the efficacy of this process is modulated by whether or not the protein is bound to calcium.

For us this practically condenses down to the facts that we are distorting our *signal of interest* – which we will now define as *the concentration of calcium ions in a small volume at a given time* – in a spatially variable manner, as the number of calcium sensors vary vastly over space (we will ignore time-variation as it is a slower process). Furthermore, we can only estimate the current state of this distorted process by illuminating it with incident light, then attempt to capture the emitted photons.

After decades of still ongoing optimisation of both the fluorescent calcium sensor protein (Whitaker 2010)(Chen et al. 2013) and the active imaging process, in the past few years the neuroscience community has largely converged on using a GCaMP6 sensor - an ultrafast rise and decay time constant variant, based on green fluorescent protein and calmodulin - illuminated by a scanning two-photon microscope and the emitted fluorescent signal photons are collected in the epi direction through the objective, then redirected to the a photomultiplier tube via a dichroic mirror. Finally this signal is digitised, and saved as a grey value of an image pixel whose location is determined based on the angle of the illumination light path's scanning mirrors. The collection of these pixels over one cycle of mirror movement forms an image, and repeated cycles result in a set of images - a video.

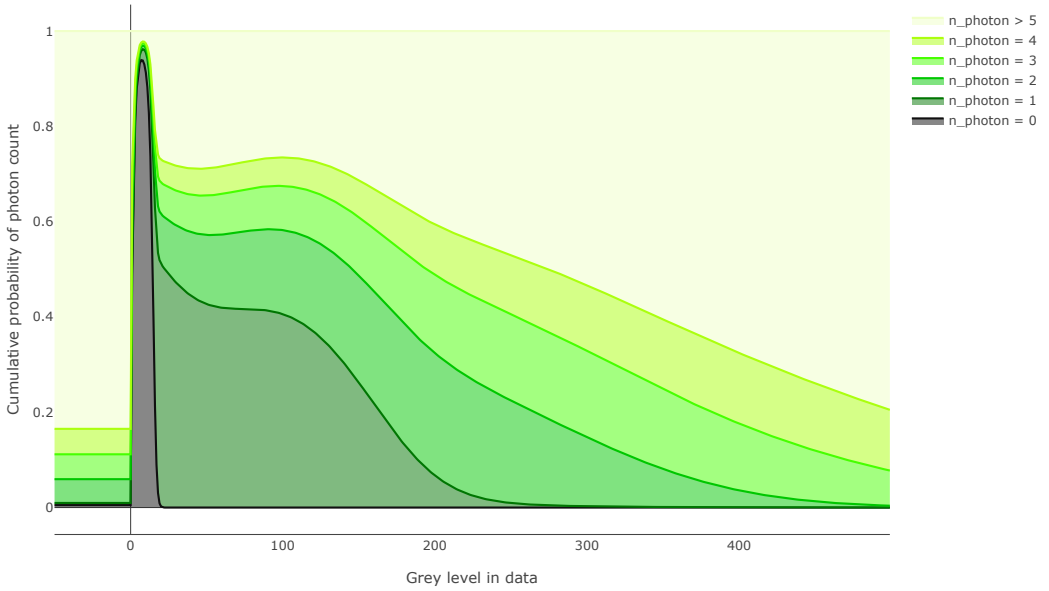


Figure 2.2: Demonstrate the relative probability of the number of photons that reached the photomultiplier and generated a photoelectron; given we have observed a grey level in the raw data. Note that although the values less than 0 on the x axis serve purely visualisation purposes as the observations are strictly non-negative, this zero level is indeed a modifiable property of the observation system.

This established process has generated thousands of hours of video and correspondingly thousands of terabytes of data over the past few years, many of them exhibiting similar features in the recorded data. As shown in figure 2.1, the data often exhibits strong background spatial non-uniformity, which is mostly caused by non-uniform distribution of the calcium sensors, uneven illumination intensities and collection efficiencies that depend on the position within the image. An other effect in the collected data is more subtle, but causes substantial difficulty in interpreting the data. This relates to what happens when the emitted fluorescent signal photons arrive at the photomultiplier. As demonstrated in figure 2.2 the relationship between observed data and the signal is non-trivial. Modelling a realistic photomultiplier is a difficult problem; what's worse, here we are faced with reverse-engineering one from data, as metadata about microscope specifications and settings are rarely shipped alongside published datasets. In order to attempt it, we need to clearly understand the individual parts of the system, and how they collectively result in the observed data. (Cite: `https://petebankhead.gitbooks.io/imagej-intro/` - This is the full book, quite good)

2.1.1 Scanning two-photon microscopy

In this section I describe, how the illumination and collection processes result in the observed signal, given a latent true state of the calcium-bound fluorescent sensors. In order to estimate this state, one needs to probe the sensors via illumination, then collect the emitted fluorescent photons (fig-

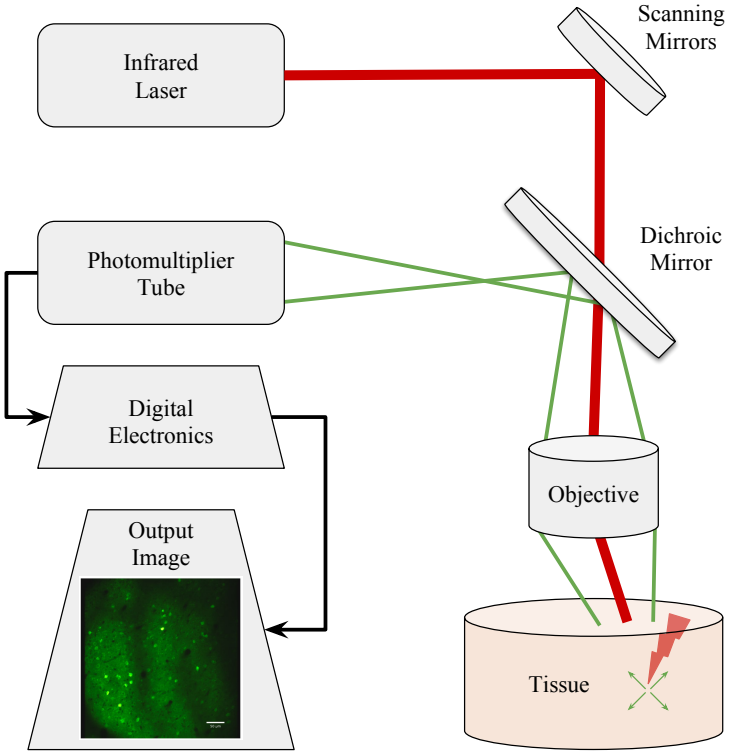


Figure 2.3: Scanning two photon microscopy schematic. The red light path indicates focused femto-second laser pulses from the illuminating laser. The green cones indicate the collection pathway of the fluorescently emitted highly scattered signal photons.

ure 2.3). Here I am giving a brief introduction and focus on the elements that might introduce spatial aberrations. For a detailed technical description of individual optical elements see Young et al. 2015.

First the illumination pathway. The main elements we need to consider are the illuminating laser, the pair of scanning mirrors, the objective, and the tissue under examination. The infrared laser emits femtosecond pulses of high wavelength light. These light packets are deflected from the optical axis by the scanning mirrors, then focused by a high numerical aperture objective, ultimately creating a tight focal point that moves within the tissue; this is deemed scanning. In order to elicit fluorescent photons, two infrared photons need to excite the same sensor molecule concurrently, explaining the name two-photon microscopy. This results in a much tighter effective point spread function around the focal spot, as the probability of evoking response scales quadratically with the field intensity, enabling two-photon microscopes to generate very little out-of-focus fluorescent signal. However, this quadratic dependence also causes small changes in the spatio-temporal field strength around focal spot to be much more pronounced in the evoked signal.

Here I am interested in discovering spatial patterns in the changes of total effective illumination. There are three main causes of spatial aberrations. The first is introduced by the scanning mirrors. In order to achieve bidirectional deflection of the focal spot, two mirrors are required, one usually implementing fast X-directional (resonant) scanning, and another the slower orthogonal Y-

directional one. As mentioned by major manufacturers (e.g. Olympus[1], Leica[2]), ideally both mirrors should be in a plane conjugate to the objective lens's back pupil, such that the laser beam can be optimally filled at all times using a so-called "4 f design", including a telescope and scan lens. For two mirrors, this is physically impossible, and compromises need to be made, usually resulting in the conjugate plane being in-between the two scan mirrors, and causing gradual loss of intensity as the scan angle increases and the focal spot moves further from the optical axis. If the mirrors are equidistantly placed from the conjugate plane, this type of aberration is radially symmetric around the optical axis, sometimes called vignetting; if the distances differ, we would notice decreasing field strength that differ about the X and the Y axes, called astigmatism. These effects are present in most datasets, due to being difficult to completely compensate for.

The second major cause for spatial aberrations is introduced by the objective itself. Although this is generally the optical element the most attention (and money) is paid towards, objective lenses are still not perfect. The X-Y tightness of the focus often widens as the focus moves away from the optical axis of the objective. In conventional microscopy this effect causes minor aberrations, but due to our quadratic dependence on field intensity in two-photon microscopy, this spatial dispersion results in an effective reduction of illumination in a radially symmetric manner about the optical axis.

Finally the tissue above the focal plane itself attenuates the focused laser beam in spatially unstructured way, due to the varying density of brain matter, in particular due to the presence of vasculature.

Once a small volume has been illuminated, the available fluorophores may get excited and subsequently emit a fluorescent photon. For fluorescent calcium sensors, the probability of such an absorption-emission event is modulated by whether or not the sensor molecule is bound to calcium, and thus serves as a proxy of the local calcium concentration. The generated fluorescent photon count is therefore depends both on the number of sensors as well as calcium ions available. The way we can distinguish between signal caused by excess sensor count as opposed to more calcium, is that calcium concentration varies rapidly in time, whereas the active sensor counts are essentially constant throughout the imaging for modern non-bleaching sensor molecules. Nevertheless, the sensor molecule availability introduces another spatially unstructured effect.

The emitted fluorescent photons then need to be collected (figure 2.3). Fluorophores emit photons uniformly with all directions, which are then scattered in the tissue. As we typically only collect photons in the epi direction through the objective, most of them are lost. Fortunately even when a large portion of the events are uniformly randomly lost, Poisson processes such as this photon emission do not change in nature, they only become a process with a correspondingly lower

[1] URL: `https://www.olympus-lifescience.com/en/microscope-resource/primer/techniques/confocal/confocalscanningsystems/`

[2] URL: `https://www.leica-microsystems.com/products/confocal-microscopes/technology/fov-scanner/`

intensity. This phenomenon is called thinning. Therefore here we only need to think of relative spatial variations, as the absolute photon loss is both difficult and unnecessary to estimate. These spatial variations in Poisson thinning during photon collection may again results from tissue density changes (although to a much smaller extent, as scattered photons are less affected than the illuminating focused beams), from the radially symmetric reduction of the solid angle of the objective, as well as the incidence location- and angle-dependence of optical pipeline elements' reflectivity and the photomultiplier's quantum efficiency.

2.1.2 Recording the optical signal

The last stage of the data creation process is the digitalisation of the signal that entered the photomultiplier tube (PMT). First, each photoelectron generated at a photocathode gets accelerated by a focusing electrode onto a successive chain of dynodes that exponentially amplify the signal via a cascade of secondary emissions. Finally the generated electron avalanche hits the anode, and generates a (temporally) sharp current pulse. The height of the pulse itself is highly variable due to the noisy amplification process. The distribution of pulse heights itself may depend upon the location and angle of the photon incident on the photocathode[3]. There are two general ways of digitising the current that arrives at the anode. For very low photon counts, the individual current pulses are unlikely to overlap in time, and therefore it is possible to extract each event that passes a threshold, and obtain an integer number. However, for moderate photon counts typical in the neuroscience video-rate imaging, generally the current pulses are preamplified with a given input offset (that thresholds DC components and low pulses) and then integrated in a time window that represents the signal photons arriving from the same location in the sample. This integration time window (with appropriate delays) is coupled with the scanning movement of the focal point in the sample, and thus the integrated and digitised signal appears as the corresponding pixel's grey level in the raw output data.

Unfortunately the noisy amplification carried out by the photomultiplier along with the thresholded integration by the subsequent electronics turn the incoming Poisson signal into the digital grey level, that possesses an arbitrary zero level alongside an effective photomultiplier gain that determines the increase in pixel intensity for each added incident photon. Furthermore, both the preamplifier offset that determines the zero level, as well as the photomultiplier voltage that determines the gain are easily accessible to the experimenters to manually maximise the apparent signal to noise ratio, and ultimately they may change not only from microscope to microscope, but even from recording to recording, making the interpretation of raw data difficult.

These are two conceptually different stages, as all processes before the photo-multiplication process affect the mean parameter of a Poisson process, but do not change the nature of it, and

[3]For an extremely detailed description of characteristics and noise sources of photomultipliers, see the premier producer Hamamatsu's excellent PMT Handbook (edition 3), URL: `https://www.hamamatsu.com/resources/pdf/etd/PMT_handbook_v3aE.pdf`

therefore maybe thought of as a purely multiplicative gain-like process applied to the calcium concentration changes. Conversely, the signal transformation carried out by the photomultiplier and subsequent electronics is a process that amplifies all entered photons. This important difference may be best summarised by saying that the pre-photomultiplier gain process changes the mean of the signal, but keeps the variance-mean ratio (the Fano factor) one; whereas the recording process carried out by the photomultiplier and electronics keeps the standard deviation-mean ratio (coefficient of variation) constant, but changes the Fano factor.

Ultimately we need to first infer the optical signal that entered the PMT by inverting the recording process, then strip it of any spatial dependence that is not caused by fast timescale changes in calcium concentration. This process may be called spatial signal equalisation.

2.2 Spatial signal equalisation via probabilistic inference

Having understood the underlying system, we need to translate it into a mathematical model, that is capable of capturing the complexities of what's known, has a reasonable behaviour when faced with the unknown, and is scalable to the tens of millions of observations that are both available and required to elucidate the latent signal. As the observations are inherently samples from an unknown probability distribution, my approach was to parametrise this probability distribution in a flexible way, where the explicit parameters are those of the actual physical systems, and all other unknown effects may either be ignored, or learned from the data in a semi-parametric fashion.

The explicit goal of the model is to estimate and correct for spatial distortions that are not simply due to the neural activity related changes of Ca^{2+} concentration. Therefore the available data will be used to estimate a gain function over spatial locations[4] that distorts some true underlying signal as summarised in section 2.1.1. The observed pixel intensities are then created from the spatially distorted optical signal that enters the photomultiplier, as described in section 2.1.2. We use the following notation:

$$\begin{array}{rl} \mathbf{x} & \text{Spatial location} \\ G(\mathbf{x}) & \text{Spatial gain} \\ Z(\mathbf{x}, t) & \text{True signal} \\ Y(\mathbf{x}, t) & \text{Observed pixel intensities} \end{array} \tag{2.1}$$

The definition of these individual elements already gives rise to structure in our probabilistic model:

[4] Throughout this thesis I make the assumption that the datasets are spatially two dimensional, as is most common in practice. However, both the model described and the code used to implement it should work for arbitrary dimensional datasets, with reasonable scaling.

$$
\begin{aligned}
p(\, Y \mid G \,) &= p(\, Y \mid G,\, Z,\, X,\, \xi \,) && \text{is the Likelihood,} \\
p(\, G \,) &= p(\, G \mid X,\, \theta \,) && \text{is the Prior over the gain,} \\
p(\, G \mid Y \,) &= p(\, G \mid Y,\, Z,\, X,\, \xi,\, \theta \,) && \text{is the Posterior over the gain,}
\end{aligned} \tag{2.2}
$$

where ξ and θ parameterise the likelihood and the prior, respectively. The prior encodes our assumptions about the structure in the spatial gain, whereas the likelihood is the probabilistic model of how the photomultiplier and subsequent electronics record the spatially distorted signal. The posterior describes our updated belief about the gain function after incorporating the current observed data as evidence.

In the following I first describe the generic parametric form of a variational inference model, then specify the forms of the prior and approximate posterior functions as an approximate Gaussian Process and finally detail a set of options for the prior and likelihood functions that correspond to our knowledge about the physical systems that implement them.

2.2.1 Variational inference

As for models corresponding to detailed physical systems, the posterior is usually non-conjugate and the normaliser is intractable, we have to turn to an approximation that we can compute. Due to its established track record along with recent developments, I chose to employ a variational sparse approximation (Titsias 2009a) to the posterior, which introduces a new parametrised - so-called variational - distribution, that is in a tractable model class. This distribution is then used to construct a lower bound to the marginal log likelihood, which is the variational lower bound:

$$q(\, G \,) = q(\, G \mid \psi \,) \approx p(\, G \mid Y \,) \qquad \text{is the Variational distribution}^5, \tag{2.3}$$

$$
\begin{aligned}
\log p(\, Y \,) &= \log \int p(\, Y \mid G \,)\, p(\, G \,)\, dG && \text{is the Marginal log likelihood.} && (2.4) \\
&= \log \int p(\, Y,\, G \,)\, \frac{q(\, G \,)}{q(\, G \,)}\, dG \\
&= \log \left(\mathbb{E}_{\mathrm{q}} \left[\frac{p(\, Y,\, G \,)}{q(\, G \,)} \right] \right) \\
&\geq \mathbb{E}_{\mathrm{q}} \left[\log \frac{p(\, Y,\, G \,)}{q(\, G \,)} \right] && \text{due to the Jensen inequality.} && (2.5)
\end{aligned}
$$

$$\mathcal{L}(q) = \mathbb{E}_{\mathrm{q}} [\log p(\, Y,\, G \,)] - \mathbb{E}_{\mathrm{q}} [\log q(\, G \,)] \qquad \text{is the Variational lower bound.} \tag{2.6}$$

Furthermore, this bound is tight on the marginal log likelihood (Cite: something), and approached as the KL-divergence between the true and the approximate posterior vanishes:

[5] Where ψ is the collection of so-called variational parameters.

$$\mathcal{L}(q) = \log p(\, Y) - D_{\mathrm{KL}}\left[\, q(\, G\,) \parallel p(\, G \mid Y\,)\right]$$

Therefore maximising the computable variational lower bound is equivalent to minimising the KL divergence between the approximating and the true posterior, and thus translates a difficult inference problem into a generic optimisation one.

A further advantage of this approach, is that the marginal likelihood defined in equation 2.4 is in fact a function of not only the gain G, but also any parameters of the illumination or observation models, that correspond to the prior and the likelihood:

$$\log p(\, Y \mid Z,\, \xi,\, \theta\,) = \log \int p(\, Y \mid G,\, Z,\, \xi\,)\; p(\, G \mid \theta\,)\; dG \tag{2.7}$$

In fact, we may treat all parameter values as uncertain, as all physical measurements and calibrations are inherently noisy, and work with the joint distribution of the observations and parameters:

$$\log p(\, Y,\, Z,\, \xi,\, \theta\,) = \log \int p(\, Y \mid G,\, Z,\, \xi\,)\; p(\, G \mid \theta\,)\; p(Z)\; p(\xi)\; p(\theta)\; dG \tag{2.8}$$

This results in an extremely flexible procedure, where knowledge about the physical systems may easily be incorporated into the prior functions of parameters. The complete model therefore defines an objective that is a function of all parameters and the gain:

$$q(\, G \mid \psi\,) \approx\; p(\, G \mid Y,\, Z,\, \xi,\, \theta\,) \tag{2.9}$$

$$\log p(\, Y,\, Z,\, \xi,\, \theta\,) =\; \log\left(\mathbb{E}_{\mathrm{q}}\left[\frac{p(\, Y,\, G,\, Z,\, \xi,\, \theta\;)}{q(\, G \mid \psi\,)}\right]\right) \tag{2.10}$$

$$\begin{aligned}
\mathcal{L}(q) =\; & \mathbb{E}_{\mathrm{q}}\left[\log p(\, Y,\, G,\, Z,\, \xi,\, \theta\;)\right] - \mathbb{E}_{\mathrm{q}}\left[\log q(\, G \mid \psi\,)\right] \\
=\; & \mathbb{E}_{\mathrm{q}}\left[\log p(\, Y \mid\, G,\, Z,\, \xi\,) + \log p(G \mid\, \theta\,) + \log p(\, Z\,) + \log p(\xi) + \log p(\theta)\right] \\
& - \mathbb{E}_{\mathrm{q}}\left[\log q(\, G \mid\, \psi)\right]
\end{aligned}$$

The final objective function for our variational inference scheme is

$$\boxed{\begin{aligned}
\mathcal{L}(q) =\; & \mathbb{E}_{\mathrm{q}}\left[\log p(\, Y \mid\, G,\, Z,\, \xi\,)\right] \\
& - D_{\mathrm{KL}}\left[\, q(\, G \mid\, \psi\,) \parallel p(\, G \mid\, \theta\,)\right] \\
& + \log p(\, Z\,) + \log p(\xi) + \log p(\theta).
\end{aligned}} \tag{2.11}$$

To fully specify our variational inference model, we need to choose the prior parameter distributions $p(Z)$, $p(\xi)$ and $p(\theta)$, the prior and variational distributions over the gain, $p(G \mid \theta\,)$

and $q(\ G\ \mid\ \psi\ )$, and the likelihood $p(\ Y \mid\ G,\ Z,\ \xi\ )$. The prior distributions are generally either uninformative ones over a range of physically plausible values, if no measurements are available, or normal distributions around some measured values, with the variance arising from the reliability of the measurement process itself. In the next subsection I detail an approximate Gaussian Process model that sets the form of the prior and variational distributions, then afterwards describe potential likelihood models.

2.2.2 Scalable approximate Gaussian Processes as prior functions

A highly scalable and yet extremely flexible model for the prior and variational distributions is a recently developed Gaussian Process approximation, that uses Structured Kernel Interpolation (Wilson and Nickisch 2015) and thus exploits the Kronecker structure inherent in imaging data, as well as has a built-in smoothness of the modelled function due to the interpolation used. In order to understand this approximation, we first need to define a vanilla Gaussian Process model, describe its sparse approximation, then choose the particular form of the approximation that is most suitable for our purposes.

A Gaussian Process over spatial locations is defined (Rasmussen and Williams 2006) by a mean function $m(\mathbf{x})$ and a positive definite covariance (or kernel) function $k(\mathbf{x},\ \mathbf{x}')$, and

$$f(\mathbf{x})\ \sim\ \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x},\ \mathbf{x}')) \tag{2.12}$$

Given a matrix of spatial locations $\mathbf{X}_{\mathrm{nd}}$, these functions may be applied element- or pairwise, resulting in the mean vector $\mathbf{m}_{\mathrm{n}} = m(\mathbf{X}_{\mathrm{nd}})$ and the kernel matrix $\mathbf{K}_{\mathrm{n\mathring{n}}} = k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}_{\mathrm{nd}})$, respectively. This then defines a *prior* normal distribution of the function values at the given locations:

$$\mathbf{f}_{\mathrm{n}}\ \sim\ \mathcal{N}(\mathbf{m}_{\mathrm{n}}, \mathbf{K}_{\mathrm{n\mathring{n}}}) \tag{2.13}$$

Observing function values $\mathbf{f}_{\mathrm{n}} = f(\mathbf{X}_{\mathrm{nd}})$ defines a *posterior* distribution over new locations $\mathbf{X}^*_{\mathrm{md}}$, such that the joint distribution of the observed and predicted values is still normal:

$$\begin{bmatrix} \mathbf{f}_{\mathrm{n}} \\ \mathbf{f}^*_{\mathrm{m}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{\mathrm{n}} \\ m(\mathbf{X}^*_{\mathrm{md}}) \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathrm{n\mathring{n}}} & k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}^*_{\mathrm{md}}) \\ k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}}) & k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}^*_{\mathrm{md}}) \end{bmatrix}, \right) \tag{2.14}$$

This then gives rise to a predictive distribution:

$$\begin{aligned} p(\mathbf{f}^*_{\mathrm{m}}\ \mid\ \mathbf{X}^*_{\mathrm{md}},\ \mathbf{X}_{\mathrm{nd}},\ \mathbf{f}_{\mathrm{n}}) = \mathcal{N}\Big(\ & m(\mathbf{X}^*_{\mathrm{md}}) + k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}})\,[\mathbf{K}_{\mathrm{n\mathring{n}}}]^{-1}\mathbf{f}_{\mathrm{n}}, \\ & k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}^*_{\mathrm{md}}) - k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}})[\mathbf{K}_{\mathrm{n\mathring{n}}}]^{-1} k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}^*_{\mathrm{md}})\ \Big) \end{aligned} \tag{2.15}$$

Unfortunately due to presence of the inverse $[\mathbf{K}_{\text{nn}}]^{-1}$, this formulation is not capable of dealing with $N \gg 10000$ locations even on modern computers, and thus we need to employ approximations to scale to larger datasets. A popular way of doing so is the use of inducing points – a set of locations $\mathbf{X}_{\text{ud}}^{\dagger}$ chosen (or optimised) in a way, such that they define an approximate kernel $\tilde{k}(\cdot,\cdot \mid \mathbf{X}_{\text{ud}}^{\dagger}) \approx k(\cdot,\cdot)$, where the computational complexity of the predictive distribution using $\tilde{k}$ is reduced to to $\mathcal{O}(U^3 + U^2 N)$ from $\mathcal{O}(N^3)$. Popular examples are the subset of regressors (SoR) method (Silverman 1985) and the fully independent training conditionals (FITC) approximation (Snelson and Ghahramani 2006), where

$$\tilde{k}_{\text{SoR}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}_{\text{ud}}^{\dagger}) = k(\mathbf{x},\mathbf{X}_{\text{ud}}^{\dagger}) \left[k(\mathbf{X}_{\text{ud}}^{\dagger},\mathbf{X}_{\text{ud}}^{\dagger})\right]^{-1} k(\mathbf{X}_{\text{ud}}^{\dagger},\mathbf{x}') \tag{2.16}$$

$$\tilde{k}_{\text{FITC}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}_{\text{ud}}^{\dagger}) = \delta_{\text{xx}'}\, k(\mathbf{x},\mathbf{x}') + (1-\delta_{\text{xx}'})\, \tilde{k}_{\text{SoR}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}_{\text{ud}}^{\dagger}). \tag{2.17}$$

In order for these types of approximation to result in large computational gains, $U \ll N$ is required, which limits predictive performance (**Wilson2014**). This limitation may be partially resolved by exploiting existing Kronecker structure in a multidimensional kernel and by imposing Cartesian grid structure on inducing points, which results in an efficiently invertible circulant Toeplitz kernel matrix for stationary k covariance functions. Given product structure in the kernel and a D-dimensional grid with $U = V^D$ grid points, these reduce the computational complexity for learning and inference from $\mathcal{O}(U^3)$ to $\mathcal{O}(DU^{1+1/D})$ by using fast matrix decomposition, fast matrix-vector products and linear conjugate gradients. For further details see (Wilson and Nickisch 2015; Wilson and Adams 2013) (Cite: wilson thesis...).

The next issue is that for N locations these approximations still require the computation of the cross-covariances $k(\mathbf{X}_{\text{nd}},\mathbf{X}_{\text{ud}}^{\dagger})$, which becomes the limiting step with $\mathcal{O}(U^2 N)$ complexity. However, this matrix may be approximated efficiently using local Structured Kernel Interpolation from the kernel matrix on the inducing points. With $\mathbf{W}_{\text{nu}}$ being a sparse interpolation weight matrix, we may approximate the cross-covariances as

$$k(\mathbf{X}_{\text{nd}},\mathbf{X}_{\text{ud}}^{\dagger}) \approx \mathbf{W}_{\text{nu}}\, k(\mathbf{X}_{\text{ud}}^{\dagger},\mathbf{X}_{\text{ud}}^{\dagger}) \tag{2.18}$$

and thus

$$\begin{aligned} k(\mathbf{X}_{\text{nd}}, \mathbf{X}_{\text{nd}}) &\approx k(\mathbf{X}_{\text{nd}}, \mathbf{X}^{\dagger}_{\text{ud}}) \left[k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}^{\dagger}_{\text{ud}})\right]^{-1} k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}_{\text{nd}}) \\ &\approx \mathbf{W}_{\text{nu}}\, k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}^{\dagger}_{\text{ud}}) \left[k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}^{\dagger}_{\text{ud}})\right]^{-1} k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}^{\dagger}_{\text{ud}})^{\top} \mathbf{W}^{\top}_{\text{nu}} \\ &= \mathbf{W}_{\text{nu}}\, k(\mathbf{X}^{\dagger}_{\text{ud}}, \mathbf{X}^{\dagger}_{\text{ud}}) \mathbf{W}^{\top}_{\text{nu}} \\ &= k_{\text{SKI}}(\mathbf{X}_{\text{nd}}, \mathbf{X}_{\text{nd}} \mid \mathbf{X}^{\dagger}_{\text{ud}},\, \mathbf{W}_{\text{nu}}) \end{aligned} \tag{2.19}$$

Therefore to finish our approximation, we need to specify the locations of the inducing points $\mathbf{X}^{\dagger}_{\text{ud}}$ and the interpolation scheme used to compute the sparse interpolation weights $\mathbf{W}_{\text{nu}}$. As suggested above the inducing point placed in a multidimensional grid over the span of the input data is crucial for scalability, thus one only needs to determine the bounds and the density of the grid. These parameters are influenced by our choice of interpolation scheme. In order to inherently impose known lengthscale spatial smoothness on the gain function inferred, I employed an interpolation scheme where the weights of the grid points are a finite-size Gaussian filter centred on the data points, then row-wise normalised:

$$\begin{aligned} w^{\text{raw}}_{\text{nu}} &= \begin{cases} \mathcal{N}\big(\sqrt{\sum_{\text{d}} (x_{\text{nd}} - x^{\dagger}_{\text{ud}})^2},\ \sigma^2_{\text{filter}}\big) & \text{if } d_{\text{x}} < d_{\max} \\ 0 & \text{otherwise} \end{cases} \\ w_{\text{nu}} &= w^{\text{raw}}_{\text{nu}} \ / \ \sum_{\text{u}} w^{\text{raw}}_{\text{nu}} \end{aligned} \tag{2.20}$$

The choice of $d_{\max}$ and especially σ_{filter} sets the smoothness of our interpolation scheme. Setting the grid spacing d_{grid} and the bounds b_{grid} fully defines our scalable model called Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP, Wilson and Nickisch 2015) for the definition of the prior on the gain function. The prior parameters include the choice and parameters of the mean function m, the covariance function k, as well as the above parameters of the approximation scheme itself. In practice this latter is not optimised. The prior, the posterior, and their KL-divergence of a function f evaluated at the input points $\mathbf{F}_{\text{n}} = f(\mathbf{X}_{\text{nd}})$ are thus defined as:

$$\theta = \left\{ m,\ \theta_{\text{m}},\ k,\ \theta_{\text{k}},\ \theta_{\text{interp}} \right\} \tag{2.21}$$

$$p(\, \mathbf{F}_{\text{n}} \mid \mathbf{X}_{\text{nd}},\ \theta\,) = \mathcal{N}\Big(\mathbf{F}_{\text{n}} - \mathbf{W}_{\text{nu}}\, m(\mathbf{X}_{\text{ud}} \mid \theta),\ k_{\text{SKI}}(\, \mathbf{X}_{\text{nd}},\, \mathbf{X}_{\text{nd}} \mid \theta\,)\Big) \tag{2.22}$$

The approximate posterior uses variational parameters ψ to define the mean and covariance structure over the inducing points, then employs the interpolation scheme to approximate the posterior over the input locations.

$$\psi = \left\{ \boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}},\ \boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}} \right\} \tag{2.23}$$

$$q(\,\mathbf{F}_{\mathrm{n}} \mid \mathbf{X}_{\mathrm{nd}},\ \boldsymbol{\psi}\,) = \mathcal{N}\left(\mathbf{F}_{\mathrm{n}} - \mathbf{W}_{\mathrm{nu}}\boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}}\ ,\ \ \mathbf{W}_{\mathrm{nu}}\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}\mathbf{W}^{\top}_{\mathrm{n\mathring{u}}}\right) \tag{2.24}$$

$$D_{\mathrm{KL}}(\,q \parallel p\,) = \frac{1}{2}\left[\begin{array}{l} -D + \log\dfrac{|k(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})|}{|\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}|} \\ + \mathrm{Trace}\left\{\left[\mathrm{k}(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1}\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}\right\} \\ + (\boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}} - m(\mathbf{X}_{\mathrm{ud}} \mid \theta))\left[k(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1}(\boldsymbol{\psi}^{\mathrm{m}}_{\mathring{\mathrm{u}}} - m(\mathbf{X}_{\mathring{\mathrm{u}}\mathrm{d}} \mid \theta)) \end{array}\right] \tag{2.25}$$

This defines the variational posterior and the KL-divergence in our variational inference scheme equation 2.11, up to the specification of the mean and covariance functions of the prior.

Probabilistic models of spatial distortions

The next step therefore is to design mean and covariance functions of the prior that correctly encode our assumptions. First and foremost, all interactions that lead from the true signal $Z(\mathbf{x},\, t)$ to the optical signal arriving at the photomultiplier may be thought of as strictly multiplicative interactions. Therefore only the product of G and Z affects the likelihood of the data:

$$p(Y \mid G,\, Z) = p(Y(\mathbf{x},\, t) \mid G(\mathbf{x})Z(\mathbf{x},\, t)). \tag{2.26}$$

As both G and Z are strictly positive quantities, we may model their multiplicative interaction as a sum in logarithmic space, $GZ = \exp(\log G + \log Z)$. This observation enables us to model the numerous multiplicative interactions (see section 2.1.1) that result in G to be modelled as an additive function describing $\log G$, with the premise that our prior and approximate posterior estimates of the gain function will now be log-normally distributed. This actually ensures the physically constrained positivity of G, and with a slight rewrite of the variational inference scheme in equation 2.11 results in:

$$\begin{aligned} F &= \log G \\ \mathcal{L}(q) &= \mathbb{E}_{\mathrm{q}}\left[\log p(\,Y \mid e^{F+\log Z},\, \xi\,)\right] \\ &\quad - D_{\mathrm{KL}}[\,q(\,F \mid \psi\,) \parallel p(\,F \mid \theta\,)] \\ &\quad + \log p(\,\log Z\,) + \log p(\xi) + \log p(\theta). \end{aligned} \tag{2.27}$$

There are numerous ways of making assumptions about the form of F, I describe examples here that are motivated by our knowledge about the illumination, the tissue and the collection system. Each distortion effect caused by the various elements enters additively into the final logarithmic gain: $F = \sum_{\mathrm{i}} F^{\mathrm{i}}$, where each F^{i} represents the logarithmic spatial gain distortion introduced by an element. Having no explicit knowledge about the particular optics and settings of a microscope system, a reasonable prior assumption is that the distortion they introduce has no well-known shape, and thus the prior mean function for each element is uniformly zero. However, as discussed previously, there are definitely assumptions we may make about the gain function, and these generally fall into two categories. One class of assumptions is the symmetries about the optical axes, and the other is the lengthscale of spatial smoothness of various distortions.

In order to capture the symmetries present in the system, I introduce the following covariance function transformations. Given a covariance function $k_{\mathrm{D}} : \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \to \mathbb{R}$ and the location of the focal point along the optical axis in space $\mathbf{x}_{\mathrm{d}}^{\mathrm{o}} \in \mathbb{R}^{\mathrm{D}}$, we may define the linearly symmetrised covariance function

$$k_{\mathrm{LinSymm}}(\, \mathbf{x}_{\mathrm{d}},\, \mathbf{x}'_{\mathrm{d}} \;\mid\; k_{\mathrm{D}},\, \mathbf{x}_{\mathrm{d}}^{\mathrm{o}} \,) = k_{\mathrm{D}}(\, |\mathbf{x}_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}|,\, |\mathbf{x}'_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}|\,), \tag{2.28}$$

where the absolute value is applied dimension-wise. Functions drawn from a Gaussian Process with a linearly symmetrised covariance function will exhibit symmetries about each individual spatial axes, which are defined in scanning microscopes as the optical axis, and its transverse plane[6] with the basis given by the scanning mirror orientations. Similarly, given a one-dimensional base function, $k_1 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, we may define a radially symmetric covariance function as

$$k_{\mathrm{RadSymm}}(\, \mathbf{x}_{\mathrm{d}},\, \mathbf{x}'_{\mathrm{d}} \;\mid\; k_1,\, \mathbf{x}_{\mathrm{d}}^{\mathrm{o}} \,) = k_1(\, \left\|\mathbf{x}_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}\right\|,\, \left\|\mathbf{x}'_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}\right\| \,). \tag{2.29}$$

Finally we need to estimate the various spatial lengthscales of the distortions and capture them with the appropriate covariance functions. I use two main classes of base covariance functions here. First, the Radial Basis Function covariance (known also as Squared Exponential, Gaussian or Exponentiated Quadratic kernel), which models diminishing strictly positive interactions that lead to smooth functions. It may be parametrised by a single lengthscale, in which case the interaction decay is isotropic; or by a vector of lengthscales, one for each dimension:

[6] It is worth mentioning here that some microscope objectives introduce a curvature as the focal point is moved off-axis, and therefore the actually imaged manifold is not truly a transversal plane with respect to the optical axis. Therefore the point cloud $\mathbf{X}_{\mathrm{nd}}$ that is usually assumed coplanar may not be, and the calculated distances are slightly overestimated. Nevertheless, the symmetries are still present, and they do not truly depend on distance from optical axis, but rather the scan angles, and thus are not affected by objective-induced sample plane curvature.

$$k_{\mathrm{RBF}}(\,\mathbf{x}_{\mathrm{d}},\,\mathbf{x}'_{\mathrm{d}}\;\mid\;l\,) = \exp\left(-\frac{\left\|\,\mathbf{x}_{\mathrm{d}}-\mathbf{x}'_{\mathrm{d}}\,\right\|^2}{2l^2}\right) \tag{2.30}$$

$$k_{\mathrm{RBF}}(\,\mathbf{x}_{\mathrm{d}},\,\mathbf{x}'_{\mathrm{d}}\;\mid\;\mathbf{l}_{\mathrm{d}}\,) = \exp\left(-\sum_{\mathrm{d}}\frac{\left(x_{\mathrm{d}}-x'_{\mathrm{d}}\right)^2}{2l_{\mathrm{d}}^2}\right) \tag{2.31}$$

A much more expressive covariance function that is also capable of modelling negative interactions is the recently introduced Spectral Mixture Kernel (Wilson and Adams 2013).

Describe spectral mixture kernel and my mexicanhat simplification

2.2.3 Photomultiplier likelihood models

As described in section section 2.1.2, the photomultiplier and the electronics translate the optical signal into an electric, then finally a digital one. Due to the limitations of photon counting, this digital signal usually represents an analog integration of the variable height current pulses induced by the incoming photons in a given time window, that corresponds to the dwell time of the optical focus on a pixel. The conversion of pulses includes a user-controllable offset of the preamplifier, that essentially thresholds the incoming signal pulses and thus sets the zero level for the Analog-Digital Converter (ADC), which in turn means the zero pixel intensity in the data. The effective gain from photon count to pixel intensity is the product of the user-configurable photomultiplier gain (set by the photomultiplier high voltage, and converts photoelectrons to mean current pulse height) and the usually fixed preamplifier gain (that converts current into voltage). The electronic components before digitalisation also introduce additive zero-mean noise into the system. These are the three most important parameters of our likelihood function: $\xi = \{\,\xi^{o}, \xi^{g},\, \xi^{\sigma_0^2}\,\}$ are the offset, the gain and the electric noise variance, respectively. These should be part of any photomultiplier model.

Let λ be the instantaneous intensity of the thinned Poisson process that generates photoelectrons at the photomultiplier cathode; then the likelihood we wish to parameterise here is

$$\lambda = e^{F+\log Z} \tag{2.32}$$

$$p(\,Y\mid\,\lambda,\,\xi\,) = \;? \tag{2.33}$$

Once we define this probability distribution p_{Y}, it is important to consider that the analog to digital conversion introduces lower and upper thresholds (cut-off and saturation), and thus given the likelihood functions described in this section (which describe the signal incoming to the analog digital converter), the true likelihoods of the digital signal are

$$
\begin{aligned}
p(\, Y_{\text{digital}} = 0 \mid p_{\text{Y}}) &= \int_{-\infty}^{0} p_{\text{Y}}(x)\, dx \\
p(\, Y_{\text{digital}} = \text{ADC}_{\text{max}} \mid \text{p}_{\text{Y}}) &= \int_{\text{ADC}_{\text{max}}}^{\infty} p_{\text{Y}}(x)\, dx \\
p(\, Y_{\text{digital}} = i \mid p_{\text{Y}},\, 0 < Y_{\text{digital}} < \text{ADC}_{\text{max}}\,) &= \int_{\text{i}-1}^{i} p_{\text{Y}}(\, x\,) dx\,,
\end{aligned}
\tag{2.34}
$$

where ADC_{max} is the number of channels on the ADC. This process is taken into account during any computations, but are omitted from the following description for brevity.

In the following I introduce three models of the photomultiplier and subsequent electronics, which best describe reality for distinct values of the light intensity λ.

Model 1 - Intensity amplification

The simplest possible model disregards the discrete nature of incoming light, and thus is only generally applicable to situations with very high incoming photon counts. Even though our data is not in this regime, this model is frequently applied, and thus worth discussing (Cite: maybe something here). It approximates the Poisson photon distribution with intensity λ as a normal distribution. The contribution of the photomultiplier is then the gain that multiplies the random variable representing the observed intensity, and the offset shifts its mean. Often a third parameter is considered, $\xi^{\sigma_0^2}$ is the variance of the zero-mean noise introduced by the electronics after the amplification process. The whole model is

$$
\nu \sim \mathcal{N}(\, \lambda,\, \lambda\,) \tag{2.35}
$$

$$
p(\, Y \mid \, \nu,\, \xi\,) \;=\; \mathcal{N}_{\text{Y}}\Big(\, \xi^{g}\,(\nu - \xi^{o}),\, \xi^{\sigma_0^2}\,\Big), \tag{2.36}
$$

therefore

$$
\xi_1 \;=\; \{\, \xi^{o}, \xi^{g},\, \xi^{\sigma_0^2}\,\} \tag{2.37}
$$

$$
p_1(\, Y \mid \, \lambda,\, \xi\,) \;=\; \mathcal{N}_{\text{Y}}\Big(\, \xi^{g}\,(\lambda - \xi^{o}),\, \xi^{g^2}\lambda + \xi^{\sigma_0^2}\,\Big). \tag{2.38}
$$

Model 2 - Gaussian single electron response

For the low intensity values observed in our data, it is of importance to take the discrete nature of incoming signal into account. The photomultiplier and subsequent electronics then amplify each photoelectron into a pixel intensity according to the single electron response (SER). This is de-

scribed by a probability distribution of the total charge generated by a single event, and is often approximated by a normal distribution. Multiple photonelectrons result in a signal that is the sum of the independent pulse charges generated.

$$\nu \sim \text{Poisson}(\lambda) \tag{2.39}$$

$$p_{\text{SER}}(C \mid \nu = 1, \xi) \approx \mathcal{N}_{\text{C}}\left(\xi^{g}, \xi^{\sigma_1^2}\right) \tag{2.40}$$

$$p(C \mid \nu, \xi) = \mathcal{N}_{\text{C}}\left(\xi^{g}\nu, \xi^{\sigma_1^2}\nu\right) \tag{2.41}$$

$$p(Y \mid C, \xi) = \mathcal{N}_{\text{Y}}\left(C - \xi^{o}, \xi^{\sigma_0^2}\right), \tag{2.42}$$

where ν is a discrete count and C is the charge generated by the photomultiplier. Therefore

$$\xi_2 = \{\xi^{o}, \xi^{g}, \xi^{\sigma_0^2}, \xi^{\sigma_1^2}\} \tag{2.43}$$

$$\begin{aligned} p_2(Y \mid \lambda, \xi) &= \sum_{\nu=0}^{\infty} p(Y \mid \nu)\, p(\nu \mid \lambda) \\ &= \sum_{\nu=0}^{\infty} \mathcal{N}_{\text{Y}}\left(\xi^{g}\nu - \xi^{o}, \xi^{\sigma_1^2}\nu + \xi^{\sigma_0^2}\right) \cdot \text{Poisson}_{\nu}(\lambda). \end{aligned} \tag{2.44}$$

Ultimately this likelihood distribution is a Poisson-weighted mixture of Gaussians, where the mixture components' mean and variance scales with the number of observed photons. In practice the infinite sum may be replaced by a finite sum, where the components with diminishingly low Poisson mixture weights are omitted.

Model 3 - Underamplified single electron response

Photomultipliers with low high-voltage settings may not amplify each photoelectron maximally, and thus the single electron response probability distribution is not well-approximated by a normal distribution anymore (Dossi et al. 2000; De Haas and Dorenbos 2010). We may describe the charge distribution of underamplified events as an exponential distribution with a given rate ξ^{e}. The single electron response distribution is a Bernoulli mixture of fully amplified and underamplified events, with probabilities $1 - \xi^{p_{\text{e}}}$ and $\xi^{p_{\text{e}}}$, respectively. Fortunately even for more than one photon, the distribution of total charges gets better and better approximated by a normal distribution, and thus we may treat only the single observed photon as a special case:

$$\nu \sim \text{Poisson}(\lambda) \tag{2.45}$$

$$p_{\text{SER}}(C \mid \nu = 1, \xi) \approx \xi^{p_e} \cdot \text{Exponential}_C(\xi^e) + (1 - \xi^{p_e}) \cdot \mathcal{N}_C\left(\xi^g, \xi^{\sigma_1^2}\right) \tag{2.46}$$

$$p(C \mid \nu, \nu \geq 2, \xi) = \mathcal{N}_C\left(\mathbb{E}_{p_{\text{SER}}}[C] \cdot \nu, \text{Var}_{p_{\text{SER}}}[C] \cdot \nu\right) \tag{2.47}$$

$$p(Y \mid C, \xi) = \mathcal{N}_Y\left(C - \xi^o, \xi^{\sigma_0^2}\right). \tag{2.48}$$

In order to derive the resulting probability distribution over Y given λ, we need to be able to describe the sum of an exponentially distributed random variable and the additive zero-mean noise with variance $\xi^{\sigma_0^2}$, which is an Exponentially Modified Gaussian (EMG) distribution, defined as

$$\begin{aligned} a &\sim \text{Exponential}(\lambda) \\ b &\sim \mathcal{N}(\mu, \sigma^2) \\ x &= a + b \\ \text{EMG}_x(\lambda, \mu, \sigma^2) &= \frac{\lambda}{2} \cdot e^{0.5\lambda(2\mu + \lambda\sigma^2 - 2x)} \cdot \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt. \end{aligned} \tag{2.49}$$

Furthermore, we need to compute the mean and variance of the underamplified SER in order to evaluate the multi-photoelectron components:

$$\begin{aligned} \mathbb{E}_{p_{\text{SER}}}[C] &= \xi^{p_e} / \xi^e + (1 - \xi^{p_e}) \xi^g \\ \text{Var}_{p_{\text{SER}}}[C] &= 2\xi^{p_e} / \xi^{e^2} + (1 - \xi^{p_e})(\xi^{g^2} + \xi^{\sigma_1^2}) - (\mathbb{E}_{p_{\text{SER}}}[C])^2 \end{aligned} \tag{2.50}$$

As a result, the complete model is

$$\xi_3 = \{\xi^o, \xi^g, \xi^{\sigma_0^2}, \xi^{\sigma_1^2}, \xi^e, \xi^{p_e}\} \tag{2.51}$$

$$\begin{aligned} p_3(Y \mid \lambda, \xi) = \; & \text{Poisson}_0(\lambda) \cdot \mathcal{N}_Y\left(-\xi^o, \xi^{\sigma_0^2}\right) \\ & + \text{Poisson}_1(\lambda) \cdot \Big(\xi^{p_e} \, \text{EMG}_Y\left(\xi^e, -\xi^o, \xi^{\sigma_0^2}\right) + \\ & \qquad (1 - \xi^{p_e}) \, \mathcal{N}_Y\left(\xi^g - \xi^o, \xi^{\sigma_1^2} + \xi^{\sigma_0^2}\right)\Big) \\ & + \sum_{\nu=2}^{\infty} \Big(\text{Poisson}_\nu(\lambda) \cdot \\ & \qquad \mathcal{N}_Y\left(\mathbb{E}_{p_{\text{SER}}}[C] - \xi^o, \text{Var}_{p_{\text{SER}}}[C] + \xi^{\sigma_0^2}\right)\Big). \end{aligned} \tag{2.52}$$

2.2.4 Heuristics for data sub-selection

Given an optical signal intensity model described in section 2.2.2 and a likelihood model from section 2.2.3, we are now equipped to carry out variational inference (section 2.2.1) and estimate the parameters of a realistic generative model of the data. There is however one latent variable that we have not yet discussed in detail, the true signal $Z(\mathbf{x},\, t)$. If no assumptions are made about this latent variable, it introduces an inherent degeneracy to the spatial model, as it would be capable of encompassing any structure we would wish to learn via G, the spatial gain. Therefore the first assumption we wish to make is that there is no spatial structure in the signal. This assumption is obviously wrong considering the whole dataset due to presence of neuronal structures (cell bodies and neurites), that do generate informative signal. In order to enable us making this assumption, we need to reject locations that are thought to have spatially structured activity – essentially we wish to find the background.

As most neural structures have a spatial extent of multiple pixels, a robust measure of identifying background pixels proved to be a low cross-correlation coefficient of a pixel with its neighbours. Given a neighbourhood averaging filter, we may estimate the local expected activity, and compute the cross-correlation of the pixel's signal with the local activity:

$$\begin{aligned} \mu(\,\mathbf{x},\, t\,) &= \mathbb{E}_{\mathring{\mathbf{x}}\in\mathrm{Neigh}(\mathbf{x})}\left[f(\mathring{\mathbf{x}})\cdot Y(\,\mathring{\mathbf{x}},\, t\,)\right] \\ \mathrm{CrossCorrCoeff}(\,\mathbf{x}\,) &= \frac{\mathbb{E}_{\mathrm{t}}\Big[\left(Y\,(\mathbf{x},\, t\,) - \mathbb{E}_{\mathrm{t}}[Y\,(\mathbf{x},\, t\,)]\right)\cdot\left(\mu(\,\mathbf{x},\, t\,) - \mathbb{E}_{\mathrm{t}}[\mu(\,\mathbf{x},\, t\,)]\right)\Big]}{\sqrt{\mathrm{Var}[\mathrm{Y}\,(\mathbf{x},\,\mathrm{t}\,)]\mathrm{Var}[\mu(\,\mathbf{x},\,\mathrm{t}\,)]}}, \end{aligned} \tag{2.53}$$

where μ represents the result of the convolution with neighborhood filter f (see **??** for further computational details). Pixels with a relatively low absolute cross-correlation coefficient with the local average are likely to be noise pixels (see figure ...), as their activity is not coordinated with nearby entities.

Add example figures here

Low cross-correlation (and information loss) in the signal may also be caused by the ADC thresholding. This means that values that are equivalent to 0 or $\mathrm{ADC}_{\mathrm{max}}$ could have represented any signals below or above these thresholds (see equation 2.34), and thus shift the mean and lower the spread during computation of the cross-correlation coefficient. Pixels with a high portion of thresholded signals were excluded.

Another effect of the data recording process that was not discussed so far is the presence of missing observations, which are unfortunately represented by 0 values in the dataset, rather than being marked as missing. During recording of the data, the raw digitalised observations generated in real time are stored in finite-size memory buffers, which are then read and flushed by the controlling computer asynchronously. Unfortunately due to the operation of modern computers, this is often

not carried out with as regular a timing signal as other operations in the microscope, which leads to presence of missing observations. We however may tell apart such missing pixels from true zero pixel intensities by the spatial structure in them. Long aligned streaks of zero values (whose spatial alignment is due to their scanning's temporal continuity) are significantly more likely to have been caused by missing data than by truly observing clipped-off signals, provided the data has been recorded with recommended gain and offset settings. The detection of aligned zeros may be carried out by convolution with a set of neighbourhood filters. For detecting any x-direction streaks of length at least k, generate filters

$$f^i = [\, \mathbf{0}_{\min(-i,0)},\, \mathbf{1}_k,\, \mathbf{0}_{\min(i+1,0)} \,], \quad i \in [-k+1, k-1], \tag{2.54}$$

where the vectors of 0s and 1s are concatenated in the direction of fast scanning. If any of the filters return a zero value, that means that there has been a streak of 0s that the current pixel is part of, and thus it should be treated as missing data rather than a true 0 valued observation. We may also notice and detect similar streaks of 0s in the slow scanning direction especially near the edges, which may be due to slight overlap of memory buffering and flushing. See figure 2.5 for an example of identifying missing values in real data. Once missing observations have been flagged, they may either be removed from the dataset, or imputed. Imputation is a common statistical operation in data preprocessing, and it means replacing the missing values with its expectation according to some metric. As a number of algorithms cannot deal with missing observations, I will refer to the imputed datasets that had its missing values replaced with the mean of its 3x3x3 spatio-temporal neighbourhood.

A final, computationally crucial assumption we wish to make about the selected (background, non-saturated) pixels, is that all their apparent temporal activity is purely due to noise. If this is true, our model may be further significantly simplified, as not only will $Z(\mathbf{x}, t)$ lack spatial structure (by selecting spatially unstructured background pixels via equation 2.53), but also temporal one, thus enabling us to model it with a single constant value $Z(\mathbf{x}, t) = z$. Again, in some datasets this assumption definitely cannot be made, as there are clear brightness changes over time. However, these brightness changes are often due to either a non-spatially specific general increase in signal; or a slight movement of the whole sample relative to the microscope, as the animal carries out a certain behaviour. Lateral movement artefacts are usually corrected before publishing the data and should be done by the user before applying this method (see Greenberg and Kerr 2009, Chen et al. 2012 or Pnevmatikakis and Giovannucci 2017 for examples of this extensively studied operation), but these do not correct for the induced overall temporal changes. As the change affects all pixels similarly, these behaviour-elicited temporal changes may be readily identified and removed from the data via singular value decomposition. Therefore as a final heuristic preprocessing step, we may apply the following:

$$\mathbf{Y}(\mathbf{x},\, t) = \sum_{\mathrm{i}=1}^{K} s_{\mathrm{i}}\, \mathbf{U}_{\mathrm{i}}(\mathbf{x}) \mathbf{V}_{\mathrm{i}}(t) \tag{2.55}$$

$$\mathbf{Y}_{\mathrm{corr}}(\mathbf{x},\, t) = \mathbf{Y}(\mathbf{x},\, t) - \sum_{\mathrm{i}=1}^{K_{\mathrm{shared}}} s_{\mathrm{i}}\, \mathbf{U}_{\mathrm{i}}(\mathbf{x}) \mathbf{V}_{\mathrm{i}}(t), \tag{2.56}$$

where $K_{\mathrm{shared}} \ll K$ is the number of globally shared temporal components, and is usually set to 1 or 2, based on the singular value spectrum $\{s_{\mathrm{i}}\}$.

To summarise, after applying the above defined heuristics, we may assume that the data Y only contains background pixels whose spatial distortions are purely due to the effective spatial gain we wish to model, and any temporal variation of the signals is only due to the various sources of noise present in the system. This means that we can further simplify our variational inference scheme of equation 2.27, assuming that the temporal samples at the same location are independent and identically distributed, and are fully described by a a single intensity value that only depends on the modelled spatial gain function and a constant z true signal:

$$\begin{aligned}
F(\mathbf{x}) &= \log G(\mathbf{x}) \\
\lambda(\mathbf{x}) &= e^{F(\mathbf{x}) + \log z} \\
\log p(\, Y \mid \lambda,\, \xi\,) &= \sum_{\mathrm{t}} \log p(\, Y(\mathbf{x},\, t) \mid \lambda(\mathbf{x}),\, \xi\,) \\
\mathcal{L}(q) &= \mathbb{E}_{\mathrm{q}}[\log p(\, Y \mid \lambda,\, \xi\,)] \\
&\quad - D_{\mathrm{KL}}[\, q(\, F \mid \psi\,) \,\|\, p(\, F \mid \theta\,)\,] \\
&\quad + \log p(\, \log z\,) + \log p(\xi) + \log p(\theta).
\end{aligned} \tag{2.57}$$

Equipped with this objective function, using the spatial gain function model described in section 2.2.2 and a likelihood model from section 2.2.3, we may optimise all parameters $(\, \psi, \xi, \theta, z\,)$ for a typical dataset of 512x512 spatial extent and hundreds of temporal frames within hours on modern computer GPU.

2.3 Results

In the following I apply the above discussed algorithm to real datasets, evaluate the validity of the assumptions that I made, and show that the fitted model indeed describes the data well. Afterwards I show how the pixel intensity data may be interpreted in terms of an instantaneous intensity that generated it, and how that intensity may be corrected from spatial distortions we estimated via the model fit. This results in a standardised baseline intensity, enabling consistent interpretation of images regardless of location within the image or even the microscope that recorded the data, as all these variations are accounted for within the model.

2.3.1 Datasets and model initialisation

In order to show the general applicability of the method, as well as to produce easily verifiable and reproducible results, I chose to use the dataset openly published as part of the Neurofinder[7] project. As part of an effort for establishing a set of calcium imaging analysis challenges, datasets were collected by a number of established labs using similar (but different) two-photon microscopy equipment to image various areas of the mouse brain in vivo. In all instances the mice were awake and head-fixed, and the fluorescent calcium sensor has been a variant of GCaMP6. See table 2.1 for further details of the individual datasets.

Code	Contributors	Brain region	Data Size (XxYxT)	Pixel size	Frame frequency
00.00	Simon Peron (Svoboda Lab)	vS1	512x512x3024	0.87μm	7 Hz
01.00	Adam Packer, Lloyd Russell (Hausser Lab)	V1	512x512x2250	1.25μm	7.5 Hz
02.00	Nicholas Sofroniew (Svoboda Lab)	vS1	512x512x8000	0.87μm	8 Hz
03.00	Jeff Zaremba (Losonczy Lab)	dHPC CA1	512x512x3024	0.59μm	7.5 Hz
04.00	Matthias Minderer (Harvey Lab)	Hindlimb S1	512x512x3000	0.87μm	6.75 Hz

Table 2.1: Basic characterisation of Neurofinder datasets

In the following I carry out a more extensive characterisation of dataset 00.00, which process is used to motivate and explain the statistics I compute for each dataset and summarise in table 2.2.

[7] URL: `https://github.com/codeneuro/neurofinder`

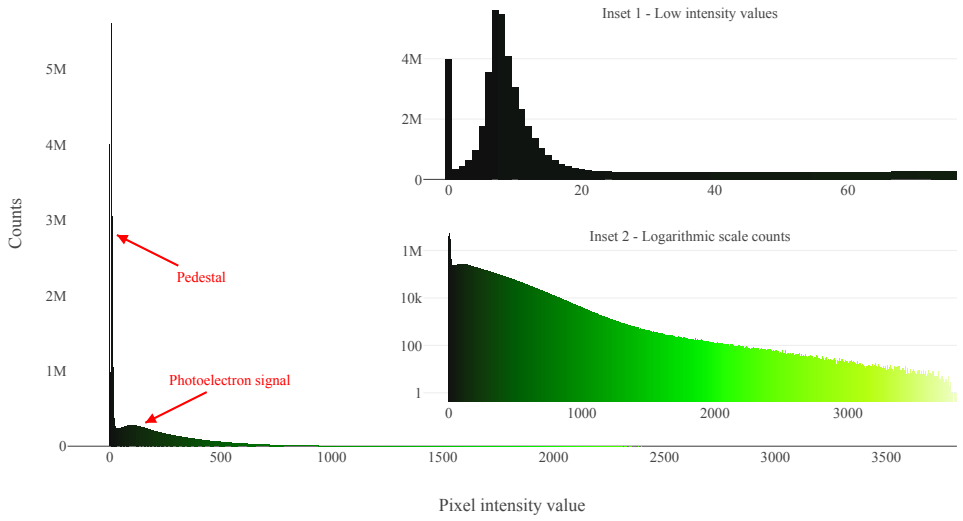


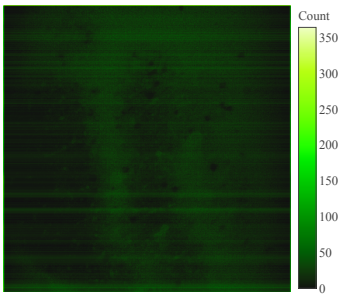
Figure 2.4: Pixel intensity histogram for the first 500 frames of dataset 00.00. The insets provide different visualisations of the same data. The separation of pedestal and signal pixels are clearly visible in the main figure. Inset 1 points out the separation of threshold and pedestal, whereas inset 2 shows the presence of high intensity observations. The colors of the individual bars were set to match those of image visualisations and to draw attention to the changes in axis ranges, but do not convey additional information.

These statistics are also used in the heuristic data exclusion step, or to initialise the model for fitting. As I only used the first 500 frames of each dataset for the model fit to demonstrate data efficiency and reduce run times, the statistics computed here will be accessing only those frames as well. The remainder of the data time points are held out for validation of the generality of results.

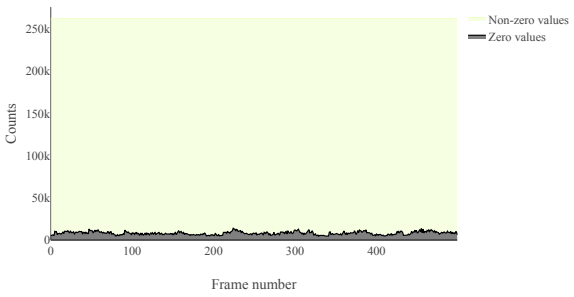
Figure 2.4 demonstrates the distribution of observed intensities over the whole imaging area and recording time. The distinct characteristics include the presence of a pedestal peak, which represent time windows with no photoelectrons observed, and therefore the minor fluctuation is solely due to the electronic noise on and after the photomultiplier anode. The photoelectronic signal peak is much wider due to the large noise in the amplification process itself, as well as the fact that it includes additive signals from multiple photons within the same gating time window. The third characteristic is the peak exactly at 0, which is caused by the ADC cutoff – every signal below a certain threshold results in a 0 value. The excessive presence of 0s in most datasets is however not only caused by thresholded real signals; when the data is simply not recorded, instead of flagging it as missing data, the digital value is still often reported as 0. In figure 2.5 I showcase the characteristics of

[8] It may also be high due to increased spatial correlation in illumination at the left and right edges, caused by the resonant galvo mirror being in the less linear regime of its sinusoidal curve. Our imputation process with local spatiotemporal mean further increases this measure, but that just means our foreground rejection heuristic is overly conservative for locations with missing data.

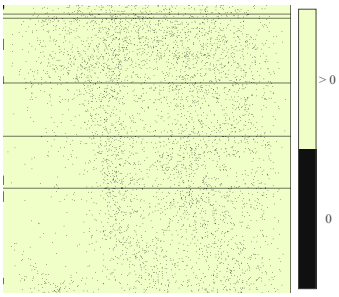
Figure 2.5: Identifying missing values and true observations.
(a-b) showcase the raw data over space and time, (c-d) provide examples and motivate the missingness heuristic described in equation 2.54, and (e-f) demonstrate the efficacy of the identification and removal of missing data.



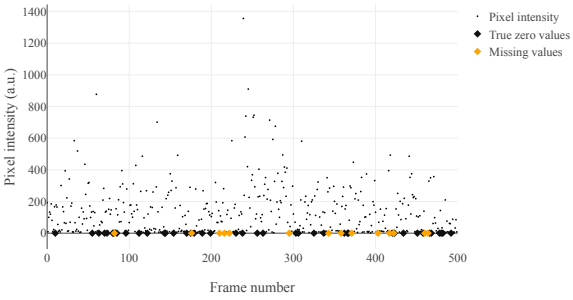
(a) Total number of zero values at each pixel over 500 frames. Note the high counts at the edges and the x-direction streaks.



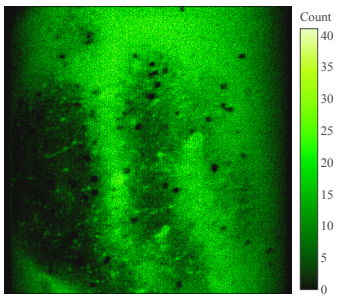
(b) The total number of zero values in each frame. Note the apparent high number and fluctuations.



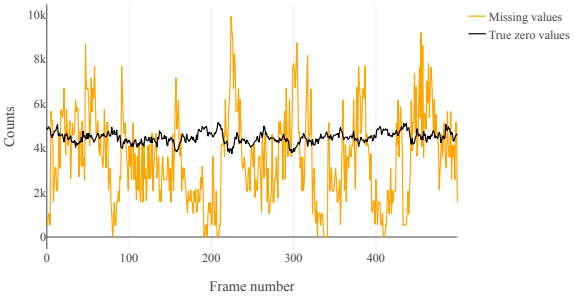
(c) Typical example frame, chosen as the one with the median number of zero observations (frame 44, 7959 zeros). Note the completely missing line-scans, as well as the missing values in the first and last columns.



(d) Example pixel, chosen as the one with the highest number of true zero observations (pixel 307x-61y, 55 total zeros). Without neighbourhood information, it would be impossible to tell apart true 0s from missing data.



(e) Number of true zero observations after removing all zeros flagged as missing data.



(f) Number of true zero and missing data observations. Note that the number of true zeros per frame has much smaller fluctuation, more consistent with sampling variance.

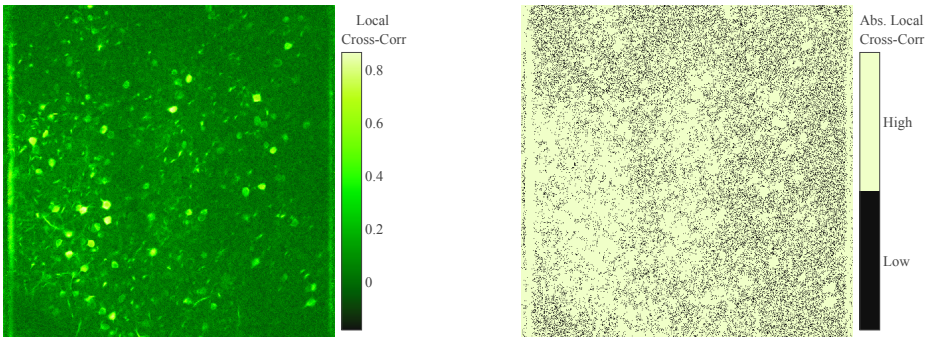


Figure 2.6: (Left) Local cross-correlation computed according to equation 2.53. This measure is robustly high for active neural structures[8], and thus may be used as a means to identify background pixels.
(Right) Thresholded absolute signal – at 0.014 as the threshold, chosen as the 0.15 quantile of all absolute local cross-correlations over space. The black pixels constitute our accepted background locations.

the distribution of zero observations, and the application of the missingness identification procedure described in equation 2.54. In other datasets one needs to worry about saturated pixels as well, whose appearance may be detected by the overpopulation of the histogram's maximum-valued bin.

The next step is to select the locations of background pixels in the dataset, that are expected to act as homogeneous Poisson source of photons, whose temporally constant intensity depends on the location. As described in section section 2.2.4, we may identify neural structures via the shared signal in their multi-pixel extent, as evidenced by the local cross-correlation measure (figure 2.6). Rejecting pixels with high local cross-correlations results in the identification of the required background pixels. As the time courses of individual Poisson sources recorded via extremely noisy machinery is impossible to identify without a model, the assumption of homogeneity will only be examined posthumously, once the model has been fit, which is why I call this background selection a heuristic procedure. In some other datasets the presence of temporally structured low-dimensional signal violating the temporal homogeneity assumption is obvious. This is often a behaviour-induced side-effect, sometimes an undesired one of physical brain movement relative to the microscope, or a desired one of increased brain activity at certain experimental triggers. As described earlier, the presence of such strong global signal should be checked for and removed via SVD or other matrix decomposition methods before this background selection takes place, as neither kind of signal is what we want to correct the data for.

Finally we wish to get simple estimates of certain model parameters that we may use to initialise the model fits. A common method of gain estimation given Poisson inputs is to chart the variance against the mean for each observation. This Fano factor should be 1 for the input Poisson process, therefore any other slope must be caused by the photomultiplier and subsequent electronics. Fit results for dataset 00.00 are shown in figure 2.7 and reported for each dataset in table 2.2.

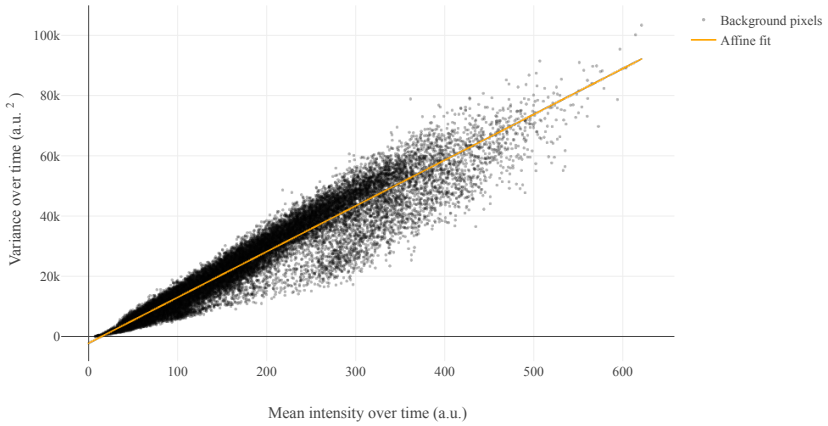


Figure 2.7: Estimation of the photomultiplier and electronics gain and offset using an affine least squares fit to observed variance against mean of each background pixel, excluding missing data.

Code	Missing%	# Training locations	Pedestal Peak±width	Linear offset	Linear gain
00.00	1.346%	39322	7.59 ± 8.27	13.48	149.72
01.00	1.036%	39322	135.11 ± 207.78	136.68	1534.01
02.00	32.15%	39322	0.63 ± 12.84	28.13	299.67
03.00	0.112%	36603	34.20 ± 223.27	-326.03.	990.81
04.00	6.48%	39322	2.12 ± 9.06	6.12	24.43

Table 2.2: Derived statistics of the datasets used

2.3.2 Model fits and dataset correction

Our goals were two-fold; firstly we need to ensure that we estimated parametric model of the photomultiplier well, which enables us to transform the recorded data into photon flux estimates that changes linearly with respect to the calcium reporter signal. Secondly, we aimed to learn a useful description of the spatial gain non-uniformity, which can then be applied as as a divisive correction to the photon flux, resulting in a corrected dataset, which should have a spatially uniform background signal level. Importantly, this correction should remove differences not only within a single field of view, but even across datasets from different microscopes, brain regions, or animals; ultimately making subsequent data analysis methods more standardised ubiquitously applicable, and the scientific conclusions drawn more robust.

To show the extent to which these two goals are achieved by the proposed model as well as the necessary fitting process, I fitted the model to the first 500 frames of each dataset, and evaluated the correction on frames 1001-2000, to minimise the effects of temporal noise correlation.

First I describe in detail the resulting model fit for dataset 00.00, as well as the correction process: first applying the maximum likelihood based non-linear transformation to the data to obtain photon flux estimates from recorded grey values, then applying the spatial gain correction to reduce non-uniformity within the field of view. Afterwards in **??** I show the fitted likelihood parameter values for all datasets, as well as the effect of the corresponding non-linear transformation in table 2.4. Finally for each dataset I show a set of figures that showcase the effects of the grey-to-photon conversion, display the learned spatial gain function, apply the correction and discuss its results in figures 2.8 to 2.12.

The transformation itself ends up being relatively simple due to using maximum a posteriori estimates of both the likelihood and the the spatial gain non-uniformity, but keep in mind that during learning we propagate the full noise distributions through the whole model, leading to a robust fitting procedure. To illustrate this, figure 2.8a shows that given a single observed grey level, what is the conditional probability distribution $p(\nu \,\|\, Y(\mathbf{x},t))$ over the number of photons incident at the photomultiplier (see equation 2.45 for the likelihood model). Given this discrete distribution over photon numbers, we find the maximum a posteriori estimate of the rate of the Poisson distribution that generated the photons, $\nu \sim \text{Poisson}(\lambda)$. This λ rate I will now call 'photon flux', to indicate it is not a discrete photon count, but rather our rate estimate of the process that generated the photons $\nu(\mathbf{x},t)$, and ultimately the observation $Y(\mathbf{x},t)$. This grey level to photon flux estimate is shown in figure 2.8b, where the errors were assumed additive Gaussian in the output space (representing the electrical noise), and again computed as a MAP inverse of the likelihood.

Next I investigated the effect of the resulting non-linear transformation on the data, mainly whether or not the resulting process became more Poisson-like, having a Fano factor of 1 (despite being rate estimate rather than actual photon count ones). I show the mean-variance plots for the original and the transformed data in figure 2.8c-d, and also show the parameters of the affine fits in table 2.4, that shows we can achieve reasonable standardisation across multiple, very different datasets, created in different labs by different equipments and settings.

Lastly I show the need for spatial gain normalisation within a single field of view, by showing the mean image in figure 2.8e, and the mean photon flux image in figure 2.8f. The latter already reveals darker areas better, by learning the strong non-linearity present near the photomultiplier pedestal. These low light levels are typical in biological scanning two-photon imaging application, and thus most datasets are affected by this non-linearity. However, there usually still remains a lot of gain variation present across the field of view, the strongest of which is often induced by non-homogeneity of the scattering tissue (vasculature). This is well illustrated in the learned spatial gain non-uniformity of dataset 00.00, shown in figure 2.8g. an other important effect is circularly symmetric illumination and collection fall-off away from the optical axes towards the edges of the image, which is better shown by figures 2.9 to 2.11g. Finally there often is strong non-uniformity aligned along the scanning direction, shown in figure 2.12e-g. We wish to design priors that are

capable of correcting non-uniformities that we do wish to correct (such microscope artefacts or vasculature shadowing), without inadvertently explaining away features of the data that we did not want to. As discussed earlier in section 2.2.2, our main computational tools for prior design for Gaussian Processes include choice of kernel type, inducing symmetries and fixing (or setting strong hyperpriors) for spatial lengthscales.

The mean images of the spatially gain corrected and photon flux transformed datasets are shown in figures 2.8 to 2.12h, that showcase the strength and potential weaknesses of the method. Note that I applied the exact same hyperparameter settings, initialisation and priors to all five vastly different datasets. The too long spatial lengthscales used were unable to perfectly capture the strong edge effect in dataset 04.00, resulting in overamplification near the scan-aligned edges, but otherwise the results are fairly uniform.

To more precisely describe the effects of the spatial gain normalisation, I evaluated its effects only at the background pixel locations. Furthermore, as the single-observation photon-flux transform induces a difficult-to-understand noise distribution, I instead estimated a single $\lambda(\mathbf{x})$ rate per background location, calculating the maximum a posteriori estimate $\hat{(}\lambda(\mathbf{x}))$ given all test frames $Y(\mathbf{x},t \in [1001,2000])$. This makes again our original assumption, that the only variation in background observations are due to Poisson and additive noise, but the signal is generated at a fixed rate $\lambda(\mathbf{x})$. The histogram of the estimated $\hat{(}\lambda(\mathbf{x}))$ values over all $\mathbf{x}$ background locations is shown in figure 2.8i. To illustrate the effects of the spatial gain correction, I also overlay the histogram of the gain-corrected $\hat{(}\lambda(\mathbf{x}))/G(\mathbf{x})$ values, where I matched the mean of the histograms to enable direct comparison. We can see that in all datasets the spatial gain correction significantly reduces the spread of the histogram, showing that the spatially gain-corrected background is indeed much more uniform.

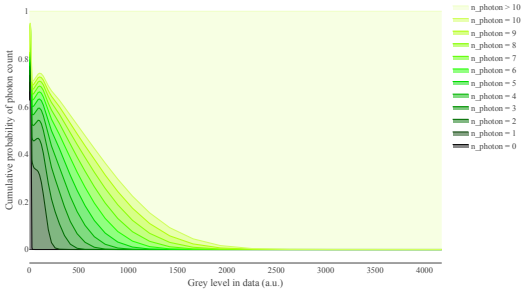
Code	Pedestal ± Pedestal Noise std	Under-amplification probability * Amplitude	Gain per photon ± Noise std	Saturation grey value (photon flux)
00.00	8.67 ± 4.87	$0.30 * 51.56$	94.89 ± 53.23	3822 (40.19)
01.00	137.30 ± 64.44	$0.30 * 142.00$	699.02 ± 561.91	8191 (11.52)
02.00	2.77 ± 9.64	$0.32 * 91.00$	180.45 ± 85.82	5872 (32.53)
03.00	36.45 ± 69.12	$0.05 * 128.24$	777.39 ± 353.17	8191 (10.49)
04.00	4.13 ± 7.37	$0.03 * 22.26$	36.11 ± 10.11	6418 (177.64)

Table 2.3: Learned likelihood parameters for underamplified Poisson likelihood

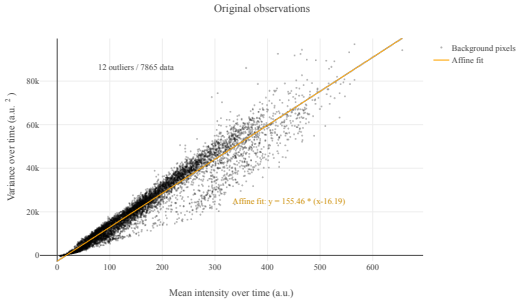
Code	Orig Offset	Orig Gain	Photon offset	Photon Gain
00.00	16.19	155.46	0.00	1.35
01.00	137.17	1524.11	-0.29	1.49
02.00	35.27	331.94	0.16	1.62
03.00	-246.82	1024.69	-0.47	1.14
04.00	19.29	27.89	0.26	0.71

Table 2.4: Original and likelihood-corrected (photon) gain and offset estimates

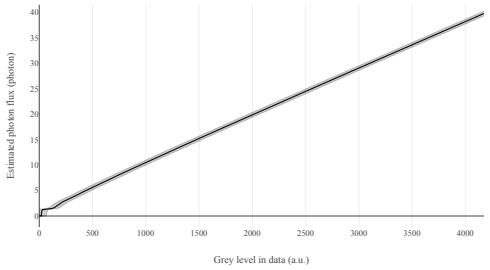
Figure 2.8: Evaluating model-based data correction for dataset 00.00.
(a) Conditional distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data (excluding ”missing” data).
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image (excluding missing data).
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all test frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matches histograms are shown.



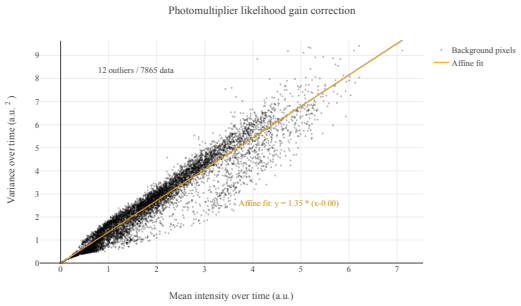
(a) Conditional distribution over photons at grey levels.



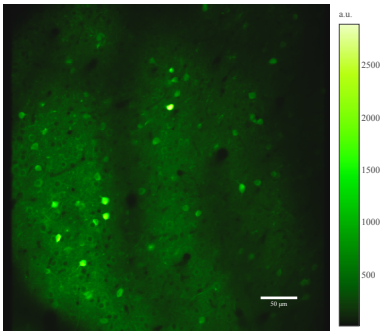
(c) Mean-variance plot in original data per background pixel.



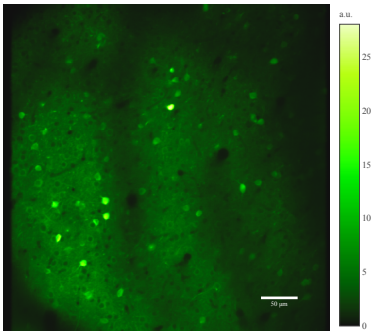
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



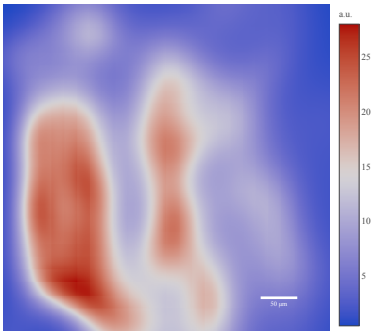
(d) Mean-variance plot of photon flux estimates per background pixel.



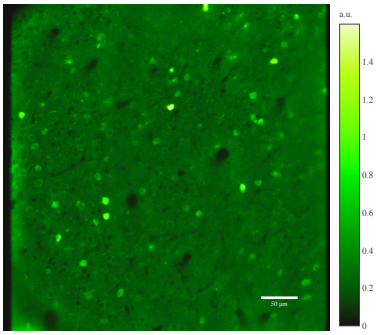
(e) Original mean image over frames (excluding missing data)



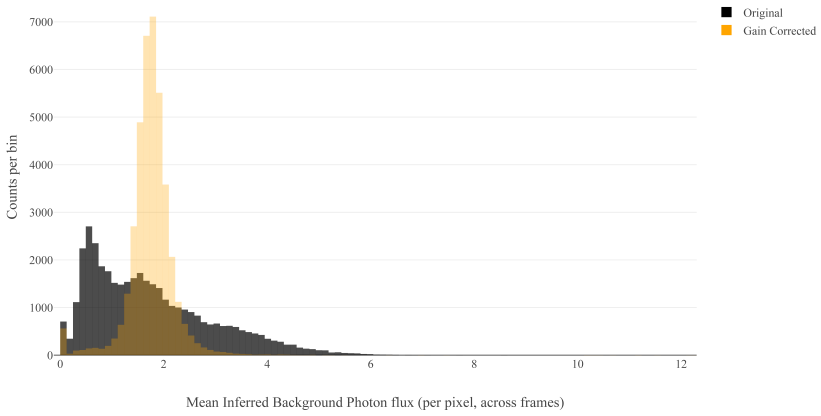
(f) Temporal mean after transformation into photon flux estimates



(g) Learned spatial gain non-uniformity.

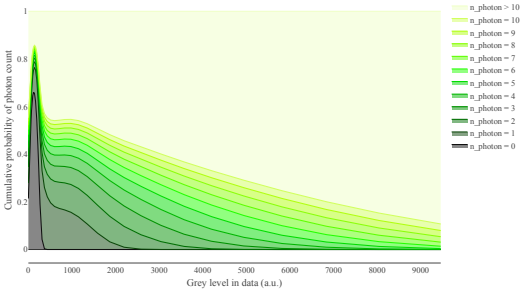


(h) Temporal mean after both likelihood and spatial gain transformation.

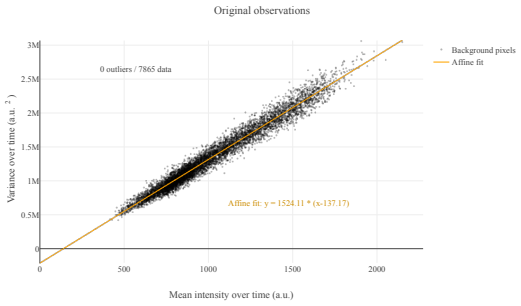


(i) Mean-matched histograms that show the effect of spatial gain correction.

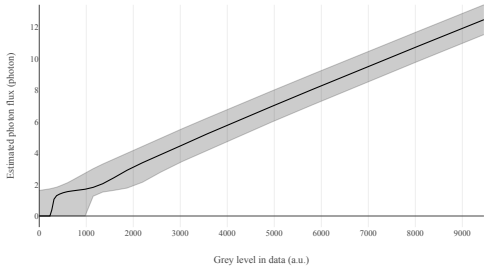
Figure 2.9: Evaluating model-based data correction for dataset 01.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data (excluding "missing" data).
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image (excluding missing data).
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all test frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matches histograms are shown.



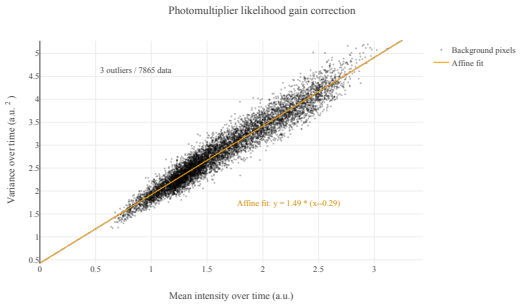
(a) Marginal distribution over photons at grey levels.



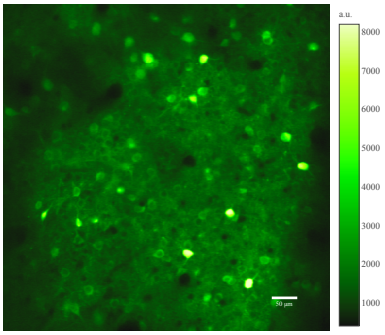
(c) Mean-variance plot in original data per background pixel.



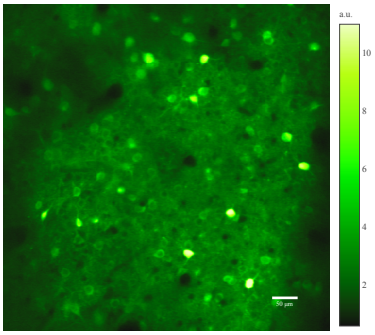
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



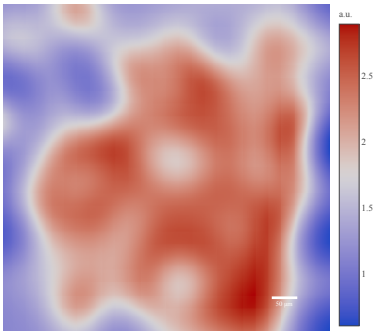
(d) Mean-variance plot of photon flux estimates per background pixel.



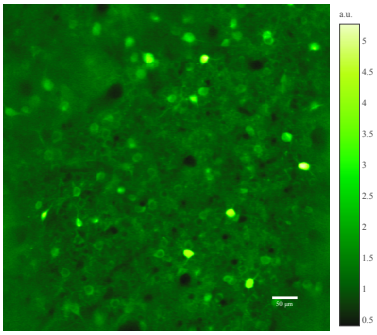
(e) Original mean image over frames (excluding missing data)



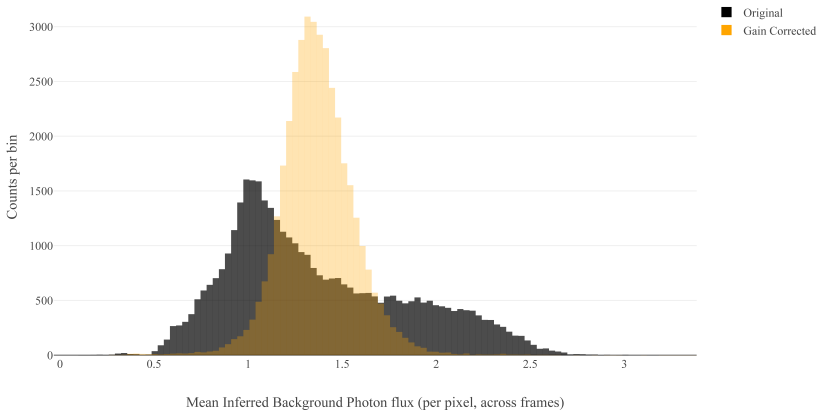
(f) Temporal mean after transformation into photon flux estimates



(g) Learned spatial gain non-uniformity.

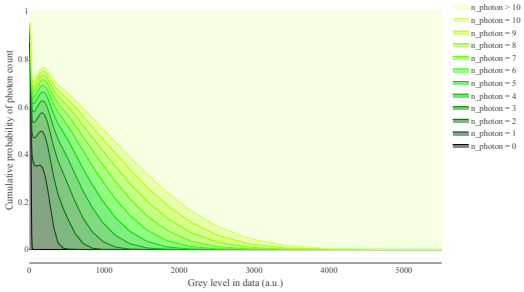


(h) Temporal mean after both likelihood and spatial gain transformation.

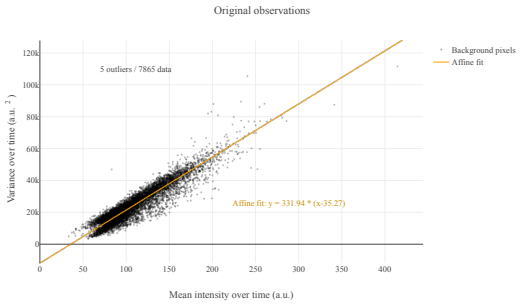


(i) Mean-matched histograms that show the effect of spatial gain correction.

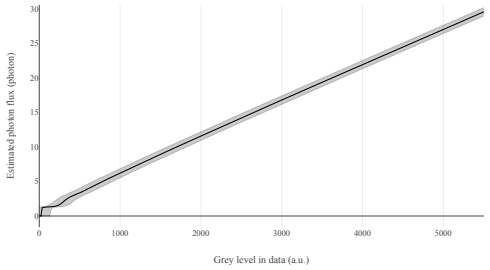
Figure 2.10: Evaluating model-based data correction for dataset 02.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data (excluding "missing" data).
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image (excluding missing data).
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all test frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matches histograms are shown.



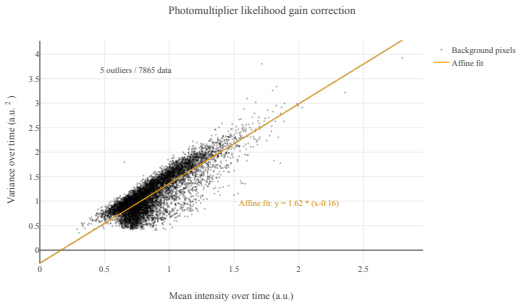
(a) Marginal distribution over photons at grey levels.



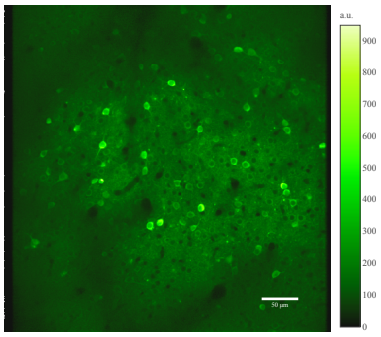
(c) Mean-variance plot in original data per background pixel.



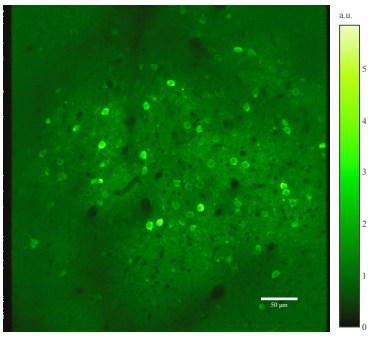
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



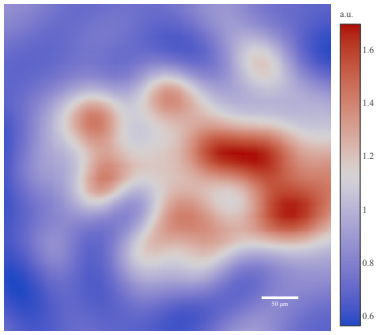
(d) Mean-variance plot of photon flux estimates per background pixel.



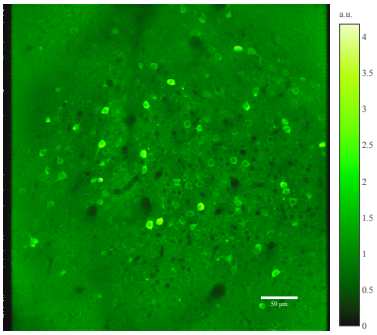
(e) Original mean image over frames (excluding missing data)



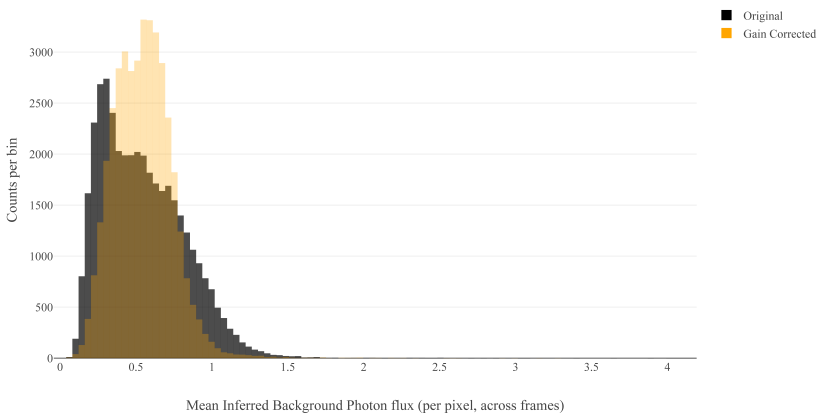
(f) Temporal mean after transformation into photon flux estimates



(g) Learned spatial gain non-uniformity.

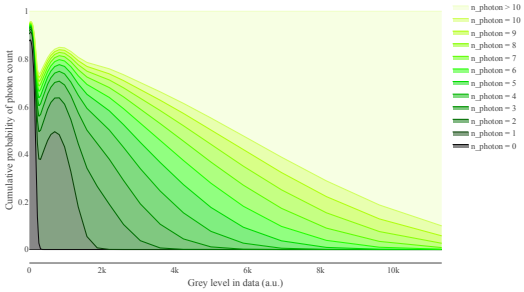


(h) Temporal mean after both likelihood and spatial gain transformation.

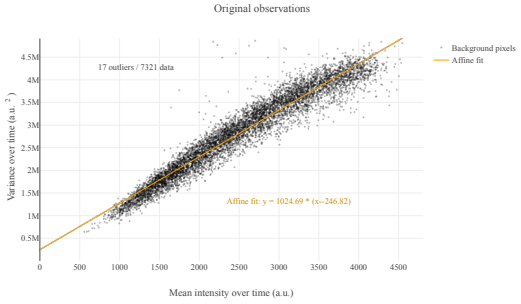


(i) Mean-matched histograms that show the effect of spatial gain correction.

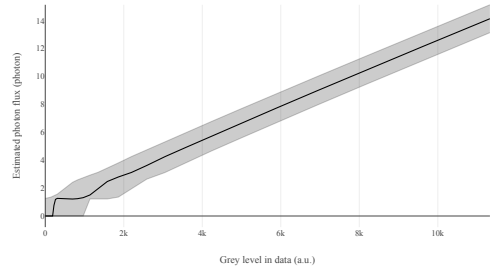
Figure 2.11: Evaluating model-based data correction for dataset 03.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data (excluding "missing" data).
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image (excluding missing data).
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all test frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matches histograms are shown.



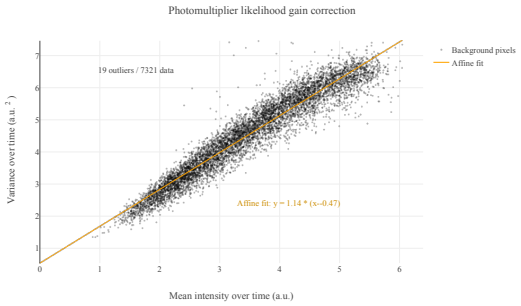
(a) Marginal distribution over photons at grey levels.



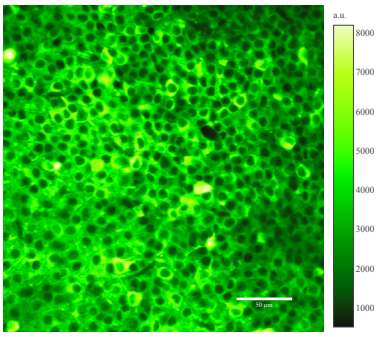
(c) Mean-variance plot in original data per background pixel.



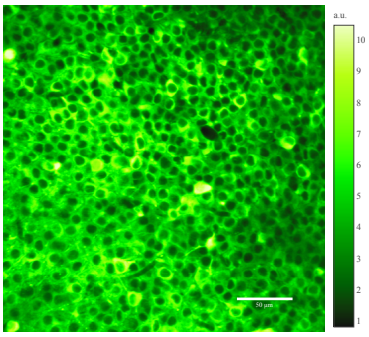
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



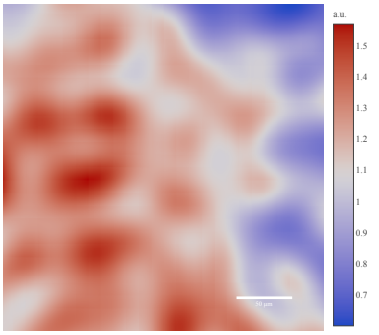
(d) Mean-variance plot of photon flux estimates per background pixel.



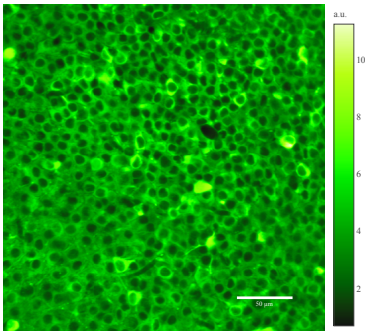
(e) Original mean image over frames (excluding missing data)



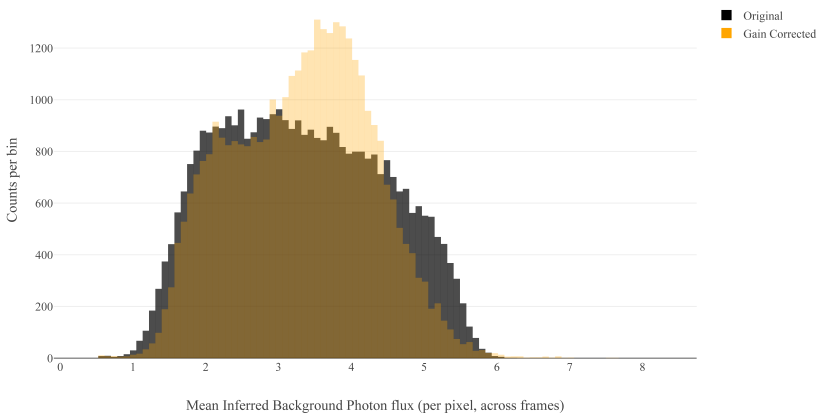
(f) Temporal mean after transformation into photon flux estimates



(g) Learned spatial gain non-uniformity.

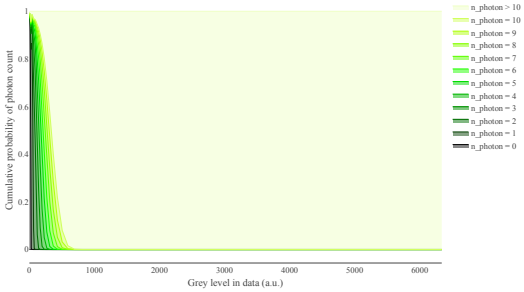


(h) Temporal mean after both likelihood and spatial gain transformation.

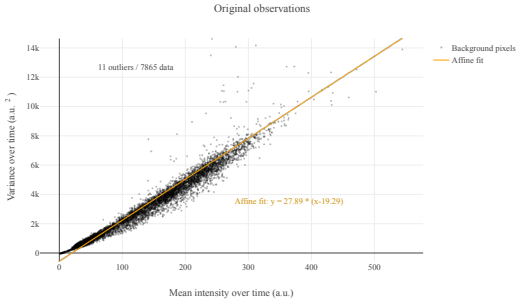


(i) Mean-matched histograms that show the effect of spatial gain correction.

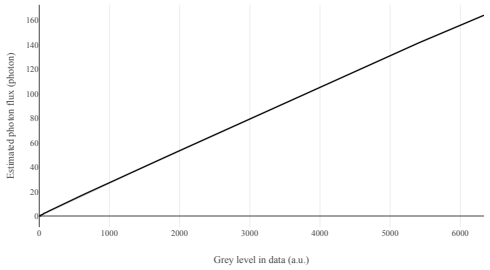
Figure 2.12: Evaluating model-based data correction for dataset 04.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data (excluding "missing" data).
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image (excluding missing data).
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all test frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matches histograms are shown.



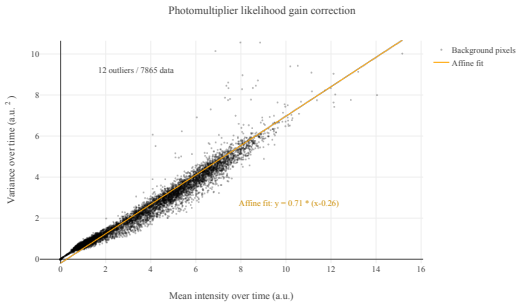
(a) Marginal distribution over photons at grey levels.



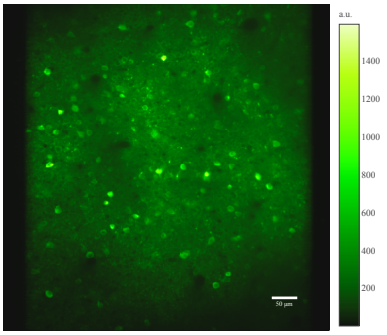
(c) Mean-variance plot in original data per background pixel.



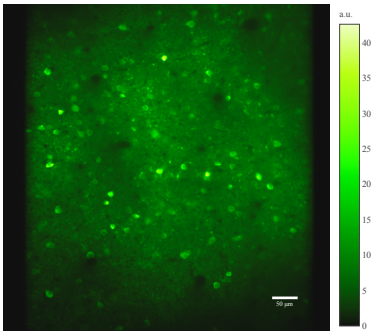
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



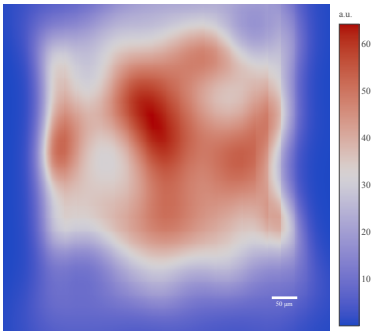
(d) Mean-variance plot of photon flux estimates per background pixel.



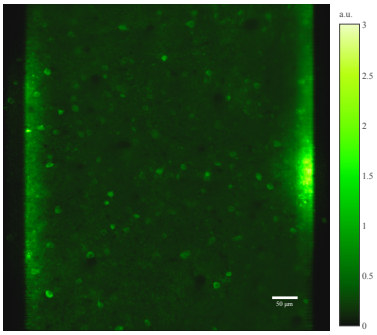
(e) Original mean image over frames (excluding missing data)



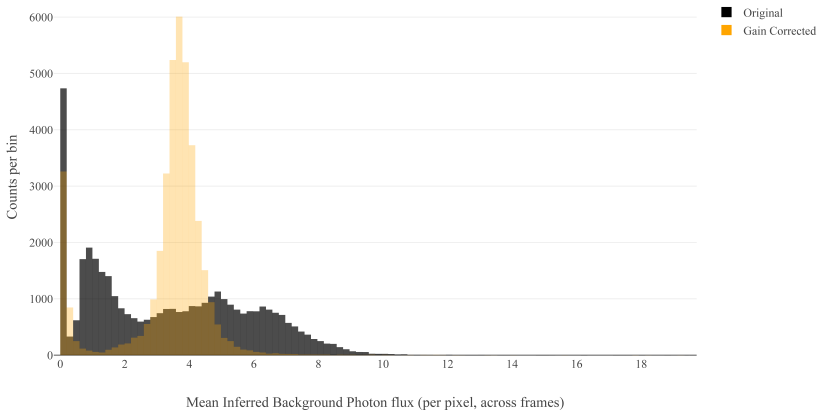
(f) Temporal mean after transformation into photon flux estimates



(g) Learned spatial gain non-uniformity.



(h) Temporal mean after both likelihood and spatial gain transformation.



(i) Mean-matched histograms that show the effect of spatial gain correction.

2.4 Discussion

Talk about

- model fit discrepencies
- Measuring parts of the system and putting them in as strong priors to the model
- Validating the model fits on physical measurements in labs

Chapter 3

Segmentation of neural cell bodies via Convolutional Higher Order Matching Pursuit (CHOMP)

3.1 Calcium imaging and segmentation of individual neurons

A major goal at the interface of computational and experimental neuroscience is to gain direct and robust access to single neurons' activity, in the hopes of testing existing models of neural computations, constraining the space of possible computational models, and potentially sparking the development of new ones via discovering novel phenomena implemented via neural population codes. Providing such access has long been a joint effort between developing novel experimental techniques, as well as the matching data analytic tools which can robustly extract single neurons' activity from the raw datasets provided. An excellent discussion of such collaboration for extracellular electrophysiological recordings and the corresponding analytic technique of spike sorting is found in (Cite: Past, present and future of spike sorting techniques spike sorting Rey 2015 https://www2.le.ac.uk/centres/csn/publications-1/Publications/publications/publication-pdfs-2015/past-present-future-spike-sorting). Here I focus on a more recent, but now wide-spread experimental technique, two-photon imaging of genetically encoded fluorescent calcium reporters, or *calcium imaging* for short. Similarly to the spike sorting question, calcium imaging has spawned its own set of analytic needs. A recent review of the current algorithmic needs – including motion correction and spike deconvolution – can be found in (Cite: Carsen-Marius recent review), and in this chapter I address the question of localising single neurons and extracting their activity[1] over time, assuming the datasets have already been sufficiently motion corrected, and potentially the

[1] It should be noted that this activity is not necessarily a direct result of electrical activity, but instead reflects the concentration of calcium ions in the cytoplasm. Although there indeed is strong correlations between electrical activity, the amount of free calcium and the emitted fluorescent signal, cells are also known to strongly regulate calcium concentrations via intracellular stores. Therefore the insights gained from calcium imaging may differ from those obtained via direct electrical recordings, and should be interpreted with care.

various non-uniformities have been mitigated, as in chapter 2.

This process is generally called the extraction of Regions of Interest (ROIs), with the name reflecting the standard procedure of converting a calcium imaging field of view into scalar activity estimates belonging to individual neurons: Small binary masks are placed onto the field of view by a human marker, each reflecting the spatial extent of a neuron, then pixel values within these ROIs are summed to reflect the neural activity. The two main issues with this procedure are that a.) observed pixel values within ROIs often contain signal from other neurons or out-of-focus fluorescence due to the imperfect optical sectioning of the microscope, and b.) that human markers processing datasets have various learned biases, and can also be strongly influenced by changing visualisations (the receiver operating characteristic curve of manual segmentation across multiple human markers is shown in (Cite: marius 2013 nips donuts)). These issues led to the ongoing development of various algorithms attempting to automatically decompose the observed fluorescence into sums of signals generated by individual – potentially overlapping – neurons, as well as background or other contaminations, these are also called 'demixing' algorithms in the field. The need for widely applicable algorithms for extracting ROIs from calcium imaging recordings is well illustrated by the existence and popularity of the Neurofinder[2] challenge: A collaborative effort from multiple experimental labs submitting calcium imaging datasets and segmentations[3] into the public domain, along with convenient automatic testing of ROI extraction algorithms' performance on all provided datasets.

I thus first provide an overview of existing methods, and discuss their main ideas, strengths and weaknesses. Then I introduce my take on the problem, Convolution Higher Order Matching Pursuit (CHOMP), a demixing algorithm based on a novel extension to earlier matching pursuit methods. I presented its neural application at the 2016 Gatsby Tri-Center Meeting, and the inference method at MLSP2016 along with the supporting conference paper (see (Cite: MLSP paper)).

Overview of calcium imaging ROI extraction algorithms

The algorithms presented here generally agree upon the fact that one needs to simultaneously estimate the spatial extent as well as the activity time courses of 100s of neurons, in the face of significant noise and signal contamination. However, they also all acknowledge the fact that this problem is both computationally intractable and the solution is not well regularised, making it a very difficult problem. Therefore what these algorithms do differ in, is the type of approximations they make to arrive at a tractable problem, and the types of constraints they put on the potential spatial extents as well as the time courses characteristics of individual neurons.

[2] `http://neurofinder.codeneuro.org/`

[3] It is unknown how these target segmentations were created, but it is likely that some datasets were segmented by multiple human markers and the ground truth was agreed upon, whereas in others a genetically encoded nuclear marker may have led to cell localisation.

Spatio-temporal constraints. The constraints placed on the spatio-temporal characteristics fit into three broad categories. Some methods place no constraints (such as PCA, SVD), but they tend to arrive at mixtures of sources rather than separating them. Placing spatial and temporal sparsity constraints leads to the family of independent component analysis methods (stICA, see (Cite: Mukamel2009, also Hyvarinen FastICA 2000)), whereas non-negativity constraints led to non-negative matrix factorisation methods (NMF (Cite: NMF based methods)). These ideas help with separating signals from neurons that behave differently, however still mixes signals from correlated sources, as it doesn't take into account the physical shape of neurons. One may add more constraints to the matrix factorisation style methods to further limit the potential spatial filters' shapes ((Cite: constrained NMF)), whereas the use of templates or basis functions with limited spatial support lead to 'convolutional' methods ((Cite: donuts, suite2p, conv MP in general)). Finally, a popular form of temporal constraining includes representing only certain features of the individual pixels' activity, such as the maximum, the mean ((Cite: donuts)), or the cross-correlation with nearby pixels ((Cite: suite2p))..

check if examiners have published similar stuff, make sure to cite if yes

Tractability. Such feature representations are not only useful to reduce noise or represent desirable neural characteristics, but in fact they significantly reduce the computational complexity of the methods. Indeed, separating spatial source localisation and temporal reconstruction into two subsequent problems is a common way of achieving tractability. As mentioned above, source localisation is often carried out a reduced feature set, but is nevertheless a difficult problem to simultaneously detect all sources that each adhere to all spatio-temporalconstraints. Often the source detection problem comes down to minimising a cost function that is the weighted sum of a reconstruction loss and the number of sources used, essentially an L0-regularised regression. Again, that is a difficult cost function to find global optima of, and therefore two types of approximate inference methods exist. The so-called L1-relaxation results in solvable problems via replacing the L0 cost with an L1 one on reconstruction weights, and is applicable to both matrix factorisation methods ((Cite: L1 regularised ICA and NMF)) as well as template-based convolutional methods ((Cite: Group Lasso)). An other way of approximating the L0-regularised regression problem is via greedy iterative reconstruction of sources that maximise reconstruction loss, and in general called (convolutional) matching pursuit methods ((Cite: donuts, suite2p)). As my novel algorithm – Convolutional Higher Order Matching Pursuit – belongs to this family of methods, I now review in more detail matching pursuit and its various extensions that make it more applicable to the neural source detection problem.

Check that spelling is consistent everywhere

3.1.1 Matching pursuit methods

The idea of basic matching pursuit is that given a recorded signal $\mathbf{y} \in \mathbb{R}^{\mathcal{I}}$, and multiple templates $\{\mathbf{b}^k \in \mathbb{R}^{\mathcal{I}}\}_{k=1}^{K}$ with the same dimensions as the observations, can we find the optimal reconstruction coefficients $\hat{\mathbf{x}} \in \mathbb{R}^{K}$, such that

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\|\mathbf{y} - \mathbf{x}^\top \mathbf{B}\|^2 + \lambda \|\mathbf{x}\|_0 \right), \tag{3.1}$$

where $\mathbf{B}$ is the collection of the $\mathbf{b}^{\mathrm{k}}$ templates (also called basis functions or dictionary elements). Matching pursuit methods approximates this problem by initialising $\mathbf{x} = \mathbf{0}$ and activating elements x_{k} one by one, by iteratively solving for the best k and x_{k} that maximally reduces the reconstruction cost:

Algorithm 3.1 Matching Pursuit algorithm pseudocode

1: Let $\hat{\mathbf{x}}^0 = \mathbf{0}$ be the empty vector of basis coefficients, $j = 0$ the current iteration, and $\mathbf{B}$ the template matrix.
2: **repeat**
3: $j \leftarrow j + 1$.
4: Calculate the current residual $\tilde{\mathbf{y}}^{\mathrm{j}} = \mathbf{y} - (\hat{\mathbf{x}}^{\mathrm{j}-1})^\top \mathbf{B}$.
5: For each unused k, solve for the best reconstruction of the residual via $x_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}$, and compute potential reconstruction cost change $\Delta\mathcal{C}^{\mathrm{reconst}}_{\mathrm{jk}}$, were we to use the $k-th$ basis function.

$$\begin{aligned} \forall_{\mathrm{k}} \text{ where } \hat{x}^{\mathrm{j}-1}_{\mathrm{k}} &= 0 \\ x^{\mathrm{j}}_{\mathrm{k}} &= \arg\min_{\mathrm{x_k}} \|\tilde{\mathbf{y}}^{\mathrm{j}} - x_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}\| \\ \Delta\mathcal{C}^{\mathrm{reconst}}_{\mathrm{jk}} &= \|\tilde{\mathbf{y}}^{\mathrm{j}} - x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}\|^2 - \|\tilde{\mathbf{y}}^{\mathrm{j}}\|^2 \\ &= (x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}})^\top (x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}) - 2(x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}})^\top \tilde{\mathbf{y}}^{\mathrm{j}} \end{aligned} \tag{3.2}$$

6: Let $\hat{x}^{\mathrm{j}}_{\mathrm{k}} = \arg\min_{x^{\mathrm{j}}_{\mathrm{k}}} \Delta\mathcal{C}^{\mathrm{reconst}}_{\mathrm{jk}}$ be the newly activated basis function on iteration j
7: **until** $\Delta\mathcal{C}^{\mathrm{reconst}}_{\mathrm{jk}} + \lambda > 0$ for all k, and thus adding a source does not decrease the total cost $\mathcal{C}$.
8: Return $\hat{\mathbf{x}}^{\mathrm{j}-1}$ as the approximate matching pursuit solution to equation 3.1.

Although a rather general algorithm, this basic matching pursuit formulation is not well-applicable to the neural source segmentation problem. The main issue comes from defining a suitable collection of basis functions $\mathbf{B}$, that the algorithm keeps fixed, and relies upon. This dictionary of basis functions needs to be able to sufficiently represent the data, but we would also like each $\mathbf{b}^{\mathrm{k}}$ template to describe exactly one neuron. We first need to address the issue, that due to an unknown, but large number of neurons present in the data at a priori unknown locations, and potentially with varying shapes. If we'd require a new basis $\mathbf{b}^{\mathrm{k}}$ for each possible neuron shape and each possible location, we'd end up with an extremely large K, leading to an intractable algorithm. Lastly, this algorithm assumes we have a single sample $\mathbf{y}$ to reconstruct, not the typically available $\mathbf{y}(t)$ time course of activity. This last issue is generally addressed by a feature representation suited to the individual problem, where $\mathbf{Y} \in \mathbb{R}^{\mathcal{I} \times \mathrm{d_f}}$, with d_{f} being the number of features, and basis functions are extended to cover the whole of this space.

The other issue of needing a very large number of K basis to represent all potential source

settings is addressed by two more formal extensions to the basic matching pursuit algorithm, *convolutional* and *block* matching pursuit ((Cite: CMP and block CMP - donuts)). The convolutional extensions assumes that the support of the individual basis functions ($\mathcal{P}$) is smaller than the observation space $\mathcal{I}$, and the full dictionary $\mathbf{B}$ is effectively created as a convolution of the small basis function with the potential locations. However, computationally we never need to represent the full $\mathbf{B}$ convolutional dictionary, but rather organise the reconstruction coefficients $\mathbf{x}$ such that $\mathbf{x} \in \mathbb{R}^{\mathcal{I}\times K}$, ie. we represent the basis function activations at every potential location[4]. The reconstruction in step 4 of algorithm 3.1 is now computed as $\sum_k \mathbf{b}_k * [\hat{\mathbf{x}}^{j-1}]_{\cdot k}$, and we need to accordingly adjust computing x_k^j and $\Delta\mathcal{C}_{jk}^{reconst}$ in step 5, but otherwise the algorithm remains the same. Block matching pursuit is a both mathematically and computationally straightforward addition, which changes the L0 penalty term of equation 3.1 to $\|\sum_k \mathbf{x}_{\cdot k}\|_0$. Conceptually, however, it allows us to further reduce the dictionary size K by enabling us to use multiple basis activations at the same location without additional cost. This means that potential cell shapes at a location are not described by a single $\mathbf{b}^k$ template (requiring many templates to account for all possible shapes), but rather as a linear combination of the templates, $\sum_k \mathbf{x}_{\cdot k}\mathbf{b}^k$, capable of describing numerous shapes with only a few templates. The complete cost function of convolutional block matching pursuit is thus

$$\mathcal{C} = \left(\left\| \mathbf{y} - \sum_k \mathbf{b}_k * [\mathbf{x}]_{\cdot k} \right\|^2 + \lambda \left\| \sum_k \mathbf{x}_{\cdot k} \right\|_0 \right) \tag{3.3}$$

and the approximate matching pursuit solution $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathcal{C}$ can still be computed by algorithm 3.1, with small modifications to step 4 and 5.

My algorithm, Convolutional Higher Order Matching Pursuit – described in the next section – uses this idea in essence, with a feature set $\mathbf{Y}$ specifically designed to match interesting neural characteristics, while enabling us to exploit the computational advantages of Convolutional Block Matching Pursuit. The requirement of a – now much simpler – dictionary of basis functions $\mathbf{B}$ as input still stands, and I discuss in section 3.2.2 how one might initialise and update it via dictionary learning.

[4] It is worth mentioning that the set of potential source locations need not coincide with the observation locations $\mathcal{I}$ for convolutional matching pursuit to be applicable. To my knowledge this has not been derived before, and thus I introduce this novel possibility in section 3.2.

3.2 Convolutional Higher Order Matching Pursuit and Dictionary Learning for segmenting neurons

3.2.1 Convolutional Higher Order Matching Pursuit

The aim of Convolutional Higher Order Matching Pursuit (CHOMP) is to carry out sparse signal decomposition of multidimensional, repeated signal measurements, where the measured signal dimensions have some built-in structure; in the case of our main application – calcium imaging – this would be pixels or voxels organised in 2 or 3 dimensions on an grid. Therefore from here onwards I will refer to this structure as measurement 'space' or 'spatial structure' and the individual measured dimensions as being 'locations', although the structure of measurements may well be of spatiotemporal nature, in which our repeated measurements are multiple time series.

In this section I first discuss the algorithm rather generally, without explicitly specifying this spatial structure. However, as our main application concerns imaging, and thus evenly spaced grids as our measurement structure, my most mature implementation indeed assumes such grids and I will describe in detail the computational tricks I use to significantly speed up the algorithm, if such assumptions can be made.

Generative model

In order to understand the problem this algorithm solves, I first describe the generative model. This is based on the idea of the measurements are generated by a set of sources, that have fixed locations and small spatial extents[5], but varying activity across samples. The maximum spatial extent of a source I call a 'patch', and we assume that the true signal within the individual patches is well-described by a linear combination of basis functions – or so-called 'basis-patches' – that correspond to valid signal 'modes', assumed to be caused by the unknown spatial 'shape' of the signal-generating source.

Each signal sample is thus made up of a sum of the signals inside the individual patches around the source locations. Measurements of this total signal are then taken at each measurement location, corrupted by additive noise. Note that the signal locations and the measurement locations do not necessarily correspond to one another. This measurement process – illustrated in figure 3.1 – is then repeated multiple times, resulting in multiple measured samples of the underlying signal. The individual samples are assumed to be independent from one-another[6].

Check footnote with Maneesh

In technical terms, let the data be generated by a set of S sources $\mathsf{O} = \{O^s\}_{s=1}^{S}$ located within

[5] Just noting one last time, that in general these locations and extents may be in space-time rather than just space only, but that would complicated the language more, and do not apply in our main application or implementation regarding calcium imaging.

[6] This is not generally the case in time series measurements, such as our calcium imaging videos. However, if the time-varying signal is stationary over time and sampled less frequently than its auto-correlation length, our independence assumption is fairly satisfied. We can achieve this by sufficiently downsampling high-frequency time-series measurements.

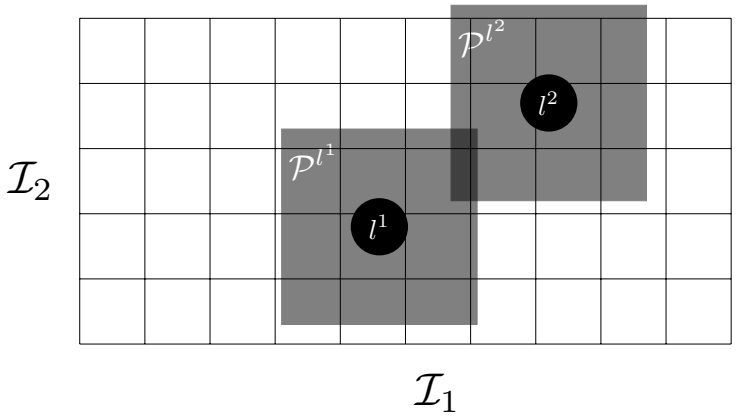


Figure 3.1: Source locations and patches embedded in a grid measurement space. The patches might overlap, or extend over the edges of the measurement space, as illustrated. Black circles indicate the source locations, the grey boxes show the patches that the individual sources affect and the dark grey area indicates overlap between the patches, where the signals from the two sources are added. The grid represents a measurement grid, and the observed data is the signal collected within each square, corrupted by additive noise.

a D-dimensional space. Each source generates a measurable signal in a finite extent D-dimensional patch around its centre $\mathbf{l}^{\mathrm{s}} \in \mathbb{R}^{\mathrm{K}}$. The signal is described by a linear combination of K known basis elements $\mathsf{B} = \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}}$ with weights $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$ for each basis patch. The bases $b^{\mathrm{k}} : \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}$ are each stationary functions with finite cuboidal support $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times \ldots \times \mathcal{P}_{\mathrm{D}}$, where $\mathcal{P}_{\mathrm{i}} = [-m_{\mathrm{i}}, m_{\mathrm{i}}]$ are closed intervals around zero, and thus

$$b^{\mathrm{k}}(\mathbf{r}, \mathbf{l}^{\mathrm{s}}) = \begin{cases} b^{\mathrm{k}}(\mathbf{r} - \mathbf{l}^{\mathrm{s}}) & \text{for } (\mathbf{r} - \mathbf{l}^{\mathrm{s}}) \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

We obtain T noisy sample measurements $\{\mathbf{y}^{\mathrm{t}}\}_{\mathrm{t}=1}^{\mathrm{T}}$, with conserved object locations but possibly variable signals. Our goal is to recover the source locations and infer the corresponding (time-)varying[7] signals $\{\mathbf{x}^{\mathrm{s,t}}\}$ for each source $O^{\mathrm{s}} = \{\mathbf{l}^{\mathrm{s}}, \mathbf{x}^{\mathrm{s,t}}\}$.

We write $\mathbf{y} \in \mathbb{R}^{|\mathcal{I}| \times \mathcal{T}}$ for the entire collection of measurements, where $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \ldots \times \mathcal{I}_{\mathrm{D}}$ is a set of D-tuples indexing discrete measurement locations in the D-dimensional data space [8]. Our measured data $\mathbf{y}$, given the bases B and the sources O, is thus described as:

[7] From here onwards I will refer to the signals changing over samples as them having time courses (as they do in calcium imaging), but this need not generally be the case.

[8] Note that in experimental setups the measurement locations themselves would have an effective spatial extent within the space, rather than being point measurements. However, in current technical description, the effective spatial area of signal collection can be represented within the basis elements $b^{\mathrm{k}}(\cdot)$ themselves – effectively a convolution of the true signal mode with the measurement extent – and thus we can think of the measurements, as if collected at point-like locations.

$$\begin{aligned} &\forall \mathbf{l} \in \mathcal{I}, \quad t \in \{1 \dots T\} \\ &\tilde{y}^{\mathbf{l},\mathrm{t}}(\mathsf{O},\mathsf{B}) = \left(\sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{k}=1}^{\mathrm{K}} b^{\mathrm{k}}(\mathbf{l}-\mathbf{l}^{\mathrm{s}}) x_{\mathrm{k}}^{\mathrm{s,t}} \right) \\ &\mathbf{y}^{\mathrm{t}} = \tilde{\mathbf{y}}^{\mathrm{t}}(\mathsf{O},\mathsf{B}) + \boldsymbol{\varepsilon}^{\mathrm{t}}, \end{aligned} \tag{3.5}$$

where $\boldsymbol{\varepsilon}^{\mathrm{t}} \sim \mathcal{N}(\mathbf{0},\ \mathbf{C} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|})$ is additive Gaussian noise, potentially correlated across measurement locations $\mathcal{I}$, but independent over samples t.

Check with Maneesh

Cost function

Based on the generative model described, for a given number of sources, the inference problem can be described as two joint inferences between finding the source locations and then estimating the corresponding signal time courses at each location. A further issue is estimating the unknown number of sources in the first place.

As described earlier, matching pursuit methods find balance between the reconstruction error and the penalty for number of objects used by starting with 0 sources and maximum error, and adding sources iteratively, until the cost of adding a new source is not offset anymore by a large enough reduction in reconstruction error, so that it reduces the total error.

Let our cost function thus be the sum of the reconstruction error – that is defined on the original data $\mathbf{y}$ and the current estimate $\hat{\mathbf{y}}$ – and the cost of the used sources – which is purely on the current setting of $\hat{\mathsf{O}}$:

$$\mathcal{C} = \mathcal{C}^{\text{reconst}}(\mathbf{y}, \hat{\mathbf{y}}) + \mathcal{C}^{\text{sources}}(\hat{\mathsf{O}}). \tag{3.6}$$

Notice, that in this formulation the inference of spatial location of sources and the time courses of their signals have to be done jointly each time we add a new source, making this a rather complicated, and often intractable problem. My main observation within this problem setting was that existing algorithms simplify the problem by either disregarding the multidimensional spatial structure (describing the observation dimensions as a single vector) and doing sparse matrix decomposition on the resulting space-time matrix (Cite: PCA, NMF-like methods); or separating the problem of joint inference of location and time course into first finding the source locations based on a simpler feature representation, and then given the source locations they solve the now much simpler full reconstruction problem. However this simpler representation so far has been achieved by reconstructing only the mean across all samples (Cite: marius' paper), and therefore sources with low mean signal may be ignored by the source localisation carried out on the mean representation. This is especially a crucial issue for neural applications, where neurons often fire very sparsely, and thus may end up with low mean signals over long recordings, and subsequently missed during source localisation.

Convolutional Higher Order Matching Pursuit (CHOMP) creates a framework based on these

earlier ideas, that suits neural segmentation more closely based on the known biophysical properties of neurons in calcium imaging recordings. We wish to keep the confined spatial representation provided by small basis functions, and also the linear combination of those bases to describe the variations of signal modes within neurons. However, as the full joint localisation and time course reconstruction is generally intractable, CHOMP also utilises the idea of localising sources via reconstructing a simplified feature representation. As pointed out, using solely the mean signal as our feature to find our neurons may lead to missing a number of them, and thus I use higher order features that are inherent to time courses of neural calcium reporters. We expect the cytoplasmic calcium signal in active neurons to exhibit higher variance than background signals, as well as being more skewed towards higher values, and more kurtotic due to the sparse firing. CHOMP thus extracts these features from the provided samples, and finds source locations that display the expected neural properties, rather than just a high mean signal.

Therefore we first need to extract these higher order feature statistics from the data. As we wish to retain the additivity of the individual signals for spatially overlapping sources, our feature representation needs to operate in the space of cumulants. Furthermore, due to the non-pointlike nature of our sources, we need to retain not only the pointwise cumulants at each observation location, but also the joint cumulant tensors (such as the covariance matrix) for observations that are within the same patch $\mathcal{P}$ and thus could be jointly influenced by a single source. The unbiased, minimum-variance empirical cumulant estimator is the multivariate K-statistics (Cite: relevant paper). In order to compute it, we first need to define the unnormalised, non-central vector moments of the observed data, as

$$\mathbf{S}_{\mathrm{r}} = \sum_{\mathrm{t}=1}^{\mathrm{T}} (\mathbf{y}^{\mathrm{t}})^{\otimes \mathrm{r}}, \tag{3.7}$$

where $(\cdot)^{\otimes \mathrm{r}}$ is the rth generalised (tensor) outer product $(\cdot)\otimes(\cdot)\otimes\ldots\otimes(\cdot)$. We use the first four K-statistics[9] for T samples:

$$\begin{aligned}
\mathbf{Y}^1 &= \frac{\mathbf{S}_1}{T} \\
\mathbf{Y}^2 &= \frac{T\mathbf{S}_2 - \mathbf{S}_1^{\otimes 2}}{T(T-1)} \\
\mathbf{Y}^3 &= \frac{T^2\mathbf{S}_3 - 3T\mathbf{S}_2\otimes\mathbf{S}_1 + 2\mathbf{S}_1^{\otimes 3}}{T(T-1)(T-2)} \\
\mathbf{Y}^4 &= \frac{T^2(T+1)\mathbf{S}_4 - 4T(T+1)\mathbf{S}_3\otimes\mathbf{S}_1 - 3T(T-1)\mathbf{S}_2^{\otimes 2} + 12\mathbf{S}_2\otimes\mathbf{S}_1^{\otimes 2} - 6\mathbf{S}_1^{\otimes 4}}{T(T-1)(T-2)(T-3)}.
\end{aligned} \tag{3.8}$$

Note that this retains the spatial structure of the original data, meaning if our observations are on a plane ($D=2$), then the mean $\mathbf{Y}^1$ will be a 2nd order tensor, the variance $\mathbf{Y}^2$ is a 4th order one,

[9] See (Cite: Stuart1987a) for higher-order expressions, and (Cite: DiNardo2008) for a general discussion.

and so on. Similarly for $D = 3$ dimensional datasets the relevant scaling is 3rd, 6th etc dimensional tensors. Although the formulas given above compute these tensors for the complete observation space, note that due to our generative model and small basis functions we will only be able to reconstruct elements of these tensors that are within our interaction length, as defined by the patch size $\mathcal{P}$. Therefore we need not even compute or store most tensor elements, significantly reducing both computational and storage costs. Added to the fact that these cumulant tensors – similarly to covariance matrices – are symmetric[10], we can efficiently compute the 'reconstructable' elements via dynamic programming. This feature extraction procedure is illustrated in figure 3.2, which shows the resulting local parts of the cumulant feature tensors for two 1-dimensional sources with identical shapes, but different signals.

Having defined and extracted the neurally relevant features from the data that we wish to reconstruct, we have two tasks left to define the feature reconstruction cost $\mathcal{C}_{\mathbf{X}}^{\text{reconst}}$. First we need to compute the reconstruction of the features $\hat{\mathbf{Y}}$ given a current setting of the sources $\hat{\mathrm{O}}$. Secondly, we need to define a distance metric between the reconstructed and original features.

Starting with the feature reconstruction, we need to remember that our main purpose with the feature representation was not having to compute all time-varying basis function weights $\{\mathbf{x}^{\mathrm{s,t}}\}$ for each each source. This means we don't have access to the complete reconstruction $\hat{\mathbf{y}}$ at all time points, but rather we wish to represent our reconstruction only within the feature space available. To this end we define a new set of basis function weights $\{\mathbf{X}^{\mathrm{r}}\}_{\mathrm{r}=1}^{\mathrm{R}}$, which represents not the reconstruction of a single sample (such as $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$ did), but rather represents the reconstruction of the first R cumulants. Although $\mathbf{X}^{\mathrm{r}}$ is generally a $K \times K \times \ldots \times K$ r-th order tensor, with the $X_{\mathrm{k}_1\mathrm{k}_2,\cdots\mathrm{k}_{\mathrm{r}}}$ element being the linear combination weight of the tensor product of the respective bases: $\otimes_{\mathrm{i}=1}^{\mathrm{r}} b^{\mathrm{k}_{\mathrm{i}}}(\cdot)$, these reconstruction tensors need to be symmetric to represent valid signal cumulants, which significantly reduces the number of free weight parameters in each $\mathbf{X}^{\mathrm{r}}$, with their degrees of freedom being $\binom{K+r-1}{r} \ll K^{\mathrm{r}}$. To illustrate the tractability achieved through this feature representation, let's calculate the number of free parameters in CHOMP versus reconstructing the full time course in a realistic scenario of $K = 4$ bases, $R = 4$ cumulant orders and $T = 1000$ samples:

$$\sum_{\mathrm{r}=1}^{\mathrm{R}} \binom{K+r-1}{r} = 69 \ll 4000 = T * K \tag{3.9}$$

Note that this nearly 100-fold reduction is to be understood per source, and in calcium imaging we often expect 100s of sources within a single dataset. Thus CHOMP achieves on the order of 10000 times less free parameters to estimate during inference of source location, compared to previous methods that would reconstruct the complete time course at the same time. On the other hand, CHOMP retains much more information about the time course itself than the previous method which operates only on the mean of all samples (admittedly with only 4 parameter per source), while

[10] This is also called supersymmetric in some communities. See (Cite: Comon 2008 symmetric tensors) for a detailed introduction to a pure mathematician's perspective on multilinear calculations.

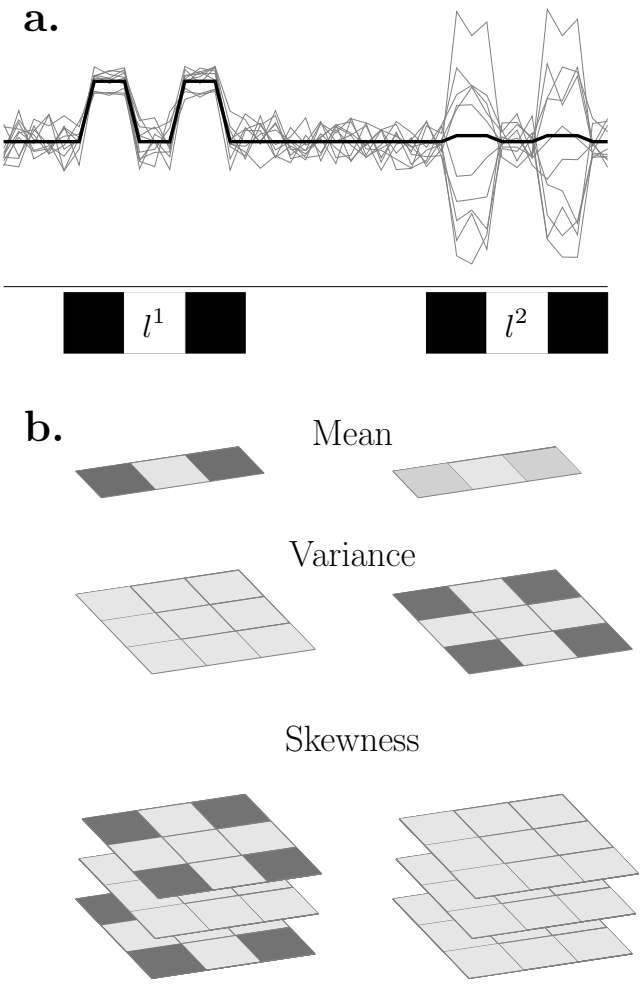


Figure 3.2: Feature extraction procedure in CHOMP for two sources with identical shapes. **a.** Two sources in a 1D space, l^1 with high mean and skewness, l^2 with high variance. The thick line is the true mean across the whole space, the thin ones are individual samples. The squares below indicate observation locations in 1D and the black colour shows the shape of the sources at those locations. **b.** The first three resulting cumulant tensors. Darker colour indicates higher value. The source at l^1 would have been found by an algorithm reconstructing mean values only, but at l^2 the variance feature is likely necessary for successful source localisation.

also directly encompassing the mean-only method, when we simply set $R = 1$. Having defined this new higher order feature reconstruction using $\{\mathbf{X}^{\mathrm{r}}\}$ as coefficients, we can compute the full reconstruction of each tensor order r, similarly to equation 3.5, but now computing the reconstructions of feature tensors rather than individual samples.

$$\begin{aligned} &\forall\, \mathbf{l}^{\mathrm{i}} \in \mathcal{I}, \quad r \in \{1 \dots R\} \\ &\widehat{Y}^{\mathrm{r}}{}^{\mathbf{l}^1,\cdots,\mathbf{l}^{\mathrm{r}}}(\mathsf{O}_{\mathbf{X}}, \mathsf{B}) = \left(\sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{k}_1,\mathrm{k}_2,\cdots,\mathrm{k}_{\mathrm{r}}}^{\mathrm{K},\mathrm{K},\cdots,\mathrm{K}} \left(X^{\mathrm{s,r}}_{\mathrm{k}_1\mathrm{k}_2\cdots\mathrm{k}_{\mathrm{r}}} \prod_{\mathrm{i}=1}^{\mathrm{r}} b^{\mathrm{k}_{\mathrm{i}}}(\mathbf{l}^{\mathrm{i}} - \mathbf{l}^{\mathrm{s}}) \right) \right), \end{aligned} \tag{3.10}$$

where $\mathsf{O}_{\mathbf{X}}$ represents the set of object reconstructions with locations and feature weights

$$O^{s}_{\mathbf{X}} = \left\{ \mathbf{l}^{s} \in \mathbb{R}^{D}, \quad \left\{ \mathbf{X}^{s,r} \in \mathbb{R}^{K \times K \times \ldots_{r} \times K} \right\}_{r=1}^{R} \right\}. \tag{3.11}$$

Furthermore, note again that each reconstructed feature tensor $\widehat{\mathbf{Y}^{r}}$ is a $\mathcal{I} \times \mathcal{I} \times \ldots_{r} \times \mathcal{I}$ space indexed by r 'locations' $\{\mathbf{l}^{i} \in \mathcal{I}\}_{i=1}^{r}$, while the reconstruction weight tensors $\mathbf{X}$ are indexed by r integers $\{k_{i}\}_{i=1}^{r}$. These integers also select the appropriate basis functions $b^{\cdot}(\cdot)$.

The final step in fully defining the feature reconstruction cost $\mathcal{C}^{\text{reconst}}_{\mathbf{X}}$, is to set a distance metric between the reconstructions and the observations. I give here a generic definition that showcases the flexibility of CHOMP. Afterwards I shortly discuss what results various choices lead to, and potential extensions that would take into account further considerations.

Firstly, the reconstruction orders are completely independent due to the use of cumulant feature representation, and thus the most important choice is balancing the weight of the reconstructing individual feature orders with R weights, $\mathbf{c} \in \mathbb{R}^{R}_{\geq 0}$. Secondly, we need to define R distance metrics $f^{r}(\cdot,\cdot) : \mathcal{I}^{r} \times \mathcal{I}^{r} \mapsto \mathbb{R}_{+}$, acting on the observed and reconstructed features. Then simply

$$\mathcal{C}^{\text{reconst}}_{\mathbf{X}} = \sum_{r=1}^{R} c_{r} f^{r}\left(\mathbf{Y}^{r}, \widehat{\mathbf{Y}^{r}}\right), \tag{3.12}$$

but this simple form gives us a wealth of options. If only one c_{i} is non-zero, then this results in simple pursuit algorithms, with $c_{1} \neq 0$ reconstructing the mean signal as done previously, but also $c_{2} \neq 0$ favouring locations with higher spatially structured variance, or only $c_{4} \neq 0$ carrying out pure kurtosis pursuit, pinpointing sparsely firing cells. Having multiple non-zero coefficients in $\mathbf{c}$ effectively interpolates between these options.

The other choice is of course the distance metric itself, with which one needs to be more careful. The simplest option is an entrywise norm, such as the Frobenius norm, for each r order. However it should be noted that for a given dataset $\mathbf{y}$, where noise is independent and identically distributed across samples, the feature noise levels we expect in the individual entries vary both with the order of the feature tensor, as well as whether or not the entry is a diagonal or an off-diagonal element within the tensor. This may be further complicated by having spatial autocorrelation within individual $\mathbf{y}^{t}$ samples, which should also be taken into account in the error metric used to compute the reconstruction cost. A generally applicable procedure, assuming additive Gaussian noise on samples, is to estimate noise levels and spatial autocorrelation lengths in $\mathbf{y}^{t}$ samples, applying Isserlis' theorem (Cite: chomp paper refs 7,8) to compute the expected noise in the various tensor entries, then applying an appropriately weighted entrywise Frobenius norm to the residual after reconstruction:

$$f^{r}\left(\mathbf{Y}^{r}, \widehat{\mathbf{Y}^{r}}\right) = \frac{1}{N^{r}(\mathcal{P})} \left\| \frac{1}{\boldsymbol{\sigma}^{r}(\mathbf{y})} \odot \left(\mathbf{Y}^{r} - \widehat{\mathbf{Y}^{r}}\right) \right\|_{F}^{2}, \tag{3.13}$$

where $\boldsymbol{\sigma}^{r}(\mathbf{y})$ computes the expected entrywise standard deviations based on the noise levels in $\mathbf{y}$, and $N^{r}(\mathcal{P})$ is the number of elements in a feature patch of order r, to ensure higher reconstruction

orders are not overrepresented in the cost function due to the higher number of reconstructable entries.

Having written down the reconstruction cost, we now only need to define the cost of sources used, $\mathcal{C}_{\mathbf{X}}^{\text{sources}}(\hat{\mathsf{O}})$, to arrive at a full description of the cost function we wish to minimise. In basic matching pursuit algorithms, this is simply a weighted function of the number of sources used in the reconstruction (the cardinality of $\hat{\mathsf{O}}$), but again, CHOMP provides more options due to its convolutional nature. Importantly for neural applications, as we believe our sources occupy physical space, we wish to discourage the algorithm from finding highly overlapping sources. We don't want to completely disable this behaviour, especially in $D = 2$ dimensions, as overlapping sources can be the result of imperfect optical sectioning during two-photon calcium microscopy. Generally a good sources cost is

$$\mathcal{C}_{\mathbf{X}}^{\text{sources}}(\hat{\mathsf{O}}_{\mathbf{X}}) = \lambda \|\hat{\mathsf{O}}_{\mathbf{X}}\| + \sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{s}'=1}^{\mathrm{s}} g(\hat{O}_{\mathbf{X}}^{\mathrm{s}}, \hat{O}_{\mathbf{X}}^{\mathrm{s}'}), \tag{3.14}$$

where λ is the cost per source and $g(\hat{O}_{\mathbf{X}}^{\mathrm{s}}, \hat{O}_{\mathbf{X}}^{\mathrm{s}'})$ is an overlap penalty between sources s and s'. In it's simplest form it operates on distances between source locations $g(\cdot,\cdot) \propto \|\mathbf{l}^{\mathrm{s}} - \mathbf{l}^{\mathrm{s}'}\|$, and is decreasing with distance, being zero for sources further away than the patch size $\mathcal{P}$. However, one may also wish to take into account the reconstructions of nearby sources via the coefficients $\mathbf{X}^{\mathrm{s}}$ and $\mathbf{X}^{\mathrm{s}'}$, penalising nearby sources with complementary reconstructed shapes less.

To conclude, the complete cost function we wish to minimise is built from a reconstruction error and the cost for sources used, in the following parametric form:

$$\mathcal{C}_{\mathbf{X}} = \mathcal{C}_{\mathbf{X}}^{\text{reconst}}\left(\mathbf{Y}^{\mathrm{r}}, \widehat{\mathbf{Y}^{\mathrm{r}}}\right) + \mathcal{C}_{\mathbf{X}}^{\text{sources}}(\hat{\mathsf{O}}_{\mathbf{X}})$$

$$\begin{aligned} \mathcal{C}_{\mathbf{X}}^{\text{reconst}} &= \sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}} f^{\mathrm{r}}\left(\mathbf{Y}^{\mathrm{r}}, \widehat{\mathbf{Y}^{\mathrm{r}}}\right) \\ f^{\mathrm{r}}\left(\mathbf{Y}^{\mathrm{r}}, \widehat{\mathbf{Y}^{\mathrm{r}}}\right) &= \frac{1}{N^{\mathrm{r}}(\mathcal{P})} \left\| \frac{1}{\boldsymbol{\sigma}^{\mathrm{r}}(\mathbf{y})} \odot \left(\mathbf{Y}^{\mathrm{r}} - \widehat{\mathbf{Y}^{\mathrm{r}}}\right) \right\|_{\mathrm{F}}^{2} \end{aligned} \tag{3.15}$$

$$\mathcal{C}_{\mathbf{X}}^{\text{sources}} = \lambda |\hat{\mathsf{O}}_{\mathbf{X}}| + \sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{s}'=1}^{\mathrm{s}} g(\hat{O}_{\mathbf{X}}^{\mathrm{s}}, \hat{O}_{\mathbf{X}}^{\mathrm{s}'}),$$

where the main user-adjustable parameters are $\mathbf{c}$, the weights of individual reconstruction orders, and λ, the cost for each source. On a lower level, one can further customise the designs of the

metrics for reconstruction quality $f^{\mathrm{r}}(\cdot,\cdot)$ and for source overlap penalty $g(\cdot,\cdot)$.

Inference via matching pursuit

The main advantage of matching pursuit based source localisation, is that we turn an intractable simultaneous inference problem of an unknown number of sources into an iterative process. We add sources one-by-one, selecting the new source location greedily, such that it maximally decreases the cost function, until no potential source would decrease the cost anymore. Computing the potential decrease of the cost at a proposed new location in the above defined cost function is a computationally cheap procedure, as the reconstruction cost change depends only on a few of the observations (the ones within the patch size), and we can solve for all $\mathbf{X}^{\mathrm{r}}$ coefficients tensors in closed form. Furthermore, the change in source cost only depends on a simple function of the previous sources, and even though the computational cost of the overlap-cost increases linearly throughout the inference with each additional source, it is still negligible for the typical number of few 100 sources.

However, in order to select where the next source should be placed, we essentially need to carry out global optimisation at every iteration across all possible source locations (that is approximately the convex hull of observation locations extended by the patch size, ie. $\mathrm{Conv}(\mathcal{I} \pm \mathcal{P})$). Although this would of course be computationally extremely expensive and thus intractable, we can exploit two characteristics of the cost function to carry out this procedure. Firstly, the reconstruction cost function itself is as smooth as the basis functions $b^{\mathrm{k}}(\cdot)$ are, meaning we can first compute the cost decrease on a grid, and use a few promising locations to carry out local optimisation to find the next source location with maximum cost decrease[11]. Secondly, all reconstruction cost changes are local around a proposed location. This means two things: *a.)* we can massively parallelise computations on modern computational clusters, spreading both data and computation across nodes at will and *b.)* we only need to carry out the computation on the full grid once, since in subsequent iterations we can update this grid only around the previously added source.

After we have selected where to add the next new source, updating the current reconstruction and the residual is simply a byproduct of the selection process, and computationally free. The only computational cost is updating the grid which serves to provide initialisation for the next set of local optimisations.

In pseudocode, the algorithm proceeds as follows:

Observations on a grid

As discussed above, an efficient solution to finding the global optima of the cost function involves first computing the cost surface on a grid of potential source locations L. From here onwards, I am also going to assume that my observations also lie on a grid, which is typically the case in two-photon

[11] This is of course not necessarily the global optimum of cost decrease, but it still results in a reproducible greedy algorithm

Algorithm 3.2 High level inference pseudocode for Convolutional Higher Order Matching Pursuit

1: Let $\hat{\mathsf{O}}_{\mathbf{X}} = \{\}$ be the empty set of inferred sources

2: At a set of locations (L) spanning the observation space $\text{Conv}(\mathcal{I} \pm \mathcal{P})$ compute the optimal reconstruction coefficients $\mathbf{X} = \{\mathbf{X}^{\mathrm{r}}\}_{\mathrm{r}=1}^{\mathrm{R}}$ that (locally) minimise the reconstruction error:

3:

$$\begin{aligned} &\forall\, \mathbf{l} \in \mathsf{L}: \\ &\quad \tilde{\mathbf{Y}}(\mathbf{l}) = \hat{\mathbf{Y}}(\hat{\mathsf{O}}_{\mathbf{X}} \cup O^{\mathrm{l}}_{\mathbf{X}}), \quad \text{where } \; O^{\mathrm{l}}_{\mathbf{X}} = \{\mathbf{X}, \mathbf{l}\} \\ &\quad \tilde{\mathbf{X}}(\mathbf{l}) = \arg\min_{\mathbf{X}} \; \mathcal{C}^{\text{reconst}}_{\mathbf{X}}\left(\mathbf{Y}, \tilde{\mathbf{Y}}(\mathbf{l})\right)\big) \end{aligned} \tag{3.16}$$

4: **repeat**

5: At each location $\mathbf{l} \in \mathsf{L}$ compute $\Delta\mathcal{C}_{\mathbf{X}}$ based on $\tilde{\mathbf{X}}(\mathbf{l})$

6: At a few 'promising' locations, initialise gradient descent jointly in $\mathbf{l}$ and $\mathbf{X}$ on $\mathcal{C}_{\mathbf{X}}$ to find the local minima, and subsequently select the global minimum location and reconstruction $\tilde{O}^{\text{new}}_{\mathbf{X}}$.

7: If adding $\tilde{O}^{\text{new}}_{\mathbf{X}}$ decreases the cost, expand the solution set, $\hat{\mathsf{O}}_{\mathbf{X}} := \hat{\mathsf{O}}_{\mathbf{X}} \cup \tilde{O}^{\text{new}}_{\mathbf{X}}$.

8: Update $\tilde{\mathbf{X}}(\mathbf{l})$ affected locally by the new source, where $\mathbf{l} - \mathbf{l}^{\text{new}} \in 2\mathcal{P}$

9: **until** $\tilde{O}^{\text{new}}_{\mathbf{X}}$ does not decrease the cost $\mathcal{C}_{\mathbf{X}}$.

10: Given the locations in the solution set $\hat{\mathsf{O}}_{\mathbf{X}}$, calculate the individual sample reconstruction coefficients $\mathbf{x}^{\mathrm{s,t}}$, and thus the full reconstruction of the original data, $\hat{\mathsf{O}}$.

calcium recordings (be it 2 or 3 dimensional), my main application of interest. This assumption allows for a further set of simplifications when computing $\tilde{\mathbf{X}}(\mathbf{l})$ in step 3 of algorithm 3.2.

Let a local patch around an observation at location $\mathbf{l}$ be $\mathcal{P}_{\mathbf{l}}$, and let the observation grid's step size be such that a patch covers a total of M observations. Therefore we can represent a local patch of observations as a D-th order tensor with M elements $\mathbf{y}^{\mathrm{t}}_{\mathbf{l}}$. Similarly, we may represent the individual basis functions $b^{\mathrm{k}}(\cdot)$ as D-th order tensors with a total of M elements, let $\mathbf{b}^{\mathrm{k}} = b^{\mathrm{k}}(\mathcal{P})$ denote this tensor[12].

When computing the cumulant features $\mathbf{Y}^{\mathrm{r}}$, we can similarly now represent local patches of them as D^{r}-th order tensors with M^{r} elements, denoting such a local feature patch tensor as $\mathbf{Y}^{\mathrm{r}}_{\mathbf{l}}$. In order to write down local reconstructions – following equation 3.10 – in a tensor format now available due to the grid structure, we can define outer products of the basis functions tensors, letting $\mathbf{b}^{\mathrm{k}_1\mathrm{k}_2\ldots\mathrm{k}_{\mathrm{r}}} = \mathbf{b}^{\mathrm{k}_1} \otimes \ldots \otimes \mathbf{b}^{\mathrm{k}_{\mathrm{r}}}$ (where $\forall i \;\; k_{\mathrm{i}} \in \{1 \ldots K\}$) denote a D^{r}-th order tensor with M^{r} elements. Let us denote the respective flattened (vectorised) tensors as $\vec{\mathbf{Y}}^{\mathrm{r}}{}_{\mathbf{l}}$ and $\vec{\mathbf{b}}^{\mathrm{k}_1\mathrm{k}_2\ldots\mathrm{k}_{\mathrm{r}}}$ being M^{r} long vectors, and $\mathbf{B}^{\mathrm{r}} \in \mathbb{R}^{\mathrm{M}^{\mathrm{r}} \times \mathrm{K}^{\mathrm{r}}}$ being the matrix collecting all possible variations of the vectorised basis function outer products as columns. Then our local reconstruction problem boils down to

[12] This description is applicable when the approximating grid L is chosen with a grid size that is an integer multiple of the observation grid size. This is typically a good choice for calcium imaging applications, where neurons span multiple pixels/voxels. For approximation grid sizes that are rational multiples of the original, e.g. $1/n$ step sizes, one would need n different sets of $\mathbf{b}^{\mathrm{k}}$ tensors, applying the appropriate one at each sub-grid offset value during the convolutional procedure described in the following. This finer initial approximation increases the computational complexity n^{D}-fold initially, and n^{Dr}-fold during the local updates, while the procedure remains highly parallelisable, and the gradient descent optimisation will likely converge faster due to the better initialisations.

$$\forall\, r: \quad \hat{\mathbf{X}}^{\mathrm{r}}_{\mathbf{l}} = \arg\min_{\mathbf{X}^{\mathrm{r}}} f^{\mathrm{r}}\left(\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}},\ \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}}\right). \tag{3.17}$$

For clarity, let's assume here that $f^{\mathrm{r}}(\cdot,\cdot)$ is simply the Frobenius norm, but this easily generalises for the entrywise weighted versions suggested above as well. Therefore the solution is

$$\begin{aligned}\hat{\mathbf{X}}^{\mathrm{r}}_{\mathbf{l}} &= \arg\min_{\mathbf{X}^{\mathrm{r}}} \left\| \vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} - \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}} \right\|^2_{\mathrm{F}} \\ &= \left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1}\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} \qquad \text{where} \quad \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} = \mathbf{B}^{\mathrm{r}\top}\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}}\,.\end{aligned} \tag{3.18}$$

Here $\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} \in \mathbb{R}^{\mathrm{K}^{\mathrm{r}}}$ represents the vectorised projection of the r-th order local patch cumulant tensor onto the r-th order basis function tensors. Having solved for the optimal reconstruction coefficients $\hat{\mathbf{X}}_{\mathbf{l}}$, we can compute the maximum decrease in the reconstruction cost given a new source at $\mathbf{l}$ as

$$\begin{aligned}\Delta\mathcal{C}^{\mathrm{reconst}}_{\mathbf{X}}(\mathbf{l}) &= \sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}}\left(\left\|\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} - \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}}\right\|^2_{\mathrm{F}} - \left\|\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}}\right\|^2_{\mathrm{F}}\right) \\ &= \sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}}\left(-2\vec{\mathbf{X}}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}\top}\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} + \vec{\mathbf{X}}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}}\right) \\ &= -\sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}}\left(\mathbf{Z}^{\mathrm{r}\top}_{\mathbf{l}}\left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1}\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}}\right)\end{aligned} \tag{3.19}$$

This means that we do not even need to store the feature reconstruction coefficients $\mathbf{X}$ throughout the steps 3-5 of algorithm 3.2, as using only the projections of the data onto the expanded bases enables us to compute the potential reconstruction cost change at all locations $\mathbf{l} \in \mathsf{L}$. Furthermore, these projections can be computed very efficiently by higher order convolutions as shown in equation 3.18, and then to calculate the cost change, we only need to compute the inverse $\left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1}$ once, resulting in fast quadratic form computations.

To carry out steps 6 and 8 of algorithm 3.2, the optimisation and the update, we can similarly complete the whole procedure in the projection space of $\mathbf{Z}^{\mathrm{r}}$, noting that both of these steps again require evaluation the $b^{\mathrm{k}}(\cdot)$ basis functions at arbitrary locations, rather than using their $\mathbf{b}^{\mathrm{k}}$ tensor representations.

Solutions on a grid

In order to further simplify the inference process, we may wish to sacrifice a bit of localisation accuracy, and decide that our source locations may only lie on the grid L. Note that this represents an approximation to the original generative model described in equation 3.5, where sources locations were in a D dimensional Euclidean space, rather than at discrete subset thereof.

However, with a fine enough grid we lose little in localisation, but completely eliminate the expensive gradient descent optimisation in step 6 of algorithm 3.2 and may significantly simplify the local coefficient update in step 8. Specifically, with this approximation we never need to evaluate the bases $b^{\mathrm{k}}(\cdot)$ at arbitrary locations, and therefore we do not even need to represent the complete basis functions, only the M-element tensors $\mathbf{b}^{\mathrm{k}}$.

The local update after adding a new source to the solution set $\hat{\mathsf{O}}_{\mathbf{X}}$ generally requires computing the residual in feature space $\mathbf{Y}$ by subtracting the reconstructed $\mathbf{BX}$, as the cost function change associated with the next source depends on the proposed source's ability to fit this residual. However, as data and residuals only enter the cost change calculations via the basis projections $\mathbf{Z}$ (see equation 3.19), we do not need to explicitly represent the residual, we may just compute the change in the projections directly, after adding a new source $O^{\mathrm{s}} = \{\hat{\mathbf{X}}^{\mathrm{s}}, \mathbf{l}^{\mathrm{s}}\}$. For clarity I first show the changes for $r = 1, 2$ using the basis function formulation – that is applicable for arbitrary $\mathbf{l}^{\mathrm{s}}$ source locations – then discuss how the tensor representation may be used to speed up the computations.

$$\begin{aligned}
\Delta_{\mathrm{s}}[\mathbf{Z}^1_{\mathbf{l}}]_{\mathrm{k}} &= \int_{\mathbf{l}'} b^{\mathrm{k}}(\mathbf{l}' - \mathbf{l}) \left(\sum_{\mathrm{k}'} b^{\mathrm{k}'}(\mathbf{l}' - \mathbf{l}^{\mathrm{s}}) \, [\hat{\mathbf{X}}^{\mathrm{s},1}]_{\mathrm{k}'} \right) \\
\Delta_{\mathrm{s}}[\mathbf{Z}^2_{\mathbf{l}}]_{\mathrm{k}_1\mathrm{k}_2} &= \int \int_{\mathbf{l}'_1, \mathbf{l}'_2} b^{\mathrm{k}_1}(\mathbf{l}'_1 - \mathbf{l}) \, b^{\mathrm{k}_2}(\mathbf{l}'_2 - \mathbf{l}) \left(\sum_{\mathrm{k}'_1, \mathrm{k}'_2} b^{\mathrm{k}'_1}(\mathbf{l}' - \mathbf{l}^{\mathrm{s}}) \, b^{\mathrm{k}'_2}(\mathbf{l}' - \mathbf{l}^{\mathrm{s}}) \, [\hat{\mathbf{X}}^{\mathrm{s},1}]_{\mathrm{k}'_1\mathrm{k}'_2} \right)
\end{aligned} \tag{3.20}$$

Therefore in order to carry out the update directly in $\mathbf{Z}$ space, we essentially need to compute the pairwise convolutions of the basis functions with each other in every order, then weight the convolutions by the reconstruction coefficients, that are linear functions of $\mathbf{Z}$ themselves (see equation 3.18). Therefore we can define a single linear operator $U^{\mathrm{r}}(\Delta\mathbf{l} \mid \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}})$ that is a function of distance and the basis functions, and maps the projection vector $\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}}$ at newly added source to the projection vector at nearby locations $\mathbf{l}$:

$$\Delta_{\mathrm{s}}\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} = U^{\mathrm{r}}(\mathbf{l} - \mathbf{l}^{\mathrm{s}} \mid \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}}) \cdot \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}} \tag{3.21}$$

When the solution locations $\mathbf{l}^{\mathrm{s}}$ are restricted to the L grid, this linear update operator can be represented by a $M_* \times K^{\mathrm{r}} \times K^{\mathrm{r}}$ tensor $\mathbf{U}^{\mathrm{r}}$, and only need to be computed once. Here M_* is the number of grid points within a $2\mathcal{P}$ patch, which is the area where $\mathbf{Z}$ projections are affected by adding a new source. Therefore updating all projections around the newly added location can be written as:

$$\Delta_{\mathrm{s}}\mathbf{Z}^{\mathrm{r}}_{2\mathcal{P}(\mathbf{l}^{\mathrm{s}})} = \mathbf{U}^{\mathrm{r}} \, \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}}, \tag{3.22}$$

where $2\mathcal{P}(\mathbf{l}^{\mathrm{s}})$ selects the appropriate M_* locations affected.

A visual summary of algorithm 3.2 using the vectorised projections $\mathbf{Z}$ is shown in figure 3.3.

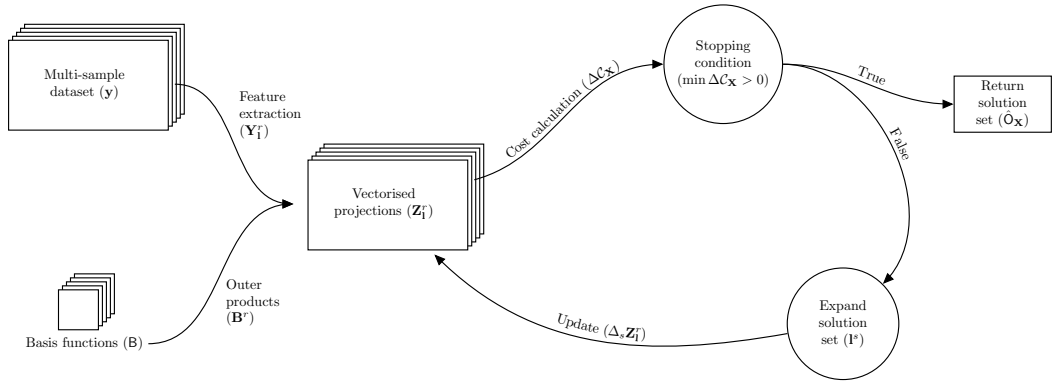


Figure 3.3: Algorithm flowchart for Convolutional Higher Order Matching Pursuit. Showcases the initial computation of projecting the data features onto the basis functions, then the iterative update loop, which involves only spatially local computations.

3.2.2 Dictionary Learning

The inference algorithm described in the previous section relies on a fixed set of basis functions B – often called a dictionary – that collectively are capable of describing the temporal changes of the various spatial signal modes of cells of interest. Although in some signal processing applications these modes may indeed be known, or at least stationary over multiple recordings using the same observation process and similar signal generators, it is nevertheless useful to discuss how one may initialise a set of basis functions, as well as how to iteratively update them based on the inference results. I first discuss the update procedure, as it is generally applicable to arbitrary types of data, then provide potential initialisations that are more specific to two-photon calcium imaging datasets.

In order to update the basis functions, we have to select appropriate training data that admits to sparse decomposition resulting in the new bases. As the basis functions' support is the patch $\mathcal{P}$, our first task is to select patches around the inferred source locations, let these be $\mathcal{P}(\mathbf{l}^s)$. In the general case, where $\mathbf{l}^s$ are arbitrary locations, these extracted patches will result in location-value pairs ($\mathbf{u} \in \mathcal{P}$ and $v \in \mathbb{R}$), where the values are scalar observations $y^t_\mathbf{l}$, $\mathbf{l} \in \mathcal{I}$, and the locations are shifted relative to the current source location, with $\mathbf{u} = \mathbf{l} - \mathbf{l}^s$ spanning the patch $\mathcal{P}$. We then use these $\{\mathbf{u}, v\}$ pairs as training data to any generalised additive model capable of learning K components which explain the training data well, and represent the basis functions that can be evaluated at arbitrary new locations within the patch $\mathcal{P}$. Such models are described in (Cite: vincent's Scalable GAM using sparse variational Gaussian processes, plus some references from that paper). One difficulty during the preparation of this training data is that multiple inferred sources may affect the same observation location due to overlapping patches. In these cases care must be taken to either omit these affected locations from the training data, or - when affected by n different sources - represent them as n separate samples with values $\{v^*_i\}^n_{i=1}$, where v^*_i is the observed value v minus the reconstructions of all except the i-th source affecting that location.

When both observations and solutions lie on a grid, we do not have to represent the bases as functions, but we can rather use their D-th order tensor representation $\mathbf{b}^{\mathrm{k}}$. Similarly the observation matches $\mathbf{y}^{\mathrm{t},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$ are D-th order tensors, with the same size as the bases, with the $*$ again suggesting to explain away other sources at overlaps. Updating this tensor representation of the bases is a much more standard problem, as we can vectorise the patches, collect them into a matrix as columns, and apply standard matrix factorisation methods to recover the top K (singular) vectors which best explain the data. Applicable matrix factorisation methods are Singular Value Decomposition (SVD), Independent Component Analysis (ICA) or Non-negative Matrix Factorisation (NMF). That is

$$\begin{aligned}[\mathbf{V}]_{\cdot,\mathrm{s}} &= \vec{\mathbf{y}}^{\mathrm{t},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})} \qquad \text{for all } O^{\mathrm{s}} \in \mathsf{O} \\ \left\{\mathbf{b}^{\mathrm{k}}_{\mathrm{new}}\right\}_{\mathrm{k}=1}^{\mathrm{K}} &= \text{MatrixFactorisation}_{\mathrm{K}}(\mathbf{V})\end{aligned} \tag{3.23}$$

Decomposing feature tensors. Alternatively, especially when we are trying to ignore some orders of signal cumulants during source localisation and not even aiming to represent the full temporal signal, we may wish to update the basis functions based on the feature tensors, rather than based on the original observation samples. In this case we can similarly collect the feature tensors $\mathbf{Y}^{\mathrm{r},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$, which are now D^{r}-th order tensors computed from the noisy data. Learning sparse decomposition from tensors of multiple orders is still an active research area, especially when we have multiple noisy samples from each order. Although my main aim with the calcium imaging application was to represent the signal modes over time, I did not thoroughly explore this space, two methods that are capable of tensor-only learning would be Higher Order SVD (HOSVD) (Cite: hosvd paper) and Multi-Tensor Factorisation (MTF) (Cite: Khan 2015).

The former, HOSVD, essentially relies upon flattening of tensors using each faceas the base, and averaging across their decompositions. This is very simply applicable for symmetric tensors, as we only have to do a single flattening, but to my knowledge has only been derived for a single tensor of a single given order. My attempt at extending HOSVD to multiple tensor samples of multiple orders proceeded as follows. For each tensor order r, and each sample s flatten the symmetric feature tensor $\mathbf{Y}^{\mathrm{r},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$ into an $M \times M^{\mathrm{r}-1}$ matrix, and stack the S samples as columns, resulting in an $M \times S * M^{\mathrm{r}-1}$ matrix, that I call $\mathbf{V}^{\mathrm{r}}$. If we now stacked tensors of different orders as columns again, the lower orders would be significantly underrepresented, as each additional tensor order yields M-times as many samples as the previous one. Therefore we have need to ensure we are representing each tensor order sufficiently well before doing the decomposition. I found two heuristic options for renormalising the columns of these $\mathbf{V}^{\mathrm{r}}$ matrices that seem to work in practice, but was yet unable to prove their domain of applicability. The first option is to simply divide each entry in each matrix by M^{r}/M, the number of entries in the tensor divided by M, the number of elements in the basis functions. I found however, that this could now lead to under-representing the effect of higher order tensors. The second option comes from my earlier discussion of the number of free elements in a

Look up proper terminology

symmetric tensor (see equation 3.9), and thus I use the number of free entries in each symmetric tensor, rather than the total number of entries $\binom{M+r-1}{r/M}$. This penalises higher orders significantly less, as $\binom{M+r-1}{r\ll M^r}$ for the typical large M and $r > 2$.

The alternative, Multi-Tensor Factorisation (MTF) method defines a principled joint Bayesian prior for the weighing between multiple tensors orders, but I found the resulting computational complexity and the running times of the implementation prohibitive to use with the typical parameters in calcium imaging applications with $M > 50$, $R > 2$ and $S > 100$. Therefore, if dictionary updates are required, I for now recommend the use of equation 3.23 with well-understood matrix factorisation methods, until multi-sample, multi-order tensor factorisation methods enter a more maturely applicable phase.

Application to neural data. Next I discuss the specific application of CHOMP inference and dictionary learning to neural two-photon calcium imaging datasets. CHOMP relies on basis functions capable of describing neurons well. One way of initialising the CHOMP dictionary is to use the source locations indicated by human markers as training data and then apply the dictionary learning shown in equation 3.23. Although as discussed previously, this does introduce unnecessary human bias, typically the most prominent 10-50 cells per field of view are agreed upon by all human markers, and do not significantly change in repeated experiments (that use the same microscope with similar settings and probe the same brain areas in the same species). These human-segmented datasets are typically available or easy to collect in most experimental laboratories, and thus can be used as an initialisation to CHOMP. The patch size $\mathcal{P}$ of basis functions is chosen to ensure including the segmented cells' full extent.

That said, I both prefer reducing the human involvement in data analysis, and one does not necessarily have access to previously segmented data of interest, and thus I describe here an initialisation that is purely driven by a model of neural signal generation and the observed data. Firstly, the most popular type of recordings, and thus the ones I mainly focused on, are two-dimensional optical cross-sections of superficial cortex layers containing pyramidal neuron cell bodies. The typical shapes that correspond to the cell bodies in such recordings are filled circles and so-called 'donuts'. As the fluorescent calcium sensors are designed to mainly be expressed in the cytoplasm, but not the nucleus, the filled shape corresponds to cells for which the optical cross-section does not include the nucleus, or in which the calcium sensor protein is overexpressed also in the nucleus (these latter cells are typically dead). Conversely, donuts correspond to cells in which the darker middle section is the nucleus, with the halo of cytoplasm around it. Other active neural elements in the field of view are neural processes (axons or dendrites). The ones that are perpendicular to the field of view show up as speckles in the image, whereas the non-perpendicular ones show up as long thin lines.

Lastly, non-neural elements typically correspond to vasculature, either perpendicular to the field of view and show up as larger 'holes', or non-perpendicular vasculature above the image shows up as 'shadows', affecting a larger area in the field of view.

Maybe put images here showcasing these ideas

To initialise a set of basis functions capable of describing such data, we first to determine the maximum cell size. This is probably best done by visual inspection (as slightly overestimating only increases computational cost, but is not otherwise detrimental), but one may utilise D-dimensional discrete Fourier transform to find the spatial frequencies present in the data, or finding the typical circle size via circle Hough Transform. Once this has been done, a good initialisation for the bases includes a small disk with $K-1$ increasing radii donuts, the linear combination of which is capable of describing speckles, donuts and filled circles of various sizes. After this initialisation, one may proceed iteratively with inference and dictionary updates, and can discover non-centrally symmetric signal modes as well.

Basis function symmetries. The symmetries of the basis functions both during initialisation and the updates is an interesting question. As discussed in chapter 2, we can easily design Gaussian Process function descriptions with arbitrary symmetries, and similarly restricting matrix decomposition methods to find basis modes only with pre-designed symmetries is a matter of reparameterising the locations $\mathbf{u}$ of the training data, then arranging the vectorised values $\vec{\mathbf{v}}^*$ to reflect the symmetrising transformation done in location space. In my experience however, a number of interesting signal modes may not be symmetric about the center location of the source, and thus artificially limiting the dictionary updates to discover only symmetric modes may render the bases unable to explain interesting neural behaviour and thus potentially miss sources during inference. This can especially be the case for large apparent cell sizes and higher frame rates, in cells whose processes are not aligned perpendicularly to the optical section will show a calcium release wave propagating through the cell body. Similarly, in high frame-rate 3-dimensional datasets we would always expect such waves to be present, and should not forcefully introduce symmetries into bases function.

3.2.3 Iterative inference and dictionary learning across datasets

In order to end up with a good reconstruction of the dataset, we generally need to carry out inference and dictionary update steps iteratively, until the cost function $C_{\mathbf{X}}$ defined in equation 3.15 does not decrease anymore at the end of the full inference step, meaning the inference-update procedure has converged.

However, this may be an expensive procedure and one may end up with a different set of bases functions for each dataset. If the sole purpose of the bases is to provide good source localisation, and in subsequent data processing the only output used will indeed be the source locations, then this is not a problem. However if the bases are to be used also in determining the full extent of single-cell

regions of interest (ROIs) or even during the extraction of the neural signal[13], it is desirable that the learned bases remain stable over multiple data analysis sessions.

We can achieve this by performing the inference simultaneously across multiple datasets (recorded by the same microscope with the same settings, but potentially in different animals, tasks or even brain regions), then doing the dictionary update on all patches collected from all datasets. When this iterative procedure on the sum of all costs from individual datasets $C_{\mathbf{X}}^{\text{total}} = \sum_{\text{datasets}} C_{\mathbf{X}}^{\text{dataset}}$ converges, we can ensure that further data processing steps may rely on this fixed, stable set of basis functions regardless of the dataset. This also has the advantage that once such a fixed dictionary is learned, one only needs to perform a single inference step on new, similar datasets, significantly reducing computational needs. Furthermore, as the update tensors $\mathbf{U}^{\text{r}}$ in equation 3.22 are a function of the bases only, they can be computed once and stored for subsequent inferences, leading to further speedup.

3.3 Validation and experimental results

3.3.1 Validation of CHOMP-based location inference

In order to thoroughly evaluate the impact of incorporating higher order cumulants in convolutional matching pursuit, I created a simulation using a broad range of signal distributions, evaluating the localisation accuracy, and comparing to a group LASSO implementation (Cite: what implementation?). These results are published in (Cite: MLSP paper) and reproduced here.

Data were simulated from the generative model (equation 3.5) in a $D = 1$ dimensional space, with $|\mathcal{I}| = 512$ observations on an equidistant grid, and true source locations restricted to the observation grid, thus $\mathbf{l}^{\text{s}} \in \mathsf{L}$ and $\mathsf{L} = \mathcal{I}$, with 26 sources per field of view (≈ 0.05 occupancy rate, ensuring overlapping sources). Each simulation was based on a new random set of basis functions – provided as input to the tested inference algorithms – with $M = 11$, $K = 2$, entries chosen from a unit variance normal distribution, then the bases are symmetrised about the center and normalised to unit norm. Signal distributions were modelled as mixtures of Gaussians, with mixture parameters selected by non-linear least-squares to match an intended set of cumulants. I explored symmetric distributions with means and variances spanning multiple orders of magnitudes ($\mu \in [0, 10], \sigma^2_{\text{signal}} \in [0.01, 10]$) and a number of excess kurtosis values $(3, 10, 50)$. For each source, signal coefficients $\mathbf{x}^{\text{s}}_{\text{k,t}}$ were sampled ($T = 1000$) from a distribution with given cumulants, and I explored three scenarios for sources placed within the same field of view. In the *no mixing* case, all sources within the same field of view share signal coefficients sampled from the same distribution, whereas in the *uniform mixture*, the sources' signal distributions were sampled uniformly randomly from all possible settings. Lastly, in the *realistic mixture* case, I first sample a single set of cumulant settings, then the indi-

[13] For example, instead of summing up all signal within a binary ROI to represent the signal of that ROI, we can use the optimal reconstruction of the signal that both reduces additive noise and rejects signals resulting from source overlaps

vidual sources' distribution parameters are distributed log-normally around the single setting with one order of magnitude standard deviation. This is called the realistic case, as generally we expect sources' signal distributions to vary slightly within the same field of view, but not multiple orders of magnitude.

The source locations are chosen uniformly randomly within the field of view – only ensuring no exact co-localisation, overlaps are however common – and the final observations are created as the centred basis functions multiplied by the appropriate source signal coefficients – summed additively for overlapping sources – and corrupted by zero-mean, unit-variance ($\sigma^2_{\text{noise}} = 1$) additive Gaussian noise. For all three signal distribution mixture types within the field of view, I simulated and analysed $n = 10000$ fields of view.

Inference of source locations was carried out as described in figure 3.3, with a stopping condition modified to correspond to the true number of sources. The values of $\boldsymbol{\sigma}^{\text{r}}$ in the reconstruction cost distance metrics $f^{\text{r}}(\cdot)$ (see equations 3.12 and 3.13) were set as described there, resulting in $(\boldsymbol{\sigma}^{\text{r}})^2 = (M^{\text{r}} - K^{\text{r}}) * (\sigma^2_{\text{noise}}/T)$. For comparability I used the true value for $\sigma^2_{\text{noise}} = 1$, instead of estimating it from the data as suggested for real datasets.

figure out why this result?

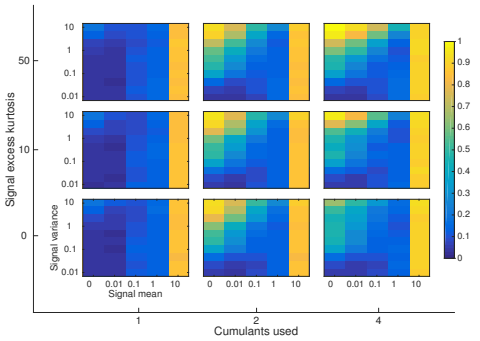
A natural evaluation metric is the frequency with which the algorithm correctly locates the sources, applied using increasing orders of cumulants. We find that as long as the signal distributions contain significant higher order structure, it is indeed feasible to attempt to reconstruct those tensors and they do increase localisation performance (see figure 3.4a). A further feature of greedy algorithms in general, including the current one, is that they provide a natural ordering of the sources found. We thus define the Area Under the Recall Curve as

$$\begin{aligned} \text{AURC}(\mathsf{O}, \hat{\mathsf{O}}) &= \frac{1}{S} \sum_{\text{s}=1}^{\text{S}} \frac{\text{NumCorrect}[\hat{\mathsf{O}}_{1:\text{s}}] - \text{Chance}_{\text{s}}}{s - \text{Chance}_{\text{s}}} \\ \text{Chance}_{\text{s}} &= \frac{1}{\binom{|\mathcal{I}|}{s}} \sum_{\text{s}'=1}^{\text{s}} s' \binom{s}{s'} \binom{|\mathcal{I}| - s}{s - s'} \end{aligned} \tag{3.24}$$

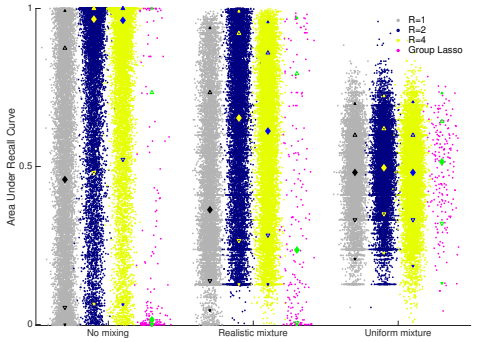
and estimate the performance using this metric, which weights the correctness of earlier sources higher. This is of interest especially in the case when the field of view contains signals from multiple distributions, such as the realistic or the uniform mixtures (see figure 3.4b). Finally I looked at how much higher order features offer on a case-by-case basis (section 3.3.1) and found that for all practical cases, higher order estimators are substantially beneficial.

I also compared the proposed CHOMP method to a group lasso implementation by (Cite: Boyd2010), that corresponds to the L_1-relaxation of . To this I provided as input the full $|\mathcal{I}| \times K|\mathcal{I}|$ spatial design matrix as well as the grouped time courses belonging to the corresponding locations. Estimated time courses were sorted by their norms to obtain the ordered $\hat{\mathsf{O}}$ and the AURC evaluated as above. CHOMP outperformed the group lasso (figure 3.4b) in source localisation while being

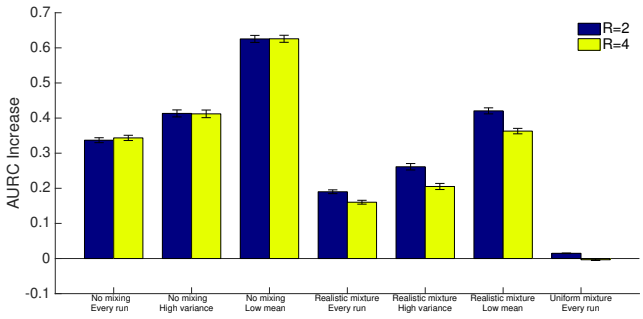
refer to conv match pursuit eq from intro



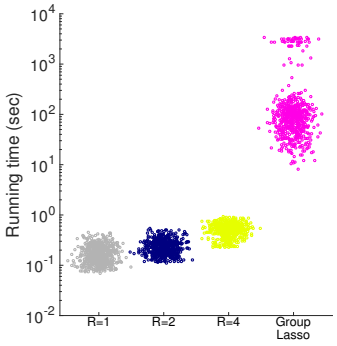
(a) Localisation performance for varying average signal cumulants, while $\sigma^2_{\text{noise}} = 1$ kept constant.



(b) Performance levels of source signal distribution mixtures within a single field of view



(c) The use of higher order features substantially increases localisation performance



(d) Substantial gain in running time compared to matrix factorisation

Figure 3.4: **a.** CHOMP incorporating higher order cumulants offers substantial gain in localisation performance when the corresponding structure is present in the signal distribution. Each cell shows the fraction of sources correctly localised in $n = 10000$ runs for different signal parameters (exact parameters were distributed normally with a standard deviation of one order of magnitude around the specified value). **b.** Localisation performance expressed as AURC for all runs ($n = 3 \times 10000 + 600$) with varying mixtures of sources within a single run. *No mixing*: All sources share the same signal distribution. *Realistic mixture*: as in (**a**). *Uniform mixture*: Signal distributions may vary up to 4 orders of magnitude within the same field of view. Means, 1σ and 2σ quantiles indicated. **c.** Assessing the improvement within a single field of view gained by incorporating higher order cumulants. Bars are the mean gains in AURC over the first order method, runs selected by signal distribution criterion ($n <= 10000$). *High variance*$>= 1$, *Low mean*$<= 0.1$. Error bars are SEM. **d.** Comparison of running times ($n = 4 \times 600$).

over two orders of magnitude faster[14] (see figure 3.4d).

3.3.2 Applications to neural data

Having validated CHOMP on data simulated from its generative model, we need to see how well and what it does on real neural recordings. To this end I will use the Neurofinder datasets carefully characterised in chapter 2, and unless otherwise stated, the data used as input to CHOMP has been transformed into photon flux estimates, divided by the non-uniform spatial gain, as the preprocessing suggests. Finally the resulting non-integer data has been stretched to the uint16 range and is represented by integers, as the expected format of most neural recordings. Note that although these transformations indeed equalise the expected background signal level, as shown in section 2.3.2, they can not fundamentally change the signal-to-noise ratios across the images, which should be taken into account during the reconstruction of higher order cumulants.

I first examine the characteristics of neural data in section 3.3.2.1, whether it indeed includes detectable co-cumulants generated by single neurons of small spatial extents, as the CHOMP generative model suggests. Next, in section 3.3.2.3, I show that CHOMP is indeed capable of reconstructing both the mean signal and higher order cumulants, and these can be used in localising single neural cell bodies. Finally I discuss the results of CHOMP on the neurofinder challange in section 3.3.2.4.

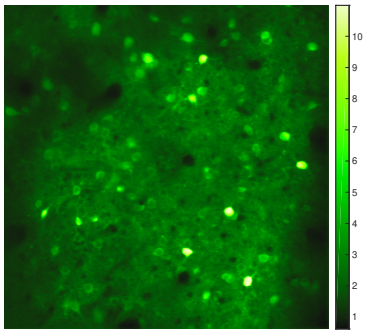
3.3.2.1 Higher order co-cumulants are present in the data

For a detailed example on what higher order cumulants and joint cumulants are present in the data, I will use the 01.00 dataset. Therefore first in figure 3.5, I characterise how the dataset have been transformed via preprocessing methods described in chapter 2 to achieve uniform background signal over the field of view. Furthermore, in figure 3.5b and e, one may see that there indeed is structured pixel-wise variance present, with visible shapes clearly matching cell-like objects in the mean image. Similar structures are present in the standardized 3rd and 4th pixel-wise univariate cumulants – skewness and kurtosis – as well, shown in figure 3.6. These pixel-wise higher order cumulants I will call *diagonal cumulants*, as they represent the diagonal of the joint cumulant tensors.

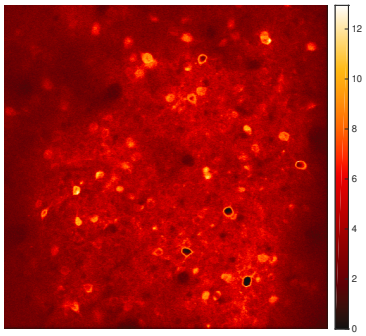
Although these diagonal higher order cumulants already show cell-like spatial structures, and thus may be beneficial to use as features during segmentation, the main idea of using higher order cumulants contains representing how activities of pixels vary jointly. Such joint variation then in principle enable one distinguish seemingly overlapping cells and segment them correctly. To illustrate this, let us first zoom in on the mean and variance images, and see how covariance tensors may be interpreted in figure 3.7. The ultimate difficulty that CHOMP attempts to solve, but is hard for humans to interpret, is to retain a sense of spatial closeness of pixels, while still representing their joint cumulants. CHOMP achieves this by using higher order tensors, such that the covariances of

[14] Note that I could evaluate the group lasso method only on smaller sample of $n = 600 \ll 10000$ due to prohibitively slow running times.

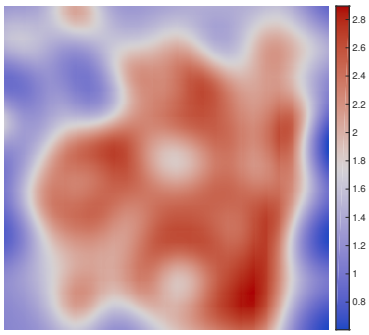
Figure 3.5: How spatial correction affects mean and variance of dataset 01.00. Although the background mean signal is indeed correctly equalised, as shown in the different between (a) and (d), the signal-to-noise ratio does not change by divisive normalisation, and thus originally darker areas - generally with lower signal to noise ratios - now appear to have higher pixel-wise variance in (e), compared to (b). The numeric differences are caused by (a) and (b) representing actual photon flux estimates, whereas (d) and (e) have been stretched, such that the integer dataset utilises the uint16 range optimally; the stretch factor was 3747.



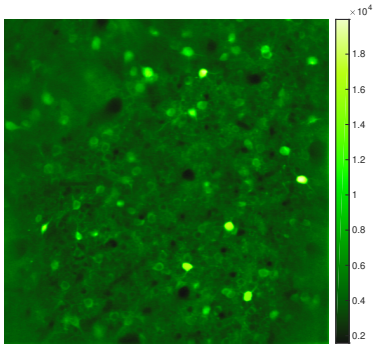
(a) Mean over time of photon fluxes estimated at each pixel and frame.



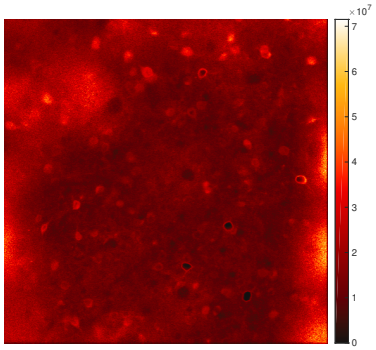
(b) Variance over time of photon fluxes estimated at each pixel and frame.



(c) Spatial gain non-uniformity, used for correction.

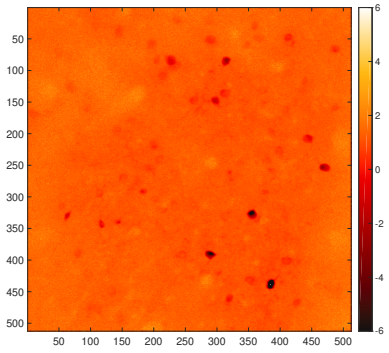


(d) Mean over time of photon fluxes estimated at each pixel and frame and corrected for spatial non-unformity.

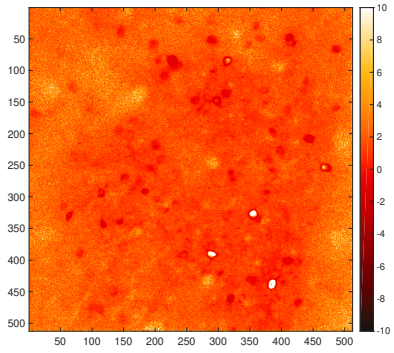


(e) Variance over time of photon fluxes estimated at each pixel and frame and corrected for spatial non-unformity.

Figure 3.6: How spatial correction affects skewness and kurtosis of dataset 01.00. Both images represent standardised cumulants, that is the raw n-th cumulant divided by σ^n, where σ is the standard deviation of the same pixel. Standardised cumulants are unaffected by spatial gain correction.



(a) Standardised 3rd cumulant (skewness) over time of photon fluxes estimated at each pixel and frame.

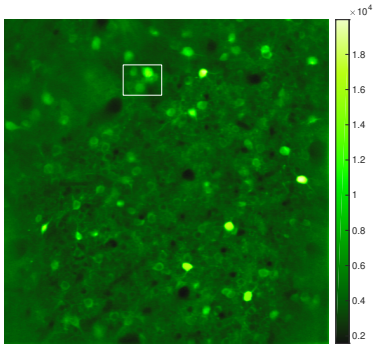


(b) Standardised 4th cumulant (kurtosis) over time of photon fluxes estimated at each pixel and frame.

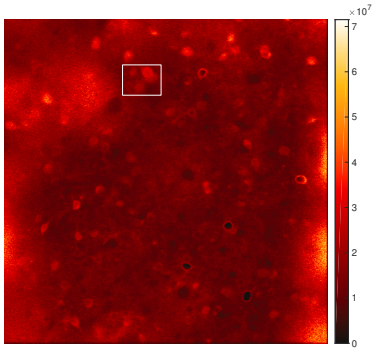
$D = 2$ dimensional data would be a $D \times D = 4$-th order tensor, whereas their joint kurtosis could only possibly be represented by a $D \times D \times D \times D = 16$-th order one. Although as discussed earlier, these tensors are symmetric, with few unique elements, that doesn't help understanding what they represent, or to visualise them. Visualisation is best done in 2D via flattening the tensors[15], and getting used to the interpretation. As we are more familiar with covariance matrices over vectors, which do retain spatial closeness, I first show in figure 3.7 (e-h) the covariance matrix along a single vertical line. Firstly, (g) shows that the covariance matrix has a strong diagonal, due to the independent Poisson noise for each pixel, and although off-diagonal structure is present, it is difficult to interpret. In order to visualise the cell-induced covariances, I indicate the presumed cell locations along the line in (f), and show a Gaussian-filtered ($\sigma_{\text{filt}} = 1.2$ pixels) covariance matrix, from which the diagonal was removed before filtering and treated as missing data. The two overlapping cells now indeed have visible and distinguishable covariance structures despite the high degree of overlap, and also illustrate the reason we decided to use cumulants - the overlapping area has clearly higher covariances than the individual cells, but covariances are additive, and thus ideal for matching pursuit type 'explaining away' modelling, such as CHOMP. Others have also recognised the importance of such off-diagonal joint variation, such as using cross-correlation metrics to find co-varying pixels(Cite: Marius suite2p). I do believe other metrics are generally less robust due to their lack of additivity, and would advise the use of joint cumulants for separation of additive overlapping sources.

[15] In this chapter all flattening was carried out in Matlab (R), therefore using Fortran-style (column first) reshaping.

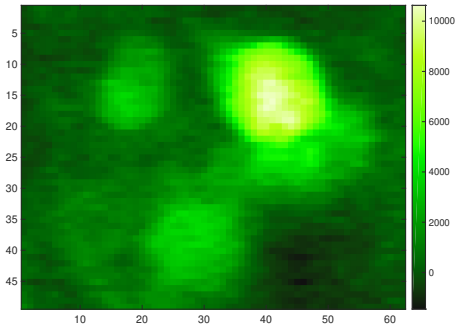
Figure 3.7: Co-variances are present in the data. The mean and variance images in (c)-(d) are the patch indicated in (a) and (b). (e)-(h) illustrate covariances along a vertical line, (i)-(l) along a horizontal line, and (n)-(q) in a small patch. (m) shows spatial autocorrelation.



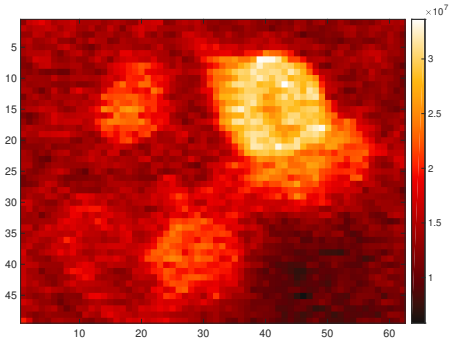
(a) Mean over time of photon fluxes estimated at each pixel and frame.



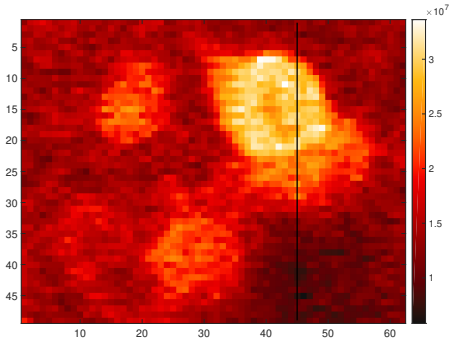
(b) Variance over time of photon fluxes estimated at each pixel and frame.



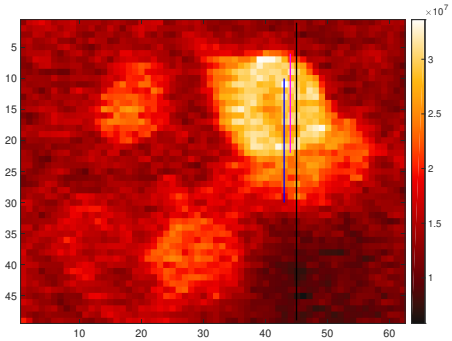
(c) Mean in a smaller area.



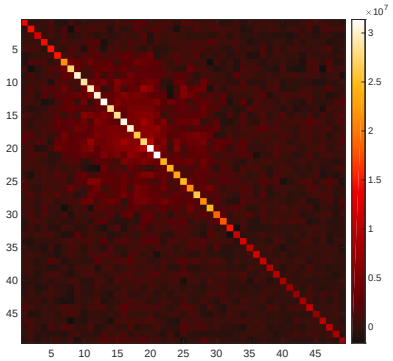
(d) Variance in a smaller area.



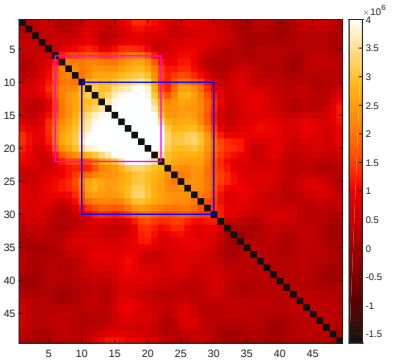
(e) Black line indicates selected pixels for (g).



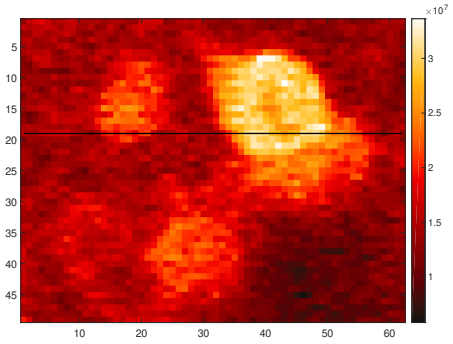
(f) Colored lines indicate highlighted regions in (h).



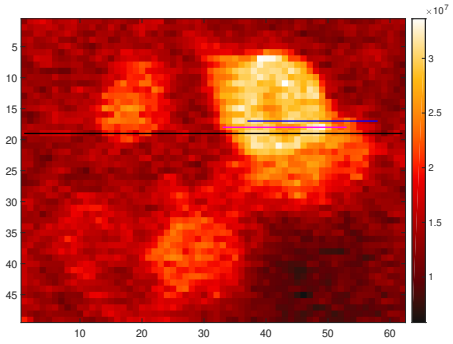
(g) Covariance matrix of the selected pixels, diagonally dominant and noisy.



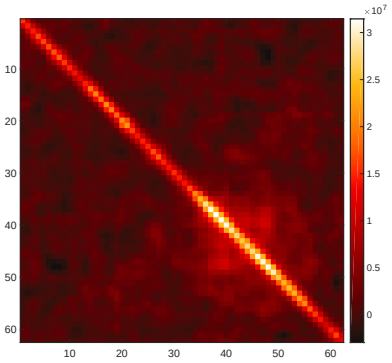
(h) Smoothed covariance matrix with diagonal removed, cell covariances are indicated.



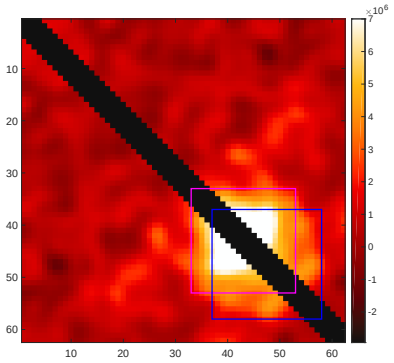
(i) Black line indicates selected pixels for (k).



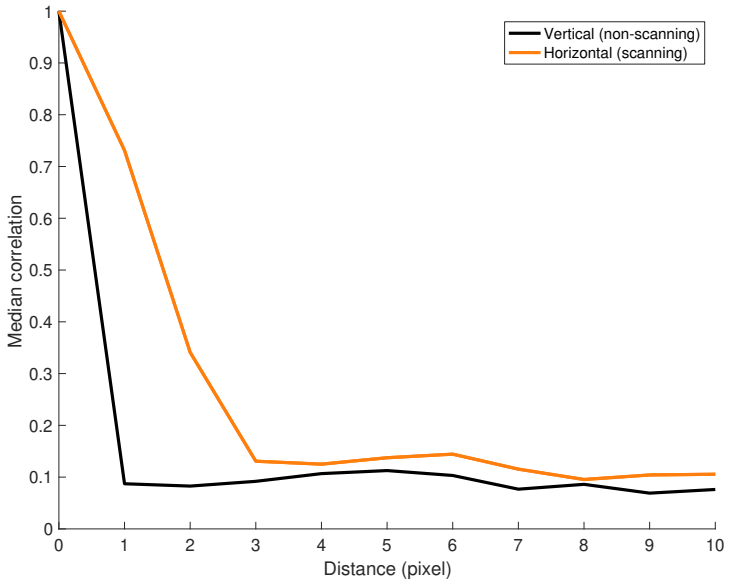
(j) Colored lines indicate highlighted regions in (l).



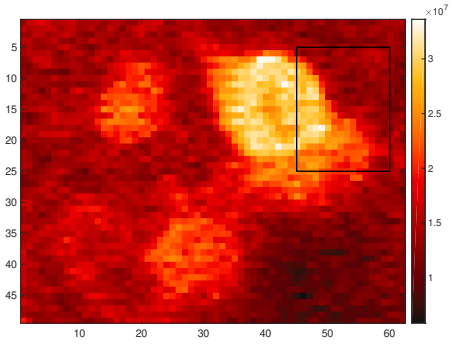
(k) Covariance matrix of the selected pixels, shows spatial autocorrelation (multiple strong diagonals).



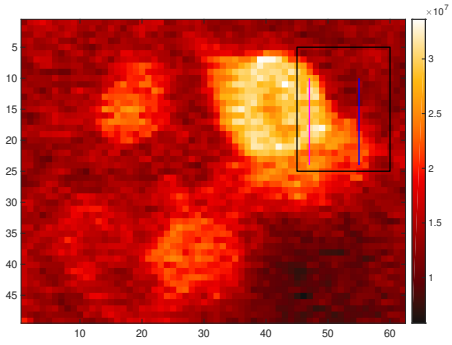
(l) Smoothed covariance matrix with diagonals removed, cell covariances are indicated.



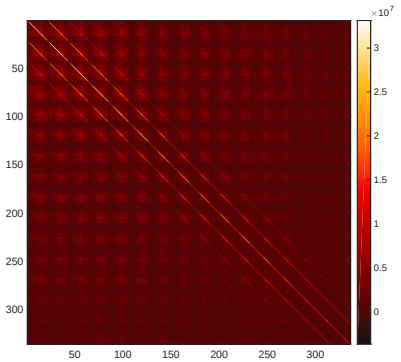
(m) Median correlation of pixel activities decays with distance. In the scanning direction we can see significant spatial auto-correlation, caused essentially by temporal autocorrelation of the two-photon imaging setup. In the non-scanning (vertical) direction, autocorrelation is completely absent.



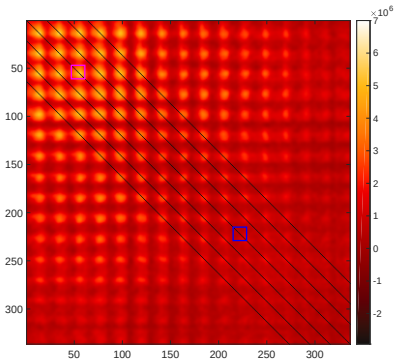
(n) Black box indicates selected pixels for (p). All pixels within the box contribute.



(o) Colored lines representing pixels, whose covariances are highlighted in (q) as blocks.



(p) Covariance matrix of the selected pixels. The blocked structure arises from the fact that this covariance matrix no longer retains spatial closeness in the horizontal direction. This is well illustrated by the fact that the strong off-diagonals (the results of scanning autocorrelation) are offset by 21 pixels – the height of the box in (n).



(q) Smoothed covariance matrix with diagonals removed. The highlight boxes indicate the small covariance boxes for the pixels sub-selected in (o), but also note the co-covariance blocks between those sets of pixels, due to both lines overlapping with the rightmost cell.

Regardless of metric, one must be careful about potentially unexpected artefacts in the data, which is introduced by the experimental system. One such common artefact, that strongly corrupts our mental model of cell-induced covariances, is shown in figure 3.7 (i-l). By simply examining the covariances matrix along a horizontal direction, we find that instead of a single strong diagonal representing independent pixel variances, there are multiple strong diagonals in (k), indicating spatial auto-covariance. The fact that this only shows up in the horizontal direction, but not the vertical one, is a strong indicator that this is in fact not a spatial effect, but rather the temporal auto-correlation induced by the two-photon microscope system, strictly in the scanning direction. In order to discover truly cell-induced covariance structures in the scanning direction, with thus need to estimate this autocorrelation length (m), and remove the appropriate diagonals to reveal the structure of off-diagonal covariances (j,l).

Thus far we did not need to worry about retaining the sense of spatial closeness, as we were

examining covariances along a single line. My last example highlights the difficulty of visualisation and interpretation in the realistic case, in which one attempts to reveal covariances within a full image patch. In figure 3.7 (n-q) I first indicate a patch of pixels and its covariance matrix in (n,p), revealing a blocked structure due to the flattening process disturbing the sense of spatial closeness in the horizontal direction. This is well illustrated by the fact that the strong off-diagonals (the results of scanning autocorrelation) are offset by 21 pixels, which is the height of the selected patch in (n). Finally to aid with the understanding of this blocked structure, two colored vertical lines are indicated in (o), whose covariance blocks are shown in (q). Note the existing higher cross-covariances between the two lines of pixels (found at the intersection of coordinates of individual blocks), which is induced by the rightmost cell contributing signal to both sets of lines.

Finally, as an illustration that such co-cumulant tensors may be computed for higher order cumulants as well, the resulting co-cumulants of different orders are shown in figure 3.8 for 5 pixels along the vertical line shown in figure 3.7e. It is difficult to see structure in this by eye, it merely demonstrates the exponential growth of tensor size. This is the reason while CHOMP never explicitly represents these co-cumulant tensors, merely their projections onto the basis functions, and even then it exploits the tensor symmetricity to reduce both storage and computation costs; without these reductions, the algorithm would be significantly slower and potentially impossible to run without using out-of-memory storage.

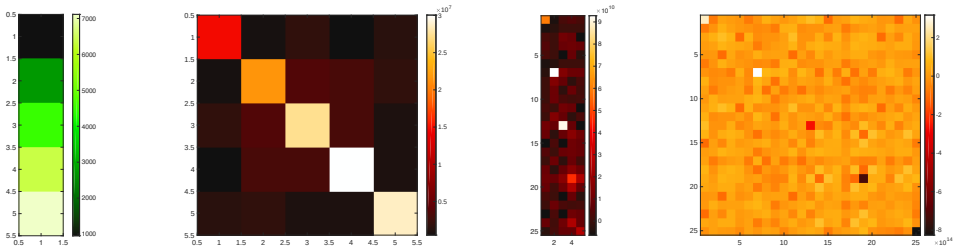


Figure 3.8: Illustration of co-cumulants of different orders. From left to right we see flattenned representations of 5 pixels' co-cumulants in orders 1 through 4. Although structure is present, it is progressively more difficult to understand it as the tensor order is increasing.

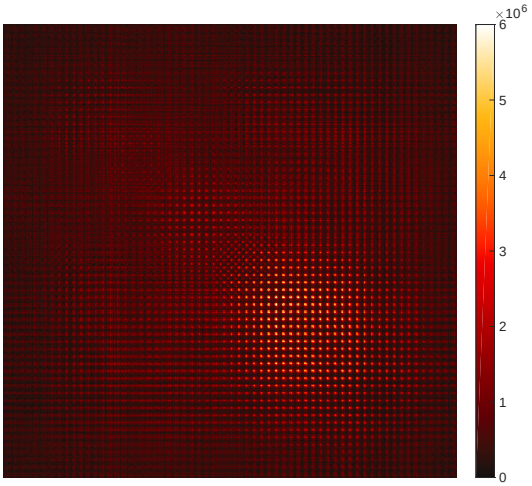
3.3.2.2 CHOMP reconstruct higher order co-cumulants

Having discovered that the data indeed contains interesting higher order structure, I now show that CHOMP is capable of reconstructing such structure, and thus these features can be used to localise regions of interest that resemble active neurons.

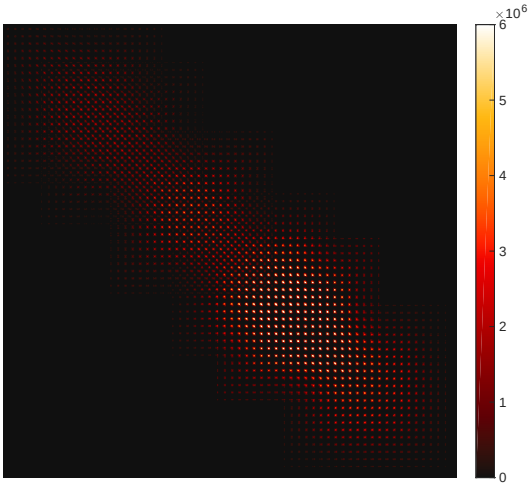
3.3.2.3 CHOMP enables various definitions of what a neuron is

MAYBE Add this

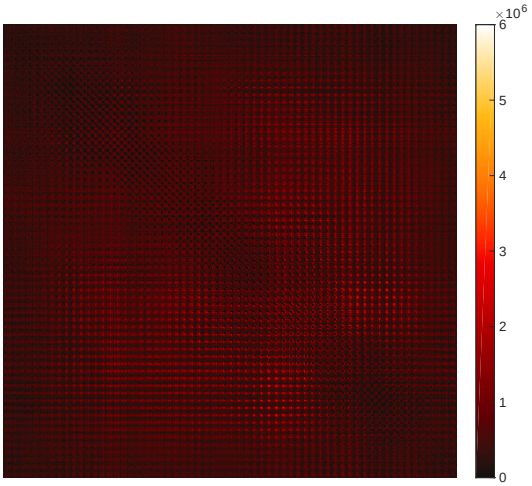
Figure 3.9: CHOMP reconstructs the full off-diagonal patch covariance. (a-c) show the full covariance for the whole zoomed region in figure 3.7, but is difficult to interpret. Therefore (d-g) and (h-k) show more interpretable subsets of the CHOMP reconstruction, along the horizontal and vertical lines used before.

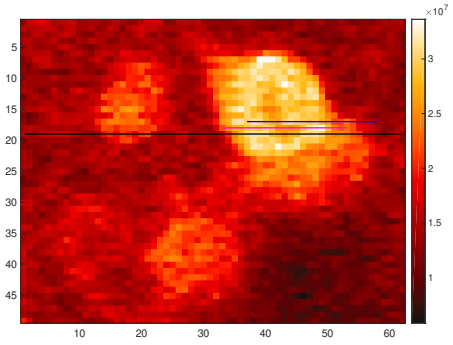


(a) Full flattened covariance tensor of the 49x62 pixel region shown in figure 3.7c-d.

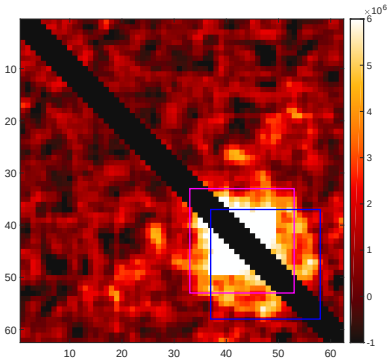


(b) Reconstruction of (a) by CHOMP, using 7 sources with a patch size of 23x23. The localisation and reconstruction was based on the off-diagonal covariances only, no diagonals or other cumulant orders were used.

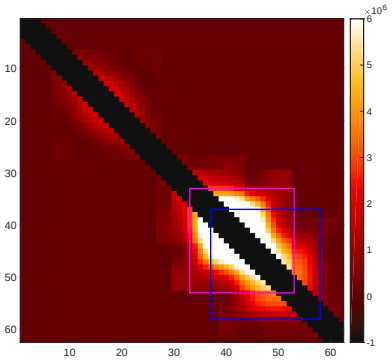




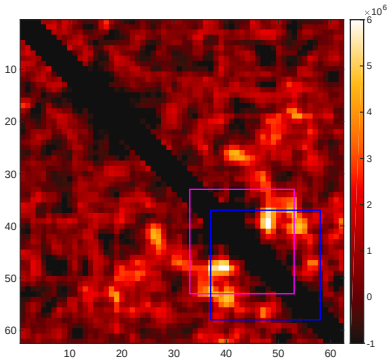
(d) Pixelwise variance, with black line selecting pixels for the covariance matrix in (e) and colored lines indicate boxes in (e-g). Reproduction of figure 3.7j.



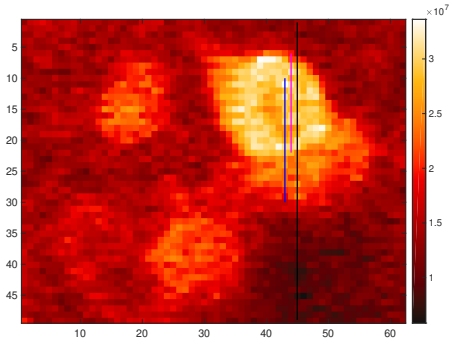
(e) Original covariance along line of pixels.



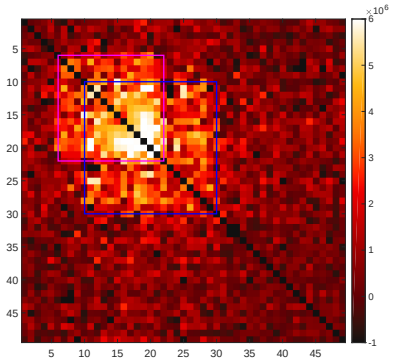
(f) CHOMP reconstruction of covariance.



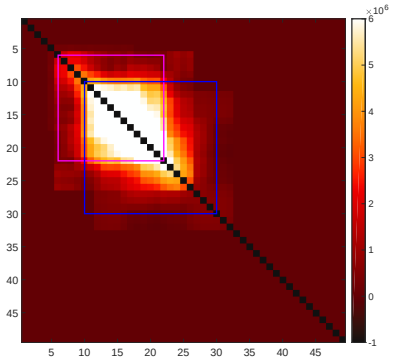
(g) Residual covariance.



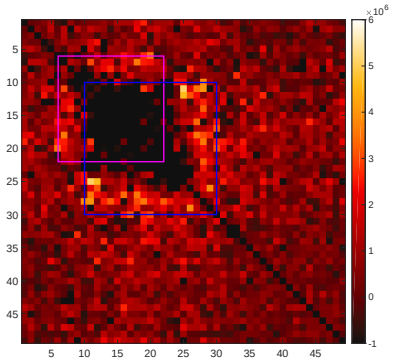
(h) Pixelwise variance, with black line selecting pixels for the covariance matrix in (i) and colored lines indicate boxes in (i-k). Reproduction of figure 3.7f.



(i) Original covariance along line of pixels.



(j) CHOMP reconstruction of covariance.



(k) Residual covariance.

CHOMP encompasses numerous algorithms, and provides a flexible definition of what a neuron is. By changing the definitions, it can reprodicibly reorder the cells found according to how well they match our current definition.

3.3.2.4 Results on the Neurofinder challenge

Finally to show the applicability of preprocessing and CHOMP across numerous datasets, I discuss the results on the Neurofinder challenge. The datasets were introduced in Chapter 1....

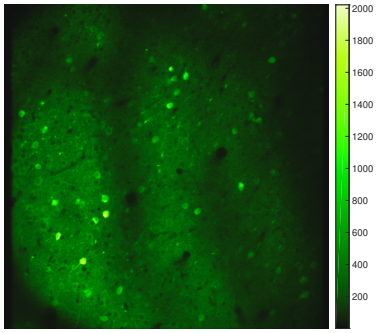
The next task is of course to validate whether the extra ROIs found by preproc+CHOMP could indeed be cells, and why are some - that are amongst the neurofinder labels - missed by the proposed algorithm. In .

write more here about 00.00 results

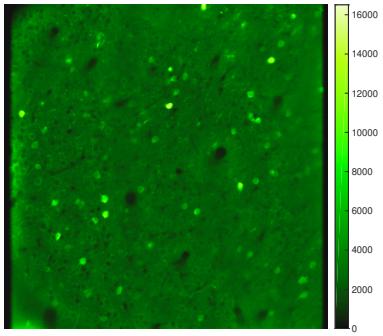
write more here about 00.00 results

3.4 Discussion

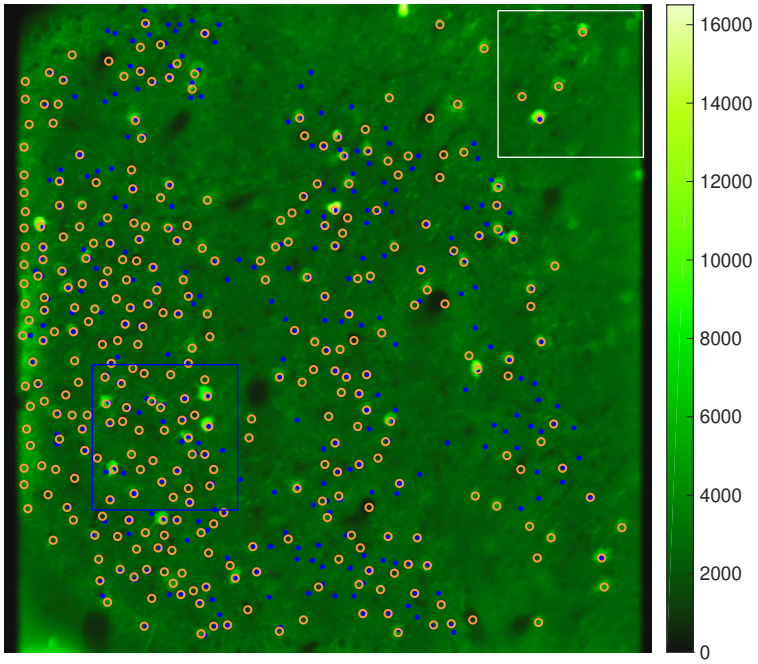
Figure 3.10: CHOMP results on dataset 00.00.



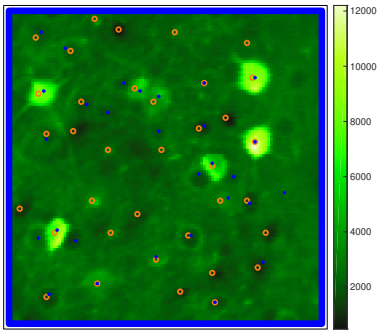
(a) Original mean image



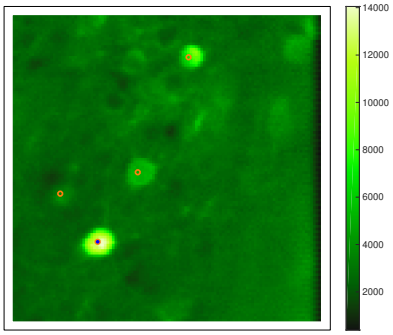
(b) Preprocessed mean image



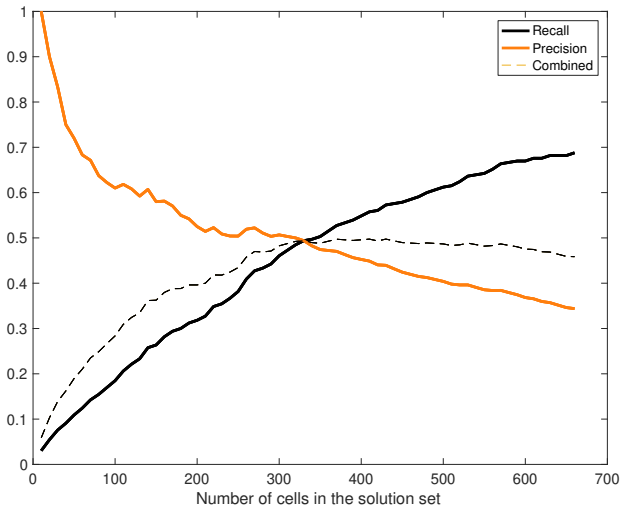
(c) Preprocessed mean image. Neurofinder-labelled (blue) and CHOMP-identified (orange) cell locations are superimposed. The markers represent the centre of mass of individual binary region of interests. The 330 ground truth locations, and the first 330 CHOMP identified locations are shown.



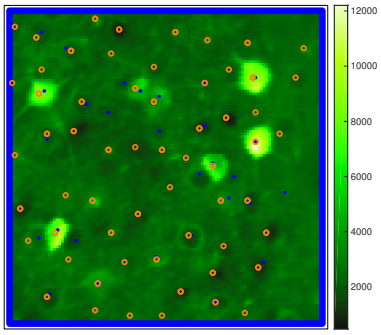
(d) Zoomed in block, indicated by blue outline in (c).



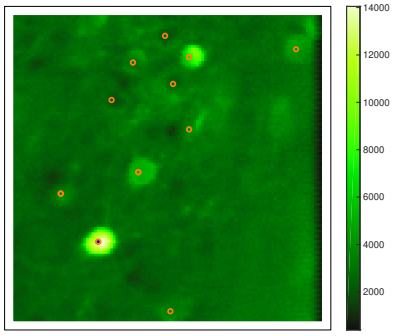
(e) Zoomed in block, indicated by white outline in (c).



(f) Precision-recall curve calculated by the external neurofinder script, given the 330 ground truth locations and the first N locations identified by CHOMP.

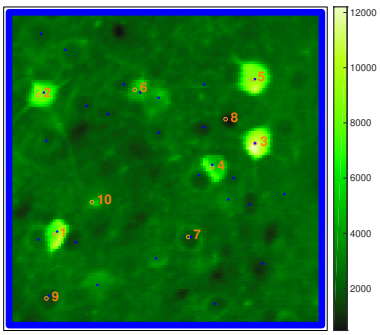


(g) As in (d), but using double the number of cells in the CHOMP solution set.

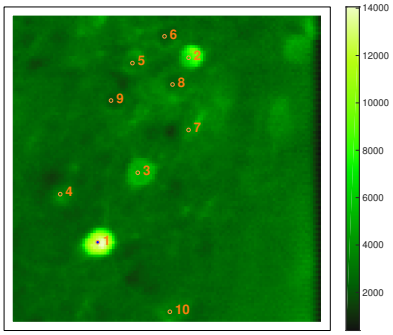


(h) As in (e), but using double the number of cells in the CHOMP solution set.

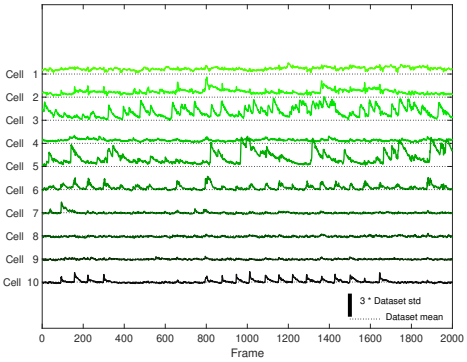
Figure 3.11: Discussing CHOMP results on dataset 00.00. CHOMP identifies additional ROIs that have cell-like activity (b,d). The preprocessing makes signals more comparable across the field of view (a-f), revealing these additional cells (d,f). CHOMP misses some inactive ROIs close to highly active ones (g-h). Supplemental videos are available for the zoomed in areas in (a) and (b).



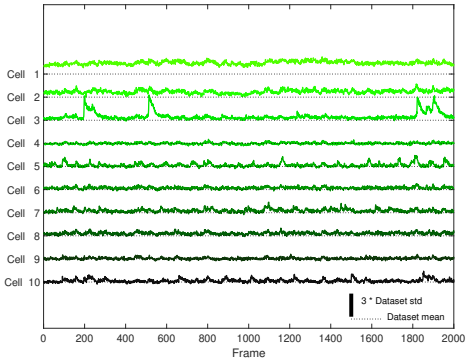
(a) As in figure 3.10d, but showing only the top 10 ROIs suggested by CHOMP.



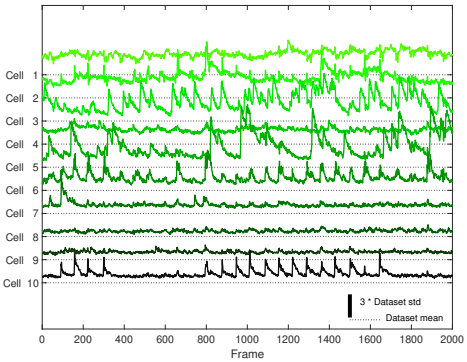
(b) As in figure 3.10e, but showing only the top 10 ROIs suggested by CHOMP.



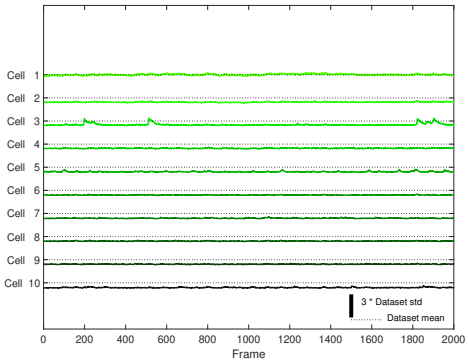
(c) Activity time courses from the 10 numbered ROIs in (a), identified by CHOMP.



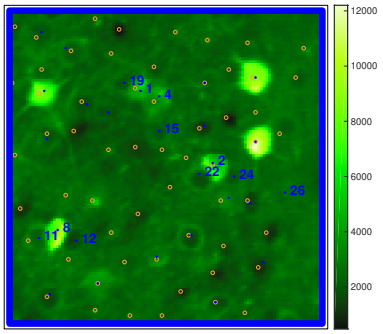
(d) Activity time courses from the 10 numbered ROIs in (b), identified by CHOMP.



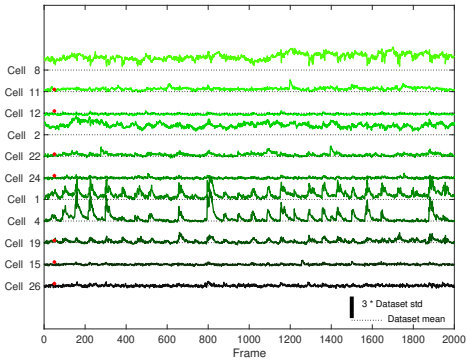
(e) As in (c), but extracting from the original dataset instead of the preprocessed one.



(f) As in (d), but extracting from the original dataset instead of the preprocessed one.



(g) Numbering Neurofinder ROIs that were missed by CHOMP, or are near the missed ROIs.



(h) Time courses of ROIs in (g), with the red asterisk indicating cells that were not found by CHOMP.

Chapter 4

Learning interpretable models of latent stochastic dynamical systems

4.1 Understanding the dynamics of neural recordings

When a neuroscience experiment is set up, we often aim to investigate the computation that a brain has to carry out in order to successfully solve an experimenter-defined task. Examples include probing premotor cortical areas for information relating to planned movement (Churchland et al. 2012) or looking for integration of evidence for decision making within the prefrontal cortex (Mante et al. 2013). Ultimately we often assume that neurons collectively represent so-called 'latent' or 'computational' variables related to the task, and much of the recorded signal may be explained by successfully uncovering the computational variables the neurons represent. To ensure that our understanding of the system's complexity is well-captured by our recovered latent variables, we also want to ensure that they represent a stochastic, time-homogeneous Markovian state space - that is, given the current state of the system, the next state will not depend on the past history of states. This makes our set of recovered computational variables in some sense *complete*: such a set of variables enables us to reason about the individual system states, as well as paths through the state space, without having to consider non-(temporally)-local effects. Furthermore, it lets us investigate the dynamics of the state space, so that we can make probabilistic predictions of future states of the system and its representation within the recorded neurons, thus enabling us to estimate the error inherent in any model.

Sketching the dynamical portrait of even noiseless, simple systems is no easy task. In order to glean insight into the noisy, subsampled data recorded of an extremely complicated system carrying out an experimenter-defined task, we need to make certain choices and compromises.

One popular approach investigators often take, is to use a simple model to fit to available data, and examine how this simplified description changes over time, or as we vary experimental parameters. Examples include fitting linear dynamical systems to monkey reach data, where the change in

dynamics is parameterised by the reach direction (Soldado Magraner 2018); or our work, in which the linear dynamics' parameters are allowed to slowly vary over time via building a hierarchical model with a Gaussian Process prior (Park, Bohner, and Macke 2015). Although linear dynamics can be learned efficiently in closed form and their changes can be thoroughly interpreted (Soldado Magraner 2018), a frequent criticism of using linear dynamical systems for explaining the medium timescale evolution of systems, is that such descriptions do not adequately capture the data itself, so one has to be very careful when interpreting results. However, allowing for linear dynamics' parameters to vary more rapidly in time (such as Duncker et al. 2017), make their interpretations difficult.

An alternative, more recent approach has been to fit models with complex, non-linear Markovian dynamics[1] to multivariate time series data. Examples are latent recurrent neural networks (RNN, see e.g. LFADS Sussillo et al. 2016; Pandarinath et al. 2018a) and Gaussian Process State Space Models (GPSSM, see e.g. Damianou, Titsias, and Lawrence 2011; Eleftheriadis et al. 2017, without neural data applications). Although these methods are powerful at inferring the complex and noisy trajectories of the computational variables encoded by the recorded neural population activity, and they do contain explicit description of the dynamical transition function, the results only really provide insight into 'what' the system does (following those inferred trajectories), rather than the 'how' it does it (via implementing interpretable dynamics that lead to those inferred trajectories). These problems have been recognised by others too, and they turned to the classic method of describing the learnt model dynamics by 'opening the black box', and finding the fixed points of the model (Sussillo and Barak 2013; Golub and Sussillo 2018).

The idea of utilising fixed points and locally linearised dynamics as human-interpretable descriptions of complex dynamical systems is indeed a powerful one, and has a long history in physics. Bifurcation theory concerns itself with identifying the number and type of fixed points in a system. Its application to deterministic systems is part of the standard curriculum, however identifying - or even defining - fixed points or attractors in stochastic dynamical system is in its early stages (Arnold and Crauel 1991; Diks 2006; Wang, Chen, and Duan 2018), and thus random fixed point or stochastic bifurcation theory is not easily applicable to experimental data recorded from unknown systems. Therefore the approach used by the methods above (e.g. Golub and Sussillo 2018) was to simulate trajectories from the learned systems (either noiseless, or averaging noisy ones), and find deterministic descriptions of the fixed points of the system.

I felt that this methodology - fitting a stochastic dynamical system to data, then finding fixed

[1] There also seems to be a lot of confusion in the terminology of what people call a 'dynamics' or 'dynamical model'. In my definition, also stated as above, a latent dynamical system is described by not only a latent state that evolves over time, but I put a further requirement of having a well-defined Markovian state transition function as part of the model. In my view, this requirement disqualifies several methods that implement arbitrary non-Markovian time correlation structures, such as GPFA (see Yu et al. 2009; Duncker and Sahani 2018), including extensions such as Gaussian Process Latent Variable Models (e.g. Wu et al. 2017) and Variational Latent Gaussian Processes (Zhao and Park 2016). Although these are powerful at inferring latent trajectories, they do not provide explicit descriptions of the latent dynamics, and depending on the GP kernel choices, the trajectories likely do not exhibit first-order Markovian dynamics.

points using the model only - could be improved by directly including deterministic fixed point descriptions as parameters of the model, and fitting the observed data while simultaneously learning point estimates or posteriors over the fixed points and local linearisations. This directly results in a human-interpretable description of an unknown, noisy dynamical system, while ensuring that our uncertainty about the number, location or properties of the fixed points is propagated correctly, and taken into account as part of the model fit.

The main idea of conditioning the transition dynamics on fixed points is described in section 4.2, then I discuss the case when the transition dynamics function has a Gaussian Process prior, leading to a conditioned Gaussian Process State Space Model, in section 4.2.1. My first specific implementation – submitted to ICML 2018 (Bohner and Sahani 2018) and presented at COSYNE 2018 (Sahani 2018) – is described in section 4.2.2, whereas a subsequent, more flexible algorithm – the result of a collaboration with Lea Duncker and Julien Boussard and published at ICML 2019 (Duncker et al. 2019) – is shown in section 4.2.3. Finally the two described algorithms are applied to simulated systems in section 4.3, to demonstrate their ability to both infer latent trajectories, and provide correct and interpretable descriptions of the dynamics given moderate datasets, including ones mimicking typical neural recordings in sections 4.3.4 and 4.3.5.

4.2 Modeling latent stochastic dynamics with fixed points

In order to build up towards the idea of finding the fixed points within dynamical maps fitted to repeated time-series recordings, we need to first describe the individual elements of a model that could implement such a task, then find a way of combining them together. I will discuss a very general model at first, and show the particular choices I made gradually - keeping in mind that there are many such choices, which all lead to a different specific algorithm, some potentially better or more broadly applicable than mine. My aim here was not to explore the complete space of all such potential algorithms, or argue my choices are the best possible ones; but rather to describe and share a working implementation of an idea that I hope will spark a novel way of looking into dynamical systems fit to neural recordings.

First, let us define the data we wish to model. It consists of repeated measurements (trials) of a time series

$$\mathbf{y}(t) \in \mathbb{R}^{\mathrm{N}}, \;\; t \in [1, T] \,, \tag{4.1}$$

where the output dimensionality N often corresponds to the number neurons recorded from, and T could be the number of video frames in calcium imaging, or time bins in spike sorted electrical recordings. In the simplest case, each trial will consist of the same number of neurons and time points, and thus the collection of trials may be described as either the set $\mathcal{Y} = \{\mathbf{y}^{\mathrm{m}}(t)\}_{\mathrm{m=1...M}}^{\mathrm{t=1...T}}$ or

simply the tensor $\mathbf{Y}_{\text{ntm}}$. Note that for the algorithm described here, we do not require the trials to have the same number of time points in trials, but it greatly simplifies the mathematical description. We do - for now - require the time points to be equally spaced in time (which is usually the case with microscopy and time-binned data), but in an extension described later (which was part of a collaborative effort to improve my initial model), that requirement will also be dropped (see section 4.2.3).

Our first modeling step involves dimensionality reduction of the time series data, by assuming there is a lower dimensional latent dynamical space, which is capable of capturing the interactions between the often large number of recorded neurons. Let us denote timeseries within this latent space as $\mathbf{x}(t) \in \mathbb{R}^{\text{D}}$, where D is the latent dimensionality. Therefore the first modelling step of dimensionality reduction is

$$\mathbf{y}(t) = g(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\text{y}}(t) \ , \tag{4.2}$$

where our goal is to infer latent time series $\mathbf{x}$ and learn an output mapping function $g : \mathbb{R}^{\text{D}} \mapsto \mathbb{R}^{\text{N}}$, such that the discrepancies $\boldsymbol{\varepsilon}^{\text{y}}(t)$ are minimised (according to some metric and aggregation[2]).

Next, we wish to learn about the dynamical evaluation of the latent time series. At our current level of understanding most real biological process are 'inherently' noisy, therefore we need to describe our dynamical evaluation as a stochastic differential equation $d\mathbf{x} = f'(\mathbf{x})\,dt + \sqrt{\boldsymbol{\Sigma}^{\text{x}}}\,d\mathbf{w}$, where $\mathbf{w}(t)$ is a Wiener noise process. However, given we are assuming our data is on a fixed step time grid, we can use the significantly simpler language of stochastic state-space models, a discrete time analogue of stochastic differential equations (SDEs). Therefore we can write

$$\mathbf{x}(t+1) = f(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\text{x}}(t) \ , \tag{4.3}$$

where we need to learn a dynamical transition map $f : \mathbb{R}^{\text{D}} \mapsto \mathbb{R}^{\text{D}}$ and the level of transition noise $\boldsymbol{\varepsilon}^{\text{x}}(t)$.

Finally, to enhance our insight into the dynamical process, we wish to explicitly model the locations of the fixed points in the transition map $f(\cdot)$ and the behaviour of the system around them through local linearisations. Let $\mathbf{s}^{\text{l}} \in \mathbb{R}^{\text{D}}$ be a fixed point in f, and $\mathbf{J}^{\text{l}} \in \mathbb{R}^{\text{D}\times\text{D}}$ the Jacobian matrix of f evaluated at that point $\left(\text{ie. } \mathbf{J}^{\text{l}} = \nabla_{\text{x}} f(\mathbf{x})\,|_{\mathbf{x}=\mathbf{s}^{\text{l}}}\right)$. If we then choose a parametrisation of f that depends explicitly on these fixed points and local linearisations, we can learn these parameters as part of our model fitting process.

The full model is then

[2] In the simplest case the squared error summed over all neurons, time points and trials, $\sum_{\text{n,t,m}} \varepsilon^{\text{y}}_{\text{nm}}(t)^2$. Minimising this results in the optimal solution if we assume the observations have uniform isotropic Gaussian noise.

$$\begin{aligned}\mathbf{x}(t+1) &= f(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\mathrm{x}}(t)\\ \mathbf{y}(t) &= g(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\mathrm{y}}(t)\end{aligned} \tag{4.4}$$

$$\text{such that } \forall\, \mathbf{s}^{\mathrm{l}}, \mathbf{J}^{\mathrm{l}}$$

$$f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}} \text{ and } \nabla_{\mathrm{x}} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{s}^{\mathrm{l}}} = \mathbf{J}^{\mathrm{l}}$$

In such a model, we succeed to fit the data, and to gain direct scientific insight into the behaviour of a system that could have generated the observations, in a single step. What parameterisation of f may enable us to include the fixed point parameters and still fit datasets with various output and noise models? Recent work on Gaussian Process State Space Models (GP-SSMs) has shown great promise in modelling stochastic latent non-linear discrete time dynamical systems in a variety of applications (e.g. Frigola, Chen, and Rasmussen 2014; McHutchon 2014); furthermore, functions drawn from a Gaussian Process also lend themselves to be conditioned on their fixed points and derivatives. As it uniquely combines these strengths, I chose to use a Gaussian Process parameterisation for the transition map function f. This choice informs both the inference and learning steps of any algorithm, and thus will now be discussed in detail.

4.2.1 A Gaussian Process prior with fixed point parameters

Let the D-valued transition map function be a collection of D independent scalar-valued functions, $f(\cdot) = [f^1(\cdot), f^2(\cdot), \ldots, f^{\mathrm{D}}(\cdot)]$, a common technique when representing multiple-output functions via Gaussian Processes.

A Gaussian Process prior over a scalar-valued function $f^{\mathrm{d}}(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ with given mean and covariance functions, $m(\cdot) : \mathbb{R}^{\mathrm{D}\times\mathrm{P}} \mapsto \mathbb{R}^{\mathrm{P}}$ and $k(\cdot,\cdot) : \mathbb{R}^{\mathrm{D}\times\mathrm{P}} \times \mathbb{R}^{\mathrm{D}\times\mathrm{Q}} \mapsto \mathbb{R}^{\mathrm{P}\times\mathrm{Q}}$ defines a joint distribution of function values for an arbitrary set of input points $\mathbf{X} \in \mathbb{R}^{\mathrm{D}\times\mathrm{P}}$, such that

$$f^{\mathrm{d}}(\mathbf{X}) \sim \mathcal{N}\left(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X})\right) \tag{4.5}$$

In addition to the mean and covariance functions[3], we now wish to incorporate the fixed points $\mathbf{S} = \left[\mathbf{s}^1, \ldots, \mathbf{s}^{\mathrm{L}}\right] \in \mathbb{R}^{\mathrm{D}\times\mathrm{L}}$ and the local linearisations $\mathbf{J} = \left[\mathbf{J}^1, \ldots, \mathbf{J}^{\mathrm{L}}\right] \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}\times\mathrm{L}}$ where L is the number of fixed points. A Gaussian Process conditioned on these implies a joint distribution of not only the function values at given inputs, but also the fixed point and derivative parameters (which are mathematically equivalent to observations in a GP, but their values may change during the model estimation algorithm). In order to describe this joint distribution, we first need to define the derivatives of the mean and covariance functions, as they couple the Jacobians (essentially equivalent to derivative observations) to function value observations.

[3] that may different for each output dimension, resulting in a set of m^{d} and k^{d} functions

Fortunately the derivative of a Gaussian Process is a Gaussian Process by itself, with the new mean function being the derivative of the mean function $\nabla m(\cdot)$, and the covariance function being the gradient of the covariance function with respect to both inputs $\nabla_1\nabla_2 k(\cdot,\cdot)$ (see in appendix B). Furthermore, the cross-covariance of a Gaussian Process and its derivative can also be computed, by taking the gradient of the original covariance function with respect to only one of its inputs, $\nabla_1 k(\cdot,\cdot)$ or $\nabla_2 k(\cdot,\cdot)$. Therefore the complete joint normal distribution of the $f^{\mathrm{a}}(\cdot)$ function values at given locations $\mathbf{X}$ induced by a Gaussian Process that includes fixed points $\mathbf{S}$ and their local linearisations $\mathbf{J}$ can be written as

$$\begin{bmatrix} \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ f^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m^{\mathrm{s}} \\ \nabla m^{\mathrm{s}} \\ m^{\mathrm{x}} \end{bmatrix}, \begin{bmatrix} k^{\mathrm{ss}} & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{ss}} & \nabla_1\nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.6}$$

where $\mathbf{S}_{\mathrm{d}\cdot}$ and $\mathbf{J}_{\mathrm{d}\cdot\cdot}$ selects the d-th entry along the first dimension, and the superscripts indicate the inputs to the functions m and k, with s representing the fixed point locations $\mathbf{S}$, whereas x corresponds to the locations of interest $\mathbf{X}$. Note that the locations of the Jacobians always correspond to that of a fixed point, therefore for example $\nabla_1 k^{\mathrm{sx}} = \nabla_1 k(\mathbf{S},\mathbf{X})$ indeed represents the cross-covariance of the (mean-centred) Jacobians and potential function value observations, with the Jacobians and gradient function outputs appropriately reshaped[4].

An equivalent, alternative view of this prior is instead of examining the joint distribution of fixed-point related parameters and function values as in equation 4.6, we rather define a conditional Gaussian Process on the function values:

$$p(\, f(x)\,) = \mathcal{GP}(\, f(x) \mid \forall_1 f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}}\,,\ \nabla f(\mathbf{s}^{\mathrm{l}}) = \mathbf{J}^{\mathrm{l}}\,), \tag{4.7}$$

where the m mean and k covariance functions are omitted from the notation for clarity, but are implied by any GP. Note that the algebraic computations (including the conditioning itself) is generally done based on the joint view in equation 4.6, but I do feel that this idea of thinking of these conditional GPs provides additional insight.

As we are establishing a powerful prior over a completely unknown function that we wish to estimate from data, we need to insert a little more flexibility into it. The main concern is a priori fixing the number of fixed points. In order to automatically select the number of fixed points during the inference and learning processes described later, I choose the strategy of setting the number

[4]The gradient operation increases the tensor order of the function, which then need to be flattened to represent the corresponding block of the covariance matrix

of fixed points large, but augmenting the prior with an extra learnable variance parameter σ^{l} for each fixed point, such that it doesn't constrain the function value exactly, but $f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}} + \sigma^{\mathrm{l}}\boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim (N)(0,I)$. This then results in Automatic Relevance Determination (ARD) style procedure in which the system is allowed to learn large σ^{l} values for fixed points that are 'unnecessary', therefore not only learning the location of true fixed points, but also pruning the extra ones. Note that there a number of alternative options available for determining the cardinality of a set of unknown parameters (such as life-death processes as in Pnevmatikakis et al. 2013 or standard model selection via a held out validation dataset), but they tend to be both computationally more expensive and more prone to initialisations than the ARD procedure used here.

Mathematically, these extra variance parameters enter the joint prior through the k^{ss} block of the covariance matrix in the joint (equation 4.6) , such that $k^{\mathrm{ss}} = k(\mathbf{S},\mathbf{S}) + \mathrm{diag}([\,\sigma^{1},\ldots,\sigma^{\mathrm{L}}\,])$. In the conditional prior view of equation 4.7, these added variance parameters enter much more clearly, by specifying the function value at a fixed point as a normally distributed random variable $p(\,\tilde{\mathbf{s}}^{\mathrm{l}}\,) = \mathcal{N}(\mathbf{s}^{\mathrm{l}}, \sigma^{\mathrm{l}} I)$ and integrating over these intermediate random variables:

$$p(\,f(x)\,) = \int \cdots \int \mathcal{GP}(\,f(x) \mid \forall_1 f(\mathbf{s}^{\mathrm{l}}) = \tilde{\mathbf{s}}^{\mathrm{l}}\,,\, \nabla f(\mathbf{s}^{\mathrm{l}}) = \mathbf{J}^{\mathrm{l}}\,) \prod_{\mathrm{l}} p(\,\tilde{\mathbf{s}}^{\mathrm{l}}\,)\, d\tilde{\mathbf{s}}^{\mathrm{l}}. \tag{4.8}$$

Having established how our parametric Gaussian Process prior affects the distribution of functions via the fixed points and local linearisation, we now need to understand, how this idea may be applied to infer latent time series $\mathbf{x}^{\mathrm{m}}(t)$ from the data $\mathcal{Y}$, and learn the various model elements, the output mapping g, the output noise $\boldsymbol{\varepsilon}^{\mathrm{y}}$, the transition map function f along with the transition noise $\boldsymbol{\varepsilon}^{\mathrm{x}}$, and of course the parameters of interest, the fixed point locations $\mathbf{S}$, their 'certainties' $\boldsymbol{\sigma}^{\mathrm{s}}$ and the linearisations around each fixed point, $\mathbf{J}$.

4.2.2 Estimating latent Gaussian Process transition map models of time series

The main difficulty of simultaneously inferring latent space trajectories $\mathbf{x}(t)$ and learning their $f(\cdot)$ transition function given a GP prior over it, is that exact Gaussian Process posteriors are represented by their input-output relationships, however these inputs and outputs in the case of time series data are themselves not independent, and indeed their dependence is described by the function we wished to represent by them in the first place.

This issue of the dependence structure of latent time series models with a GP transition function is discussed at length in McHutchon 2014, here I merely wish to state their results. The first, and easiest way of breaking this dependence structure is to just make the assumption that the dependence structure does not exist to begin with, and treat the time series data as merely a regression problem from time t to time $t+1$, essentially ignoring the propagation of uncertainty over time. This is simple and can be effective in certain situations, but in the case of investigating neural system, the estimation and representation of uncertainty is often a key question, and as such, I decided to find an

alternative approach.

Another way, that is widely used in scaling up Gaussian Process, is that of inducing points, essentially a set of extra input-output pairs that exclusively describe the function $f(\cdot)$, without direct dependence on the data. These can either be thought of extra parameters of the complete model leading to the family of methods called sparse GPs, which include FITC and PITC; or can be thought of as variational parameters and are to be integrated over when predicting from an estimated posterior. Here we utilise both approaches in the description of two distinct methods for estimating the transition function. I will first describe in detail the sparse GP version and the algorithm it led to, and later also include the main ideas from the variational approach, which was joint work with Lea Duncker and Julien Boussard and is described in section section 4.2.3

4.2.2.1 Sparse approximation

We introduce inducing point locations $\mathbf{z} \in \mathbb{R}^{\mathrm{D}}$ and uncertain function values at those locations $f^{\mathrm{d}}(\mathbf{z}) = \tilde{u}^{\mathrm{d}}$, where $\tilde{u}^{\mathrm{d}} \sim \mathcal{N}(u_{\mathrm{d}}, \sigma^{\mathrm{u}})$ is a normal random variable with mean u_{d} and uncertainty $sigma^{\mathrm{u}}$. The reason for this uncertainty parameter is that both our measurements are noisy, and we are learning a function in an unknown, transformed latent space, so we do not expect to have the same uncertainty (noise variance) everywhere in the latent space. This representation for a sparse approximation of a heteroscedastic function follows the idea in Snelson and Ghahramani 2012.

As each dimension of the $f(\cdot)$ function is independent, we can also collect the u_{d} scalar values into a vector, and write $f(\mathbf{z}) = \tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}} \sim \mathcal{N}(\mathbf{u}, \sigma^{\mathrm{u}})$. Whether or not to use a different uncertainty level σ^{u} for each output dimension is a choice one can make, I decided to use a single value per inducing point location, therefore it represents how much the $\mathbf{u}$ should constrain the function at that location, rather than some form of non-isotropy in latent space transitions (for which we later introduce a different parameter that is location-independent).

Of course, we need more than one inducing point to sufficiently represent the function, and thus we denote the collections of inducing point locations, mean values and uncertainties as $\mathbf{Z} \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$, $\mathbf{U} \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$ and $\boldsymbol{\sigma}^{\mathrm{u}} \in \mathbb{R}^{\mathrm{K}}$, respectively. This are treated as additional parameters of the model, and are to be estimated given the data, along with parameters described previously. Our "posterior" Gaussian Process will then really mean finding a setting of these inducing point parameters, such that the predictive distribution given the true prior and the inducing point parameters as "noisy training data" results in a GP distribution over functions that optimises the exact marginal log likelihood of the time series data. The joint over the posterior function values $\hat{f}^{\mathrm{d}}(\mathbf{X})$, the prior and the inducing values resemble the form of equation 4.6, noting the added location-wise uncertainty terms $\boldsymbol{\sigma}^{\mathrm{u}}$ and $\boldsymbol{\sigma}^{\mathrm{s}}$ in the covariance, and the added superscripts z representing the kernel's input being $\mathbf{Z}$, e.g. $\nabla_1 k^{\mathrm{sz}} = \nabla_1 k(\mathbf{S}, \mathbf{Z})$.

$$\begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ \hat{f}^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m^{\mathrm{z}} \\ m^{\mathrm{s}} \\ \nabla m^{\mathrm{s}} \\ m^{\mathrm{x}} \end{bmatrix}, \begin{bmatrix} k^{\mathrm{zz}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{u}}) & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} & k^{\mathrm{zx}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.9}$$

To simplify notation for further calculation, from this point onwards we assume (without loss of generality) that the mean function $m(\cdot)$ is uniformly zero, and we introduce the matrices $\mathbf{K}^{\mathrm{post}} \in \mathbb{R}^{(\mathrm{K+L+KL})\times(\mathrm{K+L+KL})}$ and $\mathbf{K}^{\mathrm{pred}} \in \mathbb{R}^{\mathrm{P}\times(\mathrm{K+L+KL})}$, representing the posterior and the predictive parts of the covariance matrix above:

$$\begin{aligned} \mathbf{K}^{\mathrm{post}} &= \begin{bmatrix} k^{\mathrm{zz}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{u}}) & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} \end{bmatrix} \\ \mathbf{K}^{\mathrm{pred}} &= \begin{bmatrix} k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} \end{bmatrix} \end{aligned} \tag{4.10}$$

$$\begin{bmatrix} \begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \\ \hat{f}^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}^{\mathrm{post}} & (\mathbf{K}^{\mathrm{pred}})^{\top} \\ \mathbf{K}^{\mathrm{pred}} & k^{\mathrm{xx}} \end{bmatrix} \right)$$

As these posterior predictions $\hat{f}$ are going to be the ones that enter into our optimisation scheme, it is useful at this point to define the mean and variance $\hat{f}^{\mathrm{d}}(\mathbf{x})$ given a single test input location $\mathbf{x}$, and the current setting of all model parameters. Note that due to our assumption of independence of each $f^{\mathrm{d}}(\cdot)$ function, the covariances of the predictions are 0. This trivially generalises to all d output

dimensions, and also to multiple input locations[5].

$$\mathbb{E}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\mathbf{x})] = \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} \begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \tag{4.11}$$

$$\mathrm{Var}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\mathbf{x})] = k(\mathbf{x},\mathbf{x}) - \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} (\mathbf{K}^{\mathrm{pred}})^{\top}$$

Having established how the sparse approximation can be used to describe a "parametric" GP that disconnects the learned transition function from the dependence structure of the time series data, we have two tasks left. First I will discuss how we infer the latent path given the data and a current setting of all parameters (for the prior, the inducing points, the kernel and the output mapping), Secondly, once we have inferred the latent path's distribution, we can see how the parameters may be changed to maximise the exact marginal log likelihood via gradient ascent.

4.2.2.2 Inference and learning

There are numerous inference algorithms available for latent time series data, but they all generally follow the scheme of passing massages forward and/or backward in time in the latent space, and also collecting messages from the observed outputs at each time point, by (approximately) inverting the output model. Message passing schemes that only propagate information forward - and thus may be used in real-time applications - are called filtering algorithms (examples include Wan and Van Der Merwe 2000; Ko et al. 2007; Poyiadjis, Doucet, and Singh 2011), whereas on already recorded data we may carry out 'smoothing', which also incorporates messages from the future to determine the most likely latent state distribution at a given time (for example Deisenroth et al. 2012; Deisenroth and Mohamed 2012).

Given the large number of inference schemes available, it is fortunate that Andrew McHutchon previously compared a large number of these methods on various simulated datasets McHutchon 2014 that are close to our problem setting, and found that the so-called "Direct method" performed best both in both one-step-ahead predictive performance and recovering true model parameters. Furthermore owing to its relative simplicity it is a very stable method, unlikely to find spurious trajectories. The inference scheme of this Direct method is Assumed Density Filtering (ADF, see Deisenroth, Huber, and Hanebeck 2009 or Ramakrishnan, Ertin, and Moses 2011), which I describe here in more detail.

Assumed Density Filtering proceeds via a single forward filtering for each time series in our dataset, and propagates beliefs through the current estimate of the transition function, incorporates

[5] Note that in the time series setting we generally only have to deal with single input locations at a time, representing our current location in latent space

messages from the observations, and projects any resulting non-Gaussian distributions onto a Gaussian via moment matching of the means and variances. This results in a Gaussian belief over each latent state at the end of the inference, enabling us both to represent these beliefs via only means and variances, as well as making it a convenient input to any learning scheme, regardless of the output mapping[6].

Let $\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}$ represent our belief of the latent state at time $t-1$ as a normal distribution, with

$$\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1} \sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t}-1|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}-1|\mathrm{t}-1}\right) \tag{4.12}$$

The superscripts $t-1|t-1$ indicate that we represent our belief of $\mathbf{x}$ at time $t-1$, with evidence from $\mathbf{y}$ observations incorporated up to time $t-1$ as well. In ADF, we first propagate this belief through the posterior transition function $\hat{f}(\cdot)$, and approximate the resulting distribution again as a Gaussian:

$$\begin{aligned}\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\approx \hat{f}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}) \\ \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\right)\end{aligned} \tag{4.13}$$

These moments may be computed via taking the expectations with respect to $\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}$ of the means and variances of the GP predictions shown in equation 4.11. The posterior part of the GP covariance matrix, $\mathbf{K}^{\mathrm{post}}$, does not depend on the test input $\mathbf{x}$, and thus the expectation with respect to $\mathbf{x}$ only affects $k(\mathbf{x},\mathbf{x})$ and $\mathbf{K}^{\mathrm{pred}}$. Therefore the mean and variances of $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}$ can be computed by

$$\begin{aligned}\mu_{\mathrm{d}}^{\mathrm{t}|\mathrm{t}-1} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}} \mathbb{E}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})] \\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[\mathbf{K}^{\mathrm{pred}}\right]\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\begin{bmatrix}\mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot}\end{bmatrix}\end{aligned} \tag{4.14}$$

$$\begin{aligned}\Sigma_{\mathrm{dd}}^{\mathrm{t}|\mathrm{t}-1} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}} \mathrm{Var}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})] \\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[k(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}, \tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}) - \mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}(\mathbf{K}^{\mathrm{pred}})^{\top}\right] \\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[k(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}, \tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})\right] - \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[(\mathbf{K}^{\mathrm{pred}})^{\top}\mathbf{K}^{\mathrm{pred}}\right]\mathrm{Tr}\left\{\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\right\}\end{aligned} \tag{4.15}$$

However, taking the expectation with respect to an uncertain input does introduce co-variances in the prediction, and thus we need to calculate the off-diagonal terms of $\Sigma^{\mathrm{t}|\mathrm{t}-1}$, unlike in equation 4.11, where those off-diagonal terms were zero. These calculations were originally carried out

[6] As non-conjugate output mappings can still make use of efficient Gaussian Quadrature integrators due to the Gaussianity of the input

in Deisenroth, Huber, and Hanebeck 2009, but were adapted to include the fixed points and local linearisation terms, and written in terms of expectations rather than integrals for clarity.

$$\Sigma^{\mathrm{t|t-1}}_{\mathrm{d}_1\mathrm{d}_2} = \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}}\mathbb{E}_{\hat{\mathrm{f}}}\left[\hat{f}^{\mathrm{d}_1}(\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}})\hat{f}^{\mathrm{d}_2}(\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}})\right] - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1}\mu^{\mathrm{t|t-1}}_{\mathrm{d}_2}$$

The expectation with respect to $\hat{f}$ is then expanded and rearranged as follows:

$$\begin{aligned}
&= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}}\left[\left(\mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\begin{bmatrix}\mathbf{U}_{\mathrm{d}_1\cdot}\\ \mathbf{S}_{\mathrm{d}_1\cdot}\\ \mathbf{J}_{\mathrm{d}_1\cdot\cdot}\end{bmatrix}\right)\left(\mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\begin{bmatrix}\mathbf{U}_{\mathrm{d}_2\cdot}\\ \mathbf{S}_{\mathrm{d}_2\cdot}\\ \mathbf{J}_{\mathrm{d}_2\cdot\cdot}\end{bmatrix}\right)\right] - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1}\mu^{\mathrm{t|t-1}}_{\mathrm{d}_2} \\
&= \begin{bmatrix}\mathbf{U}_{\mathrm{d}_1\cdot}\\ \mathbf{S}_{\mathrm{d}_1\cdot}\\ \mathbf{J}_{\mathrm{d}_1\cdot\cdot}\end{bmatrix}^{\top}\left[\mathbf{K}^{\mathrm{post}}\right]^{-\top}\mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}}\left[(\mathbf{K}^{\mathrm{pred}})^{\top}\mathbf{K}^{\mathrm{pred}}\right]\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\begin{bmatrix}\mathbf{U}_{\mathrm{d}_2\cdot}\\ \mathbf{S}_{\mathrm{d}_2\cdot}\\ \mathbf{J}_{\mathrm{d}_2\cdot\cdot}\end{bmatrix} - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1}\mu^{\mathrm{t|t-1}}_{\mathrm{d}_2}
\end{aligned} \tag{4.16}$$

These calculations completely determine the approximate belief at time t given data up to time $t-1$:

$$\begin{aligned}
p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}}) &\approx p(\,\mathbf{x}(t) \mid \mathbf{y}(1:t-1)\,) \\
\tilde{\mathbf{x}}^{\mathrm{t|t-1}} &\sim \mathcal{N}\left(\,\boldsymbol{\mu}^{\mathrm{t|t-1}}, \boldsymbol{\Sigma}^{\mathrm{t|t-1}}\,\right)
\end{aligned} \tag{4.17}$$

The final inference step is to incorporate the observation at time t, and compute the updated belief

$$\begin{aligned}
p(\tilde{\mathbf{x}}^{\mathrm{t|t}}) &\approx p(\,\mathbf{x}(t) \mid \mathbf{y}(1:t)\,) \\
\tilde{\mathbf{x}}^{\mathrm{t|t}} &\sim \mathcal{N}\left(\,\boldsymbol{\mu}^{\mathrm{t|t}}, \boldsymbol{\Sigma}^{\mathrm{t|t}}\,\right)
\end{aligned} \tag{4.18}$$

In order to do this, we apply Bayes' rule to the likelihood computed given our current belief:

$$p(\,\mathbf{x}(t) \mid \mathbf{y}(1:t)\,) = \frac{p(\mathbf{y}(t) \mid \mathbf{x}(t))\; p(\mathbf{x}(t) \mid \mathbf{y}(1:t-1)\,)}{p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1)\,)}$$

$$\begin{aligned}
&\approx \frac{p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\; p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\displaystyle\int p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\; p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})\; d\tilde{\mathbf{x}}^{\mathrm{t|t-1}}} \\
&= \frac{p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\; p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\mathcal{L}(t)}
\end{aligned} \tag{4.19}$$

where $\mathcal{L}(t) \approx p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1)\,)$ is the approximate marginal likelihood contribution at time t. As the above is a valid normalised probability distribution (that approximates the true posterior

belief at $\mathbf{x}(t)$), we can find its first two moments and thus update our beliefs accordingly:

$$\begin{aligned}\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}}\left[\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}\,\frac{p(\mathbf{y}(t)\mid\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})}{\mathcal{L}(t)}\right]\\ \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}}\left[\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}\,(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})^\top\,\frac{p(\mathbf{y}(t)\mid\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})}{\mathcal{L}(t)}\right] - \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}}(\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}})^\top\end{aligned} \tag{4.20}$$

Given that our belief representation $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}$ is Gaussian, for conjugate likelihoods (such as affine or GP transformations with additive Gaussian noise) these expectation may be available in closed form as in Deisenroth, Huber, and Hanebeck 2009. However, even for non-conjugate likelihoods, the resulting integrals over $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}$ are a Gaussian Quadrature, and thus these moments can be approximated efficiently using numerical methods for arbitrary observation models.

To summarise, in order to infer the latent time series $\mathbf{x}(\cdot)$, we chose to represent our beliefs approximately via normally distributed random variables $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}}$. Given an initial belief $\left\{\boldsymbol{\mu}^{0|0},\,\boldsymbol{\Sigma}^{0|0}\right\}$, we can successively for each new observation time 1.) update the belief by propagating it through the GP transition function (see equations 4.14 to 4.16) and 2.) incorporate the new observation into our belief (see equations 4.18 to 4.20).

These methods and equations are generally applicable for any choice of kernel function or observation model. However it must be stated that the kernel expectations which arise in step 1 are available in closed form only for linear, polynomial and Exponentiated Quadratic / Squared Exponential kernel functions. As the Exponentiated Quadratic kernel function is the most applicable one, I derived analytic expressions for the derivatives as well as all required expectations for it in appendix B, which enables closed form forward propagation of the belief. The expectations in step 2 are only computable in closed form for linear or GP transformations with additive Gaussian noise.

Fortunately the Exponentiated Quadratic kernel over the latent space, and linear output mapping with additive Gaussian noise defines a rather wide class of models. It restricts only the noise model, the smoothness of the transition map $f(\cdot)$ as defined by the lengthscale(s) of the kernel(s), and ensuring that the output mapping $g(\cdot)$ is monotonic[7]. I derive the filtering step for the linear output mapping in section 4.2.2.3 for this widely applicable setup.

Learning During the inference process described above, we already compute the approximate marginal likelihood terms at each observation, $\mathcal{L}(t)$, and therefore we can compute the log marginal likelihood as a function of all parameters θ .

[7] Any non-linearity in a monotonic output function can be represented by transformation of the latent space itself, and the corresponding change of the transition function. In fact, allowing for and learning complex output mappings reduces the interpretability of the inferred latent trajectories and the learned non-linear transition map, and therefore should be avoided.

$$
\begin{aligned}
l(\theta) &= \log p(\, y(1:T) \mid \theta) \\
&= \log \prod_{\mathrm{t}} p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1)) \\
&\approx \sum_{\mathrm{t}} \log \int p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})\; p(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})\; d\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \\
&= \sum_{\mathrm{t}} \log \mathcal{L}(t)
\end{aligned}
\tag{4.21}
$$

As all operations described in this section are differentiable (or the derivatives can numerically be approximated), if one keeps track of the computation graph and the individual derivatives[8], all parameters can be updated via gradient ascent on $l(\theta)$ as the objective. In practice it is useful to only update certain parameters after an inference step, such as keeping the output mapping $g(\cdot)$ (and thus the structure of the latent space) more stable while initially learning the transition map $f(\cdot)$ via its parameters $\theta_{\mathrm{f}} = \{\mathbf{S}, \mathbf{J}, \boldsymbol{\sigma}^{\mathrm{s}}, \mathbf{Z}, \mathbf{U}, \boldsymbol{\sigma}^{\mathrm{u}}\}$ and kernel hyperparameters θ_{k}. Inference and learning steps are iterated until convergence (or until some early stopping threshold based on a metric on held out validation data).

4.2.2.3 Linear Gaussian observation model

As stated above, for a linear Gaussian observation model, we can obtain closed form updates for incorporating a new observation into our belief, meaning we can fully specify equations 4.18 to 4.20. Let

$$
\begin{aligned}
p(\mathbf{y}^{\mathrm{t}} \mid \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}) &= \mathcal{N}_{\mathbf{y}}\left(g(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}), \boldsymbol{\Sigma}^{\mathrm{y}}\right) \\
g(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}) &= \mathbf{C}\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \\
\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\sim \mathcal{N}(\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1})
\end{aligned}
\tag{4.22}
$$

where $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \in \mathbb{R}^{\mathrm{D}}$ is our belief at time t given data up to time $t-1$, $\mathbf{C} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is the linear output mapping, $\boldsymbol{\Sigma}^{\mathrm{y}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is the observation noise variance. We wish to calculate the approximate marginal likelihood given our current belief, as well as update our belief in light of the new observation. We can calculate the conditional mean and variance of y given our belief as

$$
\begin{aligned}
p(\mathbf{y}^{\mathrm{t}} \mid \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}) &= \mathcal{N}_{\mathbf{y}}\left(\boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}, \boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right) \\
\boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}} &= \mathbf{C}\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1} \\
\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}} &= \boldsymbol{\Sigma}^{\mathrm{y}} + \mathbf{C}\boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}
\end{aligned}
\tag{4.23}
$$

Therefore the approximate marginal likelihood contribution at time t is

[8] for which automated methods are now widely available

$$\begin{aligned}\log \mathcal{L}(t) &= p(\mathbf{y}^{\mathrm{t}} \mid \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}) \\ &= -\frac{1}{2}\left(N\log(2\pi) + \log\|\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\| + (\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}})^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}(\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}})\right)\end{aligned} \tag{4.24}$$

And from the conditional $x|y,b$ we can update our belief via

$$\begin{aligned}\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}} &= \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1} + \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}(\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}) \\ \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}} &= \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1} - \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}\mathbf{C}\boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\end{aligned} \tag{4.25}$$

When this linear Gaussian observation model is used alongside an Exponentiated Quadratic kernel function, the inference is completely analytically computable with closed form expressions, and we can use the approximate marginal likelihood $l(\theta) = \sum_{\mathrm{t}} \log \mathcal{L}(t)$ as an objective function for parameter optimisation via gradient ascent.

In accordance with previous terminology, I will refer to this algorithm as *Fixed-point Gaussian Process Assumed Density Filtering* (FP GP-ADF) with linear Gaussian observations in further sections, and show applications in sections 4.3.2 and 4.3.5.

4.2.3 Continuous time modelling of transition flows via Variational Sparse Gaussian Process inference and learning

This section presents joint work with Lea Duncker and Julian Boussard, published in Duncker et al. 2019, resulting in the *Fixed-point Gaussian Process latent Stochastic Differential Equation* (FP GP-SDE) algorithm. I summarise here the main ideas, please refer to the paper for further details[9].

An alternative, more flexible approach to modeling dynamical systems is to identify the transition flows in the latent space, rather than relying upon discrete time transition maps. As mentioned in section 4.2.2, we may use stochastic differential equations (SDEs) to model the evolution of the latent state $\mathbf{x}$

$$d\mathbf{x} = f'(\mathbf{x})\,dt + \sqrt{\boldsymbol{\Sigma}}\,d\mathbf{w}, \tag{4.26}$$

where the function $f'(\cdot)$ now represents a transition flow rather than the transition map itself, and $\boldsymbol{\Sigma}$ is the incremental noise covariance shaping the Wiener noise process $\mathbf{w}(t)$.

The latent state is observed indirectly through noisy measurements $\mathbf{y} \in \mathbb{R}^{\mathrm{N}}$ at unevenly spaced time points t_{i}.

[9] Mind the notation and definitions in the paper is slightly different from present thesis chapter, but both are self-consistent. Furthermore, the notation within this subsection uses subscripts for naming objects, and not to be understood as indicial notation.

$$\mathbb{E}_{\mathrm{y}|\mathrm{x}}[\mathbf{y}(t_\mathrm{i})] = g(\mathbf{C}\mathbf{x}(t_\mathrm{i}) + \mathbf{d}) \tag{4.27}$$

We consider here a general, but particular form of measurement model, in which the measurements are distributed with a known parametric form and generalized linear dependence; that is the expected value is $g(\mathbf{C}\mathbf{x}+\mathbf{d})$ with a given inverse-link function g and parameters $\mathbf{C} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $\mathbf{d} \in \mathbb{R}^{\mathrm{N}}$. We seek to infer latent paths $\mathbf{x}(t)$ along with the dynamical parameters and an interpretable representation of the dynamical transition flow $f'(\cdot)$.

In the following I describe the two main differences compared to the previous FP GP-ADF algorithm. Firstly, we chose to use a different treatment of how inducing points were incorporated in the model and what objective we use to learn model parameters. In the above FP GP-ADF algorithm I treat the inducing points (represented by locations, mean values and uncertainties) as extra parameters of the model, and found maximum likelihood point estimates for all parameters, based on an approximate marginal likelihood as the objective function. Here, as shown in section 4.2.3.1, we used an alternative option for learning model parameters based on treating both the distribution over the latents $\mathbf{x}$ as well as the inducing points $\mathbf{u}$ as variational parameters following Titsias 2009b. We then maximise the variational free energy (a well-defined lower bound to the marginal likelihood rather than an approximation to it as previously) by integrating over the variational parameters[10]. Secondly, in order to make use of the continuous time representation based on SDEs, we need to employ a different inference algorithm (derived in section 4.2.3.2), one that uses the dynamical flow representation to propagate latent beliefs and can thus incorporate observations at arbitrary time points.

4.2.3.1 Variational sparse approximation

Although mathematically the introduction of fixed and inducing points to a GP prior over the transition flow function $f'(\cdot)$ looks very similar to what we derived in equation 4.9, there are some significant differences in what the parameters mean and how we proceed from there. I first state the updated representation, then discuss the differences.

[10] Note that the parametric and variational treatments of model learning are easily interchangeable and thus we could similarly derive a variational version of FP GP-ADF algorithm or a parametric version of the algorithm described here

$$\begin{bmatrix} \tilde{\mathbf{U}}_{\mathrm{d}\cdot} \\ \mathbf{0} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ f'_{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k^{\mathrm{zz}} & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} & k^{\mathrm{zx}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.28}$$

Firstly, as we are modelling the transition flow rather than the map, we expect the function values at fixed points to be zero, rather than identity, therefore $f'(\mathbf{s}^{\mathrm{l}}) = \mathbf{0}$ for each fixed point. Similarly the Jacobians $\mathbf{J}^{\mathrm{l}}$ now represent derivatives of the flows rather than the map. Secondly, the inducing point values used in the prior are themselves random variables, and their prior distribution is defined by the kernel governing the GP, therefore for each latent dimension d we have $\tilde{\mathbf{U}}_{\mathrm{d}\cdot} \sim \mathcal{N}(\mathbf{0}, k^{\mathrm{zz}})$ a priori, and thus there is no extra diagonal term added to k^{zz} in the joint covariance matrix. Given these changes, we can again simplify the notation for the joint, then calculate the conditional prior on each $f'_{\mathrm{d}}(\cdot)$ function, similarly to equations 4.10 and 4.11, as follows:

$$\begin{aligned} \mathbf{K}^{\mathrm{prior}} &= \begin{bmatrix} k^{\mathrm{zz}} & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} \end{bmatrix} \\ \mathbf{K}^{\mathrm{pred}} &= \begin{bmatrix} k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_{\mathrm{d}\cdot} \\ \mathbf{0} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \\ f'_{\mathrm{d}}(\mathbf{X}) \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}^{\mathrm{prior}} & (\mathbf{K}^{\mathrm{pred}})^{\top} \\ \mathbf{K}^{\mathrm{pred}} & k^{\mathrm{xx}} \end{bmatrix} \right) \end{aligned} \tag{4.29}$$

The form of the predictive mean and variance remains the same, noting that the inducing point

values are random variables, and thus this represents a true conditional prior GP over $f'(\cdot)$ rather than a parametrised posterior.

$$f'_{\mathrm{d}} \,|\, \tilde{u}_{\mathrm{d}} \sim \mathcal{GP}\left(\mathbb{E}_{\mathrm{f}' \,|\, \tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})], \mathrm{Var}_{\mathrm{f}' \,|\, \tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})]\right)$$

$$\mathbb{E}_{\mathrm{f}' \,|\, \tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})] = \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{prior}}\right]^{-1} \begin{bmatrix} \tilde{\mathbf{U}}_{\mathrm{d}\cdot} \\ \mathbf{0} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \tag{4.30}$$

$$\mathrm{Var}_{\mathrm{f}' \,|\, \tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})] = k(\mathbf{x},\mathbf{x}) - \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{prior}}\right]^{-1} (\mathbf{K}^{\mathrm{pred}})^{\top}$$

We derived an alternative sparse GP prior over the transition flow $f'(\cdot)$, conditioned on the inducing point values $\tilde{\mathbf{U}}$ and parametrised by the inducing point locations $\mathbf{Z}$, as well as fixed point locations $\mathbf{S}$, the local linearisations around them $\mathbf{J}$, and the kernel hyperparameters θ_{k}. We now need to understand how to represent the latent process $\mathbf{x}(t)$, incorporate observations $\mathbf{y}(t_{\mathrm{i}})$ recorded at arbitrary times t_{i}, propagate beliefs about $\mathbf{x}$ through the transition flow function $f'(\cdot)$ and thus infer the posterior $p(\mathbf{x}(t) \,|\, \mathcal{Y})$, and finally learn the parameters of both the transition flow $f'(\cdot)$ and the output mapping $g(\mathbf{C}\mathbf{x}+\mathbf{d})$.

4.2.3.2 Inference and learning

The problem of performing approximate inference in continuous-time SDE models has been considered previously, with the two main approaches being Expectation Propagation Cseke et al. 2016 and variational inference Archambeau et al. 2007; Archambeau et al. 2008. We first review the latter approach in this section, then derive our Variational Bayes algorithm that extends this work.

(r)$^{\top}$

Review of Archambeau approximation

Archambeau et al. 2007; Archambeau et al. 2008 consider the latent SDE model in equations 4.26 and 4.27 under linear Gaussian observations. The authors derive an approximate inference algorithm based on a variational Gaussian approximation to the posterior process on $\mathbf{x}(t)$ under the constraint that the approximate process has Markov structure, as is the case for the true posterior process. The most general way to construct such an approximation is via a linear time-varying SDE of the form

$$d\mathbf{x} = (-\mathbf{A}(t)\mathbf{x}(t) + \mathbf{b}(t))\, dt + \sqrt{\mathbf{\Sigma}}\, d\mathbf{w} \tag{4.31}$$

An alternative way to express this approximation is via a GP of the form

$$q_{\mathrm{x}}(\mathbf{x}(t)) = \mathcal{GP}\left(\mathbf{m}_{\mathrm{x}}(t), \mathbf{R}_{\mathrm{x}}(t)\right) \tag{4.32}$$

whose means $\mathbf{m}_{\mathrm{x}}(t)$ and covariances $\mathbf{R}_{\mathrm{x}}(t)$ evolve in time according to the ordinary differential equations (ODEs):

$$\begin{aligned} \frac{d\mathbf{m}_{\mathrm{x}}}{dt} &= -\mathbf{A}(t)\mathbf{m}_{\mathrm{x}} + \mathbf{b}(t) \\ \frac{d\mathbf{R}_{\mathrm{x}}}{dt} &= -\mathbf{A}(t)\mathbf{R}_{\mathrm{x}} - \mathbf{R}_{\mathrm{x}}\mathbf{A}(t)(r) + \mathbf{\Sigma} \end{aligned} \tag{4.33}$$

Archambeau et al. 2007; Archambeau et al. 2008 derive a lower bound to the marginal log-likelihood – often called the variational free energy or evidence lower bound – whose maximisation with respect to q_{x} is equivalent to minimising the Kullback-Leibler (KL) divergence between the approximate and true posterior process. The free energy has the form

$$\mathcal{F} = \sum_{\mathrm{i}} \left\langle \log p(\mathbf{y}_{\mathrm{i}}|\mathbf{x}_{\mathrm{i}}) \right\rangle_{q_{\mathrm{x}}} - \mathsf{KL}[q_{\mathrm{x}}(\mathbf{x}) \| p(\mathbf{x})] \tag{4.34}$$

The first term is the expected log-likelihood under the approximation and only depends on the marginal distributions $q_{\mathrm{x}}(\mathbf{x}(t))$. The second term is the KL-divergence between the continuous-time approximate posterior process and the prior process. Archambeau et al. 2007 show that this term can be written as

$$\mathsf{KL}[q_{\mathrm{x}}(\mathbf{x}) \| p(\mathbf{x})] = \int_{\mathcal{T}} dt \left\langle (\mathbf{f} - \mathbf{f}_{\mathrm{q}})(r) \mathbf{\Sigma}^{-1} (\mathbf{f} - \mathbf{f}_{\mathrm{q}}) \right\rangle_{\mathrm{q}} \tag{4.35}$$

where $\mathbf{f}$ is the output of the true stationary posterior transition flow $\hat{f}'(\cdot)$ evaluated at $\mathbf{x}(t)$, $\mathbf{f} = \hat{f}'(\mathbf{x}(t))$, and $\mathbf{f}_{\mathrm{q}} = \mathbf{f}_{\mathrm{q}}(t, \mathbf{x}(t)) = -\mathbf{A}(t)\mathbf{x}(t) + \mathbf{b}(t)$ is our time-varying approximation to the current flow, with the inputs omitted for both.Note that the noise covariance $\mathbf{\Sigma}$ is deliberately chosen to be equal for the SDEs in q_{x} and p, as this term would diverge otherwise.

To maximise $\mathcal{F}$ with respect to $\mathbf{m}_{\mathrm{x}}(t)$ and $\mathbf{R}_{\mathrm{x}}(t)$, subject to the constraint that the approximate posterior process has Markov structure according to equation (4.31), one can find the stationary points of the Lagrangian

$$\mathcal{L} = \mathcal{F} - \mathcal{C}_1 - \mathcal{C}_2 \tag{4.36}$$

with

$$\begin{aligned} \mathcal{C}_1 &= \int_{\mathcal{T}} dt \; \mathsf{Tr}\left[\Psi \left(\frac{d\mathbf{R}_{\mathrm{x}}}{dt} + \mathbf{A}\mathbf{R}_{\mathrm{x}} + \mathbf{R}_{\mathrm{x}}\mathbf{A}(r) - \mathbf{\Sigma} \right) \right] \\ \mathcal{C}_2 &= \int_{\mathcal{T}} dt \; \boldsymbol{\lambda}(r) \left(\frac{d\mathbf{m}_{\mathrm{x}}}{dt} + \mathbf{A}\mathbf{m}_{\mathrm{x}} - \mathbf{b} \right) \end{aligned} \tag{4.37}$$

Where Ψ and $\boldsymbol{\lambda}$ are Lagrange multipliers. Archambeau et al. 2007; Archambeau et al. 2008 derive a smoothing algorithm that involves iterating fixed point updates of this Lagrangian. These are either closed form, or require solving ODEs forward and backward in time, thus achieving linear time

complexity. In section **??**, we will modify this original algorithm in order to improve its numerical stability, and show how to incorporate it in an efficient Variational Bayes algorithm.

Novel Variational Bayes extension

We can derive an efficient Variational Bayes (VB) algorithm Attias 2000 for variational inference and learning in the model in equations 4.26 and 4.27 by maximising a variational free energy. We assume that our full variational distribution factorises as

$$q(\mathbf{x},\mathbf{f},\tilde{\mathbf{u}}) = q_{\mathrm{x}}(\mathbf{x})q_{\mathrm{f,u}}(\mathbf{f},\tilde{\mathbf{u}}) \tag{4.38}$$

Following Titsias 2009b, we choose $q_{\mathrm{f,u}}(\mathbf{f},\tilde{\mathbf{u}}) = \prod_{\mathrm{k}=1}^{\mathrm{K}} p(f_{\mathrm{k}}|\tilde{\mathbf{u}}_{\mathrm{k}},\theta)q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}})$. The variational approximation of the posterior over the inducing points are chosen to be of the form $q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}}) = \mathcal{N}\left(\tilde{\mathbf{u}}_{\mathrm{k}}|\mathbf{m}_{\mathrm{u}}^{\mathrm{k}},\mathbf{R}_{\mathrm{u}}^{\mathrm{k}}\right)$. The marginal variational distribution $q_{\mathrm{f}}(\mathbf{f}) = \prod_{\mathrm{k}} \int d\tilde{\mathbf{u}}_{\mathrm{k}} p(f_{\mathrm{k}}|\tilde{\mathbf{u}}_{\mathrm{k}},\theta)q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}})$ is also a Gaussian Process. The resulting expression for the variational free energy is of the form:

$$\mathcal{F}^* = \langle \mathcal{F} \rangle_{\mathrm{q_f}} - \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathsf{KL}[q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}}) \| p(\tilde{\mathbf{u}}_{\mathrm{k}}|\theta)] \tag{4.39}$$

The VB algorithm then iterates over an inference step, where the distribution q_{x} over the latent path is updated, a learning step where $q_{\mathrm{f,u}}$ and the parameters in the affine output mapping are updated, and a hyperparameter learning step where the kernel hyperparameters, and fixed point locations are updated.

Inference Our inference approach follows directly from the work in Archambeau et al. 2007; Archambeau et al. 2008, though we consider a wider class of observation models and include a non-parametric Bayesian treatment of the dynamics $\mathbf{f}$ under the conditioned sparse GP prior introduced in section 4.2.3.1.

After using integration by parts on the Lagrangian in (4.36) (exchanging $\mathcal{F}$ for $\mathcal{F}^*$), we take variational derivatives with respect to $\mathbf{m}_{\mathrm{x}}(t)$ and $\mathbf{R}_{\mathrm{x}}(t)$. Since our model has a rotational non-identifiability with respect to the latents $\mathbf{x}$, we fix $\mathbf{\Sigma} = I$ without loss of generality. We arrive at the following set of fixed point equations:

$$\frac{d\mathbf{\Psi}}{dt} = \mathbf{A}(t)(r)\mathbf{\Psi}(t) - \mathbf{\Psi}(t)\mathbf{A}(t) - \frac{\partial \mathcal{F}^*}{\partial \mathbf{R}_{\mathrm{x}}} \odot \mathbb{P} \tag{4.40}$$

$$\frac{d\boldsymbol{\lambda}}{dt} = \mathbf{A}^{\mathrm{T}}(t)\boldsymbol{\lambda}(t) - \frac{\partial \mathcal{F}^*}{\partial \mathbf{m}_{\mathrm{x}}} \tag{4.41}$$

$$\mathbf{A}(t) = \left\langle \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right\rangle_{\mathrm{q_x q_f}} + 2\mathbf{\Psi}(t) \tag{4.42}$$

$$\mathbf{b}(t) = \langle \mathbf{f}(\mathbf{x}) \rangle_{\mathrm{q_x q_f}} + \mathbf{A}(t)\mathbf{m}_{\mathrm{x}}(t) - \boldsymbol{\lambda}(t) \tag{4.43}$$

with $\mathbb{P}_{ij} = \frac{1}{2}$ for $i \neq j$ and 1 otherwise and $\odot$ denotes the Hadamard product. In contrast to previous work, we explicitly take the symmetric variations of $\mathbf{R}_x(t)$ into account, which leads to slightly modified equations in (4.40) compared to the work in Archambeau et al. 2007; Archambeau et al. 2008, and seems to improve the numerical stability of the algorithm. As a result, we can work with the fixed point updates (4.42) and (4.43) directly, without introducing a learning rate parameter that blends the updates with the previous value of the variational parameters $\mathbf{A}$ and $\mathbf{b}$, as was done in Archambeau et al. 2007; Archambeau et al. 2008.

The inference algorithm involves solving the set of coupled ODEs in (4.33) and (4.40)-(4.43) using the conditions $\mathbf{m}_x(0) = \mathbf{m}_{x,0}$, $\mathbf{R}_x(0) = \mathbf{R}_{x,0}$ and $\boldsymbol{\lambda}(T) = 0$, $\boldsymbol{\Psi}(T) = 0$. In principle, it is possible to use any ODE solver to do this. In this work, we choose to solve (4.33) using the forward Euler method with fixed step size Δt to obtain $\mathbf{m}_x$ and $\mathbf{R}_x$ evaluated on an evenly spaced grid. Similarly, we then solve (4.40) and (4.41) backwards in time to obtain evaluations of $\boldsymbol{\lambda}$ and $\boldsymbol{\Psi}$. The solutions from the ODEs can then be used with equations (4.42) and (4.43) to obtain evaluations of $\mathbf{A}$ and $\mathbf{b}$ on the same time-grid used for solving the ODEs.

Evaluating the expectations of the terms involving $\mathbf{f}$ with respect to q_x and q_f only involves computing Gaussian expectations of covariance functions and their derivatives. These can be computed analytically for choices such as an exponentiated quadratic covariance function. We update the initial state values $\mathbf{m}_{x,0}$ and $\mathbf{R}_{x,0}$ using the same procedure as that described in Archambeau et al. 2008. Given the function evaluations on the inference time-grid, we use linear interpolation to obtain function evaluations of $\mathbf{m}_x$ and $\mathbf{R}_x$ at arbitrary time points. Further details on the inference algorithm are given in the supplementary material.

Learning Dynamics The only terms in (4.39) that depend on parameters in $\mathbf{f}$ are the expected KL-divergence between the prior and approximate posterior processes and the KL-divergence relating to the inducing points for $\mathbf{f}$, which are jointly quadratic in the inducing points and Jacobians. Thus, given $\mathbf{m}_x(t)$, $\mathbf{R}_x(t)$, $\mathbf{A}(t)$ and $\mathbf{b}(t)$, we can find closed form updates for the Jacobians and variational parameters relating to $\mathbf{f}$. For $\mathbf{R}_u^k$, the covariance matrix of the variational approximation for the

posterior over $\tilde{\mathbf{u}}_{\mathrm{k}}$, the update is of the form

$$\mathbf{K}^{\mathrm{eff}} = \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{prior}}\right]^{-1} \tag{4.44}$$

$$\mathbf{R}_{\mathrm{u}}^{\mathrm{k}} = \left({\boldsymbol{\Omega}_{\mathrm{u}}}^{-1} + \int_{\mathcal{T}} dt \left[\left\langle \left(\mathbf{K}^{\mathrm{eff}}\right)(r) \left(\mathbf{K}^{\mathrm{eff}}\right) \right\rangle_{\mathrm{q_x}} \right]_{[\mathrm{u,u}]} \right)^{-1} \tag{4.45}$$

$$\boldsymbol{\Omega}_{\mathrm{u}} = k^{\mathrm{zz}} - \begin{bmatrix} k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \end{bmatrix} \begin{bmatrix} k^{\mathrm{ss}} \end{bmatrix}^{-1} \begin{bmatrix} k^{\mathrm{zs}} \\ \nabla_2 k^{\mathrm{zs}} \end{bmatrix} \tag{4.46}$$

where the kernel matrices are defined in equation 4.29 and the operation $[X]_{[\mathrm{u,u}]}$ selects the first $K \times K$ block of X, and K is the number of inducing points. Note that $\mathbf{K}^{\mathrm{pred}}$ and thus also $\mathbf{K}^{\mathrm{eff}}$ are functions of the latent $\mathbf{x}(t)$. The mean values of the inducing point variational approximation and Jacobians around the fixed-point locations can be updated jointly as

$$\begin{bmatrix} \mathbf{m}_{\mathrm{u}}^{1} & \dots & \mathbf{m}_{\mathrm{u}}^{\mathrm{K}} \\ \mathbf{J}_1 & \dots & \mathbf{J}_{\mathrm{K}} \end{bmatrix} = \mathbf{B}_1^{-1} \left(\mathbf{B}_2 - \mathbf{B}_3\right)$$

with

$$\begin{aligned}
\mathbf{B}_1 &= \left(\tilde{\boldsymbol{\Omega}} + \int_{\mathcal{T}} dt \left[\left\langle \left(\mathbf{K}^{\mathrm{eff}}\right)(r) \left(\mathbf{K}^{\mathrm{eff}}\right) \right\rangle_{\mathrm{q_x}} \right]_{[\mathrm{uj,uj}]} \right) \\
\mathbf{B}_2 &= \int_{\mathcal{T}} dt \left[\left\langle \mathbf{K}^{\mathrm{eff}} \right\rangle_{\mathrm{q_x}} \right]_{[:,\mathrm{uj}]} (r) \langle \mathbf{f}_{\mathrm{q}} \rangle_{\mathrm{q_x}}(r) \\
\mathbf{B}_3 &= \int_{\mathcal{T}} dt \left[\left\langle \nabla_{\mathrm{x}} \mathbf{K}^{\mathrm{eff}} \right\rangle_{\mathrm{q_x}} \right]_{[:,\mathrm{uj}]} (r) \mathbf{R}_{\mathrm{x}} \mathbf{A}(r) \\
\tilde{\boldsymbol{\Omega}} &= \begin{bmatrix} {\boldsymbol{\Omega}_{\mathrm{u}}}^{-1} & -{\boldsymbol{\Omega}_{\mathrm{u}}}^{-1} \mathbf{G} \\ -\mathbf{G}(r) {\boldsymbol{\Omega}_{\mathrm{u}}}^{-1} & \mathbf{G}(r) {\boldsymbol{\Omega}_{\mathrm{u}}}^{-1} \mathbf{G} \end{bmatrix} \\
\mathbf{G} &= \left[\begin{bmatrix} k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \end{bmatrix} \begin{bmatrix} k^{\mathrm{ss}} \end{bmatrix}^{-1} \right]_{[\mathrm{j,j}]}
\end{aligned} \tag{4.47}$$

where $[X]_{[\mathrm{uj,uj}]}$ selects the first $K \times K$ and last $LD \times LD$ block of X, $[X]_{[:,\mathrm{uj}]}$ selects the first K and last LD columns of X, and $[X]_{[\mathrm{j,j}]}$ selects the last $LD \times LD$ block of X, where L is the number of fixed points, D is the latent dimensionality, and thus the LD blocks selected by the 'j' index refer to the contribution of the collection of Jacobians around the fixed points. The reason that the fixed point contributions are not selected are due to the transition flow function is conditioned to evaluate

to zero at the fixed points, and thus their contribution falls out of these update equations.

The one-dimensional integrals can be computed efficiently using, for instance, Gauss-Legendre quadrature. Detailed derivations are given in the appendix of Duncker et al. 2019, where we also provide closed form updates for the sparse variational GP approach for modelling $\mathbf{f}$ without further conditioning on fixed points and Jacobians.

Learning Output Mapping The only term that depends on the parameters $\mathbf{C}$ and $\mathbf{d}$ in (4.39) is the expected log-likelihood. Whether or not our algorithm admits for closed form solutions depends on the choice of observation model. In the case of a Gaussian likelihood, we can find the optimal updates as

$$\mathbf{C}^* = \left(\sum_{\mathrm{t}} (\mathbf{y}_{\mathrm{t}} - \mathbf{d}) \mathbf{m}_{\mathrm{x,t}}(r)\right) \left(\sum_{\mathrm{t}} (\mathbf{R}_{\mathrm{x,t}} + \mathbf{m}_{\mathrm{x,t}} \mathbf{m}_{\mathrm{x,t}}(r))\right)^{-1} \tag{4.48}$$

$$\mathbf{d}^* = \frac{1}{T} \sum_{\mathrm{t}} \left(\mathbf{y}_{\mathrm{t}} - \mathbf{C}^* \mathbf{m}_{\mathrm{x,t}}\right) \tag{4.49}$$

Where the subscript t denotes a function evaluation at t. For other choices of observation model a closed form solution may not be available, but parameter updates can again be found by maximising the free energy using standard optimisation approaches.

Learning Hyperparameters The kernel function hyperparameters θ_{k} and fixed point locations $\mathbf{S}$ are learnt by direct optimisation of the variational free energy. The inducing point locations $\mathbf{Z}$ can also be included here, though we chose to hold them fixed on a chosen grid for all examples shown.

4.3 Applications

Having defined two novel algorithms, FP GP-ADF and FP GP-SDE that incorporate fixed points and their local linearisation as directly learnable parameters, I wish to demonstrate that these ideas and the resulting algorithms are indeed applicable for a variety of problem settings, including neural data analysis. Although truly understanding an unknown real-world dynamical system is extremely difficult, requiring years of careful analysis and probably repeated experiments and data collection to test the hypotheses arising from initial analyses, I do hope these algorithms may serve as one the tools researchers can apply to glean further insights into systems and generate truly testable hypothesis based on the crude dynamical portraits we can recover computationally.

Due to the difficulties in evaluating whether a learned portrait accurately describes an unknown system, I chose to validate the algorithms via a range of carefully simulated experiments rather than applying them to experimental data and attempting to draw superficial conclusions based on the resulting fits.

This section describes the simulated systems and shows that the algorithms are indeed capable

of recovering the true dynamical portraits in a variety of settings, including non-uniform observation times (for FP GP-SDE) and non-Gaussian noise models. The latent dynamics were restricted to 1 and 2 dimensional systems in these examples, as it is difficult to visualise and interpret the dynamical portrait as well as the type and stability of stochastic fixed points in higher dimensions. However, the implemented algorithms are readily applicable to $D > 2$ latent dimensions too. There is no such restriction put on the dimensionality of the observed space, N, as in both electrical and imaging type neural recordings we often get $N \gg 50$ simultaneously recorded neural timeseries, and thus in the experiments below we demonstrate that the algorithms can indeed make use of such data.

I first show the basic capabilities of both algorithms via two similar one-dimensional systems, one defined by its transition flow function and sampled irregularly in time (section 4.3.1), and the other sampled on a time grid with a dynamical map function gowerning its evolution (section 4.3.2). Afterwards I describe a life-like simulation of complex chemical reactor we three stable and two unstable fixed points observed in high dimensions via a 'spectroscopic' measurement process to showcase that our system is capable of learning very complex dynamical portraits (section 4.3.3).

Next I discuss neural applications, first via simulating a generically parametrised known two-dimensional SDE observed through a multivariate point process emulating electrical population recordings (section 4.3.4). Finally I simulate from a high-dimensional spiking neural model observed through a calcium microscopy like process on a regular time grid representing video frames. The parameters of the spiking system were optimised by others to fit real neural recordings which implement multi-stage 2-dimensional 'memory and decision making' dynamics, regulated by external input and experiencing bifurcation (section 4.3.5).

In all examples the Exponentiated Quadratic kernel function is used due to the analytical solutions it provides. For both methods the inducing point locations $\mathbf{Z}$ are chosen on an equidistant grid, and these locations are not optimised to reduce computational complexity and improve stability. For the GP-ADF method we initialise parameters via GP regression from time t to $t+1$. For the GP-SDE method, the inducing point means and the Jacobians are initialised as zero and the ODEs in equations 4.40 to 4.43 are solved with the forward Euler method with $\Delta t = 1msec$.

4.3.1 Double well flow

We first demonstrate the FP GP-SDE method on a the classic one-dimensional double-well example, where the latent SDE evolves with drift $f(x) = 4x(1-x^2)$. We simulate data on 20 trials with multivariate Gaussian outputs of dimensionality $N = 15$ with unknown variances 0.25, and observe the output process at 20 randomly sampled time-points per trial. We chose 8 evenly spaced inducing points in $(-3,3)$ for f. While the true dynamics have three fixed points, we condition the prior on f on four fixed points, initialise their standard deviations as 0.1 and use the ARD method to automatically adjust those to select the correct number of fixed points. The results are summarised

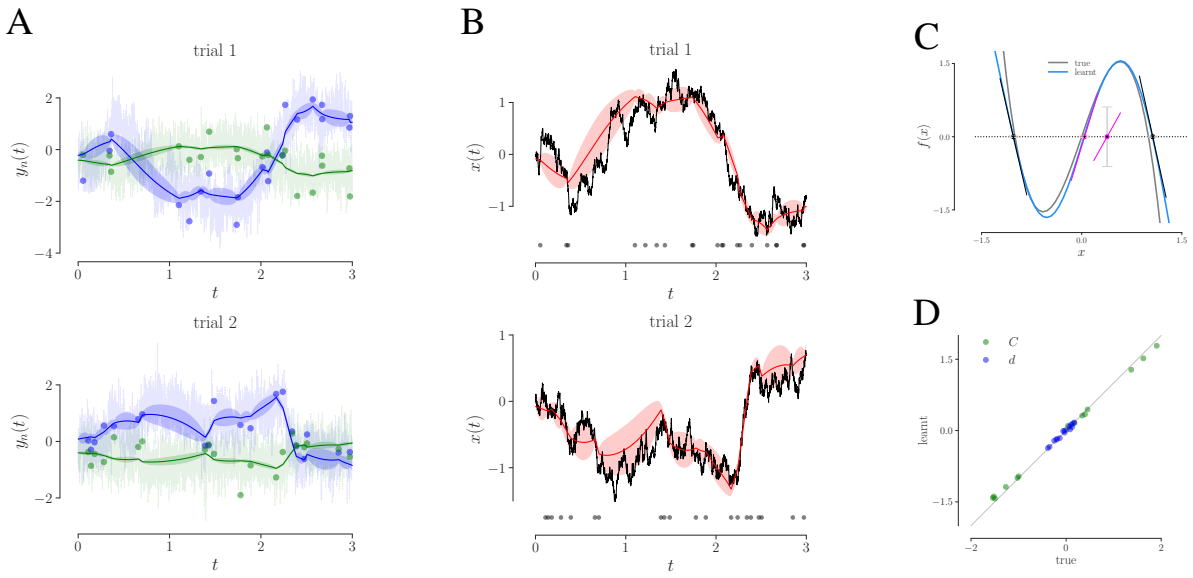


Figure 4.1: Double-well dynamics. A: Two example dimensions of the output process on two different trials. The dots represent the observed data-points of the noisy output processes plotted in faint lines. The solid blue/green traces are the inferred posterior means with ± 1 posterior standard deviation tubes around them. B: True and inferred latent SDE trajectory for the same example trials as in A. The red traces represent the posterior means with ± 1 posterior standard deviation tubes around them, black traces show the true latent SDE path. The black dots indicate the times when observations of $\mathbf{y}$ were made. C: True and learnt dynamics together with the learnt fixed-point locations and tangent lines. Stable fixed points are shown in black, unstable ones in magenta. The uncertainty about the fixed point observation is illustrated using grey error bars representing ± 1 standard deviation. Only the additional fourth fixed point is associated with high uncertainty. D: True vs. learnt model parameters $\mathbf{C}$ and $\mathbf{d}$.

in figure 4.1, demonstrating that the algorithm can successfully perform inference and interpretable learning of the SDE path and dynamics, respectively, and does not move away from the good initial location for the model parameters $\mathbf{C}$ and $\mathbf{d}$.

4.3.2 Double well map and pitchfork bifurcation

Next I demonstrate that not only can we employ these ideas to find fixed points in a single experimental dataset, but rather that FP GP-ADF applied to a set of datasets, which differ in the setting of a single parameter (often the case in experimental data), can discover qualitative changes in the dynamics as the parameter varies. This qualitative change is called bifurcation, and is identified by FP GP-ADF by correctly detecting that the number and stability of fixed points differ between datasets. Here I demonstrate this ability through a classic one-parameter system that exhibits pitchfork bifurcation, and strongly resembles the one in section 4.3.1. Instead of investigating the performance at a single setting of the parameter, I generated a set of datasets with varying parameter values.

The dynamics are defined by the stochastic map and the output is simply a noisy identity map-

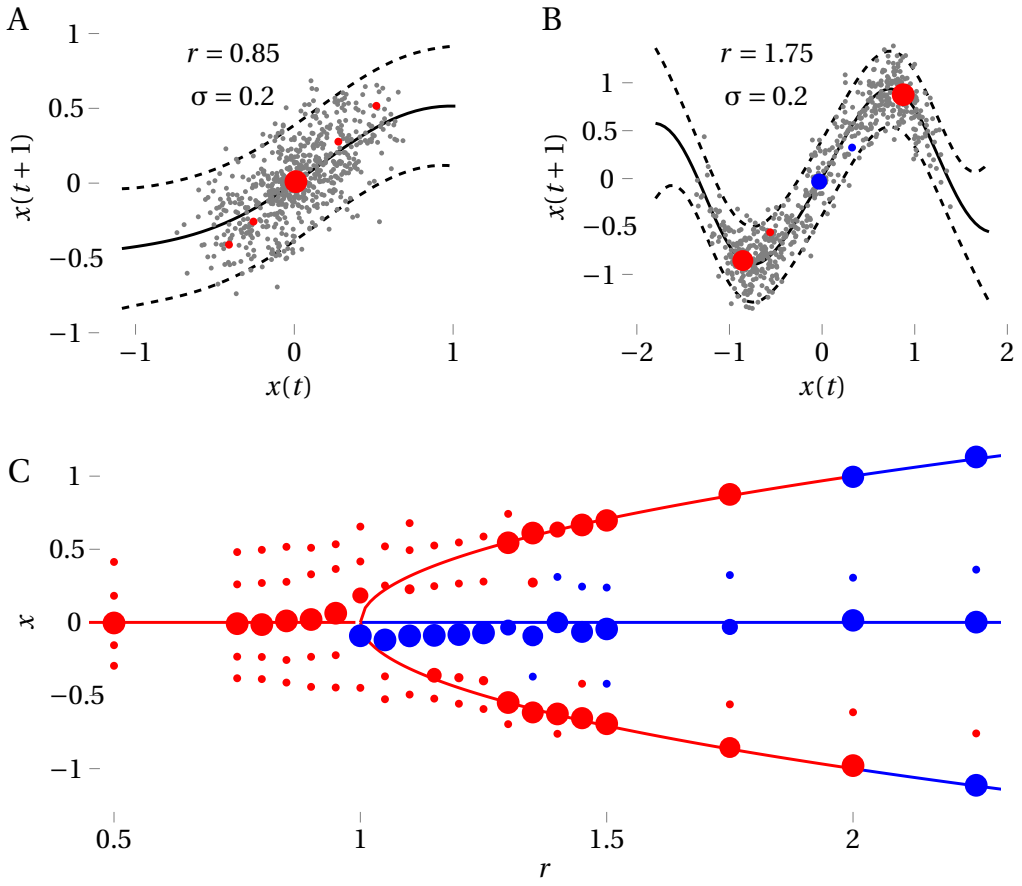


Figure 4.2: Stochastic bifurcation experiment. *A.* The posterior fit of the Fixed Point Sparse Gaussian Process on simulated data from equation 4.50. Small grey points represent the training data. The solid line is the posterior mean, with the dashed lines indicating the posterior standard deviation at $2\sigma_{\text{post}}$. The red and blue circles are the learned fixed point parameter locations. Red colour indicates stable fixed points with the learned Jacobian $|J| < 1$, and blue ones are unstable with $|J| > 1$. The circle size indicates the strength of belief in the fixed point, the larger the size, the smaller the learned σ^{s} uncertainty is. *B.* Same as *A*, with $r = 1.75$, after the bifurcation. *C.* Bifurcation plot of the stochastic system. The solid lines represent the analytic solution for fixed point locations in the noiseless system, when $\sigma_{\text{x}} = 0$. The noiseless fixed points are at $x = 0$ and when $r > 1$ at $x = \pm\sqrt{r-1}$, with the color indicating their stability. The circles represent the learned fixed point locations [y axis position], for a dataset simulated at a particular value of r [x axis position].

ping, making inference simple.

$$x(t+1) = rx(t) - x^3(t) + \varepsilon_{\text{t}}^{\text{x}} \tag{4.50}$$

$$y(t) = x(t) + \varepsilon_{\text{t}}^{\text{y}}, \tag{4.51}$$

where $\varepsilon_{\text{t}}^{\text{x}} \sim \mathcal{N}\left(0, \sigma_{\text{x}}^2\right)$ and $\varepsilon_{\text{t}}^{\text{y}} \sim \mathcal{N}\left(0, \sigma_{\text{y}}^2\right)$ iid. I then examine, how varying the parameter

r affects the learned fixed points. For each parameter setting in the range $0.5 \leq r \leq 2.25$ I trained the system using 32 trials, lasting 20 time steps each, with $\sigma_{\mathrm{x}} = 0.2$ and $sigma_{\mathrm{y}} = 0.05$, and the initial condition close to zero $x(0) \sim \mathcal{N}(0, 0.001^2)$. I then fit the FP GP-ADF model to the data with 16 inducing points, with locations initialised automatically to equidistantly span the data for each dataset; and using 5 fixed point as parameters (overestimating the true numbers of 1 or 3 depending on the value of r), letting the ARD formulation determine the number of fixed points present in the system.

I first confirmed, that the method indeed captures the available data well for various values of the bifurcation parameter r, as shown in figure 4.2 A and B. I then create the so-called bifurcation plot, figure 4.2 C, based on the learned parameter values. The fixed points identified by FP GP-ADF truthfully track the expected location and stability, while successfully recovering the true number of fixed points. Consistently with previous findings on similar systems Diks 2006, we indeed find that noise shifts the bifurcation towards larger values of r, and when the distances of the noiseless fixed points are on the order of the transition noise σ_{x}, the random fixed points are not detectable from data.

4.3.3 Chemical reactor dynamics

To test the ability of the FP GP-SDE algorithm to recover realistic and complex dynamics from well-understood real life systems, I turned towards chemistry, in particular the dynamics of the iodate-AS(III) system under imperfect mixing, described by Ganapathisubramanian 1991. This system is well-approximated by two coupled first-order ODEs describing the time evolution of the concentrations of two iodate species, I^- and IO_3^- in a two-compartment reaction chamber. For certain parameter settings this system exhibits complex multi-stable dynamics, giving rise to three stable and two unstable fixed points. To simulate realistic data, I took the absorption spectra of the two iodate species provided in Kireev and Shnyrev 2015, and the simulated output thus resembles an absorption spectrum that is a linear mixture of the individual spectra, weighted by the concentrations, and corrupted by low levels of Gaussian measurement noise.

The complete system is described in its parametric form[11] as

[11] Please refer to Ganapathisubramanian 1991 for the meaning of the individual parameters and how the below system was derived.

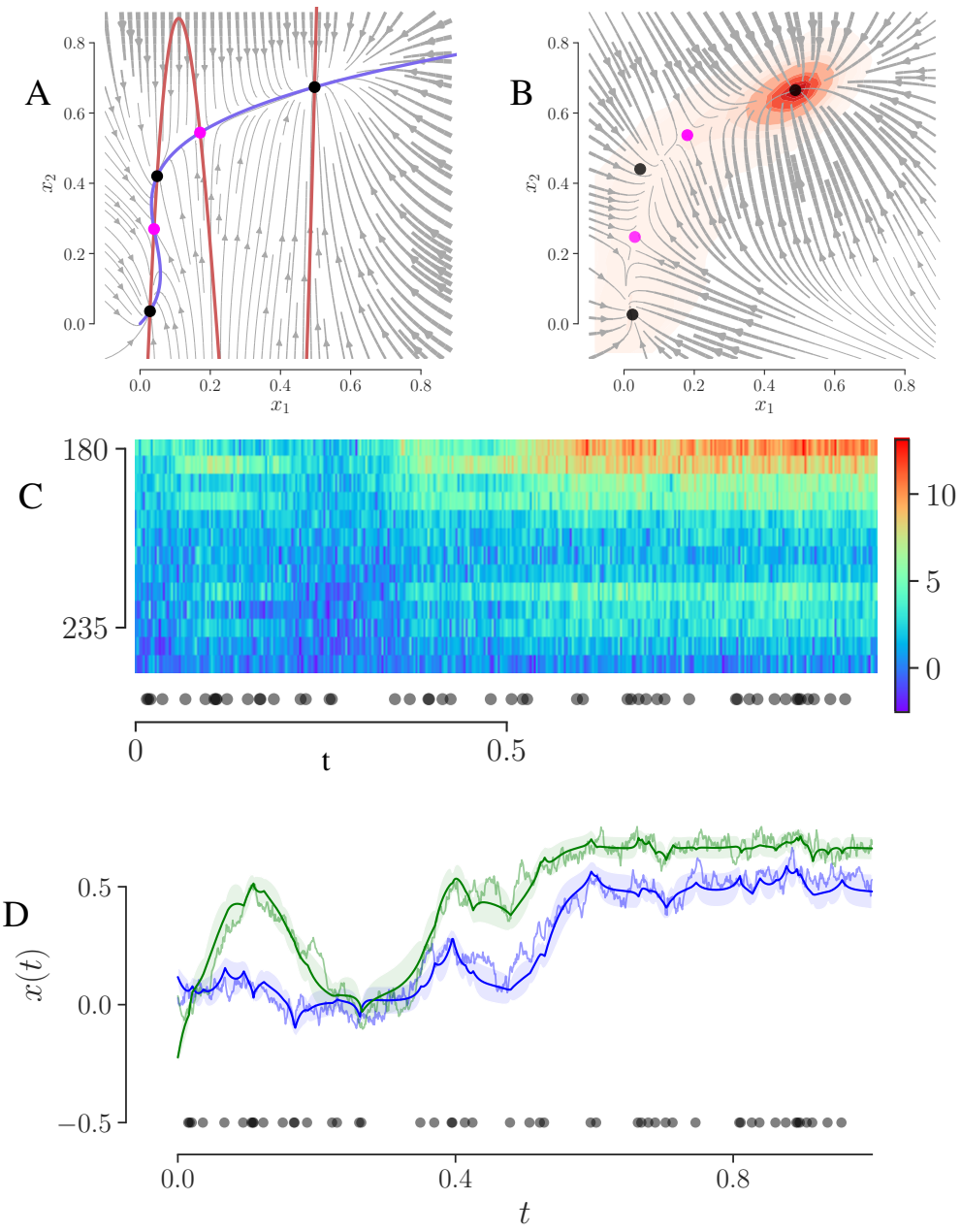


Figure 4.3: Multistable chemical reaction dynamics. A: Streamline plot of the concentration dynamics for two species of iodine, together with the nullclines and fixed points. Stable fixed points are black, unstable ones are magenta. B: Learnt dynamics and fixed points with stability determined by eigenvalues of learnt Jacobian matrices. Increasing uncertainty in the fixed-point observation is indicated by higher transparency of the dot. The red contour plot illustrates the density of latent path locations across all trials used for training. C: Example spectroscopy measurements (output process) across light wavelengths (nm). D: Example true latent path together with the inferred posterior mean and ± 1 standard deviation tubes for each latent dimension on the same trial as C. The black dots indicate the time points at which measurements were taken.

$$
\begin{aligned}
\mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}}{dt}\right] &= \left(k_{\mathrm{a}}\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}} + k_{\mathrm{b}}\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}^2\right)\left(S_0 - \lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}\right) \\
&\quad + \frac{F_1\lfloor \mathrm{I}^- \rfloor_0}{V_{\mathrm{A}}} - \frac{(F_3+F_4)\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}}{V_{\mathrm{A}}} + \frac{F_4\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}}{V_{\mathrm{A}}} \\
\mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}}{dt}\right] &= \left(k_{\mathrm{a}}\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}} + k_{\mathrm{b}}\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}^2\right)\left(S_0 - \lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}\right) \\
&\quad + \frac{F_4\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}}{V_{\mathrm{D}}} - \frac{F_4\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}}{V_{\mathrm{D}}} \\
\mathbf{x} = \begin{bmatrix} \lfloor \mathrm{I}^- \rfloor_{\mathrm{A}} \\ \lfloor \mathrm{I}^- \rfloor_{\mathrm{D}} \end{bmatrix} &= \begin{bmatrix} S_0 - \lfloor \mathrm{IO_3}^- \rfloor_{\mathrm{A}} \\ S_0 - \lfloor \mathrm{IO_3}^- \rfloor_{\mathrm{D}} \end{bmatrix} \qquad f(\mathbf{x}) = \begin{bmatrix} \mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_{\mathrm{A}}}{dt}\right] \\ \mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_{\mathrm{D}}}{dt}\right] \end{bmatrix} \\
d\mathbf{x} &= f(\mathbf{x})dt + \sigma_{\mathrm{x}} d\mathbf{w} \\
\mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \varepsilon_{\mathrm{y}} \qquad \boldsymbol{\varepsilon}_{\mathrm{y}} \sim \mathcal{N}(0, \sigma_{\mathrm{y}}\mathbf{I})
\end{aligned}
\tag{4.52}
$$

where stochasticity arises due to small volumes and low concentrations, and thus low numbers of molecules. The parameter settings of the simulation for the results shown in figure 4.3 were as follows

Noise free dynamics	$\lfloor \mathrm{I}^- \rfloor_0 = 4.4 \times 10^{-5}$	$k_0 = 2.7 \times 10^{-3}$
	$V_{\mathrm{A}} = 4 \times 10^1$	$F_4 = 3.25 \times 10^{-3}$
	$V_{\mathrm{D}} = 1$	$F_3 = k_0 V_{\mathrm{A}}$
	$k_{\mathrm{a}} = 2.1425 \times 10^{-1}$	$F_1 = \frac{1}{2}F_3$
	$k_{\mathrm{b}} = 2.1425 \times 10^4$	$F_2 = \frac{1}{2}F_3$
	$S_0 = \frac{1}{2}\left(\lfloor \mathrm{I}^- \rfloor_0 + 1.42 \times 10^{-3}\right)$	
Stochastic simulation	$dt = 1 \times 10^{-3}$	$\sigma_{\mathrm{x}} = 0.3 \times 10^{-3}$
	$\sigma_{\mathrm{y}} = 1 \times 10^{-3}$	

We finally fit the FP GP-SDE model to simulated data of 20 trials with different initial conditions, but the same parameter settings, and sampled a 13 dimensional spectroscopic observation at 50 random time points on each trial (see figure 4.3 C), resulting in a $20 \times 50 \times 13$ dimensional dataset. As a result of the fit, we recovered the correct dynamical portrait, fixed point locations and stability as shown in figure 4.3 A and B as well as inferred stochastic latent dynamics correctly, even ones that switch between the various fixed points within a single trial, as shown in figure 4.3 D.

4.3.4 Neural population dynamics

This example demonstrates the FP GP-SDE algorithm under multivariate point-process observations. We model the intensity functions of the nth output process as $\eta_{\mathrm{n}}(t) = \exp(\sum_{\mathrm{k}=1}^{\mathrm{K}} C_{\mathrm{nk}} x_{\mathrm{k}}(t) + d_{\mathrm{n}})$. Conditioned on the intensity function, the $\phi(n)$ observed event-times $\mathbf{t}^{(\mathrm{n})}$ are generated by a Poisson process with log likelihood

$$\log p(\mathbf{t}^{(\mathrm{n})}|\eta_{\mathrm{n}}) = -\int T\,\eta_{\mathrm{n}}(t)dt + \sum_{\mathrm{i}=1}^{\phi(\mathrm{n})} \log \eta_{\mathrm{n}}(t_{\mathrm{i}}^{(\mathrm{n})}) \tag{4.53}$$

In contrast to the Gaussian observation case, the first term in the log-likelihood above is continuous in $\eta_{\mathrm{n}}(t)$ and the absence of events is also informative towards the underlying intensity of the process.

This observation process is common in neural data analysis, where electrophysiological recording results in a set of spike-times of a population of simultaneously recorded neurons jointly embedded in a circuit involved in performing a computation. In fact, studying neural population activity as a dynamical system has gained increasing traction in the field of neuroscience in recent years (see Macke et al. 2011; Shenoy, Sahani, and Churchland 2013; Pandarinath et al. 2018b, including my own contribution in Park, Bohner, and Macke 2015) $f$and data analysis methods that can obtain such descriptions are thus of great interest.

We simulate a two-dimensional latent SDE using the dynamics $f_{\mathrm{k}}(\mathbf{x}) = -x_{\mathrm{k}} + \sigma_{\mathrm{k}}(w_{\mathrm{k1}}x_1 - w_{\mathrm{k2}}x_2 - z_{\mathrm{k}})$ for $k = 1,2$, where $\sigma_{\mathrm{k}}(x) = (1+\exp(-b_{\mathrm{k}}x))^{-1}$. Depending on the choice of parameters b_{k}, w_{kj} and z_{k} the dynamical system will exhibit different properties. We explore the two regimes where the system either has two stable and one unstable fixed points (Figure 4.4 C left) or exhibits a single stable spiral (Figure 4.4 C right).

We simulate data from 50 neurons on 25 trials for each of the two parameter regimes for b_{k}, w_{kj} and z_{k}. Figure 4.4 A shows example neural spike trains under the two regimes. Figure 4.4 B illustrates sample paths through the latent space under the different dynamical regimes, together with the density of latent locations visited across all trials. In both settings, we initialise our algorithm with three fixed points and inducing points placed on an evenly spaced 4×4 grid in $(-0.25, 1.25)$, and hold the parameters relating to the output mapping constant. Figure 4.4 D shows the estimated flow fields in both settings, together with the location of the fixed points and their stability as indicated by the eigenvalues of the Jacobian matrices. In both settings, the FP GP-SDE method successfully recovers the main qualitative distinguishing features of the dynamics. In the regime where the dynamics are conditioned on three fixed points but the generative system only contains one, the two additional fixed points will either be associated with higher uncertainty or move to regions where no or little data was observed, as indicated by the superimposed density plots.

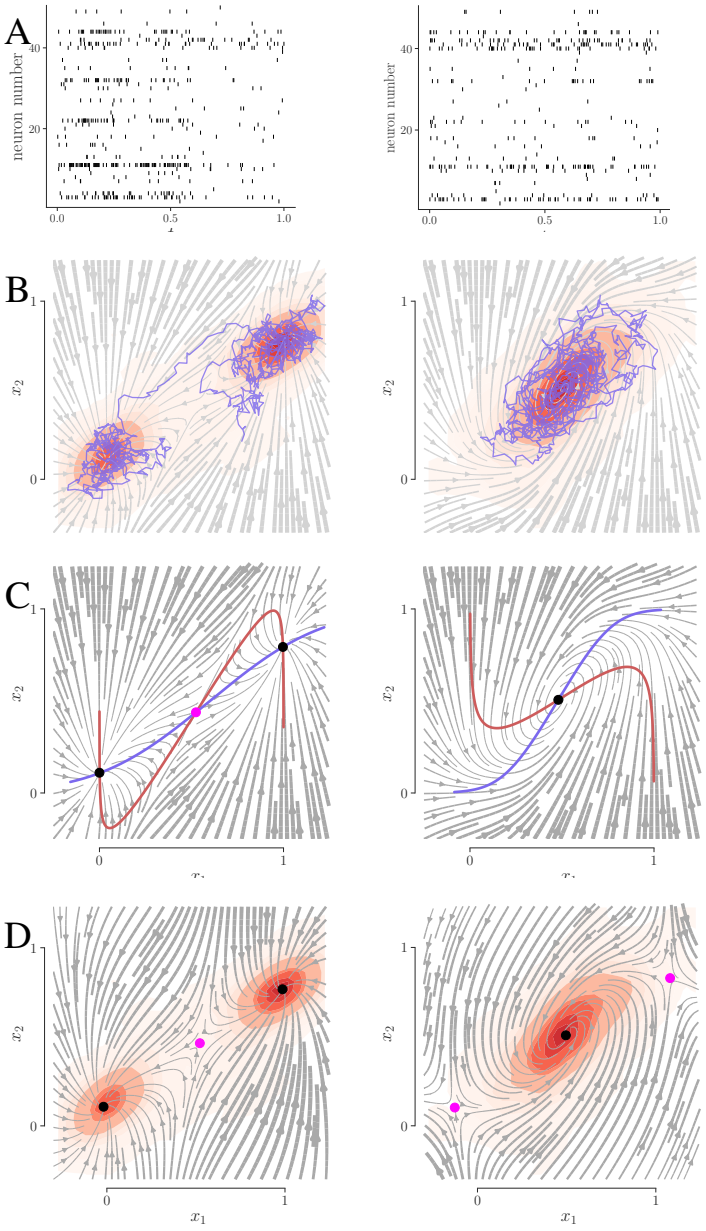


Figure 4.4: Neural population dynamics. Left: simulations with parameter settings $b_1 = 1.9, b_2 = 0.5, z_1 = 3, z_2 = 3.9, w_{11} = 10, w_{12} = 5, w_{21} = 9, w_{22} = 3$. Right: simulations with parameter settings $b_1 = 0.4, b_2 = 0.6, z_1 = 1.7, z_2 = 7, w_{11} = 20, w_{12} = 16, w_{21} = 21, w_{22} = 6$. A: Raster plot of the observed spike times for a population of 50 neurons for an example trial. B: Example paths through the two-dimensional latent space on the same trial as A, together with a density plot of latent locations visited across all trials that were used for learning the dynamics, shown in red. C: Streamline plots of the true dynamics together with their fixed points and nullclines for each latent dimension. Stable fixed points are black, unstable ones are magenta. D: Same density plots as in B together with streamline plots of the learnt dynamics and learnt fixed points. The fixed point stability is shown as indicated by the eigenvalues of the learnt Jacobian matrices.

4.3.5 Mutually inhibiting neural populations observed via calcium imaging

Lastly, I demonstrate potential use of the FP GP-ADF algorithm on the type of data that we discussed in chapters 2 and 3, recorded from a set of neurons via calcium imaging. To simulate such data that could potentially be gathered with techniques becoming available within the next couple years, I chose a model of mutually inhibiting neural population by Machens, Romo, and Brody 2005. My choice was motivated by the fact that not only does the model exhibit interesting parameterised dynamics, similarly to the examples above, but it was also carefully set up such that individual simulated neurons' characteristics matched those recorded from monkey pre-frontal cortex.

They created a simplified spiking simulation model with realistic integrate-and-fire neurons, structured inhibitory connectivity between two populations, and strictly external excitation[12] (see figure 4.5 A). This network was parametrised to be consistent with observed neural characteristics from the Rhesus monkey brain, with original neurophysiological data and the original working memory-decision making task published in Romo et al. 1999. Considering the increasing effort for making calcium imaging possible in primate brains (Seidemann et al. 2016; O'Shea et al. 2017), and that I participated in the effort (Trautmann et al. 2019) for analysing the initial results of monkey calcium imaging, I decided to simulate the observed data as if the spiking network was viewed through a calcium reporter and subsequent fluorescent imaging.

As in our study monkeys were injected with an AAV1-CamKIIA-GCaMP6f virus construct, I used the GCamp6f kinetics as identified in vivo[13] as 45 ms rise time, 142 ms decay time and 19% $\Delta F/F_0$ fluorescence increase (see Chen et al. 2013 Supplementary table 3), and thus I use the so-called fast rise, exponential decay (FRED, Norris et al. 2005) equation to describe the calcium response after a spike (similarly to Lütcke et al. 2013, and see figure 4.5 C inset for the resulting shape):

$$f^{\text{FRED}}(t - t_{\text{spike}} \mid \tau_{\text{rise}}, \tau_{\text{decay}}, A) = A * \left(e^{2\frac{\tau_{\text{rise}}}{\tau_{\text{decay}}}} \right)^{1/2} \left(e^{-\frac{\tau_{\text{rise}}}{t - t_{\text{spike}}} - - \frac{t - t_{\text{spike}}}{\tau_{\text{decay}}}} \right). \tag{4.54}$$

For the purposes of this simulation, non-linear saturation effects were not considered, to enable us focusing more precisely on discovering the latent dynamics, however for real recordings they may be taken into account both for the simulation and during model inference/learning (eg. Speiser et al. 2017). I simulated 30-30 spiking neurons from both populations A and B, then convolved their spike trains with the FRED curve, to obtain the 'noiseless' calcium traces shown in figure 4.5 C.

The simulation then takes noisy snapshots of the instantaneous fluorescence every 125 ms (8 Hz), as is typical in calcium imaging to create the final dataset (see figure 4.5 C, markers). The additive Gaussian observation noise level was set heuristically to match typical effective noise characteristics after the intracellular variations, and various data recording and preprocessing steps (see

[12] One might think of these as two groups of interneurons

[13] In mice, monkey data is unavailable, some population average hints in Seidemann et al. 2016

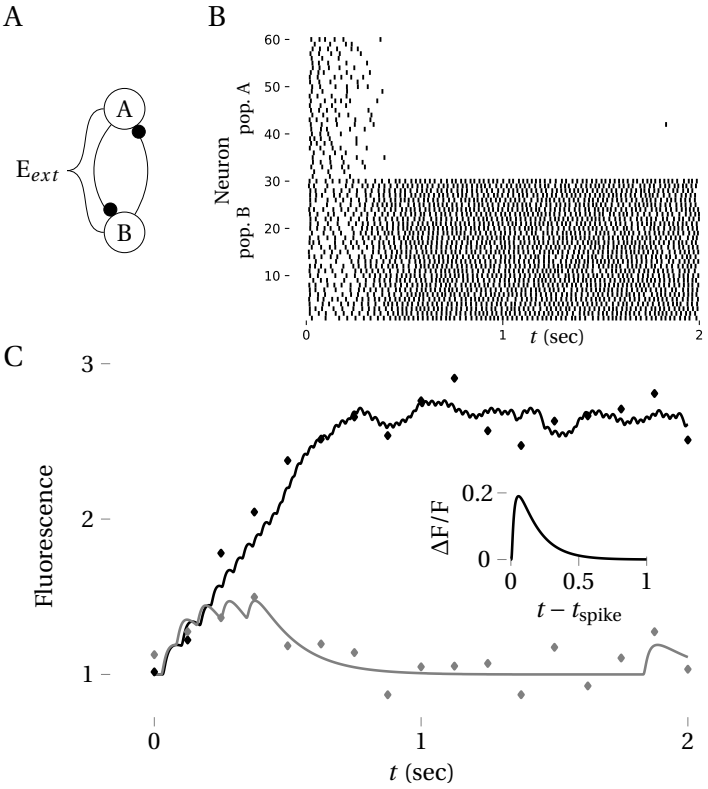


Figure 4.5: Simulation of mutually inhibiting neural populations with calcium observations. A: Circuit diagram of two mutually inhibiting neural populations under external excitation E_{ext}. B: Example raster plot during 'decision making'. Due to the mutual inhibition, in this regime we observe winner-take-all dynamics, and thus population A is silenced. C: Same trial as in B, showing two example neurons. Solid lines show the instantaneous fluorescence as a result of spikes being convolved with GCamp6f observation kinetics. The noisy and sparse observed data is shown by the markers. Inset: The fast rise exponential decay model output used as the convolution kernel.

chapter 2 for a detailed description of the real noise process). For exact details of the simulation, see the original paper (Machens, Romo, and Brody 2005), their detailed supplementary explanation and code, and my simulation and model fitting code available online[14].

The model fit results are shown in figure 4.6, and were fit to 40 simulated trials of 60 neurons' fluorescence, observed at 17 time points over a 2 second trial, with initial conditions set to be at $(1, 1)$. The model behaves differently at the two shown external excitation (E_{ext}) levels, in one case implementing a 'decision making' scheme (although here I used unbiased inputs, and only noise was integrated to serve as the basis of such decision) and in the other a 'loading' scheme, with all traces converging to a single stable fixed point (see Machens, Romo, and Brody 2005 about the naming of these behaviours). FP GP-ADF, initialised with 5 potential fixed points, successfully recovers the dynamics, the correct number of fixed points (as identified by the fixed point uncertainty parameter

[14] `https://github.com/gbohner/gpdm-fixpoints`

σ^{s}) and their stability (including learning a saddle node in figure 4.6 A), demonstrating its potential usefulness in a calcium imaging setting.

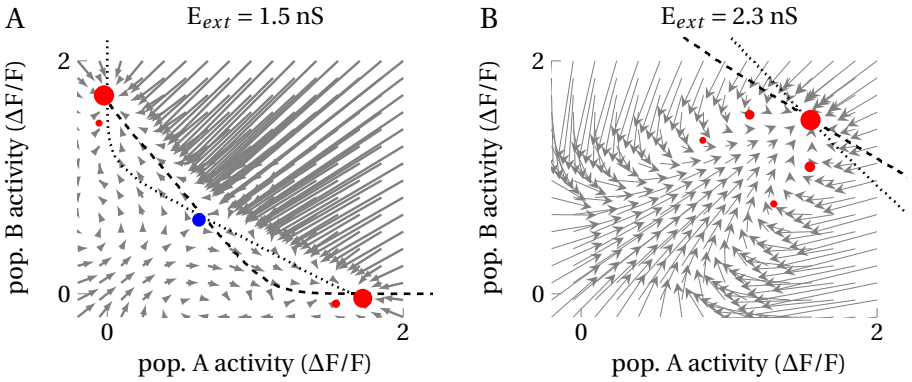


Figure 4.6: Model fits of FP GP-ADF to the simulated data. Dashed line are the numerically computed true nullclines of the simulated model. Grey arrows are the learned dynamics, and markers show the location of learned fixed points. Marker size indicates the inverse of σ^{s} (the large the point is, the more likely it is to be a fixed point). A: Under low external inputs the system exhibits winner take all dynamics. FP GP-ADF correctly identifies the 3 real fixed points, as well as their stability (red is stable, blue here is learned as a saddle node, with one stable and one unstable eigenvalue of the associated $\mathbf{J}$ matrix.). B: Higher excitatory input causes a change in model behaviour, resulting in a single stable fixed point, that is indeed correctly identified.

4.4 Discussion

In this chapter I described two novel algorithms, FP GP-ADF and FP GP-SDE that learn interpretable models of latent stochastic dynamical systems. The varied applications showcase the promise of the proposed idea of conditioning the learned transition functions on their fixed points and also demonstrate the first applications of learning GPSSM models of realistic neural-like data. Although I do hope that these initial efforts will lead to a new class of interpretable machine learning models applied widely to real neural recordings, there are still a few hurdles that need to be overcome for the widespread usage of these type of algorithms, and thus will need to be subject to further research.

Firstly, for now we assumed the dynamical systems under investigation are autonomous, that is there is no input driving the system, or at least the input is stationary over many trials, allowing us to learn the input-driven dynamics and treat changes in input as qualitative changes of the dynamics themselves, such as in section 4.3.5. In this view, the input is changing the effective transition function $f(\cdot)$, which may be undesirable. In previous work on GPSSMs used in the nonlinear control literature (e.g. McHutchon 2014), they extend the transition function to map from the state and inputs jointly to the new state, $\mathbf{x}_{\text{new}} = f(\mathbf{x}, \text{inputs})$. Although we could have added this straightforwardly, this extension assumes that the inputs are known to the model at each point in

time, as they are set by the experimenter. This is often the case in chemistry or robotics application of non-linear control. However, in many real systems, especially in neural recordings that usually investigate a small, local part of a highly interconnected network, the system that we observe is definitely non-autonomous, but we also do not have access to the inputs driving it. Simultaneously learning dynamics implemented by the measured system as well as inferring the inputs driving it is currently an interesting open research question (Soldado Magraner 2018; Sussillo et al. 2016). The major difficulty is in attempting to ensure that the inferred inputs do not explain away the dynamics that actually are implemented, and thus usually there are several constraints placed on the inferred inputs, such as they vary slowly in time or vary little across trials, or map into a particular subspace of linear dynamics (Soldado Magraner 2018; Duncker et al. 2017). A further issue with inferring inputs is the partitioning of the innovations noise. We do expect to have noise in the transitions due to inherent stochasticity and model mismatch, and allowing for the inference of inputs will explain away some of this noise. Overall, I believe inferring inputs along with learning the dynamical system is a powerful idea, but it should be implemented with care, and with very specific considerations and existing insight to the system under investigation. A potentially more generally applicable approach would be to accept the fact that unknown inputs change the learnt dynamics transition function $f(\cdot)$, and allow for it to indeed vary in time, such that we learn a full description of the transition process across trials and time, $f^{\mathrm{m}}(\cdot, t)$. This is of course a very flexible description, and may be difficult to learn, but it can be constrained in several ways, either by learning a switching process with a few fixed transition functions and a Markov Chain switching between them, or by assuming that it varies slowly in time, placing another Gaussian Process over the parameters of $f(\cdot)$, ensuring they change smoothly in time. Both of these ideas have already been implemented for linear latent dynamics (Petreska et al. 2011; Park, Bohner, and Macke 2015), and in my view they show more promise than the attempt to infer the unknown inputs to the system. With regards to their interpretability, they would result in the learning of single-trial bifurcations as we follow the now time-varying fixed points, essentially showing qualitative changes in the dynamics, just as they are being changed by the unknown inputs, and resulting in pictures similar to figure 4.2 C, but with time within trial being on the x-axis.

A second set of concerns is regarding the application of the proposed algorithms to real world datasets. Although the examples shown in section 4.3 demonstrate the reliability of the algorithms on simulated data – the results match the ground truth fixed points extremely well, and approximate the global transition function well too – this is all reliant on the available trials sufficiently exploring the whole of the dynamical portrait. This is best illustrated in figure 4.4 B-D, where the density plots showing data availability correlate strongly with 'how well' the true transition flows are estimate in different parts of the state space. Of course this is a rather general issue, it is well known we can only estimate functions where we have samples from them (unless we assume certain trends or periodicity), and Gaussian Processes are well-suited towards not giving overly confident function estimates

away from data. The larger issue is that in order to learn more complete dynamical portraits, we ideally need rather noisy latent paths exploring large chunks of the state space, whereas current experimental philosophy aims to produce as noiseless and 'reproducible' single trial traces as possible, not ideal for our purposes. A related question is a more precise definition of the 'how well' we learn transition functions. For now we only provided visual comparisons of the true and estimated dynamical portraits, and shown that the fixed point locations and stabilities are indeed what we expect them to be, based on the crossings of nullclines. However, I feel we need further clarifications on how to think of these results, especially when considering real world applications to unknown systems. The potential answer is two-fold, with both directions requiring further work. From a computational point of view, when the ground truth dynamics are known – such as in the simulated experiments above – one ideally needs to define a metric of how much the learn dynamics differ from the ground truth. This should ideally be a metric that submits to hypothesis testing, so we can make statements about learning the 'correct' dynamics matching the ground truth with some probability, given the system and the amount of data. To my knowledge such metrics are not currently available for learnt stochastic dynamical systems, and are highly non-trivial to come up with. A potential avenue would be defining similarity metrics over arbitrary dynamical systems, such as "Kernels and Dynamical Systems" did for linear dynamical systems, and should be explored in the future. Fortunately, from an experimental point of view, when we are applying our methods to real world data, the answer may be a bit clearer, although not necessarily easier to implement. I believe the correct procedure for using these computational methods would be to treat them as iterative hypothesis generators. That is, given an initial dataset, one would apply to algorithms to sketch a dynamical portrait and showing the locations and stabilities of proposed fixed points given multiple trials for each setting of experimentally available parameters (such as in sections 4.3.1 and 4.3.4). With this added information, the next set of experiments can be designed to confirm the locations and stabilities of the proposed fixed points, or to explore new areas of the latent space to get a more complete sense of the dynamics. Therefore I believe that already in their current form the proposed algorithms can play a strong role in assisting experimenters get more useful data out of their experiments by pointing out dynamically interesting features in a very directly accessible and human-interpretable way.

Thirdly, I need to discuss scalability of both the computational costs as well as the ease of interpretability. The proposed methods scale very well with regards of the dimensionality of the observation space due to the instantaneous output mapping from latent state to outputs, therefore working with even thousands of simultaneously recorded – or even stiched, as in Nonnenmacher, Turaga, and Macke 2017 – neural time series that are now becoming available due to improved experimental techniques, is not going to cause issues. However, scalability with regards to latent space dimensionality D is an open question. Computationally, the number of free parameters in the Jacobians scale quadratically with D, and could quickly lead to difficult-to-learn over-parametrised systems. What's worse, the number of inducing points required to tile a D dimensional space suffi-

ciently densely scales exponentially with D, quickly leading with huge kernel matrices k^{zz}, although this could be well-mitigated by further approximations, such as the KISS-GP – introduced in Wilson and Nickisch 2015 and also described here in section 2.2.2 – for inducing point locations $\mathbf{Z}$ on an equidistant grid. The scaling with D with regards to interpretability is potentially more worrying. All our example applications were using only up to 2 dimensional latent spaces, as the descriptive language for local linearisations around fixed point is rather complete in this case, stable or unstable nodes, saddles and spirals are well understood and intuitive to many, and also the transition functions can easily be visualised by regular $x_t \to x_{t+1}$ plots in 1D or by quiver plots in 2D. Already in three dimensions the visualisations (and thus interpretation) become difficult, and the classification of fixed point types – aside from the stable and unstable nodes – is less clear. Therefore instead of attempting to use $D \gg 2$ latent dimensionalities for the proposed methods to capture more variability in the latent space, I believe one would want to implement some of the ideas described in the paragraphs above and use a two-dimensional time-varying $f(\cdot,t) : \mathbb{R}^{2+1} \to \mathbb{R}^2$ transition function to retain the ease of interpretability.

Finally, a minor point regarding the applications of latent stochastic dynamical system to calcium-imaging data, as done in section 4.3.4. The presented application is currently a fairly standard and accepted method of working with calcium imaging observations (e.g. Khan et al. 2018), I wanted to point out that learning latent dynamics while treating the output mapping from latent space to observations as instantaneous is the wrong thing to do, when one wants to uncover the latent computational trajectories. This is due to the temporal convolution of the truly instantaneous spikes with the slower calcium reporter dynamics, as shown in figure 4.5. This time-convolution introduces correlations over time in the neural observations within the output space, whereas most existing state space models – including the ones here – assume time-correlations are only present in the latent computational trajectories, not in the observations. This issue is fairly specific to calcium imaging of neurons, and thus has not been thoroughly explored by other communities. Computational neuroscientists have indeed started working on these correlated-output models (see Speiser et al. 2017), and I do think they become very necessary for high frame-rate calcium imaging videos, where the time between frames is much less than the width of the calcium reporter's response peak, or $\Delta t \ll \tau_{\text{rise}} + \tau_{\text{decay}}$. In lower frame-rate applications, including the one presented here, this issue is much less severe, and should not affect the learnt dynamics or the inferred computational trajectories too much.

To summarise, I presented here a novel way of doing data-driven stochastic bifurcation analysis, and gaining previously unavailable scientific insight into various real-world-like systems. Although this type of analysis is still in its infancy, it can already be a useful tool for scientists to aid their iterative experimental design, and once some of the current limitations are overcome by solutions I pointed out – or even better ones – it has the potential to become an essential tool in analysing various types of real neural data.

Chapter 5

General Discussion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Appendix A

Notation

Throughout this work the reader encounters mathematical concepts of varying difficulty. As my aim is to communicate any given concept as concisely as possible without introducing any ambiguity into the equations, I felt like I need to adapt the descriptive power of the notation to the needs of the concept.

There I use a mixture[1] of the *common matrix-vector* notation for tensors of up to order two, as well as the and *sum* and *indicial* notations for computations that involve higher order tensors. To maintain clarity of which notation is in use at any given time, the reader may use the following guidelines:

- Non-bold objects with or without indices always refer to scalars. For example $\{ a, a_{\mathrm{i}}, A_{\mathrm{ij}}, A_{\mathrm{ijkl}} \}$.
- Bold objects always refer to non-zero order tensors (vectors, matrices or higher order ones). In case indices are present along a bold object, they are used as in indicial notation (see below). For example $\{ \mathbf{a}, \mathbf{A}, \mathbf{a}_{\mathrm{i}}, \mathbf{A}_{\mathrm{ij}}, \mathbf{A}_{\mathrm{ijkl}} \}$.
- For clarity, it is sometimes easier to add a subscript to an object, indicating its "name", that it is different from another object sharing the same base symbol. To avoid confusion of names with indices, I indicate names with an underline, for example $\{ \mathbf{a}_{\underline{\text{first}}}, \mathbf{A}_{\underline{\text{second}}}, \mathbf{A}^{\text{previous}}_{\underline{\text{second}}} \}$. As naming objects via superscripts can not be confused with indicial notation, underlining of superscripts as names may be ommitted.

The main features of the indicial notation, and how they relate to the common and the sum notation most readers are familiar with, are as follows:

1. In a term, the number of free indices - denoted by (possibly indexed) lower case letters of the English alphabet - represent the order of the tensor, with dimensions ordered according to order the in which indices show up in the expression. The range of an index is often not explicitly indicated, but inferred from context.

[1] Each notational style is kept consistent within a particular concept.

Common name	Tensor order	Common notation	Indicial notation
Scalar	0	a	a
Vector	1	$\mathbf{a} \in \mathbb{R}^{\mathrm{D}}$	$\mathbf{a}_{\mathrm{i}}$ i=1...D
Matrix	2	$\mathbf{A}$	$\mathbf{A}_{\mathrm{ij}}$
		$\mathbf{A}^{(\mathrm{r})}$	$\mathbf{A}_{\mathrm{ji}}$
Tensor	3	$\mathcal{A}$	$\mathbf{A}_{\mathrm{ijk}}$
	4	$\mathcal{A}$	$\mathbf{A}_{\mathrm{ijkl}}$

2. Repetition of an index within the same term is called a dummy index, and indicates summation over the range of the index. Common linear algebra operations are:

Common name	Common notation	Indicial notation	Sum notation
Scalar product	$\mathbf{c} = a\mathbf{b}$	$\mathbf{c}_{\mathrm{i}} = a\mathbf{b}_{\mathrm{i}}$	$c_{\mathrm{i}} = ab_{\mathrm{i}}$
Inner product	$c = \mathbf{a}^{(\mathrm{r})}\mathbf{b}$	$c = \mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{i}}$	$c = \sum_{\mathrm{i}=1}^{\mathrm{D}} a_{\mathrm{i}}b_{\mathrm{i}}$
	$\mathbf{c} = \mathbf{A}^{(\mathrm{r})}\mathbf{b}$	$\mathbf{c}_{\mathrm{i}} = \mathbf{A}_{\mathrm{ji}}\mathbf{b}_{\mathrm{j}}$	$c_{\mathrm{i}} = \sum_{\mathrm{j}=1}^{\mathrm{D}} A_{\mathrm{ji}}b_{\mathrm{j}}$
Outer product	$\mathbf{C} = \mathbf{a}\mathbf{b}^{(\mathrm{r})}$	$\mathbf{C}_{\mathrm{ij}} = \mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{j}}$	$C_{\mathrm{ij}} = a_{\mathrm{i}}b_{\mathrm{j}}$
Matrix trace	$c = \mathsf{Tr}(\mathbf{A})$	$c = \mathbf{A}_{\mathrm{ii}}$	$c = \sum_{\mathrm{i}=1}^{\mathrm{D}} A_{\mathrm{ii}}$
Tensor product	-	$\mathbf{c}_{\mathrm{i}} = \mathbf{A}_{\mathrm{ijk}}\mathbf{B}_{\mathrm{jk}}$	$c_{\mathrm{i}} = \sum_{\mathrm{j,k}} A_{\mathrm{ijk}}B_{\mathrm{jk}}$
	-	$\mathbf{C}_{\mathrm{jl}} = \mathbf{A}_{\mathrm{ijk}}\mathbf{B}_{\mathrm{ikl}}$	$C_{\mathrm{jl}} = \sum_{\mathrm{i,k}} A_{\mathrm{ijk}}B_{\mathrm{ikl}}$

3. Functions applied to an element in indicial notation are understood as acting on each element seperately. When a function is to be applied to the whole tensor, we indicate it by square

brackets.

$$\mathbf{A}^{-1} = [\mathbf{A}_{\mathrm{ij}}]^{-1} \neq \mathbf{A}_{\mathrm{ij}}^{-1}$$
$$eA = e^{[\mathbf{A}_{\mathrm{ij}}]} \neq e^{\mathbf{A}_{\mathrm{ij}}}$$

4. We define the following special symbols:

   The all ones tensor $\mathbf{1}_{\mathrm{ijk...}}$. This may be useful to carry out so-called broadcasting: When the free indices do not agree in terms that are being summed, they may be multiplied by the smallest order all ones tensors, such that the sum can be carried out, e.g. $\mathbf{a}_{\mathrm{i}}$ and $\mathbf{B}_{\mathrm{jk}}$ can not be summed, but we may compute $\mathbf{a}_{\mathrm{i}}\mathbf{1}_{\mathrm{jk}} + \mathbf{1}_{\mathrm{i}}\mathbf{B}_{\mathrm{jk}}$.

   The Kronecker delta tensor

$$\boldsymbol{\delta}_{\mathrm{ijk...}} = \begin{cases} 1 & \text{if } i = j = k = \ldots \\ 0 & \text{otherwise} \end{cases}$$

   Using the Kronecker delta tensor, we may extract the diagonal of a matrix $\mathbf{A}$ as the vector $\mathbf{A}_{\mathrm{ij}}\boldsymbol{\delta}_{\mathrm{ijk}}$, or represent a vector $\mathbf{a}$ as a diagonal matrix via $\mathbf{a}_{\mathrm{i}}\boldsymbol{\delta}_{\mathrm{ijk}}$. We may also implement the element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$, as $\mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{j}}\boldsymbol{\delta}_{\mathrm{ijk}}$.

   The single entry tensor, where the raised indices select a single entry from their corresponding dimension

$$\boldsymbol{\Omega}^{\mathrm{a,b,c,...}}_{\mathrm{ijk...}} = \begin{cases} 1 & \text{if } i = a,\ j = b,\ k = c,\ \ldots \\ 0 & \text{otherwise} \end{cases}$$

   This may be used to access the 3rd column of a matrix $\mathbf{A}$ as $\mathbf{A}_{\mathrm{ij}}\boldsymbol{\Omega}^{3}_{\mathrm{j}}$ or a slice from a tensor $\mathbf{A}$ as $\mathbf{A}_{\mathrm{ijkl}}\boldsymbol{\Omega}^{4,2}_{\mathrm{jl}}$.

   The n-dimensional Levi-Civita or permutation tensor

$$\boldsymbol{\varepsilon}_{\mathrm{a_1 a_2 ... a_n}} = \begin{cases} +1 & \text{if } (a_1, a_2, \ldots, a_{\mathrm{n}}) \text{ is an even permutation of } (1, 2, \ldots, n) \\ -1 & \text{if } (a_1, a_2, \ldots, a_{\mathrm{n}}) \text{ is an odd permutation of } (1, 2, \ldots, n) \\ 0 & \text{otherwise} \end{cases}$$

5. Special operators $\odot, \oplus, \ominus$ carry out entrywise operations on tensors of arbitrary, but equal orders and sizes, such as the Hadamard product:

$$c_{\mathrm{i}} = [\mathbf{a}_{\mathrm{i}} \oplus \mathbf{b}_{\mathrm{i}}]_{\mathrm{i}} = A_{\mathrm{i}} + B_{\mathrm{i}}$$
$$C_{\mathrm{ij}} = [\mathbf{A}_{\mathrm{ij}} \odot \mathbf{B}_{\mathrm{ij}}]_{\mathrm{ij}} = A_{\mathrm{ij}} B_{\mathrm{ij}}$$

Furthermore, these operators may be restricted to act only on certain slices of its inputs, and thus can be applied to two tensors of differing orders, as long as the given slice is of equal size in the two tensors. The result will be a tensor that contracts the inputs along the given slice, but keeps the other dimensions from both input tensors, effectively 'broadcasting' each tensor along the missing dimensions.

$$C_{\mathrm{dij}} = [\mathbf{A}_{\mathrm{di}} \oplus_{\mathrm{d}} \mathbf{B}_{\mathrm{dj}}]_{\mathrm{dij}} = A_{\mathrm{di}} + B_{\mathrm{dj}} = \mathbf{A}_{\mathrm{di}} \mathbf{1}_{\mathrm{j}} + \mathbf{B}_{\mathrm{dj}} \mathbf{1}_{\mathrm{i}}$$
$$C_{\mathrm{deijkl}} = [\mathbf{A}_{\mathrm{dije}} \odot_{\mathrm{de}} \mathbf{B}_{\mathrm{dekl}}]_{\mathrm{deijkl}} = A_{\mathrm{dije}} B_{\mathrm{dekl}} = (\mathbf{A}_{\mathrm{dije}} \mathbf{1}_{\mathrm{kl}}) \odot (\mathbf{B}_{\mathrm{dekl}} \mathbf{1}_{\mathrm{ij}})$$

As there is some ambiguity in the order of dimensions in the result, it is recommended to explicitly state the output dimensions of the resulting computation. Also note that

$$\begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ \cdots \\ a_{\mathrm{I}} b_{\mathrm{I}} \end{bmatrix} = \mathbf{a}_{\mathrm{i}} \odot \mathbf{b}_{\mathrm{i}} \neq \mathbf{a}_{\mathrm{i}} \mathbf{b}_{\mathrm{i}} = \sum_{\mathrm{i}} a_{\mathrm{i}} b_{\mathrm{i}}$$

6. We may use capital letters of the English alphabet to denote an ordered set of indices. If $P = \{ijk\}$, then $\mathbf{A}_{\mathrm{ijk}} = A_{\mathrm{P}} \neq \mathbf{A}_{\mathrm{jki}}$.

7. We can denote derivatives as follows. Let $A : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}^{\mathrm{D}_{\mathrm{i}_1} \times \mathrm{D}_{\mathrm{i}_2} \times \cdots \times \mathrm{D}_{\mathrm{i}_n}}$ be an n-th order tensor-valued function over a D-dimensional vector space. The gradient of A in the canonical basis is denoted by placing the coordinate dimension after a comma:

$$\nabla A \triangleq \mathbf{A}_{\mathrm{i}_1 \mathrm{i}_2 \ldots \mathrm{i}_n, \mathrm{d}} = \frac{\partial \mathbf{A}_{\mathrm{i}_1 \mathrm{i}_2 \ldots \mathrm{i}_n}}{\partial \mathbf{x}_{\mathrm{d}}} \quad d = 1, 2, \ldots, D$$

Note that the resulting tensor is now of order $n+1$. Higher order derivatives are denoted by

placing multiple indices after the comma.

$$\mathbf{A}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_\mathrm{n},\mathrm{de}} = \frac{\partial^2 \mathbf{A}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_\mathrm{n}}}{\partial \mathbf{x}_\mathrm{d}\, \partial \mathbf{x}_\mathrm{e}} \quad d,e = 1,2,\ldots,D$$

Remember, that repetition of indices still induces summation over the index, and results in a lower order tensor. We can for example represent the divergence of a scalar-valued function f as

$$\nabla^2 f \triangleq f_{,\mathrm{ii}} = \sum_{\mathrm{i}=1}^{\mathrm{D}} \frac{\partial^2 f}{\partial \mathbf{x}_\mathrm{i}\, \partial \mathbf{x}_\mathrm{i}},$$

which is indeed a scalar.

Let $k : \mathsf{U} \otimes \mathsf{V} \to \mathbb{R}$ be a scalar-valued function defined over the tensor product of real finite dimensional vector spaces. In the derivative, we may denote the identity of vector space by a raised index:

$$\mathbf{k}_{,\mathrm{d}^\mathrm{U}\mathrm{e}^\mathrm{V}} = \frac{\partial^2 k}{\partial \mathbf{u}_\mathrm{d}\, \partial \mathbf{v}_\mathrm{e}}$$

$$\mathbf{u} \in \mathsf{U} \qquad d = 1,2,\ldots,\dim(U)$$

$$\mathbf{v} \in \mathsf{V} \qquad e = 1,2,\ldots,\dim(V)$$

Appendix B

Exponentiated Quadratic kernel function, derivatives and expectations

Our aim here is to define the Exponentiated Quadratic (also known as Radial Basis Function, Squared Exponential or Gaussian) kernel function, write down its derivatives with respect to its arguments, and to derive the various expectations in the scenario, where one of the input arguments is not precisely known, but rather treated as a normally distributed random variable.

The final formulas are then used to estimate the output distribution of a function that has a Gaussian Process prior with an Exponentiated Quadratic covariance function, the function is parametrised by inducing points as well as fixed points and Jacobians around them, and the input to the function is a normally distributed random variable.

B.1 The kernel function

Let $\mathbf{x}$ and $\mathbf{z}$ be D-dimensional vectors and $k : \mathbb{R}^{\mathrm{D}} \otimes \mathbb{R}^{\mathrm{D}} \to \mathbb{R}$ be a positive definite kernel function. The exponentiated quadratic kernel function is parametrised by a positve output scale ν and positive definite covariance matrix $\mathbf{\Lambda}$:

$$k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = \nu e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^\top \mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{B.1}$$

The covariance matrix of the kernel function is often diagonal to reduce the number of parameters. As our latent space models contain rotational non-identifiability due to the flexible output mapping, we can indeed assume this diagonality without loss of generality, thus we set $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ is the D dimensional vector of so-called 'kernel lengthscales'.

The Exponentiated Quadratic kernel function is closely related to the probability density function of the normal distribution in either of its entries, and this identity will be used extensively throughout the derivations:

$$\mathcal{N}_{\mathbf{x}}(\mathbf{z},\,\mathbf{\Lambda}) = p_{\underline{\text{Normal}}}(\mathbf{x} \mid \mathbf{z},\,\mathbf{\Lambda}) \tag{B.2}$$

$$= (2\pi)^{-\mathrm{D}/2}(|\mathbf{\Lambda}|)^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^{\top}\mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{B.3}$$

$$= \frac{1}{\nu}(2\pi)^{-\mathrm{D}/2}(|\mathbf{\Lambda}|)^{-1/2} k(\mathbf{x},\mathbf{z} \mid \nu,\,\mathbf{\Lambda}) \tag{B.4}$$

Therefore also

$$k(\mathbf{x},\mathbf{z} \mid \nu,\,\mathbf{\Lambda}) = \nu(2\pi)^{\mathrm{D}/2}(|\mathbf{\Lambda}|)^{1/2}\mathcal{N}_{\mathbf{x}}(\mathbf{z},\,\mathbf{\Lambda}) \tag{B.5}$$

Furthermore, we are often required to compute the so-called kernel matrix between two sets of inputs. Let $\{\mathbf{x}^{\mathrm{n}}\}_{\mathrm{n}=1}^{\mathrm{N}} = \mathbf{X} \in \mathbb{R}^{\mathrm{DxN}}$ and $\{\mathbf{z}^{\mathrm{m}}\}_{\mathrm{m}=1}^{\mathrm{M}} = \mathbf{Z} \in \mathbb{R}^{\mathrm{DxM}}$ be collection of N and M inputs, then we can define the kernel matrix entry-wise, via common notation or via indicial notation as

$$K_{\mathrm{nm}} = k\left(\mathbf{x}^{\mathrm{n}},\mathbf{z}^{\mathrm{m}} \,\middle|\, \nu,\,\mathbf{\Lambda}\right) \tag{B.6}$$

$$\mathbf{K} = k(\mathbf{X},\mathbf{Z} \mid \nu,\,\mathbf{\Lambda}) \tag{B.7}$$

$$\mathbf{K}_{\mathrm{nm}} = k(\mathbf{X}_{\mathrm{dn}},\mathbf{Z}_{\mathrm{dm}} \,\big|\, \nu,\,\mathbf{\Lambda}_{\mathrm{d\dot{d}}})\,. \tag{B.8}$$

Although possible, computing functions of the kernel matrix is not always simple or intuitive using the entry-wise representation, and thus I first define operations assuming single inputs and using the common linear algebra notation, but then also show how the same operations may be carried out on collections of inputs using tensor calculations via the indicial notation. The single input derivatives and expectations are available at **mchutchon:2013:diffgp**, whereas for collection of inputs some of them were shown in **Girard2004** in an entry-wise fashion.

To my knowledge this is the first publicly accessible document that shows these derivatives and expectations in a tensor format that lends itself easily for computer implementations, unlike previous derivations. However, as the main ideas of these lengthy derivations follow those previously published, I will show only the end result in some cases, to avoid introducing unnecessarily complex notation. The intermediate algebraic manipulations rely mainly upon rearranging terms to produce a normalised normal probability density function under the integral sign (which then integrates to 1), and collecting terms such that we end up with sums and products of easily computable Exponentiated Quadratic kernel functions with modified output scales, lengthscales or inputs.

B.2 The derivative Gaussian Process

As we are using derivative observations as parameters for our fixed point models, we need to define the derivatives of the kernel function with respect to either or both of their inputs. For single inputs the derivatives are

$$k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) = \nu e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^\top \boldsymbol{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{B.9}$$

$$\frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) = -\boldsymbol{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) \tag{B.10}$$

$$\frac{\partial}{\partial \mathbf{z}} k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) = +\boldsymbol{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) \tag{B.11}$$

$$\frac{\partial^2}{\partial \mathbf{x} \partial \mathbf{z}} k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) = \boldsymbol{\Lambda}^{-1}(\mathbf{I} - (\mathbf{x}-\mathbf{z})(\mathbf{x}-\mathbf{z})(r)\boldsymbol{\Lambda}^{-1})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \boldsymbol{\Lambda}) \tag{B.12}$$

With these one can compute all elements of the covariance matrix between observations and derivative observations of a Gaussian Process with an Exponentiated Quadratic kernel function. Note that although the derivatives in equations B.10 and B.11 seem anti-symmetric, when we form the cross-covariance blocks of the joint covariance matrix, we compute $\frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathbf{z})$ and $\frac{\partial}{\partial \mathbf{x}} k(\mathbf{z}, \mathbf{x})$ as the blocks. As the $(\mathbf{x}-\mathbf{z})$ term also switches sign when we flip the order of kernel inputs, these terms are indeed transposes of one-another, and the covariance matrix is symmetric as expected.

We can see that the first derivatives of single inputs are D-dimensional vectors, whereas the second derivative is a $D \times D$ matrix, as derivatives of a tensor-valued function with respect to vectors increase the tensor order of the output by one. Although in the case of single inputs, $k(\cdot, \cdot)$ returns a zero-th order tensor, when the inputs are collections of points, the output is a second order tensor, the 'kernel matrix'. Therefore I now also define the derivatives for collections of points as the input, then also describe how one may flatten the resulting tensors to end up with a valid block-structured covariance matrix describing the joint distribution of observations and derivative observations.

$$\mathbf{K}_{\mathrm{nm}} = k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \tag{B.13}$$

$$\begin{aligned} \mathbf{K}_{\mathrm{d^x nm}} &= \frac{\partial}{\partial \mathbf{x}_{\mathrm{d}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= -[\boldsymbol{\lambda}_{\mathrm{d}}^{-1} \mathbf{1}_{\mathrm{nm}}] \odot [\mathbf{X} \ominus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dnm}} \odot [\mathbf{1}_{\mathrm{d}}\, k(\mathbf{X}, \mathbf{Z})]_{\mathrm{dnm}} \end{aligned} \tag{B.14}$$

$$\begin{aligned} \mathbf{K}_{\mathrm{nd^z m}} &= \frac{\partial}{\partial \mathbf{z}_{\mathrm{d}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= +[\boldsymbol{\lambda}_{\mathrm{d}}^{-1} \mathbf{1}_{\mathrm{nm}}] \odot [\mathbf{X} \ominus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dnm}} \odot [\mathbf{1}_{\mathrm{d}}\, k(\mathbf{X}, \mathbf{Z})]_{\mathrm{dnm}} \end{aligned} \tag{B.15}$$

$$\begin{aligned}\mathbf{K}_{\mathrm{d}^{\mathrm{x}}\mathrm{n}\mathring{\mathrm{d}}^{\mathrm{z}}\mathrm{m}} &= \frac{\partial^2}{\partial \mathbf{x}_{\mathrm{d}}\, \partial \mathbf{z}_{\mathring{\mathrm{d}}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu,\, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= \left[\boldsymbol{\Lambda}_{\mathrm{d}\mathring{\mathrm{d}}}^{-1}\mathbf{1}_{\mathrm{nm}} - \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} (\mathbf{X}_{\mathrm{dn}} \ominus_{\mathrm{d}} \mathbf{Z}_{\mathrm{dm}}) \odot_{\mathrm{nm}} (\mathbf{X}_{\mathring{\mathrm{d}}\mathrm{n}} \ominus_{\mathring{\mathrm{d}}} \mathbf{Z}_{\mathring{\mathrm{d}}\mathrm{m}}) \odot_{\mathring{\mathrm{d}}} \boldsymbol{\lambda}_{\mathring{\mathrm{d}}}^{-1})\right]_{\mathrm{dn}\mathring{\mathrm{d}}\mathrm{m}} \\ &\quad \odot \left[\mathbf{1}_{\mathrm{d}\mathring{\mathrm{d}}}\, k(\mathbf{X}, \mathbf{Z})\right]_{\mathrm{dn}\mathring{\mathrm{d}}\mathrm{m}} \end{aligned} \tag{B.16}$$

These equations are now directly implementable in modern computer languages with little effort, using the appropriate broadcasting operations to form tensors of correct order and size.

The last question is how to flatten these higher order tensors back to second order to form the block-structured covariance matrix of a Gaussian Process. Let $\mathbf{S}_{\mathrm{dn}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{N}}$ be the N locations of derivative observations $\mathbf{J}_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{n}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}\times\mathrm{N}}$, and $\mathbf{Z}_{\mathrm{dm}}$ the locations of regular observations $\mathbf{U}_{\mathrm{dm}}$. In such a case we obtain the tensors $\mathbf{K}^{\mathrm{zz}}_{\mathrm{mm}} = k^{\mathrm{zz}}$, $\mathbf{K}^{\mathrm{zs}'}_{\mathrm{m(dn)}} = \nabla_2 k^{\mathrm{zs}}$, $\mathbf{K}^{\mathrm{s}'\mathrm{z}}_{\mathrm{(dn)m}} = \nabla_1 k^{\mathrm{sz}}$, and $\mathbf{K}^{\mathrm{s}'\mathrm{s}'}_{\mathrm{(dn)}(\mathring{\mathrm{d}}\mathring{\mathrm{n}})} = \nabla_1\nabla_2 k^{\mathrm{ss}}$. The higher order tensors should be flattened as follows to obtain the joint distribution of regular and derivative observations $\mathbf{U}_{1\cdot}$ and $\mathbf{J}_{1\cdot\cdot}$ (setting the output dimension in question arbitrarily to $d = 1$):

$$\left[\begin{array}{c} U_{11} \\ \vdots \\ U_{1\mathrm{M}} \\ \hline J_{111} \\ J_{121} \\ \vdots \\ J_{1\mathrm{D}1} \\ J_{112} \\ \vdots \\ J_{1\mathrm{DN}} \end{array}\right] = \mathcal{N}\left(\mathbf{0},\, \left[\begin{array}{ccc|ccccccc} & \mathbf{K}^{\mathrm{zz}}_{\mathrm{mm}} & & & & & \mathbf{K}^{\mathrm{zs}'}_{\mathrm{m(dn)}} & & & \\ \hline K^{\mathrm{s}'\mathrm{z}}_{(11)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(11)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(11)} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(21)} & \cdots & K^{\mathrm{s}'\mathrm{s}'}_{(11)(\mathrm{D}1)} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(12)} & \cdots & K^{\mathrm{s}'\mathrm{s}'}_{(11)(\mathrm{DN})} \\ K^{\mathrm{s}'\mathrm{z}}_{(21)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(21)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(21)(11)} & \ddots & & & & & \\ & \vdots & & & & & & & & \\ K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{D}1)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{D}1)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{D}1)(11)} & & & \mathbf{K}^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{dn})(\mathrm{dn})} & & & \\ K^{\mathrm{s}'\mathrm{z}}_{(12)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(12)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(12)(11)} & & & & & & \\ & \vdots & & & & & & & \ddots & \\ K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{DN})1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{DN})\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{DN})(11)} & & & & & & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{DN})(\mathrm{DN})} \end{array}\right]\right) \tag{B.17}$$

B.3 Expectations with respect to noisy input

Let $\mathbf{Z} \in \mathbb{R}^{\mathrm{D}\times\mathrm{M}}$ be a collection of known inputs and $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$ a noisy input, such that $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a random variable distributed according to a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. As described in chapter 4, this $\mathbf{x}$ represents a belief, and we wish to propagate this belief through a transition function represented by a Gaussian process represented by inducing points, fixed points and derivative observations around the fixed points. In order to compute our updated belief representation, we need to calculate the expected mean and the co-variance of the output $f(\mathbf{x})$ with respect to the noisy input $\mathbf{x}$. These expressions (see equations 4.14 to 4.16) contain various expectations

of kernel matrices (defined equation 4.10), and in turn require us to calculate expectations of various functions involving the kernel function $k(\cdot,\cdot)$. Fortunately all these expectation are available in closed form for the Exponentiated Quadratic kernel function, and thus here I state the solutions that enable us to propagate latent beliefs forward analytically. The superscripts on the resulting tensor identify what did we take the expectation of, whereas the subscripts denote the correct order and size of the tensor.

$$\begin{aligned}\mathbf{E}_{\mathrm{m}}^{\mathrm{k}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= k\left(\boldsymbol{\mu}, \mathbf{Z} \;\middle|\; \frac{|\boldsymbol{\Lambda}|^{1/2}}{|\boldsymbol{\Lambda}+\boldsymbol{\Sigma}|^{1/2}}\nu,\, \boldsymbol{\Lambda}+\boldsymbol{\Sigma}\right)\end{aligned} \tag{B.18}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}}^{\mathrm{xk}} &= \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\, k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= \mathbf{E}_{\mathrm{m}}^{\mathrm{k}} \odot_{\mathrm{m}} \left[(\boldsymbol{\Lambda}^{-1}+\boldsymbol{\Sigma}^{-1})^{-1}_{\mathring{\mathrm{d}}\mathrm{d}} \left([\boldsymbol{\Lambda}^{-1}\mathbf{Z}]_{\mathrm{dm}} \odot_{\mathrm{d}} [\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}]_{\mathrm{d}}\right)\right]_{\mathrm{dm}}\end{aligned} \tag{B.19}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}}^{\mathrm{dk}} &= \mathbb{E}_{\mathrm{x}}\left[\nabla_2\, k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} \left(\mathbf{E}_{\mathrm{dm}}^{\mathrm{xk}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}_{\mathrm{m}}^{\mathrm{k}}\right)\end{aligned} \tag{B.20}$$

$$E^{\mathrm{v}} = \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{x},\mathbf{x} \mid \nu,\, \boldsymbol{\lambda})\right] = \nu \tag{B.21}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{Z},\mathbf{x} \mid \nu,\, \boldsymbol{\lambda})\; k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= k\left(\mathbf{Z}, \mathbf{Z} \;\middle|\; \frac{1}{2^{\mathrm{D}/2}}\nu,\, 2\boldsymbol{\Lambda}\right) \odot k\left([\mathbf{Z} \oplus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dm}\mathring{\mathrm{m}}}, \boldsymbol{\mu}_{\mathrm{d}} \;\middle|\; \frac{|\boldsymbol{\Lambda}|^{1/2}}{\left|\frac{\boldsymbol{\Lambda}}{2}+\boldsymbol{\Sigma}\right|^{1/2}}\nu,\, \frac{\boldsymbol{\Lambda}}{2}+\boldsymbol{\Sigma}\right)\end{aligned} \tag{B.22}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}\mathring{\mathrm{m}}}^{\mathrm{xkk}} &= \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\, k(\mathbf{Z},\mathbf{x} \mid \nu,\, \boldsymbol{\lambda})\; k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= \mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \left[\left(\left(\frac{\boldsymbol{\Lambda}}{2}\right)^{-1}+\boldsymbol{\Sigma}^{-1}\right)^{-1} \left(\left(\frac{\boldsymbol{\Lambda}}{2}\right)^{-1} \odot_{\mathrm{d}} \left([\mathbf{Z} \oplus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dm}\mathring{\mathrm{m}}} \oplus_{\mathrm{d}} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right)\right]_{\mathrm{dm}\mathring{\mathrm{m}}}\end{aligned} \tag{B.23}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{m}(\mathrm{d}\mathring{\mathrm{m}})}^{\mathrm{kdk}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{Z},\mathbf{x} \mid \nu,\, \boldsymbol{\lambda})\; \nabla_2\, k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right] \\ &= \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} \left(\mathbf{E}_{\mathrm{dm}}^{\mathrm{xkk}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}}\right)\end{aligned} \tag{B.24}$$

$$\mathbf{E}_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{xxkk}} = \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\,\mathbf{x}\, k(\mathbf{Z},\mathbf{x} \mid \nu,\, \boldsymbol{\lambda})\; k(\mathbf{x},\mathbf{Z} \mid \nu,\, \boldsymbol{\lambda})\right]$$

$$= \left[\mathbf{E}^{\mathrm{kk}}_{\mathrm{m}\mathring{\mathrm{m}}} \left(\left(\frac{\boldsymbol{\Lambda}}{2} \right)^{-1} + \boldsymbol{\Sigma}^{-1} \right)^{-1}_{\mathrm{d}\mathring{\mathrm{d}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} + \left[\mathbf{E}^{\mathrm{xkk}}_{\mathrm{dm}\mathring{\mathrm{m}}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{xkk}}_{\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \tag{B.25}$$

$$\begin{aligned} \mathbf{E}^{\mathrm{dkdk}}_{(\mathrm{dm})(\mathring{\mathrm{d}}\mathring{\mathrm{m}})} &= \mathbb{E}_{\mathrm{x}} \left[\nabla_1 \, k(\mathbf{Z}, \mathbf{x} \mid \nu, \, \boldsymbol{\lambda}) \, \nabla_2 \, k(\mathbf{x}, \mathbf{Z} \mid \nu, \, \boldsymbol{\lambda}) \right] \\ &= \left[\boldsymbol{\lambda}^{-1}_{\mathrm{d}} \boldsymbol{\lambda}^{-1}_{\mathring{\mathrm{d}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}} \\ &\quad \odot_{\mathrm{d}\mathring{\mathrm{d}}} \left[\mathbf{E}^{\mathrm{xxkk}}_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}^{\mathrm{xkk}}_{\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} - \mathbf{Z}_{\mathring{\mathrm{d}}\mathring{\mathrm{m}}} \odot_{\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{xkk}}_{\mathrm{dm}\mathring{\mathrm{m}}} + \mathbf{Z}_{\mathrm{dm}} \mathbf{Z}_{\mathring{\mathrm{d}}\mathring{\mathrm{m}}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{kk}}_{\mathrm{m}\mathring{\mathrm{m}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \end{aligned} \tag{B.26}$$

Bibliography

Archambeau, Cedric, Dan Cornford, Manfred Opper, and John Shawe-Taylor (2007). "Gaussian process approximations of stochastic differential equations". In: *Journal of machine learning research* 1, pp. 1–16 (cit. on pp. 106–109).

Archambeau, Cédric, Manfred Opper, Yuan Shen, Dan Cornford, and John S Shawe-taylor (2008). "Variational inference for diffusion processes". In: *Advances in Neural Information Processing Systems*, pp. 17–24 (cit. on pp. 106–109).

Arnold, Ludwig and Hans Crauel (Jan. 1991). "Random dynamical systems". In: pp. 1–22 (cit. on p. 90).

Attias, Hagai (2000). "A variational baysian framework for graphical models". In: *Advances in neural information processing systems*, pp. 209–215 (cit. on p. 108).

Bohner, Gergo and Maneesh Sahani (2018). "Empirical fixed point bifurcation analysis". In: *arXiv preprint arXiv:1807.01486* (cit. on p. 91).

Chen, Taoyi, Zhong Xue, Changhong Wang, Zhenshen Qu, Kelvin K. Wong, and Stephen T C Wong (2012). "Motion correction for cellular-resolution multi-photon fluorescence microscopy imaging of awake head-restrained mice using speed embedded HMM". In: *Computerized Medical Imaging and Graphics* 36.3, pp. 171–182 (cit. on p. 30).

Chen, Tsai-Wen, Trevor J. Wardill, Yi Sun, Stefan R. Pulver, Sabine L. Renninger, Amy Baohan, Eric R. Schreiter, Rex A. Kerr, Michael B. Orger, Vivek Jayaraman, Loren L. Looger, Karel Svoboda, and Douglas S. Kim (2013). "Ultrasensitive fluorescent proteins for imaging neuronal activity". In: *Nature* 499.7458, pp. 295–300 (cit. on pp. 12, 120).

Churchland, Mark M., John P. Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy (June 2012). "Neural population dynamics during reaching". In: *Nature* 487, 51 EP – (cit. on p. 89).

Cseke, Botond, David Schnoerr, Manfred Opper, and Guido Sanguinetti (2016). "Expectation propagation for continuous time stochastic processes". In: *Journal of Physics A: Mathematical and Theoretical* 49.49, p. 494002 (cit. on p. 106).

Damianou, Andreas C., Michalis K. Titsias, and Neil D. Lawrence (2011). "Variational Gaussian Process Dynamical Systems". In: pp. 1–9 (cit. on p. 90).

De Haas, J. T M and P. Dorenbos (2010). "Methods for accurate measurement of the response of photomultiplier tubes and intensity of light pulses". In: *IEEE Nuclear Science Symposium Conference Record* 58.3, pp. 205–210 (cit. on p. 27).

Deisenroth, Marc and Shakir Mohamed (2012). "Expectation propagation in Gaussian process dynamical systems". In: *Advances in Neural Information Processing Systems*, pp. 2609–2617 (cit. on p. 98).

Deisenroth, Marc Peter, Marco F. Huber, and Uwe D. Hanebeck (2009). "Analytic moment-based Gaussian process filtering". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 225–232 (cit. on pp. 98, 100, 101).

Deisenroth, Marc Peter, Ryan Darby Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl Edward Rasmussen (2012). "Robust filtering and smoothing with gaussian processes". In: *IEEE Transactions on Automatic Control* 57.7, pp. 1865–1871 (cit. on p. 98).

Diks, C (2006). "A weak bifurcation theory for discrete time stochastic dynamical systems". In: *Tinbergen Institute Discussion Paper No. 06-* June, pp. 1–30 (cit. on pp. 90, 115).

Dossi, R, A Ianni, G Ranucci, and O Ju Smirnov (2000). "Methods for precise photoelectron counting with photomultipliers". In: *Nuclear Instruments and Methods in Physics Research A* 451, pp. 623–637 (cit. on p. 27).

Duncker, L, DJ O'Shea, W Goo, KV Shenoy, and Sahani M (2017). *Low-rank non-stationary population dynamics can account for robustness to optogenetic stimulation*. COSYNE talk (cit. on pp. 90, 123).

Duncker, Lea and Maneesh Sahani (2018). "Temporal alignment and latent Gaussian process factor inference in population spike trains". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 10466–10476 (cit. on p. 90).

Duncker, Lea, Gergo Bohner, Julien Boussard, and Maneesh Sahani (2019). "Learning interpretable continuous-time models of latent stochastic dynamical systems". In: *Proceedings of the 36th International Conference on Machine Learning* (cit. on pp. 91, 103, 111).

Eleftheriadis, Stefanos, Tom Nicholson, Marc Deisenroth, and James Hensman (2017). "Identification of Gaussian Process State Space Models". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 5309–5319 (cit. on p. 90).

Frigola, Roger, Yutian Chen, and Carl E. Rasmussen (2014). "Variational Gaussian Process State-Space Models". In: pp. 1–9 (cit. on p. 93).

Ganapathisubramanian, N (1991). "Tristability in the iodate–As (III) chemical system arising from a model of stirring and mixing effects". In: *The Journal of chemical physics* 95.4, pp. 3005–3008 (cit. on p. 115).

Golub, Matt and David Sussillo (2018). "FixedPointFinder: A Tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks". In: *Journal of Open Source Software* 3.31, p. 1003 (cit. on p. 90).

Greenberg, David S. and Jason N D Kerr (2009). "Automated correction of fast motion artifacts for two-photon imaging of awake animals". In: *Journal of Neuroscience Methods* 176.1, pp. 1–15 (cit. on p. 30).

Khan, Adil G, Jasper Poort, Angus Chadwick, Antonin Blot, Maneesh Sahani, Thomas D Mrsic-Flogel, and Sonja B Hofer (June 2018). "Distinct learning-induced changes in stimulus selectivity and interactions of GABAergic interneuron classes in visual cortex". In: *Nature neuroscience* 21.6, pp. 851–859 (cit. on p. 125).

Kireev, SV and SL Shnyrev (2015). "Study of molecular iodine, iodate ions, iodide ions, and triiodide ions solutions absorption in the UV and visible light spectral bands". In: *Laser Physics* 25.7, p. 075602 (cit. on p. 115).

Ko, Jonathan, Daniel J Klein, Dieter Fox, and Dirk Hähnel (2007). "GP-UKF: Unscented kalman filters with Gaussian process prediction and observationmodels". In: *Iros*, pp. 1901–1907 (cit. on p. 98).

Lütcke, Henry, Felipe Gerhard, Friedemann Zenke, Wulfram Gerstner, and Fritjof Helmchen (2013). "Inference of neuronal network spike dynamics and topology from calcium imaging data". In: *Frontiers in Neural Circuits* 7, p. 201 (cit. on p. 120).

Machens, Christian K., Ranulfo Romo, and Carlos D. Brody (2005). "Flexible control of mutual inhibition: A neural model of two-interval discrimination". In: *Science* 307.5712, pp. 1121–1124 (cit. on pp. 120, 121).

Macke, Jakob H, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani (2011). "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems*, pp. 1350–1358 (cit. on p. 118).

Mante, Valerio, David Sussillo, Krishna V Shenoy, and William T Newsome (Nov. 2013). "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503.7474, pp. 78–84 (cit. on p. 89).

McHutchon, Andrew (2014). "Nonlinear Modelling and Control using Gaussian Processes". In: August (cit. on pp. 93, 95, 98, 122).

Nonnenmacher, Marcel, Srinivas C Turaga, and Jakob H Macke (2017). "Extracting low-dimensional dynamics from multiple large-scale neural population recordings by learning to predict correlations". In: *Advances in Neural Information Processing Systems (accepted)* Nips (cit. on p. 124).

Norris, J. P., J. T. Bonnell, D. Kazanas, J. D. Scargle, J. Hakkila, and T. W. Giblin (2005). "Long-Lag, Wide-Pulse Gamma-Ray Bursts". In: *The Astrophysical Journal* 627.1, pp. 324–345 (cit. on p. 120).

O'Shea, Daniel J., Eric Trautmann, Chandramouli Chandrasekaran, Sergey Stavisky, Jonathan C. Kao, Maneesh Sahani, Stephen Ryu, Karl Deisseroth, and Krishna V. Shenoy (2017). "The need for calcium imaging in nonhuman primates: New motor neuroscience and brain-machine interfaces". In: *Experimental Neurology* 287. Bio-electronics and prosthetics for neurological diseases, pp. 437 –451 (cit. on p. 120).

Pandarinath, Chethan, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. (2018b). "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature methods*, p. 1 (cit. on p. 118).

Pandarinath, Chethan, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo (2018a). "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature Methods* 15.10, pp. 805–815 (cit. on p. 90).

Park, Mijung, Gergo Bohner, and Jakob H Macke (2015). "Unlocking neural population non-stationarities using hierarchical dynamics models". In: *Advances in Neural Information Processing Systems*, pp. 145–153 (cit. on pp. 90, 118, 123).

Petreska, Biljana, Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V Shenoy, and Maneesh Sahani (2011). "Dynamical segmentation of single trials from population neural data". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., pp. 756–764 (cit. on p. 123).

Pnevmatikakis, E. A., J. Merel, A. Pakman, and L. Paninski (2013). "Bayesian spike inference from calcium imaging data". In: *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 349–353 (cit. on p. 95).

Pnevmatikakis, Eftychios A. and Andrea Giovannucci (2017). "NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data". In: *Journal of Neuroscience Methods* 291, pp. 83–94 (cit. on p. 30).

Poyiadjis, George, Arnaud Doucet, and Sumeetpal S. Singh (Feb. 2011). "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation". In: *Biometrika* 98.1, pp. 65–80 (cit. on p. 98).

Ramakrishnan, Naveen, Emre Ertin, and Randolph L Moses (2011). "Assumed density filtering for learning Gaussian process models". In: *Statistical Signal Processing Workshop (SSP), 2011 IEEE*. IEEE, pp. 257–260 (cit. on p. 98).

Rasmussen, C E and C K I Williams (2006). *Regression*. MIT Press, Chapter 2 (cit. on p. 20).

Romo, Ranulfo, Carlos D. Brody, Adrián Hernández, and Luis Lemus (1999). "Neuronal correlates of parametric working memory in the prefrontal cortex". In: *Nature* 399.6735, pp. 470–473 (cit. on p. 120).

Sahani, Maneesh (2018). "Describing non-linear latent dynamics". COSYNE Workshop - RNNs: What are we doing and why? (Cit. on p. 91).

Seidemann, Eyal, Yuzhi Chen, Yoon Bai, Spencer C Chen, Preeti Mehta, Bridget L Kajs, Wilson S Geisler, and Boris V Zemelman (2016). "Calcium imaging with genetically encoded indicators in behaving primates". In: *eLife* 5.2016JULY (cit. on p. 120).

Shenoy, Krishna V, Maneesh Sahani, and Mark M Churchland (2013). "Cortical control of arm movements: a dynamical systems perspective". In: *Annual review of neuroscience* 36, pp. 337–359 (cit. on p. 118).

Silverman, B W (1985). "Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting". In: *Source: Journal of the Royal Statistical Society. Series B (Methodological)* 47.1, pp. 1–52 (cit. on p. 21).

Smola, Alexander J., Rene Vidal, and Vishy Vishwanathan. "Kernels and Dynamical Systems". find at http://alex.smola.org/papers/2004/unpubSmoVidVis04.pdf (cit. on p. 124).

Snelson, E and Z Ghahramani (2006). "Sparse Gaussian processes using pseudo-inputs". In: *Advances in Neural Information Processing Systems* 18, p. 1257 (cit. on p. 21).

Snelson, Edward and Zoubin Ghahramani (2012). "Variable noise and dimensionality reduction for sparse Gaussian processes". In: (cit. on p. 96).

Soldado Magraner, Joana (2018). "Linear Dynamics of Evidence Integration in Contextual Decision Making". In: (cit. on pp. 90, 123).

Speiser, Artur, Jinyao Yan, Evan Archer, Lars Buesing, Srinivas C Turaga, and Jakob H Macke (2017). "Fast amortized inference of neural activity from calcium imaging data with variational autoencoders". In: *Advances in Neural Information Processing Systems (accepted)* Nips, pp. 1–11 (cit. on pp. 120, 125).

Sussillo, David and Omri Barak (2013). "Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks". In: *Neural computation* 25.3, pp. 626–649 (cit. on p. 90).

Sussillo, David, Rafal Jozefowicz, L. F. Abbott, and Chethan Pandarinath (2016). *LFADS - Latent Factor Analysis via Dynamical Systems* (cit. on pp. 90, 123).

Titsias, Michalis (2009b). "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial Intelligence and Statistics*, pp. 567–574 (cit. on pp. 104, 108).

Titsias, Michalis (2009a). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Aistats* 5, pp. 567–574 (cit. on p. 18).

Trautmann, Eric, Daniel O'Shea, Xulu Sun, Jim Marshel, Ailey Crow, et al. (2019). "Two photon imaging and CLARITY for primate motor neuroscience and optical brain machine interfaces" (cit. on p. 120).

Wan, E.a. A and R. Van Der Merwe (2000). "The unscented Kalman filter for nonlinear estimation". In: *Technology* v, pp. 153–158 (cit. on p. 98).

Wang, Hui, Xiaoli Chen, and Jinqiao Duan (2018). "A Stochastic Pitchfork Bifurcation in Most Probable Phase Portraits". In: pp. 1–9 (cit. on p. 90).

Whitaker, Michael (2010). "Genetically-encoded probes for measurement of intracellular calcium". In: *Methods in cell biology* 99, pp. 153–182 (cit. on p. 12).

Wilson, Andrew Gordon and Ryan Prescott Adams (2013). "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: 28 (cit. on pp. 21, 25).

Wilson, Andrew Gordon and Hannes Nickisch (2015). "Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)". In: 37 (cit. on pp. 20–22, 125).

Wu, Anqi, Nicholas A. Roy, Stephen Keeley, and Jonathan W Pillow (2017). "Gaussian process based nonlinear latent structure discovery in multivariate spike train data". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 3496–3505 (cit. on p. 90).

Young, Michael D., Jeffrey J. Field, Kraig E. Sheetz, Randy A. Bartels, and Jeff Squier (2015). "A pragmatic guide to multiphoton microscope design". In: *Advances in Optics and Photonics* 7.2, p. 276 (cit. on p. 14).

Yu, Byron M., John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani (2009). "Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity". In: *Journal of Neurophysiology* 102.1. PMID: 19357332, pp. 614–635 (cit. on p. 90).

Zhao, Yuan and Il Memming Park (2016). "Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains". In: pp. 1–25 (cit. on p. 90).