Summer 9-1-2014

Scaling Multidimensional Inference for Big Structured Data

Elad Gilboa
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Electrical and Systems Engineering

Dissertation Examination Committee:
Arye Nehorai, Chair
Viktor Gruev
Joseph O'Sullivan
Heinz Schaettler
Kilian Q. Weinberger

Scaling Multidimensional Inference for Big Structured Data

by

Elad Gilboa

A dissertation presented to the Graduate School of Arts and Sciences
of Washington University in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

August 2014
Saint Louis, Missouri

Contents

List of Figures

List of Tables

Acknowledgments

I am sincerely grateful to my advisor, Dr. Professor Arye Nehorai, Chair, for his mentoring, guidance, trust, and support throughout my research at Washington University.

I would like to thank the McDonnell International Scholar Academy for inspiring me to continue with my education and for supporting me both financially and intellectually through out my life in the university.

I wish to thank my dissertation defence committee members, Dr. Joseph O'Sullivan, Dr. Heinz Schaettler, Dr. Kilian Q. Weinberger, and Dr. Viktor Gruev, for their valueable suggestions and comments on improving my dissertation. I also wish to thank my numerous instructors from both the Technion University in Israel and Washington University in St. Louis, for helping me construct a solid background for my research.

I further thank all my collaborators Patricio S. La Rosa, Yunus Saatci, Francesca Strappini, Andrew G. Wilson, Kendrick Kay, Viktor Gruev, Abraham Z. Snyder, and John P. Cunningham, who made this research possible and fun.

A special thanks goes to my parents for their endless love, support, and encouragement. Without them none of this would have been possible.

Elad Gilboa

Washington University in Saint Louis
August 2014

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Scaling Multidimensional Inference for Big Structured Data

by

Elad Gilboa

Doctor of Philosophy in Electrical and Systems Engineering

Washington University in St. Louis, August 2014

Professor Arye Nehorai, Chair

"In information technology, big data is a collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications" [151]. In a world of increasing sensor modalities, cheaper storage, and more data oriented questions, we are quickly passing the limits of tractable computations using traditional statistical analysis methods. Methods which often show great results on simple data have difficulties processing complicated multidimensional data. Accuracy alone can no longer justify unwarranted memory use and computational complexity. Improving the scaling properties of these methods for multidimensional data is the only way to make these methods relevant. In this work we explore methods for improving the scaling properties of parametric and nonparametric models. Namely, we focus on the *structure* of the data to lower the complexity of a specific family of problems. The two types of structures considered in this work are distributive optimization with separable constraints (Chapters 2-3), and scaling Gaussian processes for multidimensional lattice input (Chapters 4-5). By improving the scaling of these methods,

we can expand their use to a wide range of applications which were previously intractable open the door to new research questions.

Chapter 1

Introduction

As sensor and storage technologies are becoming increasingly cheaper and more widely used, modern applications increasingly involve multidimensional big datasets. The captured data is often highly complex, correlated, high dimensional, nonlinear, and stochastic. These properties make the use of sophisticated inference methods computationally prohibitive and necessitate improving the scaling ability of these methods.

In this thesis we consider and develop methods to significantly reduce the scaling burden of inference methods. In the literature, a significant amount of research has gone into improving the scaling of generic methods either by simplifying the model or by using a small subset of the data. However, these simplifying assumptions improve computational complexity at the expense of modeling accuracy, and can depend strongly on the properties of the data. In this work we consider an alternative approach. Instead of dealing with the generic case, we limit our domain to a family of problems with a useful structure that allows significantly lowering their complexity without altering the inference method. Specifically in this work we consider optimization problems with block separable constraints (Chapters 2-3) and Gaussian processes for multidimensional lattice input (Chapters 4-5).

1.1 Parallel optimization with separable constraints

The first part of this work explores parallel optimization routines for problems with block separable constraints. The assumption of block separable constraints is valid for many practical problems, such as multi-agent resource allocation, where resources are being distributed amongst several agents that influence the choice of allocation. Examples of such problems

are found in smartgrid scheduling and budgeting [162], managing multi-modal sensor networks [28], and communication networks [118].

Parallel methods are commonly used to lower the runtime of hard problems. However, parallel methods introduce new challenges, such as synchronization between subproblems, convergence issues, and communication overhead. In this work we consider a parallel distributed algorithm that uses an adaptive regularizer (PDAR) for optimization problems with separable constraints. The algorithm utilizes an adaptive step-size between iterations to synchronize the distributed subproblems. We will show the theoretical convergence properties of the proposed algorithm, and illustrate its effectiveness in simulated and real problems. An important real application of this algorithm is using measurements from a microelectrode array to find the electrical conductivity of an excitable tissue.

The electrical conductivity of an excitable tissue is important for understanding the tissue's structure and functioning. However, the inverse problem of inferring spatial conductivity from data is highly ill-posed and computationally intensive. Here, we propose a novel method to solve the inverse problem of inferring tissue conductivity from a set of transmembrane potential and stimuli measurements made by microelectrode arrays (MEA). We start by formulating a forward model of the tissue, using a reaction-diffusion model with an anisotropic inhomogeneous electrical conductivity-tensor field. Then we solve the inverse problem using a single-step approximation and a parallel optimization based on the PDAR algorithm. We analyze the performance of our algorithm, then discuss its application to real measurements obtained from smooth-muscle cardiac tissue, using data collected with a high-resolution MEA system.

1.2 Scaling Gaussian Processes regression with multi-dimensional grid inputs

The second part of this work explores scaling Gaussian process regression for problems with lattice data input. In the machine learning community, Gaussian processes (GPs) have become a popular tool for nonparametric Bayesian regression. However, naive GP regression has $\mathcal{O}(N^3)$ runtime and $\mathcal{O}(N^2)$ memory complexity, where N is the number of observations.

At ten thousand or more observations, this problem is practically intractable, given current hardware. Many algorithms for improving GP scaling approximate the covariance with lower rank matrices. Other work has exploited various structures inherent in particular covariance functions. However, these GP advances have not been well extended to the multidimensional input setting, despite the preponderance of multidimensional applications. Here we extend the initial work of [127] on multiplicative kernel GPs with inputs on a multidimensional grid. We then generalize the method to handle incomplete grids, heteroscedastic noise, and automatic pattern discovery and extrapolation on large multidimensional datasets. These advances enable the use of our GP method on a wide variety of applications.

In Chapter 5, we use our efficient GP algorithm for image interpolation and denoising in division of focal plane images (DoFP). Image interpolation and denoising are inherent to digital image acquisition as most digital cameras are composed of a 2D grid of heterogeneous imaging sensors. The sensors capture only partial information of the true scene, leading to a loss of spatial resolution as well as inaccuracy of the captured polarization information. Interpolation is a standard technique to recover the missing information and increase the accuracy of the captured polarization information. Our Gaussian process regression allows for statistical image interpolation, where estimates of sensor noise are used to improve the accuracy of the estimated pixel information. This produces significant improvements over previously published interpolation methods for polarimeters, which is most pronounced in cases of low signal-to-noise ratio (SNR).

In Chapter 6, we use our efficient GP algorithm for denoising fMRI data. Although conceptually attractive, GP use in the neuroscience community has been limited by burdensome scaling properties. Naively solving exact GP inference is limited to datasets with only a few thousands data points. In a standard fMRI experiment the number of data points (voxels) can easily reach hundreds of thousands, if not millions, making GP infeasible. Fortunately, fMRI data inputs (voxels) are lie on a 4D grid (3D space + time), which makes exact GP inference, for the first time, competitive for fMRI analysis. Our GP-based statistical denoising enables adaptive noise-based smoothing that learns its spatiotemporal structure from the data and runs in practicable time.

1.3 Contributions of this work

Distributed Optimization via Adaptive Regularization for Large Problems with Separable Constraints

Here, we propose a fully distributed parallel method to solve optimization problems over multidimensional data sets. Our method can be applied to a wide variety of nonlinear problems where the constraints are block separable. In order to coordinate among the subproblems, we introduce an adaptive regularizer term that penalizes large changes in successive iterations. Our method can be seen as an extension of the classical proximal point method (PPM) with two novel advances. First, we use PPM to coordinate among the parallel subproblems, not to handle non-differentiability. Second, we enforce coordination by using adaptive regularizers that vary across different subproblems.

Estimating Electrical Conductivity Tensors of Biological Tissues from Microelectrode Arrays Data

The contributions of this effort are two-fold. First, we introduce a discrete forward model of transmembrane potential based on a diffusion-reaction model with an anisotropic inhomogeneous electrical conductivity tensor field. Second, we propose a novel parallel optimization algorithm for solving the complex inverse problem of estimating the conductivity tensor field. Specifically, we propose a single-step approximation with a parallel block-relaxation optimization method. This combination simplifies the joint tensor field estimation problem into a set of computationally tractable problems, allowing the use of efficient standard optimization algorithms. We analyze the performance of our algorithm using numerical examples of several electrical conductivity field topologies and noise levels, and discuss its application to real measurements obtained from cardiac tissue, using a high resolution MEA system.

Scaling Multidimensional Inference for Structured Gaussian Processes

While efficient methods for structured GPs are known in the case of scalar inputs, many regression applications involve multivariate inputs. We present a novel algorithm for GPs with a multiplicative kernel structure, where multidimensional inputs are on a lattice (GP-grid). We extend our GP-grid algorithm to handle two limitations of the basic algorithm by allowing for (i) incomplete data and (ii) heteroscedastic noise. Lastly, we enhance the method by incorporating expressive kernels, which learn hidden patterns in the data. These extensions to standard GP have certainly been used to good purpose in previous GP settings,

but their success can not be replicated in the large N case without additional advances related to this specific multidimensional grid structure.

Image Interpolation and Denoising for Division of Focal Plane Sensors using Gaussian Processes
This chapter presents an efficient GP inference for improved interpolation of DoFP polarimeter data. The GP statistical inference is able learn the properties of the data, and it incorporates an estimation of the sensor noise in order to increase the accuracy of the polarization information and improve spatial resolution.

Denoising fMRI data using Gaussian Processes
This chapter introduces an efficient GP-based analysis for denoising fMRI data. GP regression is a convenient, rigorous, and powerful method that generalizes and unifies previous ideas on smoothing, temporal filtering, statistical modeling, and drift removal. We will show that our GP-based method allows for several advances: 1)It removes drift removal by learning the drifts' properties simultaneously on the entire brain. 2)It jointly learns the localized spatial and temporal correlations and the heteroscedastic voxels' noise. 3)It adaptively varies the smoothing level of the voxel. 4)It shows significant improvement for real fMRI data over fixed-width smoothers, while lessening the sensitivity/specificity tradeoff.

1.4 Organization of the dissertation

The rest of the dissertation is organized as follows. Chapters 2 and 3 consider the the problem of distributive optimization with separable constraints. In Chapter 2, we develop a new general method of parallel distributive optimization for problems with separable constraints. We provide a convergence analysis as well as experimental results on simulated data. In Chapter 3, based on our parallel method, we develop a mathematical framework for solving the inverse problem of estimating effective electrical tissue conductivities from a set of electric potentials and stimulus measurements. In Chapter 4, we consider the problem of scaling Gaussian process regression for structure problems with multidimensional input. We develop an efficient algorithm that can be extended to cases of incomplete grid, heteroscedastic noise, and expressive kernels. In Chapter 5, we apply our efficient GP method to interpolate and denoise division of focal plane images. In Chapter 6, we use our GP method as part of a

routine for denoising fMRI data. We finally summarize the dissertation in Chapter 7, and point out potential future directions.

1.5 Notations

The notational conventions adopted in this work are as follows: We write a scalar as x, a vector as $\mathbf{x}$, a matrix as $\mathbf{X}$. The ith element of a vector is in the typeface of a scalar x_i. The ith row and jth column of $\mathbf{X}$ is $X(i,j)$ or $X_{i,j}$. The ith row of $\mathbf{X}$ is $\mathbf{X}_{i,:}$ or $\mathbf{x}_i$. The ith column of $\mathbf{X}$ is $\mathbf{X}_{:,i}$ or $\mathbf{x}_i$. We write a function as $g(t)$, and a lowercase bold Roman font indicates a vector function, e.g., $\mathbf{g}_t$. Writing the time index as a subscript indicates the vector at the nth timepoint, e.g., $\mathbf{x}_n \equiv \mathbf{x}[n\Delta t]$ We represent an inclusive range between a and b as $a:b$ or $a,\ldots,b$. By standard convention, even though Gaussian Process hyperparameters form a vector, we represent them with the typeface of a scalar, θ.

Symbols used

$\mathbb{R}$	The real numbers.
$\mathbb{R}^+$	Positive real numbers.
$\mathbb{C}$	The complex numbers.
$\mathbb{Q}$	The rational numbers.
$\mathbb{Z}$	The integers.
$\mathbb{Z}^+$	Positive integers.
$x*$	The complex conjugate of x.
$\mathbf{X} \otimes \mathbf{Y}$	The Kronecker product of $\mathbf{X}$ and $\mathbf{Y}$.
$\mathbf{1}$	A vector of ones.
$\mathbf{0}$	A vector of zeros or a matrix of zero, depending on context.
$\mathbf{I}_D$	The identity matrix of size D.
$\mathbb{S}_n$	Denotes the vector space of symmetric $n \times n$.
$\mathbb{S}_{n+}$	Denotes positive semi definite matrices.
$\mathbb{S}_{n++}$	Denotes positive definite matrices.
$\|\| \cdot \|\|_{\mathrm{F}}$	The Frobenius norm, defined as $\|\|\boldsymbol{X}\|\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\|X_{ij}\|^2}$.
$\mathrm{vec}(\mathbf{X})$	The vectorization of a matrix $\mathbf{X}$.
$\mathrm{diag}(\mathbf{X})$	The diagonal of a matrix $\mathbf{X}$.
$\mathrm{tr}(\mathbf{X})$	The trace of a matrix $\mathbf{X}$.
$\mathbf{X}^T$	The transpose of a matrix $\mathbf{X}$.
$\mathrm{KL}(p\|\|q)$	The Kullback-Leibler (KL) divergence between distributions p and q.
$H(p)$	The entropy of distribution p.
$\mathbb{E}(X)$	Expectation of a random variable X.

$\mathbb{V}(X)$	Variance of a random variable X.
$\mathbb{E}_{p(\cdot)}(X)$	Expectation of a random variable X with respect to p.
$\mathbb{V}_{p(\cdot)}(X)$	Variance of a random variable X with respect to p.
$\mathrm{Cov}(\mathbf{X})$	Covariance of a vector random variable $\mathbf{X}$.
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A Gaussian distribution with specified mean $\boldsymbol{\mu}$ and (co-)variance $\boldsymbol{\Sigma}$. Random variable symbol is omitted.
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	A Gaussian distribution with specified mean $\boldsymbol{\mu}$ and (co-)variance $\boldsymbol{\Sigma}$. Random variable symbol *not* omitted.
Student-t$_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A multivariate Student's t distribution with mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$, and ν degrees of freedom.
$\Gamma(\alpha, \beta)$	A gamma distribution with shape α and inverse scale β.
Poisson(λ)	A Poisson distribution with mean λ.
$\mathcal{GP}(\mu, k)$	A Gaussian process (GP) with mean function $\mu(\cdot)$ and kernel $k(\cdot, \cdot)$.
$x \equiv y$	x defined as y.
$\mathcal{O}(\cdot)$	The big-O asymptotic complexity of an algorithm.
$\leftarrow$	An assignment operation in an algorithm.
w.r.t.	Shortening of the phrase "with respect to".
IID	Shortening of the phrase "independent, identically distributed".
p.d.f.	Shortening of the phrase "probability density function".
c.d.f.	Shortening of the phrase "cumulative distribution function".

Chapter 2

Distributed Optimization via Adaptive Regularization for Large Problems with Separable Constraints

2.1 Abstract

Many practical applications require solving an optimization over large and high-dimensional data sets, which makes these problems hard to solve and prohibitively time consuming. In this work, we propose a parallel distributed algorithm that uses an adaptive regularizer (PDAR) to solve a joint optimization problem with separable constraints. The regularizer is adaptive and depends on the step size between iterations and the iteration number. We show theoretical convergence of our algorithm to an optimal solution, and use a multi-agent three-bin resource allocation example to illustrate the effectiveness of the proposed algorithm. Numerical simulations show that our algorithm converges to the same optimal solution as other distributed methods, with significantly reduced computational time.

2.2 Introduction

With the sensor and the storage technologies becoming increasingly cheaper, modern applications are seeing a sharp increase in *big data*. The explosion of such high-dimensional and complex data sets makes optimization problems extremely hard and prohibitively time consuming [21]. Parallel computing has received a significant attention lately as an effective tool to achieve the high throughput processing speeds required for processing big data sets.

Thus, there has a been a paradigm shift from aggregating multi-core processors to utilizing them efficiently [22].

Although distributed optimization has been an increasingly important topic, it has not received sufficient attention since the seminal work by Bertsekas and Tsitsiklis until recently. In the 1980's, Bertsekas and Tsitsiklis extensively studied decentralized detection and consensus problems [17] and developed algorithms such as parallel coordinate descent [145] and the block coordinate descent (BCD) (also called the block Jacobi) [17, 144]. In 1994, Ferris *et. al.* proposed parallel variable distribution (PVD) [40] that alternates between a parallelization and a synchronization step. In the parallelization step, several sub-optimal points are found using parallel optimizations. Then, in the synchronization step, the optimal point is computed by taking an optimal weighted average of the points found in the parallel step. Although PVD claims to achieve better convergence rate than BCD, the complexity of solving optimization in both the steps make it impractical for high dimensional problems. There are other efficient distributive methods in literature, such as the shooting [46], the shotgun [22], and the alternating direction method of multipliers (ADMM) [21], however, these methods apply to only a specific type of optimization problems: ℓ_1-regularization for shooting and shotgun, and linear constraints for ADMM.

In this work, we propose a fully distributed parallel method to solve optimization problems over high-dimensional data sets, which we call the parallel distributive adaptive regularization (PDAR). Our method can be applied to a wide variety of nonlinear problems where the constraints are block separable. The assumption of block separable constraints is valid for many practical problems, such as, multi-agent resource allocation where resources are being distributed amongst several agents that influence the choice of allocation. In order to coordinate among the subproblems we introduce an adaptive regularizer term that penalizes the large changes in successive iterations. Our method can be seen as an extension of the classical proximal point method (PPM) [16] with two novel advances. First, our motivation for using the PPM framework is very different than the original. We use PPM as a means to coordinate among the parallel subproblems and not for handling non differentiability. Second, we enforce coordination by using adaptive regularizers that vary across different subproblems.

The rest of the chapter is organized as follows. In Section 2.3, we formulate the problem; in Section 2.4 we propose our parallel distributive algorithm and show convergence to an

optimum solution; in Section 2.5 we provide numerical simulations, and we conclude the chapter in Section 2.6.

2.3 Problem Formulation

Consider an optimization problem given as:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) && (2.1) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, && (2.2) \end{aligned}$$

where the objective is to find the optimal vector $\boldsymbol{x}^*$ that minimizes the function $f(\boldsymbol{x}) \in \mathbb{R}$, with $\boldsymbol{x} \in \mathbb{R}^d$. The problem is often very complex, nonlinear, and high dimensional, and solving it is prohibitively time consuming. We assume that the constraint $\boldsymbol{x} \in \mathcal{X}$ can be separated into several blocks, such that

$$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N] \text{ where, } \boldsymbol{x}_i \in \mathcal{X}_i, \tag{2.3}$$

with $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^{N} n_i = d$. Once the problem is separated into blocks, distributed iterative approaches (such as the ones mentioned in the Introduction section) can be applied. However, these methods are time consuming when the sub-problems are themselves complex.

2.4 Distributed Optimization via Adaptive Regularization

In this section, we describe our distributed optimization framework with adaptive regularization. We solve the optimization problem given by Eq. (2.1) in a parallel and iterative manner. Let k denote the iteration index and $\hat{\boldsymbol{x}}^k = (\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^k)$, with $\hat{\boldsymbol{x}}_{-i}^k = \left[\hat{\boldsymbol{x}}_1^k, \ldots, \hat{\boldsymbol{x}}_{i-1}^k, \hat{\boldsymbol{x}}_{i+1}^k, \ldots, \hat{\boldsymbol{x}}_N^k\right]$ denote the solution to the optimization problem in the k^{th} iteration. In order to obtain a solution in a distributed manner, we define a set of N augmented objective functions at each

iteration k as [1]

$$L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1}) = f(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_{-i}^{k-1}) + \lambda_i^k(\boldsymbol{h}_i^{k-1})\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i^{k-1}\|^2, \tag{2.4}$$

where $\boldsymbol{h}_i^{k-1} = \hat{\boldsymbol{x}}_i^{k-1} - \hat{\boldsymbol{x}}_i^{k-2}$ is the step taken by the i^{th} block in the $(k-1)^{th}$ iteration, and $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ is an adaptive regularization coefficient which depends on both the indices i and k. We will describe the form of this regularization coefficient shortly. After defining the objective functions $L_i^k(\cdot),\ i = 1, \ldots, N$, we solve N optimization problems in a parallel fashion:

$$\begin{aligned}
\hat{\boldsymbol{x}}_1^k &= \arg\min_{\boldsymbol{x}_1 \in \mathcal{X}_1} L_i^k(\boldsymbol{x}_1; \hat{\boldsymbol{x}}^{k-1}), \\
\hat{\boldsymbol{x}}_2^k &= \arg\min_{\boldsymbol{x}_2 \in \mathcal{X}_2} L_i^k(\boldsymbol{x}_2; \hat{\boldsymbol{x}}^{k-1}), \\
&\vdots \\
\hat{\boldsymbol{x}}_N^k &= \arg\min_{\boldsymbol{x}_N \in \mathcal{X}_N} L_i^k(\boldsymbol{x}_N; \hat{\boldsymbol{x}}^{k-1}).
\end{aligned} \tag{2.5}$$

This optimization framework is in the form of a decomposition-coordination procedure [21], where N agents are trying to minimize their own augmented objective functions, and the new joint vector $\hat{\boldsymbol{x}}^k$ is obtained by simply aggregating the N blocks. If we consider a single objective function $L_i^k(\boldsymbol{x}_i^k)$ at a single iteration k, the minimization of the objective functions is only with respect to he variables of the i^{th} block. However, since the objective function depends also on variables from other blocks, a change in them will cause a change to the objective function, namely $L_i^{k+1}(\boldsymbol{x}_i^k) \neq L_i^k(\boldsymbol{x}_i^k)$.

Next, we discuss the choice of the regularization coefficient $\lambda_i^k(\boldsymbol{h}_i^{k-1})$. We chose $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ to be of the form:

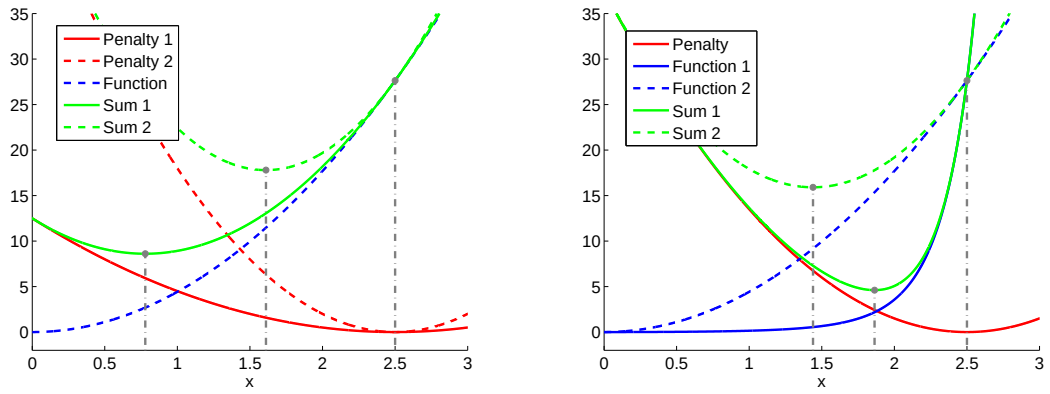
$$\lambda_i^k(\boldsymbol{h}_i^{k-1}) = \begin{cases} \max(\phi(\|\boldsymbol{h}_i^{k-1}\|), \beta) & \text{if } k < K \\ \alpha k & \text{otherwise,} \end{cases} \tag{2.6}$$

where K is a threshold on the iteration index, $\alpha > 0$, and $\beta > 0$ are parameters chosen depending on the problem. Intuitively, the threshold K divides each optimization problem into two phases. The goal of the first phase is to coordinate the parallel optimization. In this phase, each of the agents change their solution in response to the solutions of other

[1] We use a semicolon notation in Eq. (2.4) to clarify that only the variables on the left of the semicolon are allowed to change.

agents. This alternating behavior can be enforced by choosing the function $\phi(\|\boldsymbol{h}_i^{k-1}\|)$ to be a nondecreasing with respect to $\|\boldsymbol{h}_i^{k-1}\|$. This choice will increase the value of regularization coefficient, $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ as $\|\boldsymbol{h}_i^{k-1}\|$ increases. The increase in $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ will in turn enforce a smaller stepsize on the agents that had large change in the previous iteration, to allow other agents to react in the current iteration. The goal of the second phase is to fine tune the solution and to enable it to reach a local optimum.

Stepsize adaptability comes very naturally when using PDAR. As illustrated in Fig. 2.1, for the same objective function, the step size depend on the value of λ (Fig. 2.1(a)); and the objective functions rates of decrease (Fig. 2.1(b)). The adaptive stepsize is used here to regularize and coordinate the separated problems. We also mention here that a non-adaptive regularizer, e.g., $\phi(\|h_i^{k-1}\|) = C$, will cause the convergence to be highly sensitive to the choice of C. Although the choice of the function ϕ and the parameters α, and β theoretically effect the convergence of the optimization, we observed using numerical simulations that the convergence was not sensitive to these choices. In this work we choose $\phi(\|\boldsymbol{h}_i^{k-1}\|) = N^2\|\boldsymbol{h}_i^{k-1}\|$. The algorithm is summarized in Table 2.1.



(a) Step size dependence on penalty.

(b) Step size dependence on objective function

Figure 2.1: Convex functions with additive quadratic penalties at $x = 2.5$. Different combinations of convex functions and penalties result in different stepsizes. In Fig. 2.1(a), the save objective function was used with different penalties, resulting in a different step size. In Fig. fig:2fun1pen, the same penalty was used for two different objective functions. The step size changed depending on the rate of decrease of the objective functions.

Algorithm: PDAR
$k = 1$; % Iteration counter
Initialize $\boldsymbol{x}^0$ and $\lambda_i^0 \ \forall \ i$
do
parfor i **in** $1 : N$
$\hat{\boldsymbol{x}}_i^k = \arg\min_{\boldsymbol{x}_i \in \mathcal{X}_i} L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1})$
Set $\boldsymbol{h}_i^k = \hat{\boldsymbol{x}}_i^k - \hat{\boldsymbol{x}}_i^{k-1}$
Update λ_i^k
end parfor
$k := k + 1$
until $\|f(\boldsymbol{x}^k) - f(\boldsymbol{x}^{k-1})\| \leq \delta$

Table 2.1: Algorithm for Parallel Distributed Optimization

2.4.1 Discussion on the Convergence

In this section, we show that the algorithm described in the previous subsection converges to an optimum solution. Assume that the function $f(\boldsymbol{x})$ is convex. Since the augmented function $L_i^k(\cdot)$, $i = 1, \ldots, N$ is the sum of two convex functions, it is convex. We then have

$$\hat{\boldsymbol{x}}_i^k \quad = \quad \arg\min_{\boldsymbol{x}_i \in \mathcal{X}_i} L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1}). \tag{2.7}$$

Since $\hat{\boldsymbol{x}}_i^k$ is a minimizer of $L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1})$, we have by the first order necessary conditions for local optimum that

$$\begin{aligned}
& \boldsymbol{\nabla}_i L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1})\Big|_{\boldsymbol{x}_i = \hat{\boldsymbol{x}}_i^k} = 0, \\
& \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1}) + 2\lambda_i^k(\boldsymbol{h}_i^{k-1}) \underbrace{(\hat{\boldsymbol{x}}_i^k - \hat{\boldsymbol{x}}_i^{k-1})}_{\boldsymbol{h}_i^k} = 0, \\
\Rightarrow \quad & \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1}) = -2\lambda_i^k(\boldsymbol{h}_i^{k-1})\boldsymbol{h}_i^k, \\
\Rightarrow \quad & \boldsymbol{h}_i^k = \frac{-\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1})}{2\lambda_i^k(\boldsymbol{h}_i^{k-1})},
\end{aligned} \tag{2.8}$$

where the operator $\boldsymbol{\nabla}_i$ is a gradient operator with respect to $\boldsymbol{x}_i$. For $k > K$ we have $\lambda_i^k(\boldsymbol{h}_i^{k-1}) = \alpha k$, and therefore Eq. (2.8) simplifies as

$$\boldsymbol{h}_i^k = \frac{1}{2\alpha k} \underbrace{\left(-\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1})\right)}_{\boldsymbol{d}_i^k}, \tag{2.9}$$

where $\boldsymbol{d}_i^k$ is the negative gradient direction of the i^{th} agent. By concatenating all the directions into a single vector $\boldsymbol{d}^k = [\boldsymbol{d}_1^k, \boldsymbol{d}_2^k, \ldots, \boldsymbol{d}_N^k]$, we get the next iterate $\boldsymbol{x}^k$ as

$$\hat{\boldsymbol{x}}^k \;\; = \;\; \hat{\boldsymbol{x}}^{k-1} + \boldsymbol{h}^k, \tag{2.10}$$

where $\boldsymbol{h}^k = \frac{\boldsymbol{d}^k}{2\alpha k}$. We prove the convergence properties of the algorithm using the following two prepositions.

Proposition 1: For the sequence of non-stationary iterates $\hat{\boldsymbol{x}}^k$ obtained from the PDAR algorithm, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}^k < 0$. [2]

Proof: From the definition of $\boldsymbol{d}_i^k$, we have

$$\boldsymbol{d}_i^k = -\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1}). \tag{2.11}$$

Therefore,

$$\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}^k = \sum_{i=1}^{N} -\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1}) \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1}). \tag{2.12}$$

Since $\hat{\boldsymbol{x}}_i^k$ is a result of minimizing $L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1})$, the corresponding step $\boldsymbol{h}_i^k$ must be in a descending direction. Thus

$$\boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k = \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k \leq 0, \quad \forall\, i \tag{2.13}$$

However, there must exist at least one block where the strict inequality $\boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})'\boldsymbol{h}_i^k < 0$ holds. We prove this by contradiction. Assume that $\forall i$, $\boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})'\boldsymbol{h}_i^k = 0$. If $\boldsymbol{h}_i^k =$

[2] For brevity, if all the blocks in the function are from the same iteration, we will simplify the notation, i.e., $f(\hat{\boldsymbol{x}}_i^{k-1}, \hat{\boldsymbol{x}}_{-i}^{k-1}) = f(\hat{\boldsymbol{x}}^{k-1})$

$0, \forall i$, then $\hat{\boldsymbol{x}}^k$ is a stationary point which contradicts the assumption of convergence to a nonstationary point. Hence there exists some i, for which $\boldsymbol{h}_i^k \neq 0$. Now, since $L_i^k(\boldsymbol{x}^k; \hat{\boldsymbol{x}}^{k-1})$ is a convex function, it must lie above all of its tangents, i.e.,

$$L_i^k(\hat{\boldsymbol{x}}_i^k; \hat{\boldsymbol{x}}_{-i}^{k-1}) \geq L_i^k(\hat{\boldsymbol{x}}^{k-1}) + \boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{h}_i^k. \tag{2.14}$$

Since $\boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{h}_i^k = \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{h}_i^k = 0$, we have from Eq. (2.14), that $L_i^k(\hat{\boldsymbol{x}}^k) \geq L_i^k(\hat{\boldsymbol{x}}^{k-1})$. This is a contradiction, since every iterate should reduce the objective function corresponding to the block. Intuitively, this inequality implies that if the step size is perpendicular to the gradient of the objective function, then such steps do not decrease the value of the objective function. Hence there exists at least one block that satisfies inequality $\boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})' \boldsymbol{h}_i^k < 0$. Finally, since at least one block satisfies the strict inequality, their summation satisfies strict inequality:

$$\begin{aligned} & \sum_{i=1}^{N} \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^{k-1})' \boldsymbol{h}_i^k < 0, \\ \Rightarrow \quad & \sum_{i=1}^{N} \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1}) < 0, \\ \Rightarrow \quad & \boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{d}^k < 0. \end{aligned}$$

Proposition 2: Assume that f gradients to be uniformly continuous in the ℓ_2 norm, and that its gradients are bounded. The sequence $\hat{\boldsymbol{x}}^k$ converges to an optimal solution.
Proof: Formally, we need to show that for any subsequence $\{\hat{\boldsymbol{x}}^k\}$ that converges to a nonstationary point, the corresponding subsequence $\{\boldsymbol{d}^k\}$ is bounded and satisfies [16]:

$$\lim_{k \to \infty} \sup\nolimits_{k \in \mathcal{K}} \boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{d}(\hat{\boldsymbol{x}}^k) < 0, \tag{2.15}$$

where $\boldsymbol{d}(\hat{\boldsymbol{x}}^k) = -\sum_{i=1}^{N} \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^{k-1})$. Let $\epsilon > 0$, and $\{\boldsymbol{x}^k\}_{k \in \mathcal{K}}$ be an arbitrary sequence of nonstationary points such that

$$\lim_{k \to \infty} \sup\nolimits_{k \in \mathcal{K}} \hat{\boldsymbol{x}}^k = \bar{\boldsymbol{x}},$$

where $\boldsymbol{\nabla} f(\bar{\boldsymbol{x}}) \neq 0$. Then $\forall k \in \mathcal{K}$ the gradients are not equal to zero, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^k) \neq 0$, since the sequence has nonstationary points. Using Proposition 1, we have that $\forall k \in \mathcal{K}$, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})' \boldsymbol{d}(\boldsymbol{x}^k) < 0$, and specifically $\boldsymbol{\nabla} f(\bar{\boldsymbol{x}})' \boldsymbol{d}(\bar{\boldsymbol{x}}) = D_1 < 0$.

By the continuity assumption of the gradients, there $\exists\ \delta > 0$ such that $\|\nabla f(\boldsymbol{y})'\boldsymbol{d}(\boldsymbol{y}) - \nabla f(\bar{\boldsymbol{x}})'\boldsymbol{d}(\bar{\boldsymbol{x}})\| < \epsilon, \forall\ \|\boldsymbol{y} - \bar{\boldsymbol{x}}\| < \delta$. Since $\boldsymbol{x}^k \to \bar{\boldsymbol{x}}$, $\exists\ N\ \in \mathbb{N}$ such that $\forall k > N,\ \|\boldsymbol{x}^k - \bar{\boldsymbol{x}}\| < \delta$, and thus

$$\|\nabla f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) - \nabla f(\bar{\boldsymbol{x}})'\boldsymbol{d}(\bar{\boldsymbol{x}})\| < \epsilon.$$

This implies that $\nabla f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) < D_1 + \epsilon$. As $\epsilon > 0$ is arbitrary, $\lim_{k\to\infty} \sup_{k\in\mathcal{K}} \nabla f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) = D_1 < 0$. Hence the sequence of iterates $\boldsymbol{x}^k$ converges to an optimal solution.

2.5 Numerical Results

In this section, we provide numerical results to compare the convergence of the proposed distributed algorithm to those of the block coordinate descent (BCD) and parallel variable distribution (PVD) . We consider a three-bin resource allocation example for the numerical simulation. Let there be $N = 100$ agents. Each agent has fixed quantity of resources that are to be allocated among three bins. Let $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, x_{i,3}]'$ denote the allocation scheme of the i^{th} agent. Without loss of generality, let $\sum_{j=1}^{3} x_{i,j} = 1,\ \forall\ i$. The objective is to minimize the sum of the individual costs, where the cost of agents depends on their own scheme and the schemes of other agents.

Let $\boldsymbol{x} = [\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_N']'$ denote the collective scheme of all agents. The cost function of the i^{th} agent is taken as

$$f_i(\boldsymbol{x}) = \boldsymbol{x}_i' \boldsymbol{P}_i \boldsymbol{g}(\boldsymbol{x}), \tag{2.16}$$

where $\boldsymbol{P}_i = \text{diag}(p_{i,1}, p_{i,2}, p_{i,2})$ denotes the preference matrix of the i^{th} agent for each bin, and $\boldsymbol{g}(\boldsymbol{x}) = [g_1, g_2, g_3]'$ is a function dependent on the schemes of all agents, with

$$g_m = \left(\sum_{i=1}^{N} x_{i,m}\right)^2, \quad m \in \{1, 2, 3\}. \tag{2.17}$$

The goal is to solve the optimization problem:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{N} f_i(\boldsymbol{x}) \ \text{ subject to } \sum_{j=1}^{3} x_{i,j} = 1, \forall\ i. \tag{2.18}$$

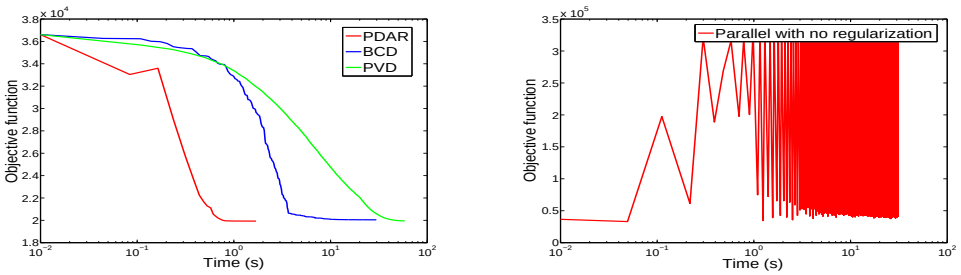
In order to find the solution to the above joint optimization problem, we solved $N = 100$ subproblems in parallel using our proposed PDAR. The optimization problem of the i^{th} agent in the k^{th} iteration is given as

$$\begin{aligned} \min_{\boldsymbol{x}_i} \quad & f_i(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_{-i}^{k-1}) + \lambda_i^k(\boldsymbol{h}_i^{k-1})\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i^{k-1}\|^2 \\ \text{subject to} \quad & \sum_{j=1}^{3} x_{i,j} = 1. \end{aligned} \tag{2.19}$$

In Fig. 2.2(a), we plot the value of the objective function as a function of the "normalized time" for BCD, PVD and our PDAR approach. We say "normalized time" to note the runtime of the parallel algorithms if we were not limited by the number of cores. In our example, we ran all the simulations on a 4 core machine; however in principle the parallel methods can run on 100 cores simultaneously. In order to make the comparison computer independent, the time axis corresponding to parallel methods was divided by 25. As illustrated, the convergence rate of our method is of an order of magnitude faster compared to BCD and PVD algorithms. The advantage comes from the fact that we can solve all the 100 optimization problems in parallel, whereas BCD is a sequential method. The PVD method, on the other hand, is worse even though it has a parallel update step. The additional time it takes to converge is due to the synchronization step, and due to the complexity of the optimization problems that are to be solved in both steps. In Fig. 2.2(b), we show the oscillatory behavior when the parallel algorithm is used with out a regularizer. This figure further emphasizes the importance of a regularizer.

2.6 Conclusions

In this chapter, we proposed a distributed optimization framework to solve large optimization problems with separable constraints. Each agent solves a local optimization problem, which is much simpler compared to the joint optimization. In order for the agents to coordinate among themselves and to reach an optimum solution, we introduced a regularization term that penalized the changes in the successive iterations with an adaptive regularization



(a) (b)

Figure 2.2: Value of the objective functions vs time for the three bin resource allocation problem. Fig. 2.2(a) shows that PDAR converges much faster compared to BCD and PVD. Fig. 2.2(b) shows the oscillatory behavior of the parallel optimization without regularization.

coefficient. We proved that our solution always converges to a local optimum, and to a global optimum if the overall objective function is convex. Numerical simulations showed that the solutions reached by our algorithm are the same as the ones obtained using other distributed approaches, with significantly reduced computation time.

Chapter 3

Estimating Electrical Conductivity Tensors of Biological Tissues from Microelectrode Arrays Data

3.1 Abstract

Finding the electrical conductivity of tissue is highly important for understanding the tissue's structure and functioning. However, the inverse problem of inferring spatial conductivity from data is highly ill-posed and computationally intensive. In this chapter, we propose a novel method to solve the inverse problem of inferring tissue conductivity from a set of transmembrane potential and stimuli measurements made by microelectrode arrays (MEA). We first formalize the discrete forward model of transmembrane potential propagation, based on a reaction-diffusion model with an anisotropic inhomogeneous electrical conductivity-tensor field. Then, we use our novel parallel optimization algorithm from Chapter 2for solving the complex inverse problem of estimating the electrical conductivity-tensor field. Specifically, we propose a single-step approximation with a parallel block-relaxation optimization routine that simplifies the joint tensor field estimation problem into a set of computationally tractable subproblems, allowing the use of efficient standard optimization tools. Finally, using numerical examples of several electrical conductivity field topologies and noise levels, we analyze the performance of our algorithm, and discuss its application to real measurements obtained from smooth-muscle cardiac tissue, using data collected with a high-resolution MEA system.

3.2 Introduction

Transmembrane potential propagation in biological tissue results when the ionic concentrations change in either the intracellular or extracellular domains. Potential propagation is correlated to the medium's conductivity, and as a mechanism of intercellular communication it plays an important role in tissue and organ functioning, e.g., exocytosis and muscle contractions [77]. However, in order to relate transmembrane potential measurements to the electrophysiological states of cells in the tissue, we depend on mathematical and computational models to capture the interaction. A classical approach to modeling spatiotemporal transmembrane potential propagation is based on the generalized cable theory, combined with dynamic models of ionic concentration gradients. This parametric model relates changes in the transmembrane potential to changes in ionic currents through the membrane, taking into account the effective electrical conductivity and geometry of the tissue. The bidomain model treats the tissue as two continuous domains, and is a macroscale model of the electrical behavior averaged over many cells, taking into account both the intracellular and extracellular current flows. Although this model has been used extensively in numerical simulations of the electrical behavior of anisotropic myocardiac tissues [111], in neuroscience it has recently been used for analyzing the non-homogeneity of the extracellular domain in nerve tissues [12, 13].

In the last few years, much progress has been made in developing high-resolution microelectrode arrays (MEA) that allow electrophysiological measurements of biological tissues with high spatiotemporal resolution [14,68]. Using this technology, it is possible to effectively and directly measure transmembrane potential propagation by parallel measurements of the tissue at different locations. By employing the biodomain model to analyze the MEA data, we gain a deeper understanding of the underlying biophysical nature of tissue. However, several model parameters must first be estimated, including ones related to the ionic current dynamics, cell geometry, and electrical tissue conductivities. Several approaches are available to model and estimate cellular ionic currents, and the interested reader is referred to the technical literature for details [35, 65, 154]. However, inferring the conductivity parameters from the data by inverse techniques is a highly challenging problem [55]. The estimation problems becomes even more significant when the tissue is assumed to be composed of multiple anisotropic inhomogeneous regions.

In this chapter, we develop a mathematical framework for solving the inverse problem of estimating the effective electrical tissue conductivities from a set of electric potentials and stimulus measurements. In particular, we formulate the problem in a system identification framework, using a parametric model based on the generalized cable theory. In this framework, experiments are performed by exciting the system and observing its input/output over a time interval [138]. Unfortunately, solving this ill-posed inverse problem is highly complex [141]. Specifically, it suffers from *high dimensionality* (as one must estimate the tensor matrix for each point in space), *nonlinearity* (due to the nonlinear extracellular field potential dynamics), and *stochasticity* (as the observations are corrupted by noise). The application of sophisticated methods, such as nonlinear filters (e.g., particle and Unscented Kalman filters [133]) or traditional constraint optimizations (e.g., augmented Lagrangian methods [103]), becomes computationally prohibitive due to the complexity of the estimation problem, especially when a high resolution grid is considered.

The contributions of this work are two-fold. First, we introduce a discrete forward model of transmembrane potential based on a diffusion-reaction model with an anisotropic inhomogeneous electrical conductivity tensor field. Second, we propose a novel parallel optimization algorithm for solving the complex inverse problem of estimating the conductivity tensor field. Specifically, we propose a single-step approximation with a parallel block-relaxation optimization method. This combination simplifies the joint tensor field estimation problem into a set of computationally tractable problems, allowing the use of efficient standard optimization algorithms. We analyze the performance of our algorithm using numerical examples of several electrical conductivity field topologies and noise levels, and discuss its application to real measurements obtained from cardiac tissue, using a high resolution MEA system.

The notational conventions adopted in this work are as follows: Italic font indicates a scalar quantity, e.g., a; lowercase boldface indicates a vector quantity, e.g., $\boldsymbol{a}$; upper case italic bold indicates a matrix quantity, e.g., $\boldsymbol{A}$. The matrix transpose is indicated by a superscript "T" as in $\boldsymbol{A}^T$, and the identity matrix of size $n \times n$ is denoted as $\boldsymbol{I}_n$. A lowercase Roman font indicates a function, e.g., $\mathrm{g}(t)$, and a lowercase bold Roman font indicates a vector function, e.g., $\mathbf{g}_t$. The set $\mathbb{S}_n$ denotes the vector space of symmetric $n \times n$ matrices, and the subsets of nonnegative definite matrices and positive definite matrices are denoted by $\mathbb{S}_{n+}$ and $\mathbb{S}_{n++}$, respectively. $||\cdot||_{\mathrm{F}}$ is the Frobenius norm, defined as $||\boldsymbol{A}||_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$. Writing

the time index as a subscript indicates the vector at the nth timepoint, e.g., $\boldsymbol{a}_n \equiv \boldsymbol{a}[n\Delta t]$, or indicates vectors from a set of timepoints, e.g., $\boldsymbol{a}_{0:n} \equiv \{\boldsymbol{a}_0, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_n\}$.

The remaining part of this work is organized as follows: In Section 3.3, we present the numerical scheme used to discretize the forward problem. Then we formulate the inverse problem in a discrete setting and propose a method to solve the complex inverse problem. Finally, in Section 3.4 we present results of the proposed method for both simulated and real data.

3.3 Materials and Methods

3.3.1 Forward Model

To model electrical propagation in biological tissue composed of elements of different conductivities, we use generalized cable theory, namely, the monodomain approach [125]. The monodomain is a specific case of the bidomain models which have been successfully used for modeling extracellular potentials in several tissues [97, 98, 107, 146]. In the monodomain model, biological tissue is reduced to a two- or three-dimensional cell grid, where the electrical behavior is governed by a set of reaction-diffusion equations [111]. The diffusion part of this model represents the spatial evolution of the transmembrane potential in a domain with changing conductivities. The reaction part models the voltage-dependent dynamics of the tissue as a function of three local currents: (i) a capacitative current through the cells' membranes, (ii) a cell-ionic current with a voltage-dependent dynamic $\mathrm{j}_{\mathrm{ion}}(v(\boldsymbol{r},t), \boldsymbol{w}, \boldsymbol{\phi}, t)$, and (iii) external or spontaneous stimulations $\mathrm{j}_{\mathrm{stim}}(\boldsymbol{r},t)$. Our system can be written as follows:

$$\underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{D}(\boldsymbol{r}) \boldsymbol{\nabla} v(\boldsymbol{r},t)}_{\text{Diffusion}} = \underbrace{a_{\mathrm{m}} \left(c_{\mathrm{m}} \frac{\partial v(\boldsymbol{r},t)}{\partial t} + \mathrm{j}_{\mathrm{ion}}(v(\boldsymbol{r},t), \boldsymbol{w}, \boldsymbol{\phi}, t) - \mathrm{j}_{\mathrm{stim}}(\boldsymbol{r},t) \right)}_{\text{Reaction}}, \tag{3.1}$$

$$v(\boldsymbol{r},0) = v_0, \tag{3.2}$$

$$\boldsymbol{\nabla} v(\boldsymbol{r},t) \cdot \boldsymbol{n}(\boldsymbol{r}) = 0, \boldsymbol{r} \in \partial C, \tag{3.3}$$

where $t \in [0, T]$, the spatial vector $\boldsymbol{r}$ belongs to $C \subseteq \mathbb{R}^p$, the domain C is a bounded Euclidean subset, $\boldsymbol{n}$ denotes the normal to the boundary, and ∂C is the boundary of domain C. In this work we consider the 2D case, where $p = 2$. Furthermore, $\mathrm{j}_{\mathrm{stim}}(\boldsymbol{r}, t)$ is the stimulus volume current density ($\mathrm{A/m^3}$); c_{m} is the membrane capacitance per unit area ($\mathrm{F/m^2}$); a_{m} is the surface-to-volume ratio of the membrane ($1/\mathrm{m}$); and $\boldsymbol{D}(\boldsymbol{r}) \in \mathbb{S}_{2++}$ is the positive definite conductivity tensor [10]. $\mathrm{j}_{\mathrm{ion}}(v(\boldsymbol{r}, t), \boldsymbol{w}, \boldsymbol{\phi}, t)$ is the ionic volume current density ($\mathrm{A/m^3}$) of a biological cell, and it can be chosen to fit a specific dynamic, with $\boldsymbol{w}$ corresponding to the internal state vector, and $\boldsymbol{\phi}$ to the model parameters. For simplicity, we consider a homogenous cell dynamic in the tissue; namely, $\boldsymbol{\phi}$ is consistent in all the cells (in the results section, we used the extended FitzHugh-Nagumo (FHN) equations [111, 124] as a fairly general and simple representation of a cell's ionic currents; however, more extensive models can be used). Eqs. (3.2) and (3.3) present the initial temporal and boundary conditions, respectively. In particular, we use the homogenous Neumann boundary condition since we assume that there will be no current through the borders of the domain, and we use the *zero state response* [117] for the initial values since we assume that the system is initially relaxed at its resting potential v_0 and cannot initiate a spontaneous response.

3.3.2 Modeling Tissue Anisotropy

In order to infer the underlying conductivity structure of the tissue, we represent biological tissue as a continuous field of conductivity tensors, $\boldsymbol{D}(\boldsymbol{r})$ in Eq. (3.1), which models local extracellular conductivities within the tissue [125]. To simplify the problem, we consider working with only a thin slice of tissue that can be represented as a 2D plane. The conductivity tensor is given by

$$\boldsymbol{D}(x, y) = \begin{bmatrix} \sigma_{\mathrm{x}}(x, y) & \sigma_{\mathrm{xy}}(x, y) \\ \sigma_{\mathrm{xy}}(x, y) & \sigma_{\mathrm{y}}(x, y) \end{bmatrix}, \tag{3.4}$$

where $\sigma_{\mathrm{x}}(x, y), \sigma_{\mathrm{xy}}(x, y), \sigma_{\mathrm{y}}(x, y)$ are the conductivity values in the horizontal, diagonal, and vertical directions, respectively. The tensor field is an indexed set of tensors in space, and is referred to as *isotropic* if all the conductivity tensors are directionally independent (symmetric), that is, $\sigma_{\mathrm{x}}(x, y) = \sigma_{\mathrm{y}}(x, y) = \sigma_0$ and $\sigma_{\mathrm{xy}}(x, y) = 0$ for all x, y. Otherwise, if some conductivity tensors in the field are directionally dependent, it is referred to as *anisotropic*.

If the conductivity tensors are constant throughout the field, then the field is referred to as *homogeneous*; otherwise, it is *inhomogeneous*.

A diffusion tensor can be expressed in terms of its eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ and eigenvectors $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ as $\boldsymbol{D} = \boldsymbol{E}^T \text{diag}(\boldsymbol{\lambda})\boldsymbol{E}$. The tensor can be represented as an ellipsoid (Fig. 3.1), whose radii (eigenvalues) represent the amount of diffusion (flow) in each of the main directions (eigenvectors) [2].

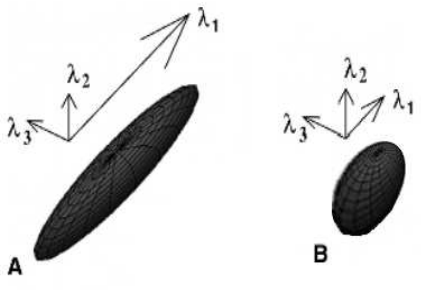
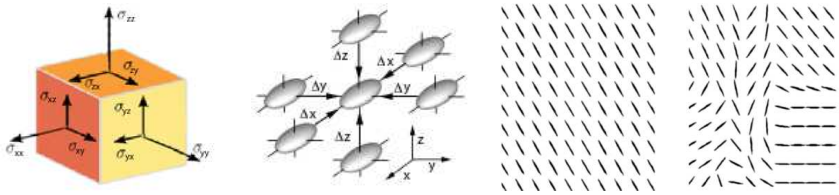


Figure 3.1: Diffusion tensors shown as ellipsoids with their corresponding eigenvectors [59].

In Fig. 3.2, we can see the general representation of a tensor and a tensor field in 3D, and examples of homogeneous and inhomogeneous anisotropic 2D tensor fields. The anisotropy inhomogeneity property of the conductivity tensors plays a major role in the spatial evolution of the wave propagation in the tissue [23]. In this work, we will focus on developing a mathematical formulation for finding the best tensor field representation to fit the electrical measurements.



(a) (b) (c) (d)

Figure 3.2: Fig. 3.2(a) depicts a general tensor in 3D space, while Fig. 3.2(b) illustrates a 3D spatially-discrete tensor field, where each tensor is represented by an ellipsoid. Figs. 3.2(c) and 3.2(d), illustrate 2D constant anisotropic-homogeneous and anisotropic-inhomogeneous 2D fields, respectively.

3.3.3 Discretization

To transfer our model from a continuous domain into a discrete vector space formulation, we will first proceed to discretize the continuous diffusion term of Eq. (3.1), which is given as

$$\boldsymbol{\nabla} \cdot \boldsymbol{D}(x,y)\boldsymbol{\nabla} v(x,y,t) = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}^T \begin{bmatrix} \sigma_{\mathrm{x}}(x,y) & \sigma_{\mathrm{xy}}(x,y) \\ \sigma_{\mathrm{xy}}(x,y) & \sigma_{\mathrm{y}}(x,y) \end{bmatrix} \begin{bmatrix} \frac{\partial v(x,y,t)}{\partial x} \\ \frac{\partial v(x,y,t)}{\partial y} \end{bmatrix}, \tag{3.5}$$

to a discrete term over the regular lattice domain $C' = [x_1, x_K] \times [y_1, y_K]$.

In particular, we apply the Finite-Difference Method (FDM) with an forward-time central-space scheme [140] to approximate the derivatives, taking into account the spatially varying tensor field of Eq. (3.5). In the following, we present the two main steps for discretization of Eq. (3.5), but for brevity we will ignore the temporal component of the extracellular voltage $v(x,y,t)$. First, we expand the matrix multiplication of $\boldsymbol{\nabla v}(x,y)$ with $\boldsymbol{D}(x,y)$, and are left with three second derivatives, which correspond to current densities: j_{x}, j_{xy}, and j_{y}. Therefore, Eq. (3.5) can be written as

$$\begin{aligned} \boldsymbol{\nabla} \cdot \boldsymbol{D}(x,y)\boldsymbol{\nabla} v(x,y) \;=\; & \underbrace{\frac{\partial \left(\sigma_{\mathrm{x}}(x,y)\frac{\partial v(x,y)}{\partial x}\right)}{\partial x}}_{j_{\mathrm{x}}} + \underbrace{\frac{\partial \left(\sigma_{\mathrm{y}}(x,y)\frac{\partial v(x,y)}{\partial y}\right)}{\partial y}}_{j_{\mathrm{y}}} \\ & \underbrace{+ \frac{\partial \left(\sigma_{\mathrm{xy}}(x,y)\frac{\partial v(x,y)}{\partial x}\right)}{\partial y} + \frac{\partial \left(\sigma_{\mathrm{xy}}(x,y)\frac{\partial v(x,y)}{\partial y}\right)}{\partial x}}_{j_{\mathrm{xy}}}. \end{aligned} \tag{3.6}$$

Second, each term of Eq. (3.6) is further expanded using a central-space difference FDM scheme. For example, j_{x} can be expanded as

$$
\begin{aligned}
j_{\mathrm{x}}(\sigma_{\mathrm{x}}, \sigma_{\mathrm{xy}}, \sigma_{\mathrm{y}}) &= \left(\left[\frac{\sigma_{\mathrm{x}}(x+\Delta x, y) - \sigma_{\mathrm{x}}(x-\Delta x, y)}{2\Delta x} \right] \left[\frac{v(x+\Delta x, y) - v(x-\Delta x, y)}{2\Delta x} \right] \right) \\
&\quad + \sigma_{\mathrm{x}}(x, y) \left(\frac{v(x+\Delta x, y) + v(x-\Delta x, y) - 2v(x, y)}{\Delta x^2} \right). \qquad (3.7)
\end{aligned}
$$

The term $j_{\mathrm{x}}(\sigma_{\mathrm{x}}, \sigma_{\mathrm{xy}}, \sigma_{\mathrm{y}})$ represents the current density along the x direction at a single point on the grid (x, y). Concatenating all points into a column stack vector, we can write Eq. (3.7) as

$$
\boldsymbol{j}_{\mathrm{x}}(\boldsymbol{\Theta}) = \boldsymbol{G}_{\mathrm{x}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{x}}) \boldsymbol{G}_{\mathrm{x}} \boldsymbol{v} + \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{x}}) \boldsymbol{G}_{\mathrm{xx}} \boldsymbol{v}, \qquad (3.8)
$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \ldots, \boldsymbol{\Theta}_K\}$ is a joint set of K tensors of the form $\boldsymbol{\Theta}_k = \begin{bmatrix} \sigma_{\mathrm{x}k} & \sigma_{\mathrm{xy}k} \\ \sigma_{\mathrm{xy}k} & \sigma_{\mathrm{y}k} \end{bmatrix}$ at location k. We will refer to this set $\boldsymbol{\Theta}$ as the discrete conductivity-tensor field. The matrices $\boldsymbol{G}_{\mathrm{x}} \in \mathbb{R}^{K \times K}$ and $\boldsymbol{G}_{\mathrm{xx}} \in \mathbb{R}^{K \times K}$ (where K is the number of nodes) are the linear operators of the first and second discrete spatial derivative operators in Eq. (3.7) [140]. The vectors $\boldsymbol{v}$ and $\boldsymbol{\sigma}_{\mathrm{x}}$ are column stack representations of the two dimensional potential function $v(x, y)$ and tensor function $\sigma_{\mathrm{x}}(x, y)$, respectively, at time t [26]. We derive $\boldsymbol{j}_{\mathrm{xy}}$ and $\boldsymbol{j}_{\mathrm{y}}$ similarly. Noting the dependence in t, we can write the three current densities as

$$
\begin{aligned}
\boldsymbol{j}_{\mathrm{x}t}(\boldsymbol{\Theta}) &= \boldsymbol{G}_{\mathrm{x}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{x}}) \boldsymbol{G}_{\mathrm{x}} \boldsymbol{v}_{\mathrm{t}} + \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{x}}) \boldsymbol{G}_{\mathrm{xx}} \boldsymbol{v}_{\mathrm{t}}, \\
\boldsymbol{j}_{\mathrm{xy}_t}(\boldsymbol{\Theta}) &= 2\boldsymbol{G}_{\mathrm{xy}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{xy}}) \boldsymbol{v}_{\mathrm{t}} + \boldsymbol{G}_{\mathrm{x}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{xy}}) \boldsymbol{G}_{\mathrm{y}} \boldsymbol{v}_{\mathrm{t}} + \boldsymbol{G}_{\mathrm{y}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{xy}}) \boldsymbol{G}_{\mathrm{x}} \boldsymbol{v}_{\mathrm{t}}, \\
\boldsymbol{j}_{\mathrm{y}_t}(\boldsymbol{\Theta}) &= \boldsymbol{G}_{\mathrm{y}} \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{y}}) \boldsymbol{G}_{\mathrm{y}} \boldsymbol{v}_{\mathrm{t}} + \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{y}}) \boldsymbol{G}_{\mathrm{yy}} \boldsymbol{v}_{\mathrm{t}}. \qquad (3.9)
\end{aligned}
$$

Next, using the forward-time central-space FDM scheme, we discretize the right side of Eq. (3.1). We obtain the following discrete form of Eq. (3.1):

$$
\boldsymbol{j}_{\mathrm{x}n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{xy}_n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{y}_n}(\boldsymbol{\Theta}) = a_{\mathrm{m}} \left(c_{\mathrm{m}} \frac{\boldsymbol{v}_{n+1} - \boldsymbol{v}_n}{\Delta t} + \boldsymbol{g}\left(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}\right) + \boldsymbol{u}_n \right). \qquad (3.10)
$$

Finally, separating $\boldsymbol{v}_{n+1}$ to left side of the equation and discretizing the boundary and initial value conditions, we arrive at the following discrete representation of the system:

$$
\boldsymbol{v}_{n+1} = \frac{\Delta t}{a_{\mathrm{m}} c_{\mathrm{m}}} \left(\boldsymbol{j}_{\mathrm{x}n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{xy}_n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{y}_n}(\boldsymbol{\Theta}) \right) - \frac{\Delta t}{c_{\mathrm{m}}} \left(\boldsymbol{g}\left(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}\right) + \boldsymbol{u}_n \right) + \boldsymbol{v}_n. \qquad (3.11)
$$

where the external stimulus input $\mathrm{j}_{\mathrm{stim}}(\boldsymbol{r}, n)$ is written as the vector $\boldsymbol{u}_n$. Further, $\boldsymbol{g}\left(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}\right)$ is a nonlinear function that depends on the cell dynamic model, model parameters $\boldsymbol{\phi}$, previous states $\boldsymbol{v}_{0:n}$, and previous internal states $\boldsymbol{w}_{0:n}$. For full derivation on state discretization in for the FHN model, please refer to the supplemental materials. To completely define the discrete system, we add the following discretized initial and Neumann boundary conditions: initially in homogenous resting potential $\boldsymbol{v}_0 = v_0\mathbf{1}$, no current through the boundaries $\boldsymbol{G}_{\mathrm{x}}\boldsymbol{v}_n = 0$, $\boldsymbol{G}_{\mathrm{y}}\boldsymbol{v}_n = 0$ for horizontal, vertical boundaries, respectively.

Note that we can write our discrete system as

$$\boldsymbol{v}_{n+1} = \boldsymbol{A}(\Delta t, a_{\mathrm{m}}, c_{\mathrm{m}}, \Delta x, \Delta y, \boldsymbol{\Theta})\boldsymbol{v}_n + \boldsymbol{g}\left(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}\right) + b\boldsymbol{u}_n, \tag{3.12}$$

where the $\boldsymbol{A}$ matrix depends on the model parameters. As can be observed from Eq. (3.12), even when we ignore the nonlinear part, the discrete system's stability is sensitive to the model parameters (a_{m} and c_{m}), the FDM parameters (Δt, Δx, and Δy), the conductive-tensor field estimation $\boldsymbol{\Theta}$, and the input values $\boldsymbol{u}_n$. Finding an analytical formulation for the stability criterion is highly challenging and will be considered in future work. However, consideration is needed when choosing parameter values, and in some cases of instability, more complex numerical approximation methods should be considered [140]. Furthermore, it is beneficial to use a numerical optimization method that can recover from instability, e.g., the standard Matlab implementation of the sequential quadratic programming method described in [104].

3.3.4 Measurement Model

In a highly dense sensor system such as an MEA, the electrodes capture the extracellular field potential, which is the product of a highly complex network of neighboring cells, making interpretation very difficult. The complexity of the system is greatly simplified when using the monodomain model, as it is a macroscopic model that describes the average behavior of the bioelectric fields on a larger scale than the size of a single cell [129]. Hence, the state $\boldsymbol{v}_n$ in our model represents the average local extracellular field, which is measured through a noisy sensor to give spatiotemporal measurements $\boldsymbol{y}_n \in \mathbb{R}^Q$. Here we will assume that $\boldsymbol{y}_n$ are corrupted with an additive white Gaussian noise of mean zero and variance γ^2, and

that the wavelengths of the dynamics are large compared to the spatial density of the MEA. Therefore, the discretized evolution and measurement models are given by

$$\boldsymbol{v}_{n+1} = \mathbf{f}(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}, \boldsymbol{\Theta}) + b\boldsymbol{u}_n, \tag{3.13}$$

$$\boldsymbol{y}_n = \boldsymbol{v}_n + \boldsymbol{\nu}. \tag{3.14}$$

Note that in this work we set $Q = K$ as we assume that the MEA is dense enough to allow complete coverage of the region in interest (ROI); however, in cases where the system exhibits subsampling, Q will be less than K.

3.3.5 Solving the Inverse Problem

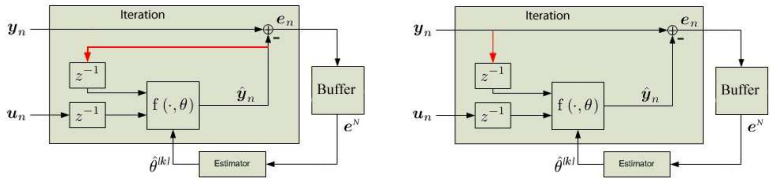
To restate the inverse problem, the goal is to estimate $\boldsymbol{\Theta}$, the discretized conductivity tensor field, from the set of measurements $\boldsymbol{y}_{0:T}$, the set of inputs $\boldsymbol{u}_{0:T}$, and the discrete system model of Eqs. (3.13) (3.14). We use an indirect method to estimate $\boldsymbol{\Theta}$, by transforming the problem into an optimization problem [141]. One indirect approach to solve this problem is using a constraint optimization scheme, specifically, the augmented Lagrangian method. This method has been used successfully in the field of geoscience for solving problems of permeability identification in convection-diffusion models [102, 103]. In the augmented Lagrangian method, the problem is written as a joint optimization routine, where the error between the model and observations is minimized (usually in a least squares sense), with the model parameters acting as constraints. However, this method can easily become computationally prohibitive. Even for efficient algorithms, one must alternate between optimization steps with high dimensionality: the number of nodes in the grid $\mathcal{O}(K)$, and the number of grid points times the number of time-steps $\mathcal{O}(KT)$. These optimizations will quickly become intractable as either the number of grid points or number of samples increases.

Another optimization approach that can be considered is the initial value approach. This method uses the fact that the model evolution function in Eq. (3.13) is deterministic, depending entirely on the initial values of the states and the known control sequence. An intuitive approach is to minimize the least-squares (LS) difference between the observations and the model trajectory by altering only the initial values $\boldsymbol{v}_0$ and $\boldsymbol{w}_0$. This Initial Value

Approach (IVA) objective function can be written as

$$\mathrm{V}_{\mathrm{IVA}}\left(\boldsymbol{v}_0, \boldsymbol{w}_0, \boldsymbol{y}^N, \boldsymbol{u}^N, \boldsymbol{\phi}, \boldsymbol{\Theta}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K}\left(\boldsymbol{y}_{n,k} - \mathbf{f}\left(\boldsymbol{v}_0, \boldsymbol{w}_0, \boldsymbol{\phi}, \boldsymbol{\Theta}\right)_{n,k} - b\boldsymbol{u}_{n,k}\right)^2, \tag{3.15}$$

where the entire deterministic trajectory of the model is calculated and compared against the measurements. This method, which has a recurrent structure as can be seen in Fig. 3.3(a), is a member of the nonlinear output error (NOE) family [135], and has the benefit of avoiding feeding the measurement noise into the nonlinear evolution function. However, the recurrence structure of the model makes it difficult to determine the stability of the solution, and increases the complexity in calculating the gradients for the model parameters estimation [135]. Furthermore, the solution is very susceptible to noise and depends sensitively on initial conditions and parameters, which makes the initial value approach inapplicable unless chosen in very close proximity to the unknown true values [134].



(a) (b)

Figure 3.3: illustrates the iterative estimation process using the discrete system model and a feedback loop. The model parameters $\boldsymbol{\theta}^{[k]}$ are estimated from the error vector $\boldsymbol{e}^N$ (difference between the measurements and the model outputs) of the previous iteration. Fig. 3.3(a), illustrates a nonlinear output error (NOE) procedure with a feedback loop on the output of the model, which makes it a recurrence structure. Fig. 3.3(b), illustrates a nonlinear autoregressive exogenous (NARX) procedure with a feedforward assembly, feeding the measurements of the model.

Rather than using such a recurrence approach, another method that shows more promising results is a recursive state space technique that evaluates the cost function in a sequential way, using the observations [134]. Filtering is used to approximate the system state variables from the noisy observations. However, traditional filtering methods such as Kalman or Extended Kalman will not be sufficient to deal with the complex nonlinear dynamics of the system [69]. A particle filter will also prove problematic since even for a very modest size grid, the system will be in a high dimensional space, which makes the particle filter ineffective. Earlier work has also been done with the Unscented Kalman Filter (UKF) for nonlinear reaction-diffusion

models [134]. However, a problem with using the UKF is approximation errors in cases of strong nonlinearity, such as in a polynomial of high degree (> 2). Another difficulty is that this approach will offer computationally tractable solutions for only low dimensional systems. For systems with even a modest grid size (e.g., 9×9), the problem becomes too complex to be solved directly, and some works suggest using a patch filter as a dimension reduction scheme [133]. However, using the dimension reduction scheme in our problem might adversely reduce the spatial resolution. As the inverse problem is too complex to allow filtering, we propose a simpler, nonrecurrent method based on the ideas of one-step prediction and block-relaxation optimization.

3.3.6 One-Step-Ahead Prediction

When dealing with invasive electrophysiological recordings, it is acceptable to assume low sensor measurement noise and use the observed data as a direct measurement of the average local field potential states, $\boldsymbol{y}_n \approx \boldsymbol{v}_n$ [68, 148]. A high signal to noise ratio (SNR) is further supported when different methods of noise reduction are used, e.g., applying a smoothing filter between consecutive observations or experiments [70]. This assumption simplifies the model by substituting the observed measurements for the hidden states, and the simplified discrete system model can be written as

$$\hat{\boldsymbol{y}}_{n+1} \quad = \quad \mathbf{f}(\boldsymbol{y}_n, \boldsymbol{w}_n, \boldsymbol{\phi}, \boldsymbol{\Theta}) + b\boldsymbol{u}_n. \tag{3.16}$$

We notice from Eq. (3.16) that this method is a feedforward predictive model, and is a member of the classical nonlinear autoregressive exogenous (NARX) family, where the nonlinear model depends on past measurements (filtered), and current and past values of the input (exogenous) series [101]. However, feeding noisy measurements to the nonlinear model will affect the stability of the system, and hence robustness to noise is an important consideration in the algorithm design. A robust design can be found by using a suitable performance bound for estimated noise, and is an important subject of future research. In this work, we will discuss the effects of noise on the algorithm, and examine possible ways to improve robustness in the Results Section (3.4.4).

The one-step-ahead prediction (OSP) score function can be formulated as the LS difference between the one-step predictions at each point in time and the measurements [135], and is given by [3]

$$\mathrm{V}_{\mathrm{OSP}}\left(\boldsymbol{y}_{0:N}, \boldsymbol{w}_0, \boldsymbol{u}_{0:N}, \boldsymbol{\phi}, \boldsymbol{\Theta}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K}\left(\boldsymbol{y}_{n,k} - \mathbf{f}\left(\boldsymbol{y}_{n-1}, \boldsymbol{w}_0, \boldsymbol{\phi}, \boldsymbol{\Theta}\right)_{n,k} - b\boldsymbol{u}_{n,k}\right)^2. \quad (3.17)$$

However, the estimation of the unknown tensors $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \ldots, \boldsymbol{\Theta}_K\}$ using Eq. (3.17) is still highly complex due to its high dimensionality and the dependencies between the components of each tensor, which must be positive definite (PD), and between different tensors in the field. The interdependencies between the tensors in the field occur because every tensor can affect the overall score function, and a change in one could result in different optimal values for the others. Solving the joint complex interdependent estimation problem requires a highly complicated semi-definite programming routine, which will often be intractable, time consuming, or converge to a local optimum. In order to simplify the problem, we propose a novel method that uses parallel relaxation optimization to separate the highly complex semi-definite optimization into a set of smaller, less complex optimizations but still maintains mutual influence in order to converge to a better overall solution.

3.3.7 Parallel Block Relaxation Optimization

To develop a computationally efficient optimization algorithm to solve the complex joint nonlinear semi-definite problem, we use a variance on the common sequential block-relaxation method [32], commonly known as alternating optimization (AO) [18]. To restate our problem, the unknown parameters, on which we are trying to perform optimization, are positively defined tensor matrices with inner dependencies in their components. Performing optimization in the tensor space is a highly nontrivial task. In recent years, a few methods have been developed to perform gradient descent in tensor space using an intrinsic geodesic marching scheme with the Riemannian framework [108]. However, these methods come with a high

[3] To simplify our inference algorithm, for certain dynamic models such as the FitzHugh-Nagumo, we can use the method of variation of parameters in order to have the evolution function dependence on $\boldsymbol{w}_{0:n}$ brought down to depend only on $\boldsymbol{w}_0$ and $\boldsymbol{v}_{0:n}$ (see supplementary materials Eq. (A-4)).

computational burden in the form of intensive use of matrix inverses, square roots, logarithms; moreover, exponentials are usually involved [4]. A simpler approach will be to break down the tensors to their components and use constraints to guarantee PD; however, this will dramatically increase the dimensionality of the problem.

In order to solve this complex problem, we turned to the block-relaxation optimization class of algorithms. In this class, a complex optimization problem is solved by iteratively solving a series of easily handled subproblems. The optimization algorithm for each subproblem is simpler since only a subset of the parameters are considered, keeping the rest of the parameters constant at their current value. The algorithm then iteratively cycles through the different subproblems and updates the parameters in each subset until convergence is reached. However, even for a modest size grid, the high-dimensionality of the problem will make the alternating optimization algorithm computationally expensive because it will have to sequentially alternate between many parameter subspaces. To improve computational costs, we vary the sequential block-relaxation optimization algorithm (i.e., AO) so that instead of simply using a sequential scheme for solving the subproblems, we parallelize the algorithm, solving them all at once and joining them after each iteration.

To use parallel block-relaxation optimization, we start with the complex joint optimization problem given by

$$\hat{\boldsymbol{\Theta}}^{[j]} \leftarrow \arg\min_{\substack{\boldsymbol{\Theta}=\{\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2,\ldots,\boldsymbol{\Theta}_K\} \\ \forall k,\boldsymbol{\Theta}_k\in\mathbb{S}_{2++}}} \mathrm{V}_{\mathrm{OSP}}\left(\boldsymbol{y}_{0:N}, \boldsymbol{w}_0, \boldsymbol{u}_{0:N}, \boldsymbol{\phi}, \hat{\boldsymbol{\Theta}}^{[j-1]}, \boldsymbol{\Theta}\right), \tag{3.18}$$

and separate it into K optimization problems of the form

$$\hat{\boldsymbol{\Theta}}_k^{[j]} \leftarrow \arg\min_{\substack{\boldsymbol{\Theta}_k=\{\sigma_{\mathrm{x}k},\sigma_{\mathrm{xy}k},\sigma_{\mathrm{y}k}\} \\ \boldsymbol{\Theta}_k\in\mathbb{S}_{2++}}} \left[\mathrm{V}_{\mathrm{OSP}}\left(\boldsymbol{y}_{0:N}, \boldsymbol{w}_0, \boldsymbol{u}_{0:N}, \boldsymbol{\phi}, \hat{\boldsymbol{\Theta}}_{i\neq k}^{[j-1]}, \boldsymbol{\Theta}_k\right)\right], \tag{3.19}$$

where $\hat{\boldsymbol{\Theta}}^{[j-1]}$ are known values from the previous iteration. The arg min function of Eq. (3.19), evaluates the new value of the tensor $\hat{\boldsymbol{\Theta}}_k$ that will optimally decrease (or increase) the objective function when only that single tensor is allowed to change.

Using a parallel scheme, however, can introduce convergence problems because the separate subproblems are greedily optimized within each iteration. This can often lead to a poor convergence rate and convergence to a local optimum. To improve the convergence properties of the algorithm we use out parallel method of Chapter 2. We introduce an additional term

to the arg min in Eq. (3.19), penalizing large changes of the parameters between consecutive iteration steps. The step-size penalty feature is added to the algorithm to reduce the probability of the algorithm becoming stuck at a local minimum that is worse than the global one. This feature is very similar to the idea behind the acceptance probability in Monte-Carlo methods, such as simulated annealing, where the acceptance probability function is usually chosen so that the probability of accepting a move decreases when it causes a large change, thus making small uphill moves more likely than large ones [116].[4] Further, having this penalty term in the optimization score function guarantees that the tensors will remain PD. In this work, we use a standard Frobenius norm as the distance function between tensors; however, there are other distance measures, e.g., Riemannian Metrics or log-Euclidean metrics, that offer a more rigorous and general framework for handling tensor operations, but they are more computationally intensive [4, 10].

The K optimization routines can then be written as

$$\hat{\boldsymbol{\Theta}}_k^{[j]} \leftarrow \arg\min_{\substack{\boldsymbol{\Theta}_k=\{\sigma_{\mathrm{x}k},\sigma_{\mathrm{xy}k},\sigma_{\mathrm{y}k}\} \\ \boldsymbol{\Theta}_k\in\mathbb{S}_{2++}}} \left[\mathrm{V}_{\mathrm{OSP}}\left(\boldsymbol{y}_{0:N}, \boldsymbol{w}_0, \boldsymbol{u}_{0:N}, \boldsymbol{\phi}, \hat{\boldsymbol{\Theta}}_{i\neq k}^{[j-1]}, \boldsymbol{\Theta}_k\right) + \lambda \mathrm{d}\left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}_k^{[j-1]}\right)\right], \tag{3.20}$$

where λ is the Lagrangian hyper-parameter. The joint parameter set at iteration j, $\hat{\boldsymbol{\Theta}}^{[j]}$, is simply the union of the solutions to the K optimization problems, and is given by

$$\hat{\boldsymbol{\Theta}}^{[j]} = \{\hat{\boldsymbol{\Theta}}_1^{[j]}, \hat{\boldsymbol{\Theta}}_2^{[j]}, \ldots, \hat{\boldsymbol{\Theta}}_K^{[j]}\}. \tag{3.21}$$

Finally, we use these ideas to write a parallel OSP algorithm (pOSP) which iterates between parallel estimation of conductivity tensors and estimation of the initial internal states $\boldsymbol{w}_0$. In order to simplify the optimization of $\boldsymbol{w}_0$, we assume that initially the system is close to the resting state and is homogeneous. This assumption lowers the dimensions of $\boldsymbol{w}_0$ since we can treat it as a scalar w_0 multiplied by a vector of ones, $\mathbf{1}$. However, in situations where this assumption does not apply, standard vector optimization methods can be used. The full iterative pOSP algorithm is presented in Alg. 1. [5]

[4] This parameter can also be viewed, in a game theoretic framework, as quantifying the amount of cooperation or "trust" the players (tensors) have in one another.

[5] Minimization is performed using a standard Matlab implementation of the sequential quadratic programming method described in [104].

Algorithm 1 Parallel OSP

procedure pOSP($\boldsymbol{\Theta}^0, \boldsymbol{w}_0^0$) ▷ $\boldsymbol{\Theta}^0 \in \mathbb{S}_{2++}, w_0^0 \in \mathbb{R}$

$\boldsymbol{\Theta}^0_{\forall k \in K} \leftarrow \boldsymbol{\Theta}^0$

$\boldsymbol{w}_0^0 \leftarrow w_0^0 \mathbf{1}$

$j \leftarrow 0$

repeat

$j \leftarrow j + 1$

parfor $k \in K$ **do**

$\hat{\boldsymbol{\Theta}}_k^{[j]} \leftarrow \arg\min_{\boldsymbol{\Theta}_k \in \mathbb{S}_{2++}} \mathrm{V}_{\mathrm{OSP}} \left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}_{i \neq k}^{[j-1]}, \boldsymbol{w}_0^{[j-1]}, \boldsymbol{y} \right) + \lambda \mathrm{d} \left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}_k^{[j-1]} \right) + \mathrm{h} \left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}^{[j-1]} \right)$

end parfor

$\hat{w}_0^{[j]} \leftarrow \arg\min_{w_0 \in \mathbb{R}} V(\boldsymbol{\Theta}^{[j]}, w_0, \boldsymbol{y})$

$\boldsymbol{w}_0^{[j]} \leftarrow \hat{w}_0^{[j]} \mathbf{1}$

until convergence

return

end procedure

The last term of the arg min is an optional spatial penalization function $\mathrm{h} \left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}^{[j-1]} \right)$, which penalizes large spatial deviations from *a priori* information of the tissue. Prior information can be provided by other sources, such as anatomical data, or from biological knowledge of the tissue properties, such as smoothness in smooth muscle tissues. We will further explore the effects of adding a spatial penalization in the Results Section 3.4.

As we elaborated above, the conductivity-tensor inference procedure developed depends on the stimulus and continuity of the tensor field. We can have situations where a particular area is stimulated and a transmembrane potential is generated, however, because some parts of the tissue might be weakly or not be completely connected, the waveform may not reach the entire domain. Therefore, when estimating the conductivity tensor at a particular point where no transmembrane potential waveform has passed through it (unexplored area), the measurement consists of noise only. In this case, also refer as no-informative measurements, the estimated tensor will be isotropic and symmetric since the noise is spatially homogeneous. On the other hand, if a transmembrane potential did pass, then we will have information to provide a conductivity tensor estimate that relates to the tissue conductivity. Hence, in order to improve the conductivity-tensor inference procedure, one straightforward strategy is to excite the domain in several locations at different instances. Since each instance of tissue stimulation will provide us with one conductivity-tensor field estimate, we need a strategy to

merge (fuse) the different tensor field estimates into a joint estimate that will better infer the conductivity-tensor field. There are several methods to utilize the information from different experiments. In the following, building on the representing of a tensor as an anisotropic ellipsoid, we will provide three criterion to fuse the set of tensor-field estimates into one.

There are several scalar indices to measure a tensor anisotropy, a simple yet effective invariant anisotropy index is the ratio of principle components (RPC) [8] and is given by,

$$R_{ij} \equiv \mathrm{RPC}\left(\boldsymbol{\Theta}_{ij}\right) = \frac{max(\boldsymbol{\lambda}_{ij})}{min(\boldsymbol{\lambda}_{ij})}. \tag{3.22}$$

where $\boldsymbol{\Theta}_{ij}$ is the tensor at node $i = 1, \ldots, K$ from experiment $j = 1 \ldots, M$. A high ratio corresponds to high anisotropy, while a ratio closer to one corresponds to an isotropic sphere. Note that there are many other anisotropy indices that exist in literature and could be incorporated here as well, e.g.,, fractional anisotropy (FA), and geodesic anisotropy (GA) [4, 9, 10].

A common method to fuse the tensor field estimates of M independent experiments is with a weighted sum of the results as follows:

$$\boldsymbol{D}_i = \sum_{j=1}^{M} \mathrm{s}\left(\boldsymbol{R}, i, j\right) \boldsymbol{\Theta}_{ij}, \tag{3.23}$$

where the experiment index is j, and the tensor position index is i. $\boldsymbol{D}_j$ is the fused tensor, $\boldsymbol{\Theta}_{ij}$ is the i tensor of experiment j, and $\mathrm{s}\left(\boldsymbol{R}, i, j\right)$ is a scalar nonnegative weight function. By using a nonnegative weight function we are guarantee that the fused tensor $\boldsymbol{D}_i$ is PD. This can be easily shown by multiplying by a general vector $\boldsymbol{x}$ on both sides,

$$\boldsymbol{x}^T \boldsymbol{D}_i \boldsymbol{x} = \sum_{j=1}^{M} \boldsymbol{x}^T \mathrm{s}\left(\boldsymbol{R}, i, j\right) \boldsymbol{\Theta}_{ij} \boldsymbol{x} = \sum_{j=1}^{M} \underbrace{\mathrm{s}\left(\boldsymbol{R}, i, j\right)}_{\geq 0} \underbrace{\boldsymbol{x}^T \boldsymbol{\Theta}_{ij} \boldsymbol{x}}_{>0} > 0 \tag{3.24}$$

where $\forall i, \exists j$ s.t. $\mathrm{s}\left(\boldsymbol{R}, i, j\right) > 0$. The fusion results highly depends on the weight function $\mathrm{s}\left(\boldsymbol{R}, i, j\right)$, we will consider five fusion schemes (three are anisotropy based and two are Euclidean distance based):

Average Anisotropy

Based on the idea that a more anisotropic estimate represents a more informative estimate, a good choice for a fusing scheme is to give a stronger weight to the anisotropic tensors. In the average anisotropy weight function, the weights of the tensors correspond to their normalized anisotropy ratio

$$\mathrm{s}\left(\boldsymbol{R}, i, j\right) = \frac{r_{ij}}{\sum_j r_{ij}}. \tag{3.25}$$

Max Anisotropy

The weight function is an indicator function for the tensor of highest anisotropic value

$$\mathrm{s}\left(\boldsymbol{R}, i, j\right) = \begin{cases} 1 & \text{if } j = \arg\max_j r_{ij}, \quad (3.26) \\ 0 & \text{otherwise.} \quad (3.27) \end{cases}$$

This is a mixture of the previous two schemes where the weights correspond to the normalized anisotropy ratio of all tensors that are bigger than a certain threshold T

$$\mathrm{s}\left(\boldsymbol{R}, i, j\right) = \begin{cases} \frac{1}{B_j} r_{ij} & \text{if } r_{ij} > T, \quad (3.28) \\ 0 & \text{otherwise.} \quad (3.29) \end{cases}$$

where B_j is a normalization constant is given by,

$$B_j = \sum_{j | r_{ij} > T} r_{ij}. \tag{3.30}$$

Euclidean Distance

The weights correspond to the Euclidean distance from the tensor center $\boldsymbol{d}_{ij}$ to the stimulus center $\boldsymbol{c}_j$

$$\mathrm{s}\left(\boldsymbol{R}, i, j\right) \;=\; ||\boldsymbol{d}_{ij} - \boldsymbol{c}_j||^{-1}. \tag{3.31}$$

fourth Order Euclidean Distance

This approach is similar to the Euclidean Distance approach when using the fourth power of the norm in order to get a more localized weighting effect around the stimuli.

$$\mathrm{s}\left(\boldsymbol{R}, i, j\right) \;=\; ||\boldsymbol{d}_{ij} - \boldsymbol{c}_j||^{-4}. \tag{3.32}$$

3.4 Results

To analyze the performance of the pOSP algorithm, we compiled a number of test simulations to examine the algorithm's ability to estimate the tensor field of different field topologies under varying noise levels. For a fairly general and simple representation of a cell's ionic current dynamics, we use the extended FitzHugh-Nagumo (FHN) equations [124]. Then, the continuous model can be written as follows:

$$\mathrm{j}_{\mathrm{ion}}(v(\boldsymbol{r},t),t) \;=\; -\frac{1}{\epsilon_1}(k(v(\boldsymbol{r},t)-v_1)(v_2-v(\boldsymbol{r},t))(v(\boldsymbol{r},t)-v_3)-w(\boldsymbol{r},t)), \tag{3.33}$$

$$\frac{\partial w(\boldsymbol{r},t)}{\partial t} \;=\; \epsilon_2(\beta v(\boldsymbol{r},t)-\gamma w(\boldsymbol{r},t)+\delta), \tag{3.34}$$

where $\mathrm{j}_{\mathrm{ion}}(\boldsymbol{r},t)$ is the ionic volume current density ($\mathrm{A/m}^3$) of a biological cell. Here, $\mathrm{j}_{\mathrm{stim}}(\boldsymbol{r},t)$ is the stimulus volume current density (Am^3) and is a known deterministic input series, and $w(\boldsymbol{r},t)$ is the internal state of the system. The rest of the model parameters and their simulation values are presented in Table 3.1. After discretizing the FHN system, we can write the nonlinear evolution function as follows:

$$\mathbf{f}(\boldsymbol{v}_{0:n}, \boldsymbol{w}_{0:n}, \boldsymbol{\phi}, \boldsymbol{\Theta}) \;=\; c\left(\boldsymbol{j}_{\mathrm{x}_n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{xy}_n}(\boldsymbol{\Theta}) + \boldsymbol{j}_{\mathrm{y}_n}(\boldsymbol{\Theta})\right) - d_n\boldsymbol{w}_0 - \mathrm{g}\left(\boldsymbol{v}_{0:n}, \boldsymbol{\phi}\right), \tag{3.35}$$

Table 3.1: In this table, we list the FitzHugh-Nagumo reaction parameters ($\boldsymbol{\phi}$) categorized by their corresponding control, and the values used for the simulations. The values of the reaction parameters for the simulations were fitted to a slow wave from a pregnant human's uterine myocyte, as presented in [125]

a_{m}	Surface-to-volume ratio of the membrane	$5.758710 \cdot 10^5$ m^{-1}
c_{m}	Membrane capacitance per unit area	0.01 Fm^{-2}
ϵ_1	Sharpness of the edges	100 $\Omega\mathrm{m}^2$
ϵ_2	Excitability duration	1 s^{-1}
γ	Control the excitability threshold of the cell	0.1
β		1
δ		0.0520 V
v_1	Control the range of $v(\boldsymbol{r}, t)$	-0.02 V
v_2		-0.04 V
v_3		-0.065 V
κ		1e4 $1/\mathrm{V}^2$

where $\boldsymbol{w}_0$ is the vector of the initial state $\boldsymbol{w}$, and $d_n = c\frac{a_{\mathrm{m}}\kappa}{\epsilon_1}e^{-\epsilon_2\gamma n\Delta t}$ is a known scalar function.

3.4.1 Step-Size Penalty λ

The hyper-parameter λ, which was introduced in the previous section, is the weight of the penalty term we added in order to control the step size of the pOSP algorithm. To better understand the influence of this parameter on the convergence properties of the algorithm, we varied λ and recorded the resulting squared error, as shown in Fig. 3.4(b). Theoretically, low λ results in greedy optimizations of each tensor and the "strongest survive" phenomenum (convergence to a local optimum). This phenomenon can be observed in Fig. 3.4(c), where a few dominating tensors appear while others are diminished. On the other hand, high λ corresponds to a slow convergence rate, as illustrated in Fig. 3.4(d), where after ten iteration the estimate is still close to the initial field. For this problem, the optimal λ value can be observed from Fig. 3.4(b) at around 10^{-4}, and the corresponding estimate is illustrated in Fig. 3.4(e). As can be observed from Fig. 3.4(e), the estimated conductivity-tensor field, although much better then the other two estimations, is still only partially accurate, namely, only on the main diagonal. This is a typical example where increasing the number of stimuli

will greatly improve the estimated results. We will further discuss this topic in the following section. In the following simulations and analysis, we used a constant $\lambda = 10^{-4}$ for all the subproblems. The effect of noise, inhomogeneity, or time-dependence on the optimal λs, and an efficient way for finding the optimal λs, are not tackled in this work and are left open for future research.

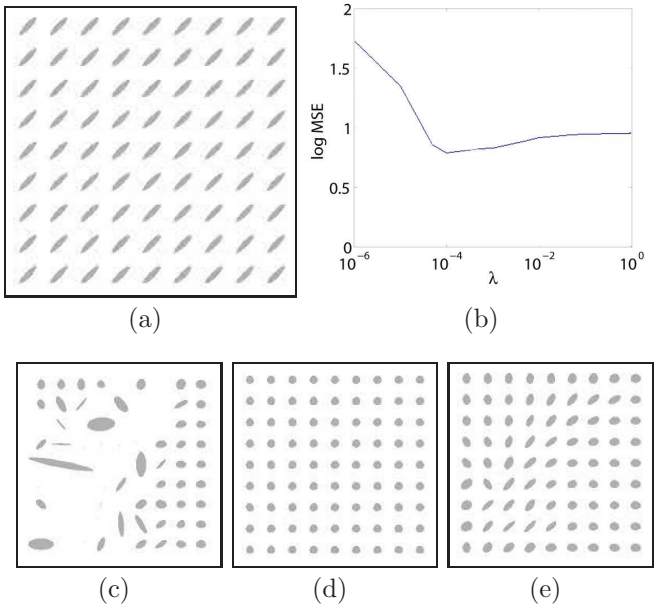
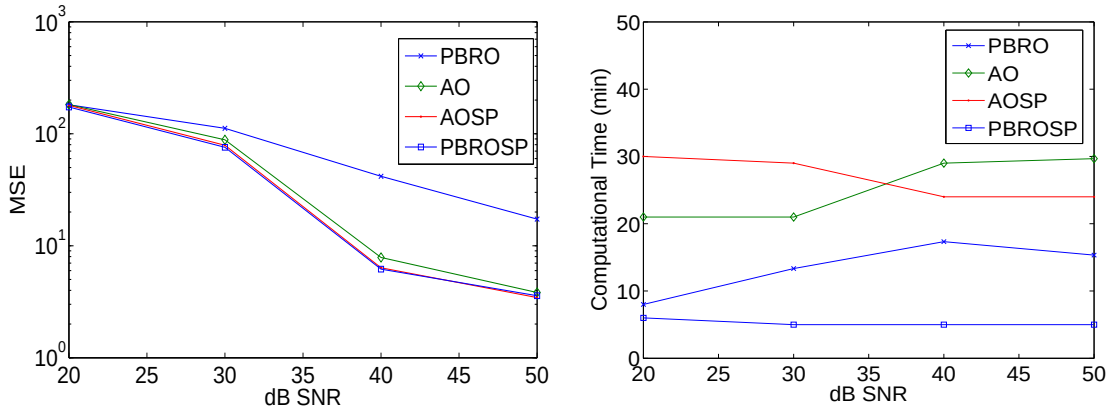


Figure 3.4: Estimation results of the pOSP algorithm on the constant 9×9 tensor field (Fig. 3.4(a)) for different values of λ. Fig. 3.4(b) shows the square error sensitivity of the algorithm to the parameter λ. All the pOSP estimations were calculated with ten iterations and same initial values. Figs. 3.4(c) and 3.4(d),3.4(e) illustrate the results of using the pOSP estimation using a low, high, and optimal (10^{-4}) λ, respectively. The plots of the tensor fields were made using the matlab tool as described in [7].

To test the significance of the step-size penalty on the optimization algorithm, we compared the standard sequential alternating optimization to the parallel block-relaxation optimization, with and without a step-size penalty ($\lambda = 10^{-4}$), for various noise levels. We compared both the accuracy and the computational time for the algorithms, using ten iterations on a toy 8×8 Gaussian mixture as illustrated in Table 3.2. From Fig. 3.5, we can observe that parallel block-relaxation optimization algorithm with step-size penalty (PBROSP) achieves

considerably better estimation results than the one without the step-size penalty. Furthermore, for all tested noise levels, the PBROSP estimation offers comparable estimation accuracy to the standard alternating optimization algorithm at a significantly lower computation time.



(a) (b)

Figure 3.5: This figure illustrates a comparison between the standard sequential alternating optimization (AO), alternating optimization with step-size penalization (AOSP), parallel block-relaxation optimization (PBRO), and parallel block-relaxation optimization with step-size penalization (PBROSP) algorithms. As can be observed from Fig. 3.5(a), addition of a step-size penalty improves the accuracy of both the parallel and sequential optimizations schemes. Both the PBROSP and AOSP produced the best estimations, and their error was almost identical for all noise levels. However, as figure 3.5(b) shows, the PBROSP had considerably faster execution time than any of the other optimization schemes, at all noise levels. (All runs were executed on a Windows7 system with parallel Matlab package.)

3.4.2 Richness of Inputs

The choice of input signals used to excite the system is a fundamental factor in the algorithm's ability to correctly estimate the spatially varying conductivity-tensor field [96, 138]. The input must possess sufficient richness in both its spatial and temporal excitations to provide enough information to fully identify the system. In this work, we do not attempt to analyze the optimal experimental design but rather present a few examples of the improvement that can be achieved by using a rich input set.

For our simulations, we considered three types of spatially varying tensor fields: constant, Gaussian mix, and circular (Table 3.2). We chose these spatial structures because they correspond respectively to a constant field, piecewise constant field with discontinuous edges, and a completely varying smooth field. Furthermore, in all fields the size of the tensors were chosen at random.

We tested the improvement of the pOSP conductivity-tensor estimation with varying numbers of stimuli (Table 3.2). The stimuli were sequential, with 150 time-steps between consecutive stimuli. We used a 8×8 grid and ran the simulation for 400 time-steps Δt, for a total of 20 seconds. The evoked stimuli were half a second each, with amplitudes of one millivolt. In Fig. 3.6, we present the resulting log mean squared error (MSE) between the estimated conductivity-tensor field and the original simulation field, for the three fields considered, as a function of a number of stimuli.For both the constant and circular fields, the MSE (using the Frobenius norm) is monotonically decreasing, meaning that the additive inputs improved the overall estimate. Furthermore, we can observe that the greatest improvement in estimation occurred when the spatial location of a new stimulus created a significantly different wave propagation compared to the waves that were created by previous stimuli. This can be seen in the constant field in Table 3.2, where the first three stimuli were along the same main direction based on the topology of the field, and as a result, they all produced a similar wave propagation. However, the fourth stimulus location was different enough to produce a new wave that allowed for more information on the structure of the top left corner of the field. The results were not as conclusive in the Gaussian mix case, where not enough time was given between sequential stimuli, causing interference in the waves, which eventually lowered the accuracy of estimation.

3.4.3 Sensitivity to Modeling Error

Although the extended FitzHugh-Nagumo equations model the extracellular field dynamics for a wide variety of excitable systems, they also present a challenge: A potentially small deviation in the parameters could cause a large modeling error. In fact, trying to infer the conductivity tensor field without properly fitting these parameters will almost certainly result in inaccurate results. We study the robustness of the algorithm by assuming an error of one parameter at a time (Fig. 3.7). The purpose of this study is to illustrate how much of

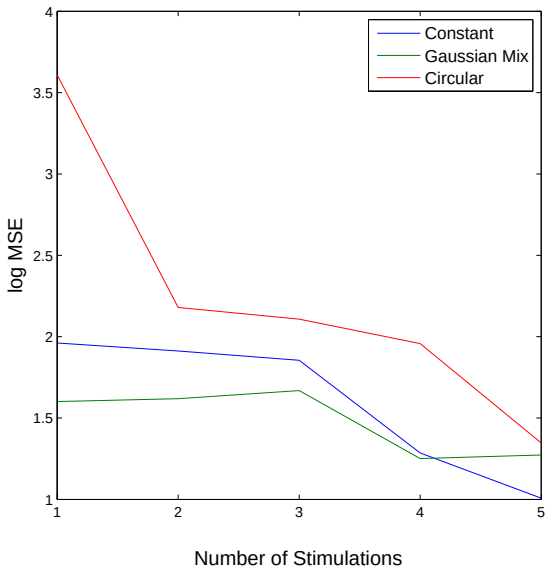
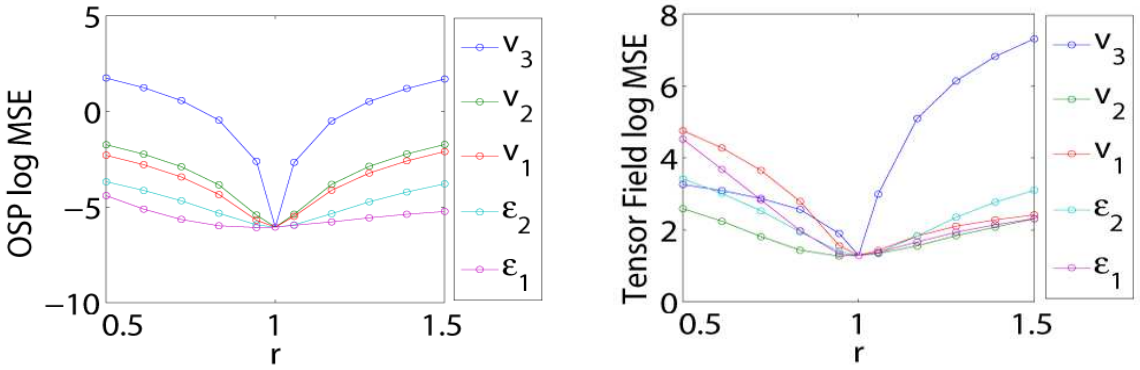


Figure 3.6: This figure illustrates the effect of an increasing number of stimuli on the log mean squared error (using the Frobenius norm), between the original conductivity-tensor field and its estimate.

an error we will have in the tensor field estimation (Fig. 3.7(a)), and in the OSP estimation (Fig. 3.7(b)) if the reaction parameters are not properly fitted. Studying the parameters v_1, v_2, v_3, ϵ_1, and ϵ_2, we can see that the algorithm is especially sensitivity to the polynomial root v_3, which controls the minimum range of the FHN model. However, v_3 is easy to estimate since the minimum range of the extracellular potential can be inferred from both data and prior knowledge. From Fig. 3.7, we can also observe that the OSP prediction error and conductivity tensor errors each have a minimum when the parameters used for the estimation were equal to the true parameters that were used to simulate the wave phenomena. Further, both errors decrease smoothly and monotonically as the parameters get closer to the true value. These results suggest that even though the estimation is sensitive to the reaction parameters, a good approximation can still be achieved by inferring the reaction parameters from the data, using the OSP error and prior knowledge.



(a) (b)

Figure 3.7: Sensitivity of the pOSP estimation to changes in the reaction parameters, using estimations over the Gaussian mix tensor field and varying one parameter each time. The reaction parameters used for estimation varied from 0.5 its true value to 3/2. Hence, if the ratio variable r is the ratio of the value chosen for estimation to the true value, then $r \in [0.5, 1.5]$. Fig. 3.7(a), illustrates the OSP prediction error from Eq. (3.17), and Fig. 3.7(b) illustrates the actual tensor field error, calculated using the Frobenius norm between the true and the estimated tensor fields.

3.4.4 Sensitivity Measurement Noise

In this section, we test the sensitivity of the pOSP (using the PBROSP optimization method) to measurement noise. We used a constant 9×9 tensor field separated into two sections by a zero tensor column in the middle (Fig. 3.8(a)). By stimulating only the left section (Fig. 3.8(c)) at different noise levels, we were able to examine noise effects on both the areas which the wave passed through (explored) and the unexplored areas where the wave did not reach (which hence contain no information). As can be seen from Fig. 3.8(f), the pOSP estimation gives accurate results for low noise levels, with high anisotropy in the explored area, and low conductivity tensors in the middle column. In the unexplored area, due to lack of information needed to improve the estimation of the conductivity tensors, the tensors remained close to their initial values. As we increase the noise, moving our system away from the assumption of high SNR (Fig. 3.8(e)), the results of the pOSP estimation begin to degrade. In Fig. 3.8(g), the estimation of the tensors in the explored areas worsen, with many tensors losing their anisotropic information and becoming isotropic. Problems also occur in the unexplored area, where the tensors exhibit a false directional anisotropic property, as can be seen in the right side of Fig. 3.8(g). This false directionality can easily mislead experimentalists into arriving at wrong conclusions about the conductivities in this area. To

improve the algorithm's robustness to noise, several filtering and constraining schemes can be employed. For clarity, we will divide the methods into three categories: (i) preprocessing, (ii) processing within pOSP, and (iii) postprocessing. In this subsection we analyze three approaches in the preprocessing step and within the pOSP algorithm, leaving discussion on postprocessing, such as fusion of conductivity-tensor field estimation, to future work.

Temporal Filter

In this method, a sliding temporal window filter is applied to the noisy measurements in the preprocessing step, prior on using them in the pOSP algorithm. The method exploits the fact that often in experiments where the wave propagation is measured, the sampling frequency is high compared to the propagation speed of the wave. Thus, consecutive measurements would be fairly similar, and averaging them will achieve higher SNR. However, yields poor results using a three time-steps window in this example, as can be seen in Fig. 3.8(i), is because the wave propagation dynamics were too rapid and were significantly degraded by the averaging.

Spatial Penalization

In this method, robustness to noise is achieved by using a constraint within the pOSP algorithm. Spatial penalization can be added to bring in prior knowledge of the tissue, such as anatomical structure, from other modalities. In this work, we will consider the less constraining assumption of smoothness as our spatial penalty. The idea is to penalize large variations between neighboring tensors, thereby directing the algorithm to choose a smoother solution for the tensor field. For the spatial penalization functions, we choose a function in the following form,

$$\mathrm{h}\left(\boldsymbol{\Theta}_k, \hat{\boldsymbol{\Theta}}^{[j-1]}\right) = \sum_{k=1}^{K} \mu \left(\boldsymbol{L}_{xy}^2 \boldsymbol{\sigma}_{\mathrm{x}} + \boldsymbol{L}_{xy}^2 \boldsymbol{\sigma}_{\mathrm{xy}} + \boldsymbol{L}_{xy}^2 \boldsymbol{\sigma}_{\mathrm{y}}\right), \tag{3.36}$$

where $\boldsymbol{L}_{xy}^2$ is the $2D$ discrete Laplace operator [26], and μ is the penalty weight. The penalization gives particularly good results in cases, such as our example, where the data was taken from a tissue that exhibits a high degree of smoothness (see Fig. 3.8(h)).

Multiple Post-Stimulus Averaging

In cases where conditions are stationary enough to conduct multiple trials of the same post-stimulus experiment, averaging the data across trials can efficiently reduce the noise without degrading the dynamics. As we can see from Fig. 3.8(j), the tensor field estimated by this method is similar to the one the in low noise case (Fig. 3.8(f)), and the method was very effective in reducing the noise and improving the results.

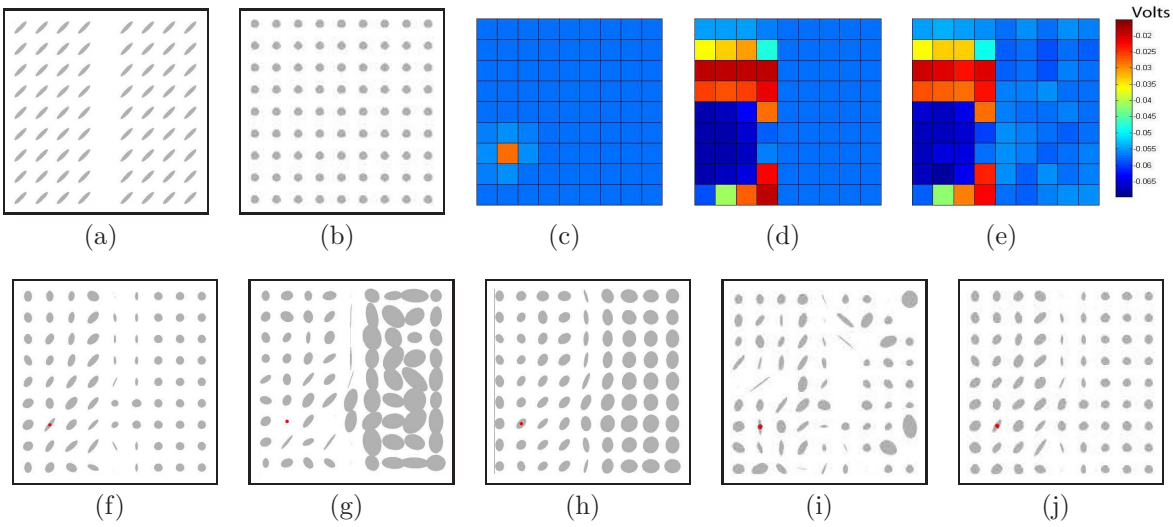


Figure 3.8: This figure illustrates the effect of noise on the pOSP algorithm in both the explored and unexplored areas. The wave propagation was simulated using the true conductivity tensor field (3.8(a)), and an isotropic tensor field as the initial field for the pOSP(3.8(b)). The stimulus in all simulations was at location $(x, y) = (2, 3)$ (3.8(c)), and the observations were corrupted with additive noise of 50dB (3.8(d)) and 35dB SNR (3.8(e)). Figs. 3.8(f) and 3.8(g) illustrate pOSP estimations for cases of low noise (50dB SNR) and high noise (35dB SNR), respectively. Figs. 3.8(h), 3.8(i), and 3.8(j), show the effects of adding a spatial penalization ($\mu = 10^{-13}$), temporal filter (three time-steps), and multiple experiments averaging (five trials), respectively.

Finally, we tested the pOSP estimation using five stimuli (an "all stimuli" experiment) under different noise levels. We considered three cases: 25dB SNR with no temporal filtering, 25dB SNR with a low pass sliding window of three time-steps, and 50dB SNR with no temporal filtering. As we can observe from the results in Table 3.3, even with high noise the pOSP can produce good estimates of the original conductive-tensor fields. Ten runs of the pOSP with differing initial fields were performed to check for sensitivity to the initial guess, all produced similar results. The estimated tensor fields from the first run are illustrated in Table 3.3.

In this section, we will explore different methods of incorporating the information of multiple experiments by fusing their tensor field estimations. As described in Section 3.3.7, there are cases where the stimuli in a certain experiment will only expose part of information needed in order to completely infer the entire tensor field. In such situations, it is useful to consider a postprocessing analysis that fuse the inferred tensor fields of different experiments in order to get a more complete estimation. To test this approach, we considered the three types of spatially varying tensor fields from before: constant, Gaussian mix, and circular. To test for noise sensitivity, we considered three cases: 25dB SNR with no temporal filtering, 25dB SNR with a low pass sliding window of five time-steps, and 50dB SNR with no temporal filtering. Every simulation was composed of five runs where in each run we stimulated the grid at a different position (as can be seen in Appendix 3.4.6). The rest of the simulation setup was the same as in the previous section.

We considered the five fusion methods from Section 3.3.7, specifically: Euclidean distance L_2, fourth order Euclidean distance L_4, average anisotropy, max anisotropy, and anisotropy threshold. For the goodness of fit criterion we calculated the squared error for each fusion method compared to the original tensor field (Table 3.4). Additionally, we also compared two additional methods that fuse the information in the preprocessing step, namely "concatenate stimuli" and "all stimuli" methods. The "concatenate stimuli" method performs the set of individual experiments, then concatenates all of the results into a single joined time series before inputting it to the pOSP algorithm. In contrast, the "all stimuli" method perform a single experiment using all the different stimuli and input its results to the pOSP algorithm. From the Table 3.4, we can see that the preprocessing fusion methods, under low noise, achieve better estimation than the posterior methods. This shows that the pOSP algorithm will converge to a solution closer to the global optimum given more information. However, although these preprocessing fusion methods achieve more accurate results, it comes with a computation price as the optimization routine of the pOSP algorithm will have to deal with a greater amount of data. Another disadvantage of using the "all stimuli" is the fact that experiments must run for a longer time period, and special care must be taken in order to make sure that the effects of the stimuli do not cancel each other. Examining the result of Table 3.4, it seems that the "all stimuli" has a very slight improvement compared to the "concatenate stimuli" method, which does not require long multi-stimulus experiments. On the other hand, the preprocessing and postprocessing do not have to be considered as alternatives, and could also be used together to further enhance the estimation. If we focus

only on comparing the results of the postprocessing fusion methods, the method that seems to give the most robust and accurate results, under both low and high SNR, is the anisotropy threshold fusion method.

The problem of insufficient information for accurate estimation is even more evident when the estimation is performed for systems with low SNR. In these systems, the strong homogeneous Gaussian noise overshadows the wave propagation signal, and results in large-isotropic tensor estimations. Another interesting phenomena that is observed from the fusion results, is that under high noise levels (e.g., 25dB SNR) some of the estimated tensors volume approach zero. This is likely due to the increase in the nonidentifiability of the estimation problem, with the algorithm converging to a less optimal local minimum. This phenomena can be reduced by adding some tensor volume constraints according to *a priori* knowledge of the tissue's expected flow dispersion and conductivities.

3.4.5 Testing on Real Data

We applied the pOSP method to a normalized data set from cardiomyocyte tissue of a newborn mouse, recorded at the Italian Institute of Technology using the high-resolution 4096-channel MEA platform of 3Brain GmbH, Switzerland.

Since the data is already normalized, and to simplify our computational effort, we scaled the data to a particular range of values between $[-0.07, -0.01]$, while the reaction parameters $\boldsymbol{\phi}$ were chosen to fit the dynamics of distinct waves of the normalized measurements. However, when possible, the method should be applied directly to the real transmembrane potential measurements. To simplify the computations, we lowered the resolution of the data into a 20×20 grid by performing block averaging. In Fig. 3.9(a), we present both snapshots of the transformed measurements and an activity measure of the tissue. The activity measure was calculated as the average potential of the five most active measurements. We can divide the observed wave phenomenon into three phases: In the first 15 milliseconds, the wave propagates from the center toward the periphery. In the second phase, between 15-23 ms, we can observe a much faster wave propagation speed. The wave propagates to the right in a funnel shape, then disperses as it reaches the boundary. In the following third phase, no

propagation is observed and the measurements are of noise only. For purposes of illustration, we estimated the conductivity tensor field corresponding to the first phase of propagation.

To use the pOSP algorithm, first the optimal reaction parameters must be inferred from the data. Finding the optimal reaction parameters is a complex task, and ideally should be estimated as an added step to the iterative pOSP algorithm, using prior biological knowledge of the tissue as constraints. There are many important questions, such as what is the optimal fitting strategy, how to handle spatial and temporal varying systems, and how to best integrate the reaction parameters' inference with the conductivity inference. However, these questions are not tackled in this work and are left for future research. In this work we limit ourself to the simplifying assumption of spatially homogeneous dynamics with time invariant parameters. Since the difference in propagation between the first and second phase suggests that there is a difference between the reaction parameters of the cells involved in these phases, we infer the reaction parameters by fitting the FHM model to the wave dynamics of single sensor measurements, which present wave propagation in the first phase only (Fig. 3.9(b)).

To define the boundary conditions of the tissue, locations where the sensors had a total cumulative activity less than the threshold were considered to be not conductive, and hence the conductivity tensors in these locations were set to zero. The initial state values were set to the theoretical resting state potential of a membrane potential $\boldsymbol{v}_0 = v_0\mathbf{1}$, and the first set of measurements was used as input to the system $\boldsymbol{u}_0 = \boldsymbol{y}_0$. We used tensor-volume constraints of $[0.3, 2]$, a spatial penalization penalty of $\mu = 10^{-12}$ (see Section 3.4.4), and a step-size penalty $\lambda = 10^{-4}$. The conductivity tensor field was estimated using the pOSP algorithm and the measurements in the first phase, and is shown in Fig. 3.9(c).

Next we checked that our estimated tensor field indeed significantly improves the model prediction results compared to using a noninformative tensor field. For the null hypothesis, we calculated the one-step-ahead prediction (OSP) errors for 30 random tensor fields, and used these results to find the mean OSP error and confidence interval. In Fig. 3.9(d), we noted the mean OSP error of random fields as "Mean Random Tensor Field", and the confidence interval, corresponding to 98%, is presented in gray. Further, we also compared the OSP errors for other noninformative tensors: an isotropic homogenous field (circles), a zero tensor field (all tensors set to zero), and a constant tensor field (Table 3.2). While all of the noninformative tensor fields tested were within the confidence interval of the random fields, we can

observe that for the first 13 ms the estimated tensor field resulted in an improvement to the OSP error which is statistically significant (3.9(d)). After 13 milliseconds, the OSP errors of the estimated and noninformative fields are relatively equal, and are within the confidence interval. This effect can be explained by the changes in both the spatial location of the wave and the change in dynamics, as explained above (Fig. 3.9(a)). These changes resulted in the estimated tensor field from the first phase of the wave becoming noninformative for the rest of the propagation.

3.4.6 Postprocessing Fusion Results

Below are the results of the pOSP algorithm corresponding to five stimuli at different spatial locations (red dot illustrate the location of the stimulus).

3.5 Discussion

Estimating the electrical conductivity from the reaction diffusion model and a set of measurements can be thought of as estimating of the functional electrical conductivity of the tissue, in the sense that we estimate the electrical conductivity tensors only from areas in the domain where the transmembrane potential wave travels (flow of information.) The benefits of our approach are: (i) It provides anisotropic heterogeneous values of conductivity, and (ii) the conductivity values represent the ability of the media to allow the passage of information, and thus, are tightly related to the underlying dynamics and functional connections of the tissue (for example, in nerve tissue, cells can be connected anatomically but not functionally.)

Our proposed method could also potentially be applied for extracellular analysis of nerve tissues. A popular macroscale approach of modeling cortical neural tissue, with its immense number of intertwined connections, is a continuous two-dimensional medium with neural field equations describing the tissue dynamics [23]. Furthermore, this approach can represent the propagation of traveling wavefronts, which play an important role in cortical information processing [14, 89]. Although the reaction-diffusion model is a simplification of the general

neural field equations as it is limited to only local interactions [89], the two models share many dynamic characteristics [149] suitable for modeling wave propagation in nerve tissue.

To achieve better results with the pOSP algorithm, a learning scheme (e.g., cross-validation) can be used in order to infer the model parameters from the data. These parameters consist of the penalty term λ, the reaction parameters (ionic current dynamics), cell geometry parameters, boundary conditions, bounds on expected conductivity tensors (how much current is reasonable), and the spatial penalty function. Setting the spatial penalty should be based on prior biological knowledge of the tissue considered. In this work we used a simple smoothness penalty as we assumed the tissue conductivity should present a smooth tensor field. However, the actual penalty value does not have a direct biological meaning ($10^{-}12$). For our real data analysis, we chose the penalty value that gave the best results for the simulated field that seem to fit our prior assumption of partwise smooth tissue (Gaussian Mix in Table II). We wish to emphasize that this is a heuristic approach, based on experiments from simulated data. How to optimally decide on this penalty term, or perhaps how to learn it from the data, is a subject of future research.

Another important part of the pOSP algorithm is the knowledge of the evoked stimuli; however, in spontaneous stimuli experiments, this knowledge is rarely available. There are several methods for dealing with the lack of information. We can either try to estimate the stimuli by using Laplacian methods that incorporate high-pass spatial filters to enhance focal activities and reduce widely distributed activity, or we can use a latent approach and infer it from the data.

The penalty hyper-parameter λ is a very important component of this algorithm, especially when a parallel scheme is utilized. Its value will affect the rate of convergence and the amount of influence each subproblem has on the others. This concept can be understood as "mutual cooperation" between the different tensors and is analogous to the game theoretic idea of trust relationship between players [94]. The main principle behind trust games is that players must cooperate to maximize their mutual gain. However, cooperation means that players must risk not playing their dominant strategy , and trust the other players to do the same. In our algorithm, the players are the tensors' matrices, and the mutual score function is the OSP function from Eq. (3.17). In every iteration, each tensor $\mathbf{\Theta}_k$ is estimated using the values of all other tensors from the previous iteration. This allows separation of the algorithm into a parallel optimization scheme which highly reduces computational costs. The "trust" concept

is introduced by adding an additional term λ, to penalize setting the tensors to their greedy individual optimum (dominant strategy). This in turn, directs them to cooperate in order to achieve a joint global optimum (see Section 3.4.1). Having the players (tensors) make their decisions in parallel is a slight variation compare to previous works which talk about sequential trust games [94], however, individually, each player makes its decision according to the other players previous actions, hence making the action sequential than parallel. In this work the "trust" parameter is constant for all players, making it more a property of the system rather than a strategic decision. In future work, we would like to explore using a separate dynamic parameter for each tensor. This approach would allow the use of the game theoretical approach of trust and reciprocity [94], thereby making the algorithm more of an active strategic game between players with a mutual score function.

Finally, we showed that for the tested simulations and initial fields used in this study, the parallel block-relaxation optimization with step-size penalty achieved better convergence behavior than the common alternating optimization algorithm (whose convergence properties are proven and well understood [18]). However, a more rigorous study must be made in order to fully understand the convergence properties of our algorithm.

In order for our model to achieve similar dynamics as presented in the data, the model parameters needed to be tweaked. A standard approach for fitting the model parameters is to disregard the diffusion part of the equation and perform least-square techniques to minimize a cost function with respect to the reaction parameters in an admissible set [1]. In our case, we did not use this approach, but rather used a simplified method of normalizing the amplitude of the measurements and fitting the reaction parameters to get model dynamics similar to a distinct short wave in the measurements. Of course, both of these approaches might not work for all dynamics, and at times, one should consider using a different reaction model than FHN. Of course, this might not work for all dynamics, and at times, one should consider using a different reaction model than FHN.

To achieve better results with the pOSP algorithm, knowledge of the evoked stimuli is needed. In spontaneous stimuli experiments, on the other hand, this knowledge is rarely available. In order to estimate the stimuli, we assume the stimulus to be an impulse effect, located at singular points in space and time. To locate these points, we use Laplacian methods that incorporate high-pass spatial filters to enhance the focal activity and reduce widely distributed activity. Once the stimuli are found, their amplitude is set by calculating the

offset between the observations values at these points and their initial resting values. It is also possible to extend the stimuli periods to achieve a longer stimulus effect.

3.6 Conclusion

We formulated a novel method for solving the inverse problem of inferring the conductivity structure of a biological tissue from a set of spatiotemporal measurements. We lowered the complexity of the optimization by using a single-step approximation employing a parallel block-relaxation optimization method of Chapter 2. This method breaks the original joint problem into a set of smaller subproblems that are solved in parallel, and avoids converging to local minima by forcing cooperation. Cooperation was achieved using a step-size penalty, and the score function was formulated using a one-step-ahead prediction. We analyzed the performance of our method using numerical examples of several electrical conductivity field topologies and noise levels, and discussed its application to real measurements obtained from a smooth cardiac mouse tissue slice, using data collected with the high-resolution 4096-channel MEA platform. In the future, we will consider optimizing model parameter fitting from the data by employing more advanced learning schemes and better utilizing prior biological information. Further, we will extend the model to nonhomogeneous reaction dynamics and establish a methodology for fusing conductivity tensor field information from different post-stimulus experiments.

3.7 Acknowledgments

MEA data set recorded at Italian Institute of Technology by using the high-resolution 4096-channel MEA platform of 3Brain GmbH, Switzerland. This work was supported in part by the McDonnell International Scholars Academy Fellowship, and also in part by a National Science found CCF-0963742. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

Table 3.2: These figures illustrate the results of the pOSP algorithm for three 8×8 conductivity-tensor fields, constant, Gaussian mix, and circular. With each sequential experiment, we introduced an additional stimulus to the simulation and observed the resultant conductivity-tensor field estimation of the pOSP. In the figures, the red dots illustrate the locations of the stimuli, and there was a period of 150 time-steps between consecutive stimuli.

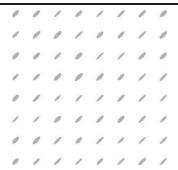
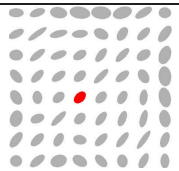
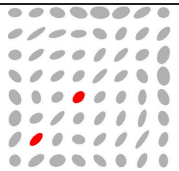
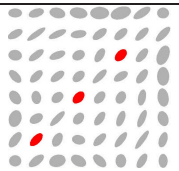
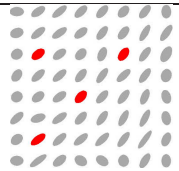
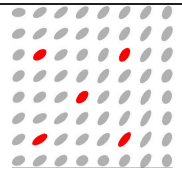
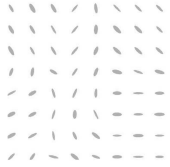
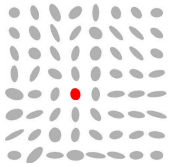
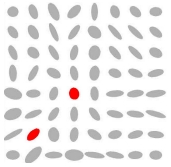
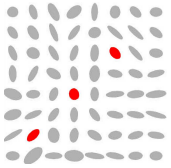
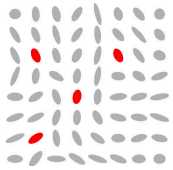
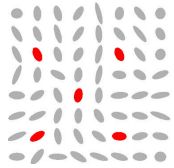
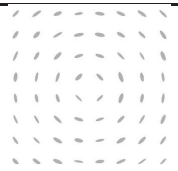
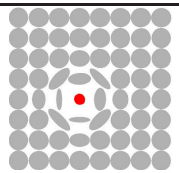
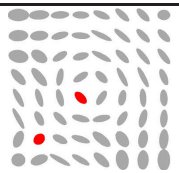
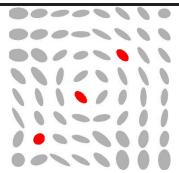
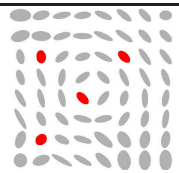
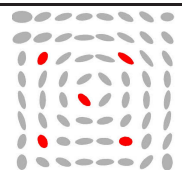
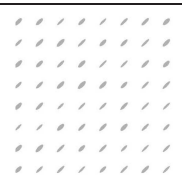
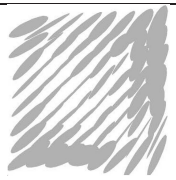
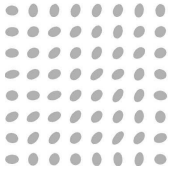
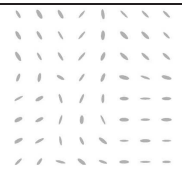
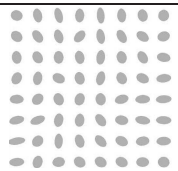
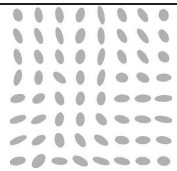
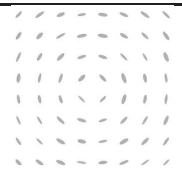
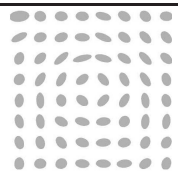
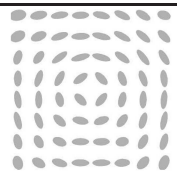
	Real Field	One Stimulation	Two Stimulations	Three Stimulations	Four Stimulations	Five Stimulations
Constant						
Gaussian Mix						
Circular						

Table 3.3: These figures illustrate pOSP estimations for "all stimuli" experiments under different noise levels. We considered three cases: 25dB SNR with no temporal filtering, 25dB SNR with a low pass sliding window of three time-steps, and 50dB SNR with no temporal filtering.

	Real Field	High Noise (25dB SNR)	High Noise (25dB SNR) with filter	Low Noise (50dB SNR)
Constant				
Gaussian				
Circular				

Field Type	Noise Level (dB SNR)	Filter	L_2	L_4	Average Anisotropy	Max Anisotropy	Anisotropy Threshold	Concatenate Stimuli	All Stimuli
constant	25	No	111.7079	72.2485	145.5941	162.2369	108.8599	129.0596	134.6183
	25	Yes	11.2345	12.1416	11.1544	10.4285	10.5909	5.5346	4.7329
	50	No	10.5327	11.4056	10.3955	11.4825	10.5987	4.0376	2.7392
Gaussian Mix	25	No	189.7867	220.9213	183.3025	115.0018	107.2119	121.5367	120.6116
	25	Yes	12.97	16.2267	12.391	13.3538	11.0813	5.9701	5.4956
	50	No	10.3222	11.6649	9.9753	11.0284	10.2125	3.7667	3.5269
circular	25	No	271.2292	266.2753	268.3646	188.0492	181.3074	272.5795	171.7289
	25	Yes	14.2658	15.9545	13.7617	12.9575	12.0746	6.214	4.8002
	50	No	11.6463	13.3173	10.7275	12.2634	11.8121	4.1814	2.9201

Table 3.4: We calculated the MSE using the Frobenius norm between the resulting estimated conductivity-tensor field after each fusion and the original field. Each row in the table correspond to a different noise level on the simulated data from the three original conductivity-tensor fields. We marked in red the fusion method that allowed for the lowest MSE with out taking into account the preprocessing methods ("Concatenate Stimuli" and "All Stimuli").

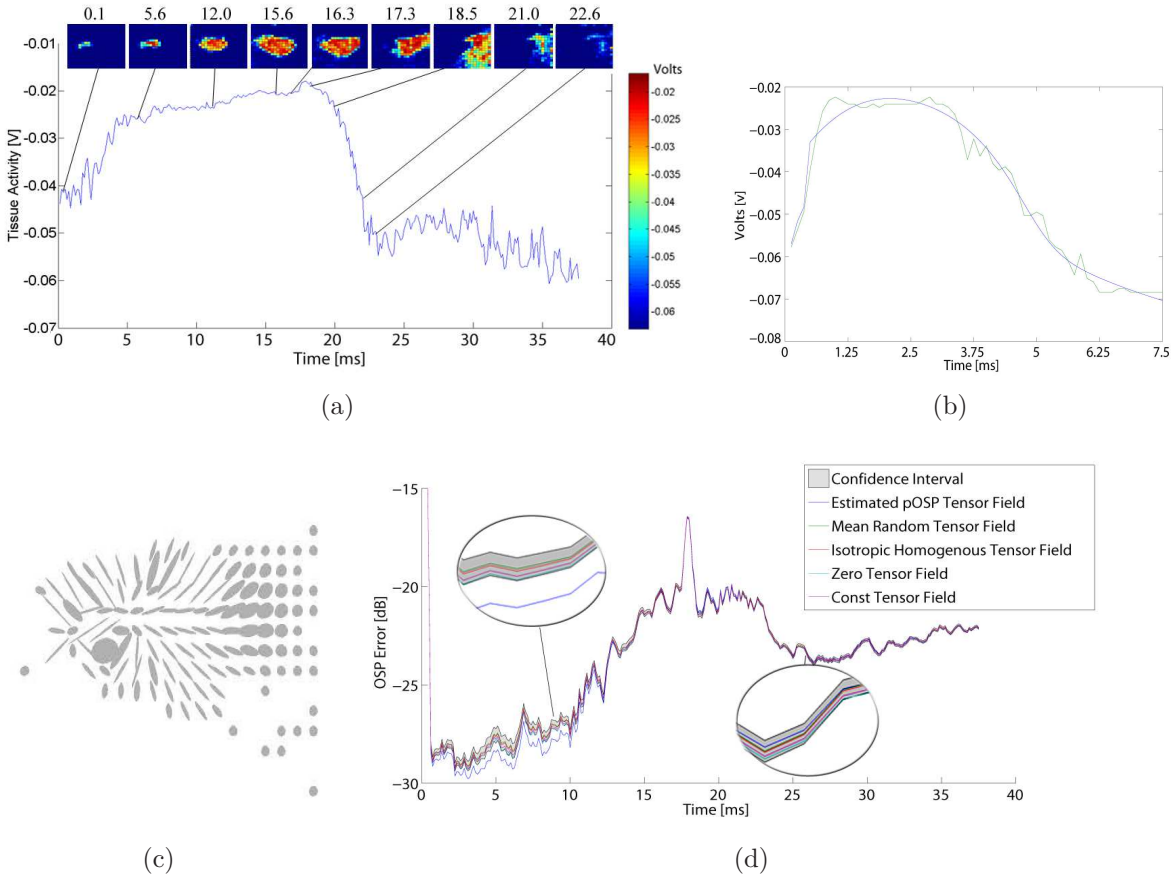


Figure 3.9: Conductivity tensor field estimation of propagating waves in a slice of cardiomyocyte tissue from a newborn mouse. Fig. 3.9(a) illustrates the instantaneous tissue activity, calculated as the average of the five most active measurements at each point in time, and depicts the instantaneous measurement grid ($\boldsymbol{y}_k$) of certain time points. The FHN reaction parameters were fitted to a wave from a single measurement sensor (3.9(b)), and the conductivity tensor field was estimated from the first 15 milliseconds of data (3.9(c)), which corresponds to the expanding wave phenomena. Fig. 3.9(d) illustrates the mean ("Mean Random Tensor Field") and confidence interval (98%, in gray) of one-step-ahead prediction (OSP) errors for 30 random tensor fields. Further, comparison of the OSP errors is shown for the estimated conductivity tensor field, an isotropic homogenous field (circles), a zero tensor field (all tensors set to zero), and a constant tensor field (Table 3.2). While all of the noninformative tensor fields tested were within the confidence interval of the random fields, it can be observed that for the first 13 ms the estimated tensor field resulted in a statistically significant improvement (3.9(d)). After 13 ms, the estimated tensor field OSP error converged with the rest of the noninformative fields. This convergence occurred because the wave's dynamics and location has changed, resulting in the estimated tensor becoming a noninformative tensor field.

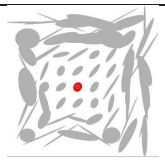
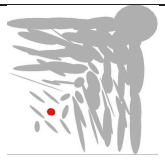
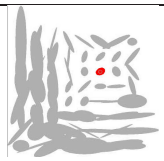
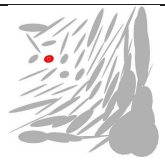
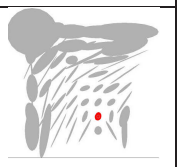
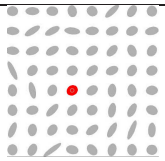
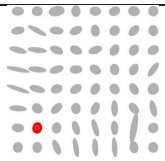
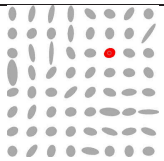
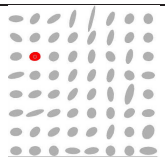
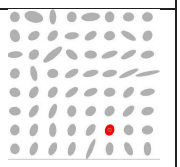
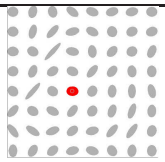
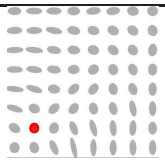
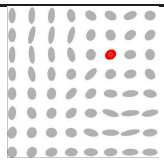
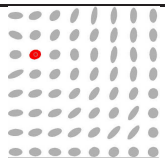
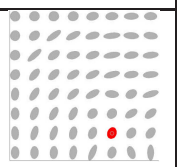
Noise Level	Filter	Stimulus 1	Stimulus 2	Stimulus 3	Stimulus 4	Stimulus 5
25 dB SNR	No					
25 dB SNR	Yes					
50 dB SNR	No					

Table 3.5: Simulations for a Constant tensor field using different noise levels. On the right are the results of the pOSP algorithm corresponding to five stimuli at different spatial locations. Red dot illustrate the location of the stimulus.

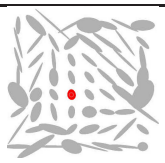
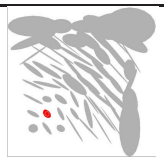
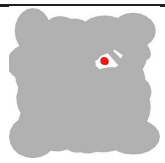
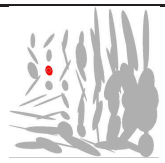
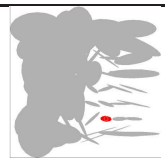
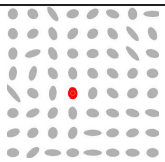
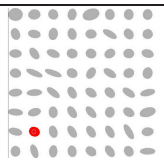
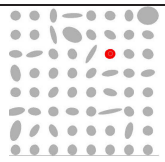
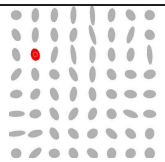
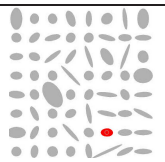
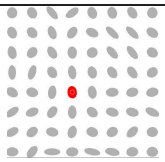
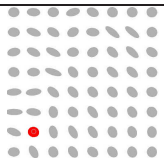
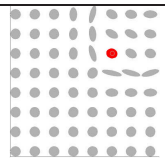
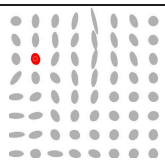
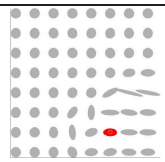
Noise Level	Filter	Stimulus 1	Stimulus 2	Stimulus 3	Stimulus 4	Stimulus 5
25 dB SNR	No					
25 dB SNR	Yes					
50 dB SNR	No					

Table 3.6: Simulations for a Gaussian mixture tensor field using different noise levels. On the right are the results of the pOSP algorithm corresponding to five stimuli at different spatial locations. Red dot illustrate the location of the stimulus.

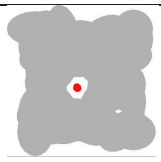
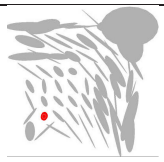
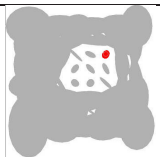
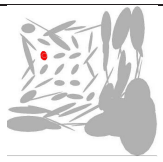
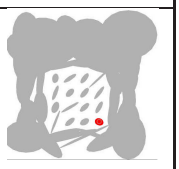
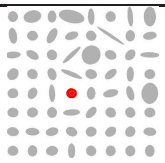
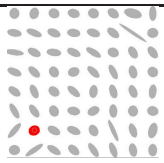
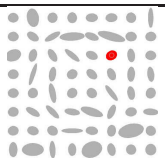
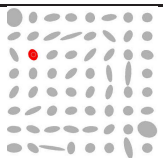
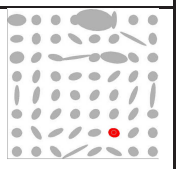
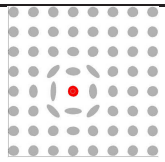
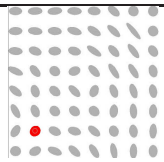
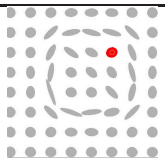
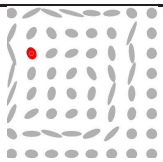
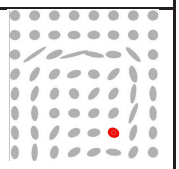
Noise Level	Filter	Stimulus 1	Stimulus 2	Stimulus 3	Stimulus 4	Stimulus 5
25 dB SNR	No					
25 dB SNR	Yes					
50 dB SNR	No					

Table 3.7: Simulations for a circular tensor field using different noise levels. On the right are the results of the pOSP algorithm corresponding to five stimuli at different spatial locations. Red dot illustrate the location of the stimulus.

Chapter 4

Scaling Multidimensional Inference for Structured Gaussian Processes

4.1 Abstract

Exact Gaussian process (GP) regression has $\mathcal{O}(N^3)$ runtime for data size N, making it intractable for large N. Many algorithms for improving GP scaling approximate the covariance with lower rank matrices. Other work has exploited structure inherent in particular covariance functions, including GPs with implied Markov structure, and inputs on a lattice (both enable $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ runtime). However, these GP advances have not been well extended to the multidimensional input setting, despite the preponderance of multidimensional applications. Here we extend the initial work of [127] on multiplicative kernel GPs with inputs on a multidimensional grid.

4.2 Introduction

Gaussian processes (GP) have become a popular tool for nonparametric Bayesian regression. Naive GP regression has $\mathcal{O}(N^3)$ runtime and $\mathcal{O}(N^2)$ memory complexity, where N is the number of observations. At ten thousand or more observations, this problem is for all practical purposes intractable, given current hardware.

A variety of approaches are suggested in the literature for improving the computational complexity of GP for large data sets. Some approximate the GP using simpler models on a lower dimensional subspace, e.g., kernel convolution [66, 161], moving averages [159], or fixed number of basis functions [29]. Other approaches enable fast computation by working

in the spectral domain or using algorithms based on the fast Fourier transform (FFT) [45, 47, 106]. Though these methods confer great advantage in the univariate case, extensions to the multivariate case often require restrictive assumptions [45]. A significant amount of research has also gone into sparse approximations, including covariance tapering [48, 71, 139], conditional independence to inducing inputs [119, 121], or a Gaussian Markov random field approximation [126]. However, the results of these algorithms depend strongly on the properties of the data [119, 126]. Since different assumptions fit different datasets, it is imperative to explore alternative avenues for attaining scalability.

While efficient methods for structured GPs are known in the case of scalar inputs, many regression applications involve multivariate inputs. We present a novel algorithm for GPs with a multiplicative kernel structure when multidimensional inputs are on a lattice (GP-grid). In Sec. 4.3.3 we extend the GP-grid algorithm to handle two limitations of the basic algorithm by allowing for (i) incomplete data, and (ii) heteroscedastic noise. Lastly, we enhance the method by incorporating expressive kernels, which learn hidden patterns in the data. Certainly these extensions to standard GP have been used to good purpose in previous GP settings, but their success can not be replicated in the large N case without additional advances related to this specific multidimensional grid structure.

4.2.1 Gaussian Process Regression

In brief, GP regression is a Bayesian method for nonparametric regression, where a prior distribution over continuous functions is specified via a Gaussian process (the use of GP in machine learning is well described in [121]). A GP is a distribution on functions $\mathbf{f}$ over an input space $\mathbf{X}$ (in the general D dimensional input space case $\mathbf{X} = \mathbb{R}^D$) such that any finite selection of input locations $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{X}$ gives rise to a multivariate Gaussian density over the associated targets, i.e.,

$$p(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)) = \mathcal{N}(\mathbf{m}_N, \mathbf{K}_N), \tag{4.1}$$

where $\mathbf{m}_N = m(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is the mean vector and $\mathbf{K}_N = \{k(\mathbf{x}_i, \mathbf{x}_j; \theta)\}_{i,j}$ is the covariance matrix, for mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot\,; \theta)$. Throughout this work, we use the subscript N to denote that $\mathbf{K}_N$ has size $N \times N$. We are specifically interested in

the basic equations for GP regression, which involve two steps. First, for given data $\mathbf{y} \in \mathbb{R}^N$ (making the standard assumption of zero-mean data, without loss of generality), we calculate the predictive mean and covariance at L unseen inputs as

$$\boldsymbol{\mu}_L = \mathbf{K}_{LN} \left(\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N \right)^{-1} \mathbf{y}, \tag{4.2}$$

$$\boldsymbol{\Sigma}_L = \mathbf{K}_L - \mathbf{K}_{LN} \left(\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N \right)^{-1} \mathbf{K}_{NL}, \tag{4.3}$$

where σ_n^2 is the variance of the observation noise (the *homogeneous* noise case, where σ_n^2 is constant across all observations), which is assumed to be Gaussian. Because the function $k(\cdot, \cdot\,; \theta)$ is parameterized by hyperparameters θ such as amplitude and lengthscale, we must also consider the log marginal likelihood $Z(\theta)$ for model selection:

$$\log Z(\theta) = -\frac{1}{2} \left(\mathbf{y}^\top (\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N)^{-1} \mathbf{y} + \log |\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N| + N \log(2\pi) \right). \tag{4.4}$$

Here we use this marginal likelihood to optimize over the hyperparameters in the usual way [121]. The runtime of full-GP regression and hyperparameter learning is $\mathcal{O}(N^3)$ due to the $\left(\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N\right)^{-1}$ and $\log |\mathbf{K}_N + \sigma_n^2 \mathbf{I}_N|$ terms, so we focus on these two objects to achieve computational efficiency.

4.3 Gaussian Processes on Multidimensional Grids

When data locations lie on a grid, such as pixels in an image, we can exploit the ordinary structure of the problem to significantly reduce the complexity of the GP algorithm. In this section, we present a novel method to perform *exact* GP inference in $\mathcal{O}(DN^{\frac{D+1}{D}})$ time for any multiplicative kernel with D-dimensional grid inputs, using properties of Kronecker products.

Much work has gone to applications of scalar lattice data, such as Toeplitz and spectral approaches [30, 47, 106]. However, these methods are very restrictive when extended to multidimensional grid input [45]. Another approach that is used extensively in the fields of computer vision and image analysis is the Gaussian-Markov random field (GMRF) model; however, this model often proves too simplistic for real data, and is known to be inconsistent over different subsets of lattice data [95]. Other work exploits separability with the

Kronecker product [152], but this approach is rarely used in practice because it requires the restrictive assumptions of noiseless, full (regular) grid measurements. Here we present a novel extension of the Kronecker method to perform exact inference in $\mathcal{O}(DN^{\frac{D+1}{D}})$ time for cases of incomplete grids, missing observations, and variable noise.

To enable our GP-grid algorithm, we revisit the two critical assumptions from Sec. 5.2. Our first key assumption is multiplicative kernel structure. One example of many is the popular squared exponential kernel, which is multiplicative across its input dimensions:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \sigma_f^2 \exp\left(-\sum_{d=1}^{D} \frac{1}{2\ell_d^2}(x_i^d - x_j^d)^2\right) = \sigma_f^2 \prod_{d=1}^{D} \exp\left(-\frac{1}{2\ell_d^2}(x_i^d - x_j^d)^2\right). \tag{4.5}$$

Our second key assumption was that of the multidimensional input space $\mathbf{X} = \mathcal{X}^1 \times \mathcal{X}^2 \times \ldots \times \mathcal{X}^D$. If our dataset contains all possible input vectors on the grid $\mathbf{X}$, we say the dataset is "grid-complete". The assumption of grid-completeness is a necessary first step but also the primary limitation of this first GP-grid algorithm (and previous works that have considered Kronecker structure [127, 152]), and we will eliminate this limitation in Sec. 4.3.3.

With these two assumptions, we can readily show (e.g., [91, 127, 152]) that the multiplicative kernel matrix decomposes to a Kronecker product over the kernel matrices of individual dimensions:

$$\mathbf{K}_N = \mathbf{K}^1_{N^{1/D}} \otimes \mathbf{K}^2_{N^{1/D}} \otimes \ldots \otimes \mathbf{K}^D_{N^{1/D}}, \tag{4.6}$$

where $\mathbf{K}^d_{N^{1/D}}$ is the covariance matrix of the input space of dimension d. Note that, while $\mathbf{K}_N$ is $N \times N$, each $\mathbf{K}^d_{N^{1/D}}$ is only $N^{1/D} \times N^{1/D}$.[6] Used correctly, this observation can enable significant computational and memory savings, because we can operate cheaply on a collection of small matrices instead of expensively on a large matrix. Realizing these savings requires three properties of Kronecker products: first, **memory complexity:** per standard practice in large scale optimization, we never store the large kernel matrix ($\mathcal{O}(N^2)$ in runtime and memory). Instead we treat it as an implicit linear operator. Then, since each $\mathbf{K}^d_{N^{1/D}}$ in Eq. (4.6) is of size $N^{1/D} \times N^{1/D}$, we only require $\mathcal{O}(DN^{2/D})$ memory, which

[6] In the general case, $\mathcal{X}_d$ has cardinality $|\mathcal{X}_d|$ and thus the implied data size $N = \prod_d |\mathcal{X}_d|$. For simplicity and clarity of the complexity analysis, in the remainder we assume equal cardinalities, namely $|\mathcal{X}_d| = N^{1/D}$ for all dimensions d, without loss of generality.

is a substantial reduction even for $D = 2$ (i.e., we now have linear memory complexity for images). Second, we will exploit **eigendecomposition:** the eigenvectors and eigenvalues matrices $\mathbf{K}_N = \mathbf{Q}_N \mathbf{V}_N \mathbf{Q}_N^\top$ can be calculated efficiently in $\mathcal{O}(DN^{3/D})$ by noting that $\mathbf{Q}_N = \mathbf{Q}^1_{N^{1/D}} \otimes \mathbf{Q}^2_{N^{1/D}} \otimes \ldots \otimes \mathbf{Q}^D_{N^{1/D}}$ and $\mathbf{V}_N = \mathbf{V}^1_{N^{1/D}} \otimes \mathbf{V}^2_{N^{1/D}} \otimes \ldots \otimes \mathbf{V}^D_{N^{1/D}}$. Each of the above two steps are standard Kronecker properties, and they will confer huge savings. Third, we exploit **matrix-vector product:** by cascading a common trick from multilinear algebra, we show in Sec. 4.3.1 that multiplication of a vector with a Kronecker matrix can be done at a reduced computational complexity of $\mathcal{O}(DN^{\frac{D+1}{D}})$, vs the full cost $\mathcal{O}(N^2)$.

4.3.1 Matrix-vector Product for Kronecker Matrices

Here we develop the matrix-vector product algorithm for Kronecker matrices which was first introduced in [127]. We first define a few operators from standard Kronecker literature. Let $\mathbf{B}$ be a matrix of size $p \times q$. The reshape$(\mathbf{B}, r, c)$ operator returns a r-by-c matrix ($rc = pq$) whose elements are taken column-wise from $\mathbf{B}$. The vec$(\cdot)$ operator stacks the matrix columns onto a single vector, vec$(\mathbf{B})$ = reshape$(\mathbf{B}, pq, 1)$, and the vec$^{-1}(\cdot)$ operator is defined as vec^{-1}(vec$(\mathbf{B})) = \mathbf{B}$. Finally, using the standard Kronecker property $(\mathbf{B} \otimes \mathbf{C})vec(\mathbf{X}) =$ vec$(\mathbf{C}\mathbf{X}\mathbf{B}^\top)$, we note that for any N argument vector $\mathbf{u} \in \mathbb{R}^N$ we have

$$\mathbf{K}_N \mathbf{u} = \left(\bigotimes_{d=1}^{D} \mathbf{K}^d_{N^{1/D}} \right) \mathbf{u} = \text{vec} \left(\mathbf{K}^D_{N^{1/D}} \mathbf{U} \left(\bigotimes_{d=1}^{D-1} \mathbf{K}^d_{N^{1/D}} \right)^\top \right), \tag{4.7}$$

where $\mathbf{U} = \text{reshape}(\mathbf{u}, N^{1/D}, N^{\frac{D-1}{D}})$, and $\mathbf{K}_N$ is the covariance matrix from Eq. (4.6). With no change to Eq. (4.7) we can introduce the vec^{-1}(vec$(\cdot)$) operators to get

$$\mathbf{K}_N \mathbf{u} = \text{vec} \left(\left(\quad \text{vec}^{-1} \left(\text{vec} \left(\quad \left(\bigotimes_{d=1}^{D-1} \mathbf{K}^d_{N^{1/D}} \right) \left(\mathbf{K}^D_{N^{1/D}} \mathbf{U} \right)^\top \quad \right) \right) \quad \right)^\top \right). \tag{4.8}$$

The inner component of Eq. (4.8) can be written as

$$\text{vec} \left(\left(\bigotimes_{d=1}^{D-1} \mathbf{K}^d_{N^{1/D}} \right) \left(\mathbf{K}^D_{N^{1/D}} \mathbf{U} \right)^\top \mathbf{I}_{N^{1/D}} \right) = \mathbf{I}_{N^{1/D}} \otimes \left(\bigotimes_{d=1}^{D-1} \mathbf{K}^d_{N^{1/D}} \right) \text{vec} \left(\left(\mathbf{K}^D_{N^{1/D}} \mathbf{U} \right)^\top \right). \tag{4.9}$$

Notice that Eq. (4.9) is in the same form as Eq. (4.7) (Kronecker matrix-vector product). By repeating Eqs. (4.8-4.9) over all D dimensions, and noting that $\left(\bigotimes_{d=1}^{D} \mathbf{I}_{N^{1/D}}\right)\mathbf{u} = \mathbf{u}$, we see that the original matrix-vector product can be written as

$$\left(\bigotimes_{d=1}^{D} \mathbf{K}^d_{N^{1/D}}\right)\mathbf{u} = \text{vec}\left(\left[\mathbf{K}^1_{N^{1/D}}, \ldots \left[\mathbf{K}^{D-1}_{N^{1/D}}, \left[\mathbf{K}^D_{N^{1/D}}, \mathbf{U}\right]\right]\right]\right) \tag{4.10}$$

$$\stackrel{\text{def}}{=} \text{kron\_mvprod}\left(\mathbf{K}^1_{N^{1/D}}, \mathbf{K}^2_{N^{1/D}}, \ldots, \mathbf{K}^D_{N^{1/D}}, \mathbf{u}\right) \tag{4.11}$$

where the bracket notation denotes matrix product, transpose then reshape, i.e., $\left[\mathbf{K}^d_{N^{1/D}}, \mathbf{U}\right] = \text{reshape}\left(\left(\mathbf{K}^d_{N^{1/D}}\mathbf{U}\right)^\top, N^{1/D}, N^{\frac{D-1}{D}}\right)$. Iteratively solving the kron_mvprod operator in Eq. (4.11) requires $\mathcal{O}(DN^{\frac{D+1}{D}})$, because each of the D bracket operations requires $\mathcal{O}(N^{\frac{D+1}{D}})$. We present the algorithm for the general D case, though throughout the results we will only consider $D = 2$ (restricting ourselves to image data), which will result in the stated complexity of $\mathcal{O}(N^{3/2})$, which is smaller than the $\mathcal{O}(N^2)$ of naive matrix-vector product.

4.3.2 GP-grid with Homogeneous Noise

Here we first introduce the GP-grid algorithm for computing exact GP inference over grid-complete data with homogeneous noise at orders of magnitude lower cost than full-GP. The introduction of *homogeneous* observation noise makes our inversion of interest be $(\mathbf{K}_N + \sigma_n^2\mathbf{I}_N)^{-1}$, which is no longer a Kronecker matrix due to the perturbation on the main diagonal. The key step is to exploit the eigendecomposition of the Kronecker matrix, such that the key GP inference operation (Eq. 4.2) is:

$$\left(\mathbf{K}_N + \sigma_n^2 I_N\right)^{-1}\mathbf{y} = \left(\mathbf{Q}_N\mathbf{V}_N\mathbf{Q}_N^\top + \sigma_n^2 I_N\right)^{-1}\mathbf{y} = \mathbf{Q}_N\left(\mathbf{V}_N + \sigma_n^2 I_N\right)^{-1}\mathbf{Q}_N^\top\mathbf{y}. \tag{4.12}$$

The right-hand side of Eq. (4.12) involves three matrix vector multiplications which, moving from right to left, can be solved efficiently with kron_mvprod in $\mathcal{O}(DN^{\frac{D+1}{D}})$, diagonal matrix-vector multiplication in $\mathcal{O}(N)$, and again kron_mvprod on the result. The logdet term can also be solved efficiently as

$$\log|\mathbf{K} + \sigma^2\mathbf{I}| = \log|\mathbf{Q}\mathbf{V}\mathbf{Q}^\top + \sigma^2\mathbf{I}| = \sum_{i=1}^{N}\log(\lambda_i + \sigma^2), \tag{4.13}$$

where λ_i are the eigenvalues of $\mathbf{K}$. Thus inference is done in $\mathcal{O}(DN^{\frac{D+1}{D}})$, which in the case of images yields $\mathcal{O}(N^{3/2})$, vs the naive cost with full-GP of $\mathcal{O}(N^3)$.

In the next section we will extend GP-grid to efficiently handle the case of $\mathbf{K} + \mathbf{D}$, where $\mathbf{K}$ is not a Kronecker matrix, and $\mathbf{D}$ is a positive diagonal matrix of known heteroscedastic noise.

4.3.3 Generalizing GP-grid for an Incomplete Grid and Heteroscedastic Noise

Unfortunately, for many applications the above assumptions - (1) a grid-complete dataset with (2) homogeneous noise $\sigma_n^2\mathbf{I}_N$ - are too simplistic, and will cause GP-grid to substantially underfit, resulting in degraded estimation power of the method (this claim will be substantiated later in Sec. 4.5). First, incomplete grids often occur due to missing values (e.g., malfunctioning sensors), or non-grid input space (e.g., a segment of an image). Second, the homogeneous noise assumption is also not valid in many real systems because sensor noise can vary between sensors or can be input-dependent. In this section we will extend GP-grid to efficiently handle both incomplete grids and heteroscedastic noise.

In [91] the authors deal with missing observations by using a sampling approach to approximate the posterior (since the implied covariance matrix lacks Kronecker structure and is computationally intractable), whereas our algorithm does inference exactly and efficiently. However, although their method improves scaling of GP inference over the full-GP, the Gibbs sampler used for learning still necessitates a high runtime cost. In [90] the authors handle missing data as a series of rank-1 updates. Their method resulted in a $\mathcal{O}(R^2N + RDN^{\frac{D+1}{D}})$ runtime and $\mathcal{O}(RN)$ memory, where R is the number of missing data points. Although this method allow for exact GP inference and is very efficient for small R, in the general case where $R = \mathcal{O}(N)$ (as in Section 4.5) this method will still incur cubic complexity.

Inference

Incomplete grids often occur due to missing values (e.g., malfunctioning sensors), or non-grid input space (e.g., a segment of an image). Previous work in literature tried to handel

missing observation for grid input using either sampling [91], or a series of rank-1 updates [90]; however, both of these methods incur high runtime cost with the increase of the number of missing observations.

We will use the notation $\mathbf{K}_M$ to represent a covariance matrix that was computed using multiplicative kernel over an input set χ^M, $|\chi^M| = M$, which do need to lie on a complete grid. Hence, $\mathbf{K}_M$ is not necessarily a non Kronecker matrix, and can be represented as $\mathbf{K}_M = \mathbf{E}\mathbf{K}_N\mathbf{E}^\top$, where the $\mathbf{K}_N$ is a Kronecker matrix computed from the set χ^N ($\chi^M \subseteq \chi^N$) of N inputs that lie on a complete grid. The $\mathbf{E}$ matrix is a selector matrix of size $M \times N$, choosing the inputs from the complete-grid space that are also in the incomplete-grid space. The $\mathbf{E}$ matrix is a sparse matrix having only M non-zero elements.

This representation is helpful for matrix vector multiplication because it allows us to project the incomplete observation vector to the complete grid space $\mathbf{y}_N = \mathbf{E}^\top\mathbf{y}_M$, perform all operations using the properties of the Kronecker matrices, and then project the results back to the incomplete space.

We use preconditioned conjugate gradients (PCG) [5] to compute $(\mathbf{K}_M + \mathbf{D})^{-1}\mathbf{y}$. Each iteration of PCG calculates the matrix vector multiplication

$$(\mathbf{K}_M + \mathbf{D})\mathbf{v} = \mathbf{E}\mathbf{K}_N\mathbf{E}^\top\mathbf{v} + \mathbf{D}\mathbf{v}. \tag{4.14}$$

The complexity of matrix vector multiplication of the diagonal and selector matrices is $\mathcal{O}(N)$, hence the complexity of the multiplication above will depend on the matrix vector multiplication of $\mathbf{K}_N$. Exploiting the fast multiplication of Kronecker matrices, PCG takes $\mathcal{O}(JN^{\frac{3}{2}})$ total operations (where the number of PCG iterations $J \ll N$) to compute $(\mathbf{K}_M + \mathbf{D})^{-1}\mathbf{y}$, which allows for exact inference.

Learning

For learning (hyperparameter training) we must evaluate the marginal likelihood of Eq. (4.4). We cannot efficiently compute the complexity penalty in the marginal likelihood $\log|\mathbf{K}_M+\mathbf{D}|$ because $\mathbf{K} = \mathbf{K}_M+\mathbf{D}$ is not a Kronecker matrix. We can alleviate this problem by replacing the exact logdet complexity with an efficient upperbound. Using an upperbound allows

to keep the computational and memory complexities low while maintaining a low model complexity. We emphasize that only the log determinant (complexity penalty) term in the marginal likelihood undergoes a small approximation, and inference remains exact.

In [41], the author showed that for a $n \times n$ hermitian positive semidefinite matrices $\mathbf{A}$, $\mathbf{B}$ with eiganvalues $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$ and $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$, respectively,

$$|\mathbf{A} + \mathbf{B}| \leq \prod_{i=1}^{n} (\alpha_i + \beta_{n+1-i}) \,. \tag{4.15}$$

These estimates are best possible in terms of the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$. Using Eq. (4.15), we can write an upperbound on the complexity penalty as:

$$\log |\mathbf{K}_M + \mathbf{D}| \leq \sum_{i=1}^{M} \log \left(\lambda_i^M + d_{M+1-i}\right) , \tag{4.16}$$

where $d_i = \text{sort}(\text{diag}(\mathbf{D}))_i$. However, finding λ_i^M, the eigenvalues of $\mathbf{K}_M$, is still $\mathcal{O}\left(N^3\right)$. We instead approximate the eigenvalues λ_i^M using the eigenvalues of $\mathbf{K}_N$, such that $\tilde{\lambda}_i^M \approx \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$ [153], which is a particularly good approximation for large M (e.g., $M > 1000$).

4.4 Kernels for Pattern Discovery

The heart of a Gaussian process model is its kernel, which encodes all inductive biases – what sorts of functions are likely under the model. Popular kernels are not often expressive enough for automatic pattern discovery and extrapolation. To learn rich structure in data, we now present highly expressive kernels which combine with the scalable exact inference procedures introduced previously in this chapter.

In general it is difficult to learn covariance structure from a single Gaussian process realisation, with no assumptions. Most popular kernels – including the Gaussian (SE), Matérn, γ-exponential, and rational quadratic kernels [122] – assume *stationarity*, meaning that they are invariant to translations in the input space x. In other words, any stationary kernel k is a function of $\tau = x - x'$, for any pair of inputs x and x'.

Bochner's theorem [19] shows that any stationary kernel $k(\tau)$ and its *spectral density* $S(s)$ are Fourier duals:

$$k(\tau) = \int S(s) e^{2\pi i s^\top \tau} ds \,, \tag{4.17}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^\top \tau} d\tau \,. \tag{4.18}$$

Therefore if we can approximate $S(s)$ to arbitrary accuracy, then we can also approximate any stationary kernel to arbitrary accuracy, and we may have more intuition about spectral densities than stationary kernels. For example, the Fourier transform of the popular SE kernel is a Gaussian centered at the origin. Likewise, the Fourier transform of a Matérn kernel is a t distribution centered at the origin. These results provide the intuition that arbitrary additive compositions of popular kernels have limited expressive power – equivalent to density estimation with, e.g., scale mixtures of Gaussians centered on the origin, which is not generally a model one would use for density estimation. Scale-location mixtures of Gaussians, however, can approximate any distribution to arbitrary precision with enough components [72], and even with a small number of components are highly flexible models.

Suppose that the spectral density $S(s)$ is a scale-location mixture of Gaussians,

$$S(s) = \sum_{a=1}^{A} w_a^2 [\mathcal{N}(s; \mu_a, \sigma_a^2) + \mathcal{N}(-s; \mu_a, \sigma_a^2)]/2 \,, \tag{4.19}$$

noting that spectral densities for real data must be symmetric about $s = 0$ [67], and assuming that x, and therefore also s, are in $\mathbb{R}^1$. If we take the inverse Fourier transform (Eq. (4.18)) of this spectral density in Equation (4.19), then we analytically obtain the corresponding spectral mixture (SM) kernel function:

$$k_{\text{SM}}(\tau) = \sum_{a=1}^{A} w_a^2 \exp\{-2\pi^2 \tau^2 \sigma_a^2\} \cos(2\pi\tau\mu_a) \,, \tag{4.20}$$

which was derived by [156], and applied solely to simple time series examples with a small number of datapoints. We extend this formulation for tractability with large datasets and multidimensional inputs.

The squared exponential kernel for multidimensional inputs, for example, decomposes as a product across input dimensions. This decomposition helps with computational tractability – limiting the number of hyperparameters in the model – and like stationarity, provides a bias that can help with learning. For higher dimensional inputs, $x \in \mathbb{R}^P$, we propose to leverage this useful product assumption for a spectral mixture product (SMP) kernel

$$k_{\text{SMP}}(\tau|\boldsymbol{\theta}) = \prod_{p=1}^{P} k_{\text{SM}}(\tau_p|\boldsymbol{\theta}_p)\,, \tag{4.21}$$

where τ_p is the p^{th} component of $\tau = x - x' \in \mathbb{R}^P$, $\boldsymbol{\theta}_p$ are the hyperparameters $\{\mu_a, \sigma_a^2, w_a^2\}_{a=1}^A$ of the p^{th} spectral mixture kernel in the product of Eq. (4.21), and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_p\}_{p=1}^P$ are the hyperparameters of the SMP kernel. With enough components A, the SMP kernel of Eq. (4.21) can model any product kernel to arbitrary precision, and is flexible even with a small number of components. We use SMP-A as shorthand for an SMP kernel with A components in each dimension (for a total of $3PA$ kernel hyperparameters and 1 noise hyperparameter).

A GP with an SMP kernel is not a finite basis function method, but instead corresponds to a finite (A component) mixture of infinite basis function expansions. By combining the fast Gaussian process framework of Section 4.3.3 with the expressive SMP kernel, we introduce a new Bayesian nonparametric framework *GPatt*, which enables automatic pattern extrapolation with Gaussian processes on large multidimensional datasets.

4.5 Results

4.5.1 Runtime Complexity

First, we compare the runtime complexity of GP-grid from Section 4.3.3 to both full-GP (naive implementation of Sec. 4.2.1 using Cholesky decomposition) and GP-grid with grid-complete and homogeneous noise. We conduct the comparison using a segment of real image data of a cone (Fig. 4.1). We consider only the input locations within the segment (size M), except for GP-grid homogeneous where we used the entire grid-complete segment (size N). At each iteration the size of the window N is increased, thereby increasing the number of input locations M (pixels we did not mask out). Fig. 4.1 illustrates the time complexity of

the three algorithms as a function of input size (pixels). For every comparison we also note the ratio of unmasked input to the total window size for each point. The time complexity presented for all algorithms is for a single calculation of the negative log marginal likelihood (NLML) and its derivatives (dNLML), which are the needed calculations in GP learning (and which carry the complexity of the entire GP algorithm). In GP-grid, the noise model is not learned but assumed to be known from the apparatus used to capture the image [110], which is:

$$\sigma_i^2 = 0.47 I_i + 56.22, \tag{4.22}$$

where at location i, σ_i^2 is the noise variance and I_i is the image intensity. Since we do not have I_i we use the measured y_i instead as an approximation, which is a common heuristic (though it technically violates the generative model of the GP) of known camera properties that we discuss later in this work. As can be seen in Fig. 4.1, GP-grid does inference exactly and scales only superlinearly with the input size, while full-GP is cubic. While the more general GP-grid (Sec. 4.3.3) does slightly increase computational effort, it does so scalably while preserving exact inference, and we will show that it has significant performance implications that easily warrant this increase. All other commonly used interpolation methods (e.g., bilinear, bicubic, and bicubic-spline) scale at least linearly with the data.

4.5.2 Application to Image Data

In this section we present the performance of GP-grid for real image data interpolation, and the improvement compared to commonly used image interpolation methods. As a reminder, GP-grid is an exact GP algorithm so the previous runtime-accuracy tradeoff comparison is not needed (since our method is always superior to Full-GP). Hence here we briefly present the application to images to show that it is a competitive method against other image processing methods. We use three novelties from Sec. 4.3.3 to test this method: the use of GP itself (enabled by GP-grid), the ability of GP to accept segmented data, and the ability of GP to accept a known noise model (Eq. (4.22)).

For comparison, we used real images acquired by a CCD imaging array,[7] where over a thousand pictures of the same four scenes were taken. We manually segmented the images into two exclusive segments, one of the object and one of the background, an example of

[7]Kodak KAI-4022 4-Mega pixel.

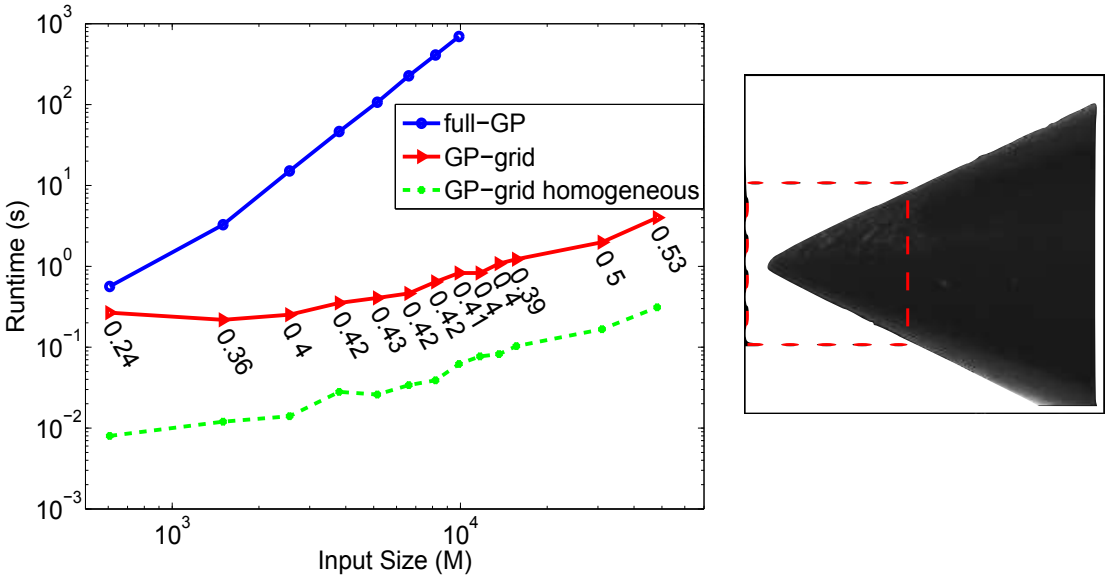


Figure 4.1: Runtime complexity of full-GP, GP-grid, and GP-grid homogeneous, for a single calculation of the negative log marginal likelihood (NLML) and its derivatives. For input, we used segmented data from the cone image of the right. At every comparison the size of the segment N (red dotted line) was increased, thereby increasing the input size M (pixels not masked out). The ratio of input size to the complete grid size (M/N) is shown next to the GP-grid plot. The slope for the full-GP is 2.6, for GP-grid is 1.0, and for GP-grid homogeneous is 1.1 (based on the last 8 points). This empirically verifies the improvement in scaling. Other interpolation methods also scale at least linearly, so the cost of running GP-grid is constant (the runtime gap is not widening with data).

which is shown in Fig. 4.2. In all pictures the empirical noise model fit reasonably well to the camera-specific noise model used in Eq. (4.22) (line shown in red). To estimate the true image, we averaged over the majority of the pictures, leaving a small subset for testing. In order to test interpolation performance, we interpolated the entire image using only a subset of the image (down-sampled by 1/4, a factor of two in both the vertical and horizontal directions). All images are 200×200 pixels, hence, even their down-sampled version will be impractical for Full-GP. The interpolated images were then compared to their corresponding averaged images for accuracy analysis. For an accuracy criterion, we compared the standardized mean square error (SMSE) between the interpolated images and the average images, as defined in [121]. Note that in all the comparisons we intentionally changed the test conditions so they would be most favorable to non-GP methods: we discarded the pixels by the border pixels (5 pixels width), and allowed non-GP methods access to the entire

image. These choices are conservative as the non-GP methods fail particularly badly at the edges, and so we discarded those results to clarify that these improvements have nothing to do with the failure modes of other methods. We chose to compare GP-grid with the common interpolation algorithms: bilinear, bicubic, bicubic-spline (Bic-sp) and NEDI [82]. Although this is by no means an exhaustive comparison, it allows for a benchmark for comparison with GP performance.

(a) (b) (c)

Figure 4.2: An example of the face image separated to an object segment (Fig. 4.2(a)) and a background segment (Fig. 4.2(b)), which are used for interpolation comparison along with their empirical noise vs. intensity model (Fig. 4.2(c)). The red line corresponds to the camera specific linear noise model in Eq. (4.22).

We ran GP-grid using both the Matérn(1/2) and Matérn(5/2) covariance functions, and learned the hyperparameters: lengthscales (l_1, l_2), signal variance (σ_f^2), and noise variance (σ_n^2) [121]. For brevity, we will use GP-grid$(\cdot)$ for GP-grid Matérn$(\cdot)$, and we will add "sph" when we used GP-grid to learn the spherical noise variance hyperparameter σ_n^2. We tested the algorithms on images such as the one from Fig. 4.2 when taken as a whole (W), object segment (O), and background segment (B). As a reference, we also added GP-grid(1/2) spherical and GP-grid(5/2) spherical. As Table 4.1 shows, the GP-grid algorithm with the camera specific noise model improved performance in all images compared with GP-grid spherical and best overall interpolation results of all the algorithms tested. The improvement over GP-grid spherical is perhaps most evident in the moose object image, where both the GP-grid spherical algorithms severely underfit the results. These improvements may be in part due to the fact that the GP-grid is the only algorithm with knowledge of a noise model (Eq. (4.22)). The GP framework is a natural choice for enabling this noise model, and its use is critically enabled by our GP-grid method (which is the point of this section).

4.5.3 Application to Temperature Data

Finally, to show the extension of our GP-grid algorithm to higher dimensional data, we show a spatio-temporal example ($D = 3$) of monthly land surface temperatures in North America.[8] Fig 4.3 (left column) shows monthly temperature readings from 1950. We used temperature readings from 9 months (excluding April, August, and December) as our training set, corresponding to $n = 24939$ data points on an irregular grid. This data comprises 44% of the full $66 \times 71 \times 12$ grid. and is irregular both in time (held-out test data) and space (incomplete land coverage). Note that this data size is already well beyond the range of a Full-GP (c.f., Fig 4.1), and the incomplete grid structure precludes the use of GP-grid spherical. Hence, our GP-grid method (Sec. 4.3.3) is a critical enabler of this application. We chose the Matérn(5/2) kernel and learned the hyperparameters (including global noise σ_n). The GP-grid inference results are shown the second column for the held-out test set of April, August and December, with their corresponding 95% confidence intervals (two standard deviations of the posterior are plotted) in the third column. Note the higher posterior variance in December (which had only past data, not past and future as in April and August), indicating that this data has significant temporal structure in addition to its spatial structure. Our GP-grid required only 4.6 seconds for inference and 5 minutes for learning, in a dataset of roughly 25 thousand points where other GP methods are intractable. The critical point of these results is that our computational advances enable GP to be applied in a new application domain where large data sets are the norm.

4.5.4 Application to Pattern Extrapolation

In our experiments we combine the SMP kernel of Eq. (4.21) with the fast exact inference and learning procedures of section 4.3.3, in a GP method we henceforth call GPatt[9,10]. As GPs have successfully been used for image denoising and interpolation, we focus on the more challenging problem of extrapolating a variety of sophisticated patterns embedded in large datasets.

[8] Data from the Joint Institute for the study of Atmosphere and Ocean (http://jisao.washington.edu/data/satmergedarctic)

[9] We write *GPatt-A* when GPatt uses an SMP-A kernel.

[10] Experiments were run on a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor.

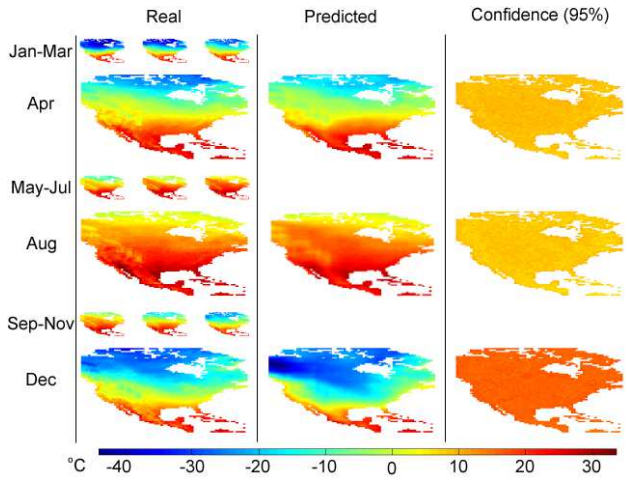


Figure 4.3: Average monthly land surface temperatures in North America in 1950. The left column presents the real measurements. Small images (9 months) were used as a training set, and April, August, and December were used as a held-out test set. The middle and right columns show the corresponding GP-grid posterior mean and 95% confidence intervals for April, August, and December.

We contrast GPatt with many alternative Gaussian process kernel methods. In particular, we compare to the recent sparse spectrum Gaussian process regression (SSGP) [80] method, which provides fast and flexible kernel learning. SSGP models the kernel spectrum (spectral density) as a sum of point masses, such that SSGP is a finite basis function model, with as many basis functions as there are spectral point masses. SSGP is similar to the recent models of [81] and [120], except it learns the locations of the point masses through marginal likelihood optimization. We use the SSGP implementation provided by the authors at http://www.tsc.uc3m.es/ miguel/downloads.php.

To further test the importance of the fast inference used in GPatt, we compare to a GP which uses the SMP kernel of section 4.4 but with the popular fast FITC [100, 137] inference, implemented in GPML[11]. We also compare to GPs with the popular squared exponential (SE), rational quadratic (RQ) and Matérn (MA) (with 3 degrees of freedom) kernels, catalogued in [122], respectively for smooth, multi-scale, and finitely differentiable functions. Since GPs with these kernels cannot scale to the large datasets we consider, we combine these kernels with the same fast inference techniques that we use with GPatt, to enable a comparison.[12]

[11] http:/www.gaussianprocess.org/gpml

[12] We also considered the model of [36], but this model is intractable for the datasets we considered and is not structured for the fast inference of section 4.3.3.

Moreover, we stress test each of these methods in terms of speed and accuracy, as a function of available data and extrapolation range, and number of components.

In all experiments we assume Gaussian noise, so that we can express the marginal likelihood of the data $p(\mathbf{y}|\boldsymbol{\theta})$ solely as a function of kernel hyperparameters $\boldsymbol{\theta}$. To learn $\boldsymbol{\theta}$ we optimize the marginal likelihood using BFGS. We use a simple initialisation scheme: any frequencies $\{\mu_a\}$ are drawn from a uniform distribution from 0 to the Nyquist frequency (1/2 the sampling rate), length-scales $\{1/\sigma_a\}$ from a truncated Gaussian distribution, with mean proportional to the range of the data, and weights $\{w_a\}$ are initialised as the empirical standard deviation of the data divided by the number of components used in the model. In general, we find GPatt is robust to initialisation.

This range of tests allows us to separately understand the effects of the SMP kernel and proposed inference methods of section 4.3.3; we will show that both are required for good extrapolation performance.

4.5.5 Extrapolating a Metal Tread Plate Pattern

We extrapolate the missing region, shown in Figure 4.4a, on a real metal tread plate texture. There are 12675 training instances (Figure 4.4a), and 4225 test instances (Figure 4.4b). The inputs are pixel locations $x \in \mathbb{R}^2$ ($P = 2$), and the outputs are pixel intensities. The full pattern is shown in Figure 4.4c. This texture contains shadows and subtle irregularities, no two identical diagonal markings, and patterns that have correlations across both input dimensions.

To reconstruct the missing region, as well as the training region, we use GPatt with 30 components for the SMP kernel of Eq. (4.21) in each dimension (GPatt-30). The GPatt reconstruction shown in Figure 4.4d is as plausible as the true full pattern shown in Figure 4.4c, and largely automatic. Without hand crafting of kernel features to suit this image, exposure to similar images, or a sophisticated initialisation procedure, GPatt has automatically discovered the underlying structure of this image, and extrapolated that structure across a large missing region, even though the structure of this pattern is not independent across the two spatial input dimensions. Indeed the separability of the SMP kernel represents only a soft prior assumption, and does not rule out posterior correlations between input dimensions.

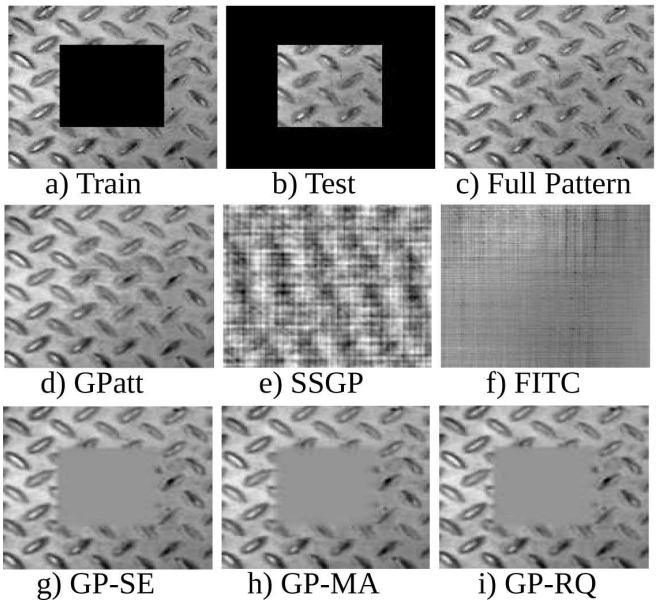


Figure 4.4: Extrapolation on a Metal Tread Plate Pattern. Missing data are shown in black. a) Training region (12675 points), b) Testing region (4225 points), c) Full tread plate pattern, d) GPatt-30, e) SSGP with 500 basis functions, f) FITC with 500 inducing (pseudo) inputs, and the SMP-30 kernel, and GPs with the fast exact inference in section 4.3.3, and g) squared exponential (SE), h) Matérn (MA), and i) rational quadratic (RQ) kernels.

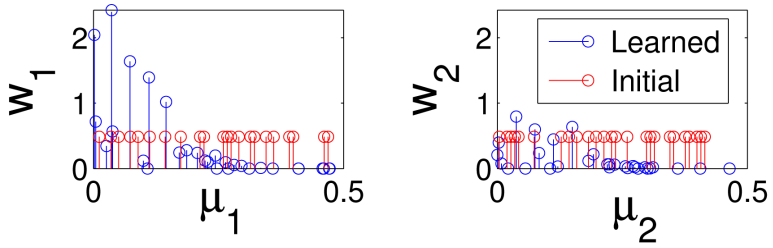


Figure 4.5: Automatic Model Selection in GPatt. Initial and learned weight and frequency parameters of GPatt-30, for each input dimension (a dimension is represented in each panel), on the metal tread plate pattern of Figure 4.4. GPatt-30 is overspecified for this pattern. During training, weights of extraneous components automatically shrink to zero, which helps indicate whether the model is overspecified, and helps mitigate the effects of model overspecification. Of the 30 initial components in each dimension, 15 are near zero after training.

The reconstruction in Figure 4.4e was produced with SSGP, using 500 basis functions. In principle SSGP can model any spectral density (and thus any stationary kernel) with infinitely many components (basis functions). However, since these components are point masses (in frequency space), each component has highly limited expressive power. Moreover, with many components SSGP experiences practical difficulties regarding initialisation, over-fitting, and computation time (scaling quadratically with the number of basis functions). Although SSGP does discover some interesting structure (a diagonal pattern), and has equal training and test performance, it is unable to capture enough information for a convincing reconstruction, and we did not find that more basis functions improved performance. Likewise, FITC with an SMP-30 kernel and 500 inducing (pseudo) inputs cannot capture the necessary information to interpolate or extrapolate. On this example, FITC had a runtime of 2 days and SSGP-500 had a runtime of 1 hour, compared to GPatt which took under 5 minutes.

GPs with SE, MA, and RQ kernels are all truly Bayesian nonparametric models – these kernels are derived from infinite basis function expansions. Therefore, as seen in Figure 4.4 g), h), i), these methods are completely able to capture the information in the training region; however, these kernels do not have the proper structure to reasonably extrapolate across the missing region – they simply act as smoothing filters. We note that this comparison is only possible because these GPs are using the fast exact inference techniques in section 4.3.3.

Overall, these results indicate that both expressive nonparametric kernels, such as the SMP kernel, and the specific fast inference in section 4.3.3, are needed to be able to extrapolate patterns in these images.

We note that the SMP-30 kernel used with GPatt has more components than needed for this problem. However, as shown in Fig. 4.5, if the model is overspecified, the complexity penalty in the marginal likelihood shrinks the weights ($\{w_a\}$ in Eq. (4.20)) of extraneous components, as a proxy for model selection – an effect similar to *automatic relevance determination* [92]. As per Eq. (4.13), this complexity penalty is a sum of log eigenvalues of a covariance matrix K. Components which do not significantly contribute to model fit will therefore be automatically pruned, as shrinking the weights decreases the eigenvalues of K and thus minimizes the complexity penalty.

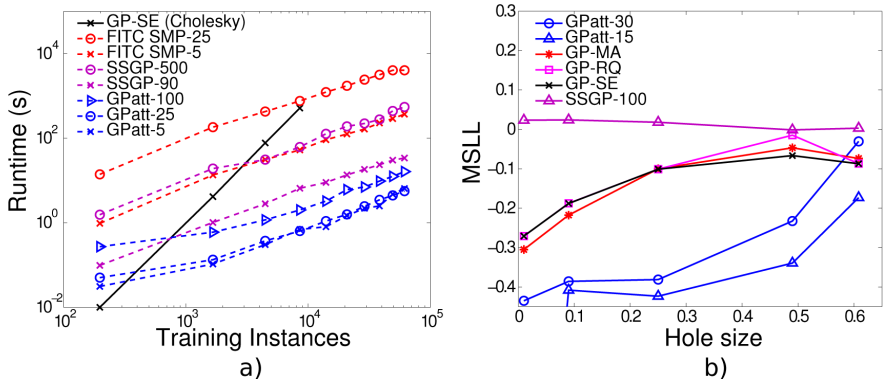


Figure 4.6: Stress Tests. a) **Runtime Stress Test**. We show the runtimes in seconds, as a function of training instances, for evaluating the log marginal likelihood, and any relevant derivatives, for a standard GP with SE kernel (as implemented in GPML), FITC with 500 inducing (pseudo) inputs and SMP-25 and SMP-5 kernels, SSGP with 90 and 500 basis functions, and GPatt-100, GPatt-25, and GPatt-5. Runtimes are for a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor, on the cone pattern ($P = 2$), shown in the supplement. The ratio of training inputs to the sum of imaginary and training inputs for GPatt (section 4.3.3) is 0.4 and 0.6 for the smallest two training sizes, and 0.7 for all other training sets. b) **Accuracy Stress Test**. MSLL as a function of holesize on the metal pattern of Figure 4.4. The values on the horizontal axis represent the fraction of missing (testing) data from the full pattern (for comparison Fig 4.4a has 25% missing data). We compare GPatt-30 and GPatt-15 with GPs with SE, MA, and RQ kernels (and the inference of section 4.3.3), and SSGP with 100 basis functions. The MSLL for GPatt-15 at a holesize of 0.01 is -1.5886.

4.5.6 Stress Tests

We stress test GPatt and alternative methods in terms of speed and accuracy, with varying datasizes, extrapolation ranges, basis functions, inducing (pseudo) inputs, and components. We assess accuracy using standardized mean square error (SMSE) and mean standardized log loss (MSLL) (a scaled negative log likelihood), as defined in [122] on page 23. Using the empirical mean and variance to fit the data would give an SMSE and MSLL of 1 and 0 respectively. Smaller SMSE and more negative MSLL values correspond to better fits of the data.

The runtime stress test in Figure 4.6a shows that the number of components used in GPatt does not significantly affect runtime, and that GPatt is much faster than FITC (using 500 inducing inputs) and SSGP (using 90 or 500 basis functions), even with 100 components

(601 kernel hyperparameters). The slope of each curve roughly indicates the asymptotic scaling of each method. In this experiment, the standard GP (with SE kernel) has a slope of 2.9, which is close to the cubic scaling we expect. All other curves have a slope of 1 ± 0.1, indicating linear scaling with the number of training instances. However, FITC and SSGP are used here with a *fixed* number of inducing inputs and basis functions. More inducing inputs and basis functions should be used when there are more training instances – and these methods scale quadratically with inducing inputs and basis functions for a fixed number of training instances. GPatt, on the other hand, can scale linearly in runtime as a function of training size, without any deterioration in performance. Furthermore, the big gaps between each curve – the fixed 1-2 orders of magnitude GPatt outperforms alternatives – are as practically important as asymptotic scaling.

The accuracy stress test in Figure 4.6b shows extrapolation (MSLL) performance on the metal tread plate pattern of Figure 4.4c with varying holesizes, running from 0% to 60% missing data for testing (for comparison the hole shown in Figure 4.4a is for 25% missing data). GPs with SE, RQ, and MA kernels (and the fast inference of section 4.3.3) all steadily increase in error as a function of holesize. Conversely, SSGP does not increase in error as a function of holesize – with finite basis functions SSGP cannot extract as much information from larger datasets as the alternatives. GPatt performs well relative to the other methods, even with a small number of components. GPatt is particularly able to exploit the extra information in additional training instances: only when the holesize is so large that over 60% of the data are missing does GPatt's performance degrade to the same level as alternative methods.

In Table 4.2 we compare the test performance of GPatt with SSGP, and GPs using SE, MA, and RQ kernels, for extrapolating five different patterns, with the same train test split as for the tread plate pattern in Figure 4.4. All patterns are shown in the supplement. GPatt consistently has the lowest SMSE and MSLL. Note that many of these datasets are sophisticated patterns, containing intricate details which are not strictly periodic, such as lighting irregularities, metal impurities, etc. Indeed SSGP has a periodic kernel (unlike the SMP kernel which is not strictly periodic), and is capable of modelling multiple periodic components, but does not perform as well as GPatt on these examples.

We end this section with a particularly large example, where we use GPatt-10 to perform learning and exact inference on the *Pores* pattern, with 383400 training points, to extrapolate

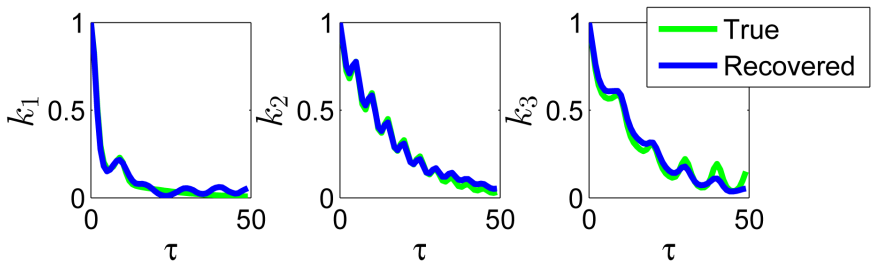


Figure 4.7: Recovering sophisticated product kernels. A product of three kernels (shown in green) was used to generate a movie of 112500 3D training points. From this data, GPatt-20 reconstructs these component kernels (the learned SMP-20 kernel is shown in blue). All kernels are a function of $\tau = x - x'$. For clarity of presentation, each kernel has been scaled by $k(0)$.

a large missing region with 96600 test points. The SMSE is 0.077, and the total runtime was 2800 seconds. Images of the successful extrapolation are shown in the supplement.

4.5.7 Recovering Complex 3D Kernels From a Video

With a relatively small number of components, GPatt is able to accurately recover a wide range of product kernels. To test GPatt's ability to recover ground truth kernels, we simulate a $50 \times 50 \times 50$ movie of data (e.g. two spatial input dimensions, one temporal) using a GP with kernel $k = k_1 k_2 k_3$ (each component kernel in this product operates on a different input dimension), where $k_1 = k_{\text{SE}} + k_{\text{SE}} \times k_{\text{PER}}$, $k_2 = k_{\text{MA}} \times k_{\text{PER}} + k_{\text{MA}} \times k_{\text{PER}}$, and $k_3 = (k_{\text{RQ}} + k_{\text{PER}}) \times k_{\text{PER}} + k_{\text{SE}}$. ($k_{\text{PER}}(\tau) = \exp[-2\sin^2(\pi\,\tau\,\omega)/\ell^2]$, $\tau = x - x'$). We use 5 consecutive 50×50 slices for testing, leaving a large number $N = 112500$ of training points, providing much information to learn the true generating kernels. Moreover, GPatt-20 can reconstruct these complex out of class kernels in under 10 minutes. We compare the learned SMP-20 kernel with the true generating kernels in Figure 4.7. In the supplement, we show true and predicted frames from the movie.

4.5.8 Wallpaper and Scene Reconstruction

Although GPatt is a general purpose regression method, it can also be used for inpainting: image restoration, object removal, etc.

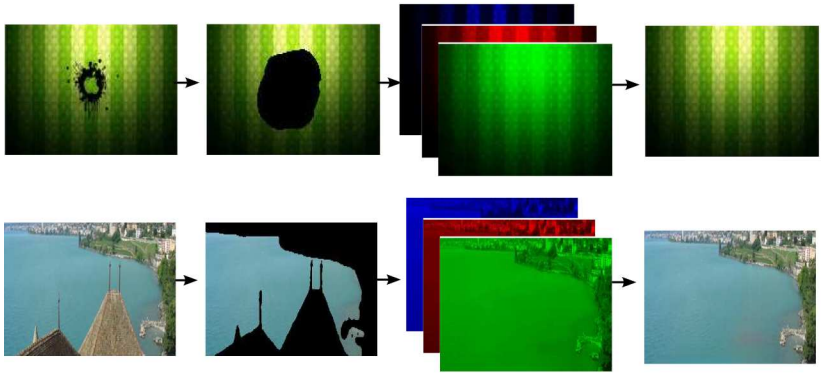


Figure 4.8: Image inpainting with GPatt. From left to right: A mask is applied to the original image, GPatt extrapolates the mask region in each of the three (red, blue, green) image channels, and the results are joined to produce the restored image. Top row: Removing a stain (train: 15047×3). Bottom row: Removing a rooftop to restore a natural scene (train: 32269×3). We do not attempt to extrapolate the coast, which is masked during training.

We first consider a wallpaper image stained by a black apple mark, shown in Figure 4.8. To remove the stain, we apply a mask and then separate the image into its three channels (red, green, and blue), resulting in 15047 pixels in each channel for training. In each channel we ran GPatt using SMP-30. We then combined the results from each channel to restore the image without any stain, which is especially impressive given the subtleties in the pattern and lighting.

In our next example, we wish to reconstruct a natural scene obscured by a prominent rooftop, shown in the second row of Figure 4.8. By applying a mask, and following the same procedure as for the stain, this time with 32269 pixels in each channel for training, GPatt reconstructs the scene without the rooftop. This reconstruction captures subtle details, such as waves in the ocean, even though only one image was used for training. In fact this example has been used with inpainting algorithms which were given access to a repository of thousands of similar images [63]. The results emphasized that conventional inpainting algorithms and GPatt have profoundly different objectives, which are sometimes even at cross purposes: inpainting attempts to make the image look good to a human (e.g., the example in [63] placed boats in the water), while GPatt is a general purpose regression algorithm, which simply aims to make accurate predictions at test input locations, from training data alone.

4.6 Open Extensions to GP-grid

In this section I will describe extensions to the GP-grid algorithm of Sec. 4.3.3. Although the extensions work well in practice, further work is necessary for understanding when these extensions will fail. I will present some of our initial directions in the hope that they will be helpful for future research.

4.6.1 Extension to Logdet Approximation for Small Datasets

As discussed in Sec. 4.3.3, the approximation $\tilde{\lambda}_i^M \approx \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$ is a particularly good approximation for large M (e.g., $M > 1000$) [153]. However, this approximation might prove problematic for complex kernels with short length scales. In these cases it might be necessary to extend the approximation in order to achieve a tighter bound. In this section we will show an extension to the case where the number of data points M is small with comparison to the amount of samples needed for fully expressing the Gaussian process kernel.

Numerical approximation of continues process

Let us start by understanding why the $\tilde{\lambda}_i^M \approx \frac{M}{N}\lambda_i^N$ approximation of [153] works. Kernel matrices are realizations of an underline continues process, where the matrix eigenvalues and eigenvectors are discretized approximations of the continues process eigenvalues and eigenfunctions [6].

$$\lambda_i \phi_i(\mathbf{x}') = \int k(\mathbf{x}, \mathbf{x}') p(\mathbf{x}) \phi(\mathbf{x}) d\mathbf{x} \simeq \frac{1}{N} \sum_{i=1}^{N} k(\mathbf{x}_l, \mathbf{x}') \phi_i(\mathbf{x}_l) \tag{4.23}$$

where the $\mathbf{x}_l$'s are sampled from $p(\mathbf{x})$. Plugging in $\mathbf{x}' = \mathbf{x}_l$ for $l = 1, \ldots, N$ we obtain the matrix eigenproblem

$$K\mathbf{u}_i = \lambda_i^{\text{mat}} \mathbf{u}_i \tag{4.24}$$

with $\phi_i(\mathbf{x}_j) \sim \sqrt{N}(\mathbf{u}_i)_j$, and

$$\lambda_i \approx \frac{1}{N}\lambda_i^{\text{mat}} = \frac{1}{N}\lambda_i^N \tag{4.25}$$

[Rasmussen 2006], where we use the convention λ_i^N for the eigenvalue of λ_i^{mat} of a matrix of size $N \times N$. In [6], the author showed that approximation λ_i^N improves as N increases, and converge to λ_i as $N \to \infty$. Since this approximation is true for every N, for a different discretization of size M we have $\frac{1}{M}\lambda_i^M \approx \lambda_i \approx \frac{1}{N}\lambda_i^N$, hence we can use $\frac{N}{M}\lambda_i^M$ as an approximation for λ_i^N.

This approximation is the basis for the Nyström method [6,153] , where a large p.s.d. matrix N (size $N \times N$) is approximated by a low-rank p.s.d. matrix M (size $M \times M$) using the top ρ eigenvectors and eigenvalues. The Nyström equations are given by:

$$\lambda_i^N \overset{def}{=} \frac{N}{M}\lambda_i^M, \qquad i = 1 \ldots, \rho \tag{4.26}$$

$$\mathbf{u}_i^N \overset{def}{=} \sqrt{\frac{M}{N}}\frac{1}{\lambda_i^M}K_{N,M}\mathbf{u}_i^M, \qquad i = 1 \ldots, \rho \tag{4.27}$$

However, for this approximation to work well we need that $\rho \leq M \leq N$, meaning that the low-rank matrix must have more samples then the underline rank of the continuous process.

For the case of $\rho \geq M$, the approximation of Eq. 4.26 can be highly inaccurate.

Improving the Logdet Approximation

To improve the logdet approximation for the case of $\rho \geq M$, we will use another equality for stationary kernels.

For a stationary kenrel $k(\cdot)$, we have:

$$\text{tr}(K^N) = Nk(0) = \sum_{i=1}^{N} \lambda_i^N, \tag{4.28}$$

for any N. In particular then:

$$k(0) = \frac{1}{N}\sum_{i=1}^{N} \lambda_i^N = \frac{1}{M}\sum_{i=1}^{M} \lambda_i^M. \tag{4.29}$$

Let us deal with the first sum, where we will multiply and divide by a sum over the first M eigenvalues of the K^N matrix:

$$k(0) = \frac{1}{N}\left(\frac{\frac{1}{M}\sum_{i=1}^{M}\lambda_i^N}{\frac{1}{M}\sum_{i=1}^{M}\lambda_i^N}\right)\sum_{i=1}^{N}\lambda_i^N, \tag{4.30}$$

which, but rearranging terms, we get:

$$k(0) = \frac{1}{M}\underbrace{\left(\frac{M}{N}\right)}_{R}\underbrace{\frac{1}{\left(\frac{\sum_{i=1}^{M}\lambda_i^N}{\sum_{i=1}^{N}\lambda_i^N}\right)}}_{\phi^{-1}}\sum_{i=1}^{N}\lambda_i^N, \tag{4.31}$$

Returning to Equation (4.32), we then get:

$$k(0) = \frac{1}{M}\sum_{i=1}^{M}\frac{R}{\phi}\lambda_i^N = \frac{1}{M}\sum_{i=1}^{M}\lambda_i^M \rightarrow \sum_{i=1}^{M}\frac{R}{\phi}\lambda_i^N = \sum_{i=1}^{M}\lambda_i^M, \tag{4.32}$$

So far all of the equations were exact, however, it is difficult to extend the results to the log determinant calculation since here we need to calculate $\sum_{i=1}^{M}\log(\lambda_i^M)$. Although it is possible to show, using the Jensen's inequality, that both $\sum_{i=1}^{M}\log\left(\frac{R}{\phi}\lambda_i^N\right)$ and $\sum_{i=1}^{M}\log|\lambda_i^M|$ are lower bounds of $M\log(k(0))$, it is still difficult to bound the error between the two terms. However, using $\frac{R}{\phi}\lambda_i^N$ as an improve approximation for λ_i^M, for $i = 1, \ldots, M$, works well in practice. Notice that in cases where $\rho \leq M \leq N$, $\phi \rightarrow 1$, which will result back in the previous approximation of $\tilde{\lambda}_i^M \approx \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$ [153].

4.6.2 Extension to Additive Multidimensional Kernels

In this section we will extend GP-grid to efficiently handle the general case of

$$\mathbf{K}_M = \mathbf{K}_{M1} + \mathbf{K}_{M2} + \ldots + \mathbf{K}_{MZ} + \mathbf{D} + \sigma^2\mathbf{I} \tag{4.33}$$

where $\mathbf{K}_M$ is composed of Z $[M \times M]$ multiplicative matrices $\mathbf{K}_{Mz}$, a positive diagonal matrix $\mathbf{D}$ representing known heteroscedastic noise, and an unknown global homogeneous noise.

Inference

To compute $(\mathbf{K}_M)^{-1}\mathbf{y}$ we will again use the PCG calculations in Eq. (4.14). The PCG iterations will now be of the form:

$$\mathbf{K}_M\mathbf{v} = \sum_{z=1}^{Z} \mathbf{E}_z\mathbf{K}_{Nz}\mathbf{E}_z^\top\mathbf{v} + \left(\mathbf{D} + \sigma^2 I\right)\mathbf{v}. \tag{4.34}$$

Hence, the complexity of will only change by a factor of Z, resulting in $\mathcal{O}(JZPN^{\frac{P+1}{P}})$ total operations (where the number of iterations $J \ll N$) to compute $(\mathbf{K}_M)^{-1}\mathbf{y}$, which allows for exact inference.

Learning

We showed that for $Z = 1$ we could use an upperbound to approximate the logdet term (Eq. (4.16)).

Next, we will handle the cases where $Z > 1$, given by

$$\log|\mathbf{K}_M| = \log|\mathbf{K}_{M1} + \mathbf{K}_{M2} + \ldots + \mathbf{K}_{MZ} + D + \sigma^2\mathbf{I}|. \tag{4.35}$$

We can not use the upperbound calculations as before on this equation since it does not generalize to multiple hermitian matrices case. However, we can still construct a tight lowerbound and use that as an approximation. As in Sec. 4.3.3, we again use the results of [41] showing that for $n \times n$ hermitian positive semidefinite matrices $\mathbf{A}$, $\mathbf{B}$ with eiganvalues $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$ and $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$, respectively,

$$\prod_{i=1}^{n}(\alpha_i + \beta_i) \leq |\mathbf{A} + \mathbf{B}|. \tag{4.36}$$

These estimates are best possible in terms of the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$.

The lower bound of Eq. 4.36, can be generalized to the case of sum over Z hermitian positive semidefinite matrices. Using Eq. 4.36, we can bound Eq. 4.35

$$\log|\mathbf{K}_M| = \log|\mathbf{K}_{M1} + \mathbf{K}_{M2} + \ldots + \mathbf{K}_{MZ} + \mathbf{D} + \sigma^2\mathbf{I}| \tag{4.37}$$

$$\geq \log \prod_{i=1}^{M} \left(d_i + \sigma^2 + \sum_{z=1}^{Z} \lambda_{zi}^{M} \right) \tag{4.38}$$

$$= \sum_{i=1}^{M} \log \left(d_i + \sigma^2 + \sum_{z=1}^{Z} \lambda_{zi}^{M} \right) \tag{4.39}$$

$$\simeq \sum_{i=1}^{M} \log \left(d_i + \sigma^2 + \sum_{z=1}^{Z} \frac{R_z}{\psi_z} \lambda_{zi}^{N} \right) \tag{4.40}$$

where $d_i = \text{sort}(\text{diag}(\mathbf{D}))_i$. The right side is an approximation to the lowerbound on the complexity of the general kernel matrix.

Solving Eq. 4.37 can be done efficiently in close to $\mathcal{O}(N)$ complexity; however, using a sum-of-kernels can results in an over-specified model. Since an over-specified model is very sensitive to noise and initial conditions, in order to gain anything from using the complicated sum-of-kernels, it is crucial to find ways to constrain the problem. Our empirical experiments showed that under all tested conditions, the SMP kernel (Sec. 4.4) was flexible enough and achieved better results than using a sum-of-kernels. However, since the sum-of-kernels is a more general prior, it might prove important under certain cases, which warrant additional research in the topic.

4.6.3 Approximations Comparison

We compare the enhanced Nyström logdet approximation that was presented in Sec. 4.6.1 to the true logdet value, and to other approximation methods. For the comparison, we ran the GP learning stage with a small number of data points ($M = 1000$), as to allow for exact calculation of the logdet and the NLML terms. For every iteration of the optimization routine we calculated the values of the exact logdet, the enhanced Nyström approximation, the Skilling approximation [165], Hadamard bound, and the Hadamard tight bound [56]. As can be seen from Fig. 4.9, the enhanced Nyström approximation allows for a tight upper-bound that fluctuates closely to the real logdet values. This allows the enhanced Nyström

approximation to be a good scalable replacement of the true logdet term for the optimization routines.

4.7 Discussion and Conclusion

Gaussian processes are perhaps the most popular nonparametric Bayesian method in machine learning, but their adoption across other fields - and notably in application domains - has been limited by their burdensome scaling properties. While important sparsification work has somewhat addressed this scalability issue, the problem is by no means closed. Here we focused on structured GP models, making nontrivial advances to existing lattice-input GP methods in order to extend structured GP techniques into the multidimensional input domain.

Notably, this GP-grid method opens up an entirely new set of applications for GP, such as image and video processing, or financial engineering applications such as implied volatility surfaces. Our future work is pursuing these application domains.

Gaussian processes are often used for smoothing and interpolation on small datasets. However, we believe that Bayesian nonparametric models are naturally suited to pattern extrapolation on large multidimensional datasets, where extra training instances can provide extra opportunities to learn additional structure in data.

The support and inductive biases of a Gaussian process are naturally encoded in a covariance kernel. A covariance kernel must always have some structure to reflect these inductive biases; and that structure can, in principle, be exploited for scalable and exact inference, without the need for simplifying approximations. Such models could play a role in a new era of machine learning, where models are expressive and scalable, but also interpretable and manageable, with simple exact learning and inference procedures.

Understanding how our existing nonparametric models can scale and be used in real data, and how these models connect to other areas of statistics, will increase the utility of machine learning algorithms in general. This is perhaps most important with Gaussian processes, which promise a wide range of useful applications. The code is available at `https://mloss.org/software/view/503/`.

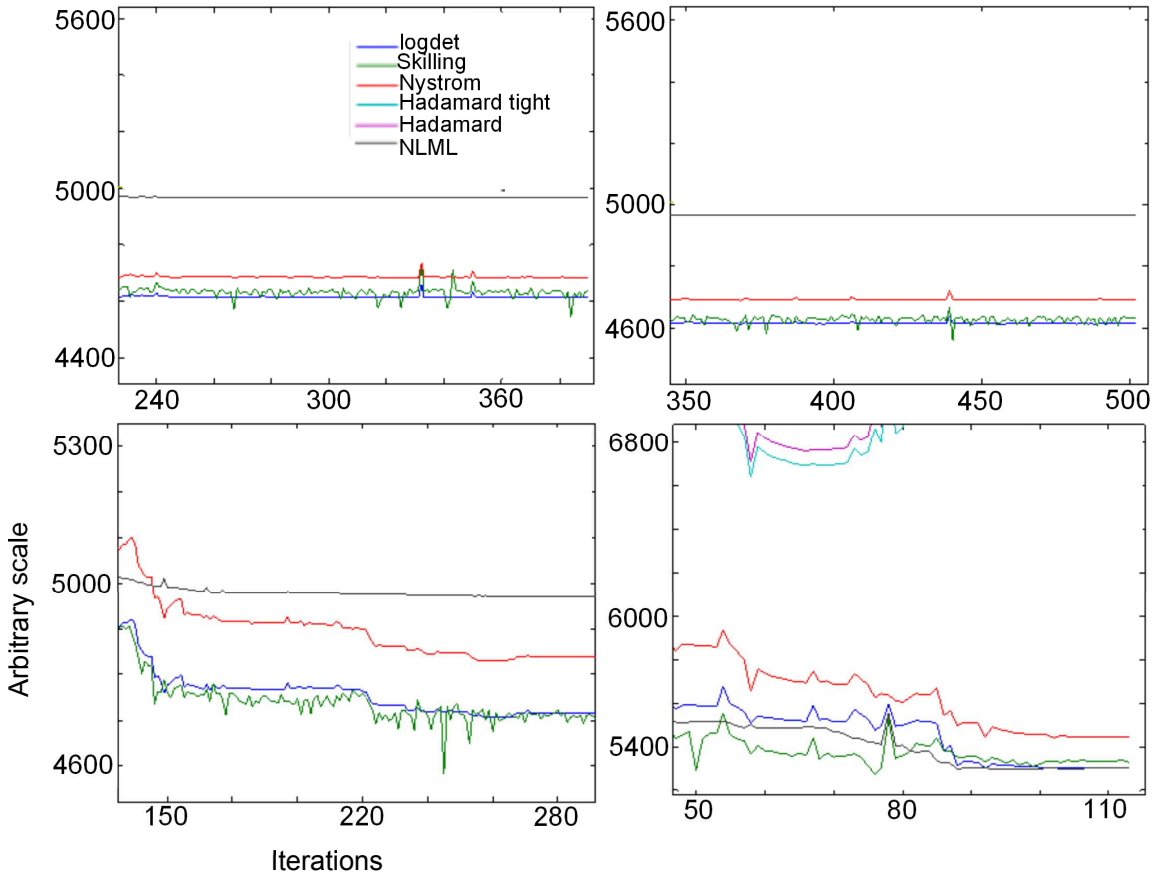


Figure 4.9: Comparison of different approximations for logdet. We present the values of the different approximations for multiple optimization iterations of the GP Algorithm with a SMP-30 kernel. We used a small number of data points ($M = 1000$) as to allow for exact calculation of the logdet and the NLML terms. As can be seen from the figures, the Skiling approximation is the tightest around the true logdet; however, its random fluctuations make it impractical to be used in an optimization routine. The Nyström method (Sec. 4.6.1) is an upperbound, and show similar fluctuations, to the true logdet. The Hadamard approximations are very loose (out of the scope of the plots) and show very different trends than the true logdet.

Table 4.1: Comparison of standardized MSE interpolation results for images with additive variable Gaussian noise. We tested each image when taken as a whole (W), object segment (O), and background segment (B).

	Sphere			Moose			Cone			Face		
Alg	O	B	W	O	B	W	O	B	W	O	B	W
GP-grid(1/2)	0.39	0.40	0.69	1.08	0.35	**0.86**	0.56	0.39	1.74	1.28	0.88	2.67
GP-grid(5/2)	**0.33**	**0.15**	0.73	1.10	**0.08**	0.91	**0.52**	**0.17**	**1.24**	**1.10**	0.70	**1.98**
GP-grid(1/2) sph	0.40	0.30	0.71	2.98	0.11	1.54	0.84	0.19	3.81	1.30	0.69	3.02
GP-grid(5/2) sph	0.38	0.23	0.92	4.37	0.08	3.05	3.33	0.19	4.99	1.73	**0.64**	2.97
Blinear	0.56	0.56	**0.68**	1.12	0.56	0.97	0.64	0.61	1.63	1.27	1.06	2.60
Bicubic	0.59	0.61	0.83	**1.01**	0.57	0.91	0.69	0.67	2.05	1.32	1.03	2.69
Bic-sp	0.74	0.76	0.80	1.16	0.76	1.06	0.80	0.82	1.24	1.23	1.12	2.04
NEDI	0.56	0.53	0.74	1.11	0.52	0.97	0.72	0.59	1.97	1.61	1.21	3.86

Table 4.2: We compare the test performance of GPatt-30 with SSGP (using 100 basis functions), and GPs using squared exponential (SE), Matérn (MA), and rational quadratic (RQ) kernels, combined with the inference of section 4.5.6, on patterns with a train test split as in the metal treadplate pattern of Figure 4.4.

	GPatt	SSGP	SE	MA	RQ
	Rubber mat		(train = 12675, test = 4225)		
SMSE	**0.31**	0.65	0.97	0.86	0.89
MSLL	**−0.57**	−0.21	0.14	−0.069	0.039
	Tread plate		(train = 12675, test = 4225)		
SMSE	**0.45**	1.06	0.895	0.881	0.896
MSLL	**−0.38**	0.018	−0.101	−0.1	−0.101
	Pores		(train = 12675, test = 4225)		
SMSE	**0.0038**	1.04	0.89	0.88	0.88
MSLL	**−2.8**	−0.024	−0.021	−0.024	−0.048
	Wood		(train = 14259, test = 4941)		
SMSE	**0.015**	0.19	0.64	0.43	0.077
MSLL	**−1.4**	−0.80	1.6	1.6	0.77
	Chain mail		(train = 14101, test = 4779)		
SMSE	**0.79**	1.1	1.1	0.99	0.97
MSLL	**−0.052**	0.036	1.6	0.26	−0.0025

Chapter 5

Image Interpolation and Denoising for Division of Focal Plane Sensors using Gaussian Processes

5.1 Abstract

Image interpolation and denoising are important techniques in image processing. These methods are inherent to digital image acquisition as most digital cameras are composed of a 2D grid of heterogeneous imaging sensors. Current polarization imaging employ four different pixelated polarization filters, commonly referred to as division of focal plane polarization sensors. The sensors capture only partial information of the true scene, leading to a loss of spatial resolution as well as inaccuracy of the captured polarization information. Interpolation is a standard technique to recover the missing information and increase the accuracy of the captured polarization information.

Here we focus specifically on Gaussian process regression as a way to perform a statistical image interpolation, where estimates of sensor noise are used to improve the accuracy of the estimated pixel information. We further exploit the inherent grid structure of this data to create a fast exact algorithm that operates in $\mathcal{O}\left(N^{3/2}\right)$(vs. the naive $\mathcal{O}\left(N^3\right)$), thus making the Gaussian process method computationally tractable for image data. This modeling advance and the enabling computational advance combine to produce significant improvements over previously published interpolation methods for polarimeters, which is most pronounced in cases of low signal-to-noise ratio (SNR). We provide the comprehensive mathematical model as well as experimental results of the GP interpolation performance for division of focal plane polarimeter.

5.2 Introduction

Solid state imaging sensors, namely CMOS and CCD cameras, capture two of the three fundamental properties of light: intensity and color. The third property of light, polarization, has been ignored by traditional sensors primarily due to the fact that our visual system is blind to polarization. However, in nature, many species are capable of sensing polarization properties of light in addition to intensity and color. The visual system in these species combines photoreceptors and specialized optics capable of filtering the polarization properties of the light field. Recent development in nanofabrication and nano-photonics has enabled the realization of compact and high resolution polarization sensors. These sensors, known as division of focal plane polarimeters (DoFP), monolithically integrate pixelated metallic nanowire filters, acting as polarization filters, with an array of imaging elements [57]. One of the main advantages of division-of-focal-plane sensors is the capability of capturing polarization information at every frame. The polarization information captured by this class of sensors can be used to extract various parameters from an imaged scene, such as microscopy for tumor margin detection [88], 3-D shape reconstruction from a single image [99], underwater imaging [128], material classification [74], and cancer diagnosis [3].

The idea of monolithically integrating optical elements with an array of photo sensitive elements is similar to today's color sensors, where a Bayer color filter is integrated with an array of CMOS/CCD pixels. The monolithic integration of optics and imaging elements has been the chief reason for the proliferation of color cameras and the dawn of proliferation of polarization imaging and applications. Another important reason for the wide acceptance of color imaging technologies is the utilization of interpolation algorithms. Since pixelated color filters are placed on the imaging plane, each sensor only observes partial information (one color), and thus it is standard practice to interpolate missing components across the sensors. Cameras using division of focal plane polarization sensors utilize four pixelated polarization filters, whose transmission axis is offset by 45 degree from each other (see Fig. 5.1). Hence, each pixel represents only 1/4 of the polarization information, severely lowering the resolution of the image and the accuracy of the reconstructed polarization information.[13] Interpolation techniques allow to estimate the missing pixel orientations across the imaging array, thereby

[13] Although the four pixels are spatially next to each other, their instantaneous field of view is different, resulting in different intensities. The pixels intensities are used to reconstruct the polarization information, such as Stokes vector, angle and degree of linear polarization.

improving both the accuracy of polarization information and mitigate the loss of spatial resolution.

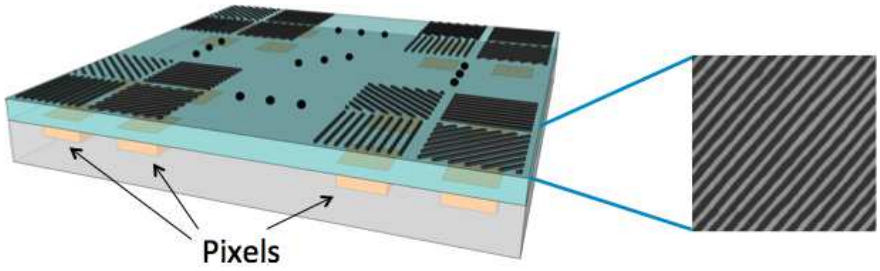


Figure 5.1: Division of focal plane polarization sensors on the imaging plane. Each pixel captures only 1/4 of the polarization information. Interpolation is needed for recovering the full camera resolution.

Similar problems were encountered in color imaging sensors, when the Bayer pattern for pixelated color filters was introduced in the 1970s [11]. In order to recover the loss of spatial resolution in color sensors and improve the accuracy of the captured color information, various image interpolation algorithms have been developed in the last 30 years. For DoFP, the common use is of conventional image interpolation algorithms such as bilinear, bicubic, and bicubic-spline, which are based on space-invariant non-adaptive linear filters [49,50,123, 147]. More advanced algorithms for color sensors use adaptive algorithms, such as the new edge-directed interpolation (NEDI) [82], or utilize the multi-frame information, such as [60]. Recently, there has been a growing interest in the use of GP regression for interpolation and denoising of image data for color sensors [64,87]. GP is a Bayesian nonparametric statistical method that is a reasonable model for natural image sources [82], and fits well the DoFP interpolation setting by accounting for heteroscedastic noise and missing data, as we will discuss next.

Whereas conceptually attractive, exact GP regression suffers from $\mathcal{O}(N^3)$ runtime for data size N, making it intractable for image data (N is the number of pixels which is often on the order of millions for standard cameras). Many authors have studied reductions in complexity via approximation methods, such as simpler models like kernel convolution [66,161], moving averages [159], or fixed numbers of basis functions [29]. A significant amount of research has also gone into sparse approximations, including covariance tapering [48,139], conditional independence to inducing inputs [119,121], sparse spectrum [79], or a Gaussian Markov random

field approximation [126]. While promising, these methods can have significant approximation penalties [27, 119]. A smaller number of works investigate reduction in complexity by exploiting special structure in the data, which avoids the accuracy/efficiency tradeoff at the cost of generality. Examples of such problems are: equidistant univariate input data where the fast Fourier transform can be used (e.g., [30]); additive models with efficient message passing routines (e.g., [37, 51, 52]); and multiplicative kernels with multidimensional grid input [52], as we discuss here.

Image data is composed of multiple pixels that lie on a two dimensional grid. However, the input data might not lie on a complete grid. This can often occur due to missing values (e.g., malfunctioning sensors), or when analyzing irregular shaped segment of the image. Furthermore, captured image data often contain heteroscedastic signal-dependent noise [110]. In fact, the heteroscedastic nature of the data is often overlooked by most of the image interpolation techniques in the literature and can significantly reduce the interpolation accuracy (see Sec. 4.5). The multidimensional grid input data induces exploitable algorithmic structures, as described in Chapter 4, that can naturally utilize the actual noise statistics of the data acquisition system. With these advances, it is possible to significantly improve both the interpolation and denoising performance over current methods.

The main contribution of this chapter is to present an efficient GP inference for improved interpolation for DoFP polarimeters. The GP statistical inference is able learn the properties of the data and incorporates an estimation of the sensor noise in order to increase the accuracy of the polarization information and improve spatial resolution.

5.3 Application to Division of Focal Plane Images

In this section we test GP-grid on real division of focal plane image data, and we demonstrate improvement in accuracy of polarization (Stokes) parameters compared to commonly-used methods. For better comparison we used four different scenes, each captured multiple times using (i) a short shutter speed resulting in low signal-to-noise ratio (SNR) images, and (ii) a long shutter speed resulting in high SNR images. We acquired hundreds of images for each scene using a CCD imaging array (Kodak KAI-4022 4MP) and a polarization filter. We used four polarization filters corresponding to angles: 0, 45, 90, and 135 (see Fig. 5.2).

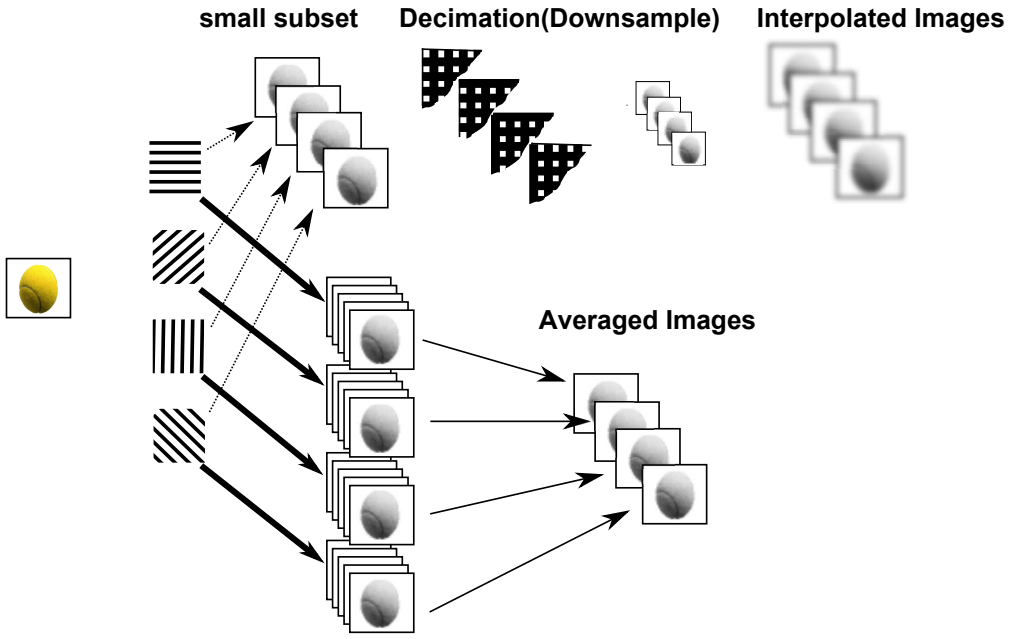


Figure 5.2: The original image on the left is passed through four polarization filters with different phases. Over a hundred filtered images are captured. A small subset of the filtered images is used for the interpolation testing and the rest are averaged to approximate the noiseless filtered images. The filtered images used for testing are downsampled by four (using different downsampling patterns) and then interpolated back to original size.

To extract the noiseless ground-truth (the basis of our comparisons), we averaged over the majority of the pictures, holding out a small subset for testing. In order to test the interpolation performance, we interpolated the entire image using only a subset of the image (downsampled by four). All images are around 40000 pixels, hence, even their down-sampled version will be impractical for the standard naive GP implementation. The interpolated images were then compared to their corresponding averaged images for accuracy analysis. The accuracy criterion we used was the normalized mean square error (NMSE) between the interpolated images and the average images, defined as:

$$\text{NMSE}(y, \bar{y}) = \frac{\frac{1}{N}\sum_i^N (y_i - \bar{y}_i)^2}{var(\bar{y})}, \tag{5.1}$$

where $\bar{y}$ is the data of averaged image.[14] Normalization is used in order to compare between the results of the low and high SNR images since they have a different intensity range.

[14]If we consider $\bar{y}$ to be our signal, the NMSE can be seen as an empirical inverse of the SNR.

We compare GP-grid with the common interpolation algorithms: bilinear, bicubic, bicubic-spline (Bic-sp) and NEDI [82]. Although this is by no means an exhaustive comparison, it does allow for a benchmark for comparison with GP performance. Note that in all the comparisons we intentionally discarded the border pixels (five pixels width) so they would be most favorable to non-GP methods as the non-GP methods fail particularly badly at the edges. Had we included the border pixels, our GP-grid algorithm would perform even better in comparison to conventional methods.

We explore real data using our improved GP-grid model. Performance of course depends critically on the noise properties of the system, which in captured images is primarily sensor noise. Other works in the literature consider additional GPs to infer a heteroscedastic noise model [53, 155], which brings additional computational complexity that is not warranted here. Instead, the simple model of Eq. (4.22) works robustly and simply for this purpose. We ran GP-grid using a multiplicative Matérn($\frac{1}{2}$) covariance function, and learned the hyperparameters: lengthscales (l_1, l_2), signal variance (σ_f^2) [121].

The first set of results presented is for the "Mug" scene (Fig. 5.3). The scene is composed of a bright mug in front of a bright background. The brightness of the images is important as a brighter image will produce higher luminance and a higher signal in the camera. The top row of the figure shows a summary of the results for short shutter speed images and the bottom row shows the results for long shutter speed images. As can be expected, the intensity range of the low SNR test image on the top is much lower than the high SNR test image on the bottom. Also, we can see that the normalized error is significantly higher for the top row (NMSE = 0.062904) than the bottom row (NMSE = 0.017713). Following the scheme presented in Fig. 5.2, we used the interpolated and averaged images to compute the Stokes parameters

$$S_0 = I_0 + I_{90}, \qquad S_1 = I_0 - I_{90}, \qquad S_2 = I_{45} - I_{135}. \tag{5.2}$$

In the left side of Fig. 5.3 we show a comparison of the normalized error between the Stokes parameters calculated using the interpolated images (using different interpolation methods) and the Stokes parameters calculated from the averaged images. It is clear that GP outperformed all the other methods in this scene.

A similar comparison was done for the three additional scenes: Horse (Fig. 5.4), Toy (Fig. 5.5), and Tennis ball (Fig. 5.6). Differently than the Mug scene analysis, here we separated the comparison for the object and the background. The reason for the separation is because the two segments have very different properties (spatial frequencies) and learning on the entire image will result in a kernel that will be suboptimal on each region separately. Close analysis of the results show two important facts. First, the improvement was higher for low SNR images than high SNR images. This is not surprising as all the interpolation methods are excepted to perform well when the noise level is low compared to the signal level. Second, S_1 and S_2 show higher improvement compared to S_0. This is because S_0 by construction is less sensitive to noise (similar to an average operator), while S_1 and S_2 are significantly more sensitive. The results of the computed Stokes parameters for the tennis ball scene in Fig. 5.7 clearly show the failure of the common interpolation algorithms for low SNR images. Improving the accuracy of S_1 and S_2 are especially important because of their nonlinear dependent to the other polarization parameters: angle of polarization $\phi = (1/2)\tan^{-1}(S_2/S_1)$ and the degree of linear polarization $\rho = \sqrt{S_1^2 + S_2^2}/S_0$.

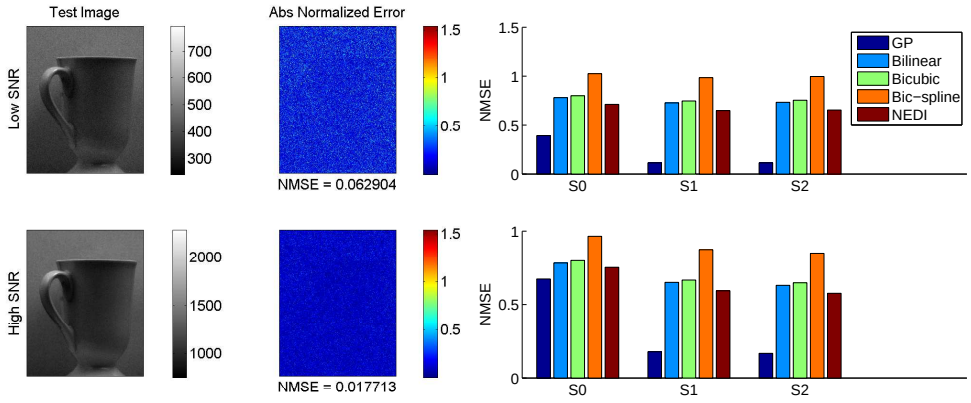


Figure 5.3: Left column shows the noisy test image before decimation (subsampling) and interpolation. Middle column illustrates the absolute normalized error for each pixel and the average NMSE. As can be seen, in the low SNR image the normalized error values are much higher than in the high SNR image. The Stokes parameters comparison is shown on the right for the five interpolation methods tested.

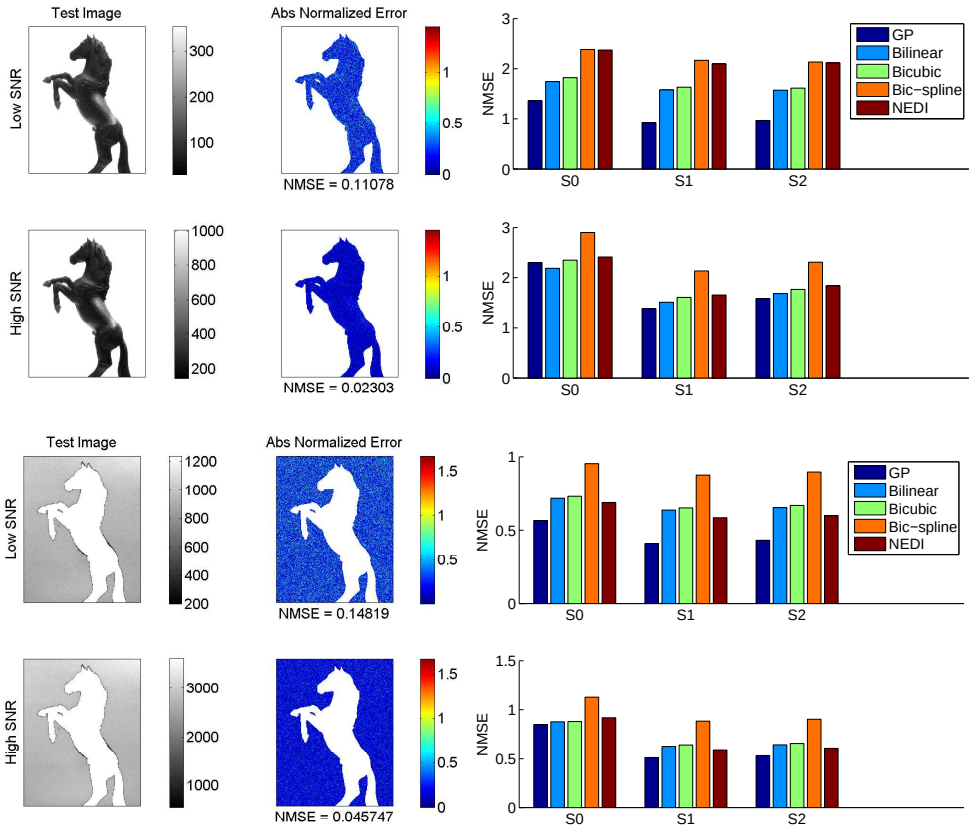


Figure 5.4: Horse Scene. See caption of Fig. 5.3.

5.4 Conclusion

Image analysis is a critical and common machine learning application, but the use of non-parametric Bayesian algorithms in this domain is infrequent. Our GP-grid algorithm importantly enables the use of exact GP technologies in this application area by greatly reducing the computational burden of full-GP. In this paper we showed that GP-grid can be generalized to incorporate incomplete grids and heteroscedastic noise, both of which contribute to significant improvement in interpolation and denoising of image data. In this work we did not consider the question of optimal segmentation for the use of GP, which is a research topic in its own right. Furthermore, other methods such as spline based interpolation could perhaps be extended to achieve similar improvements when using the camera specific noise model, but that is beyond the scope of this work. Here we have shown that GP-grid is a useful and elegant tool to enable these large performance improvements.

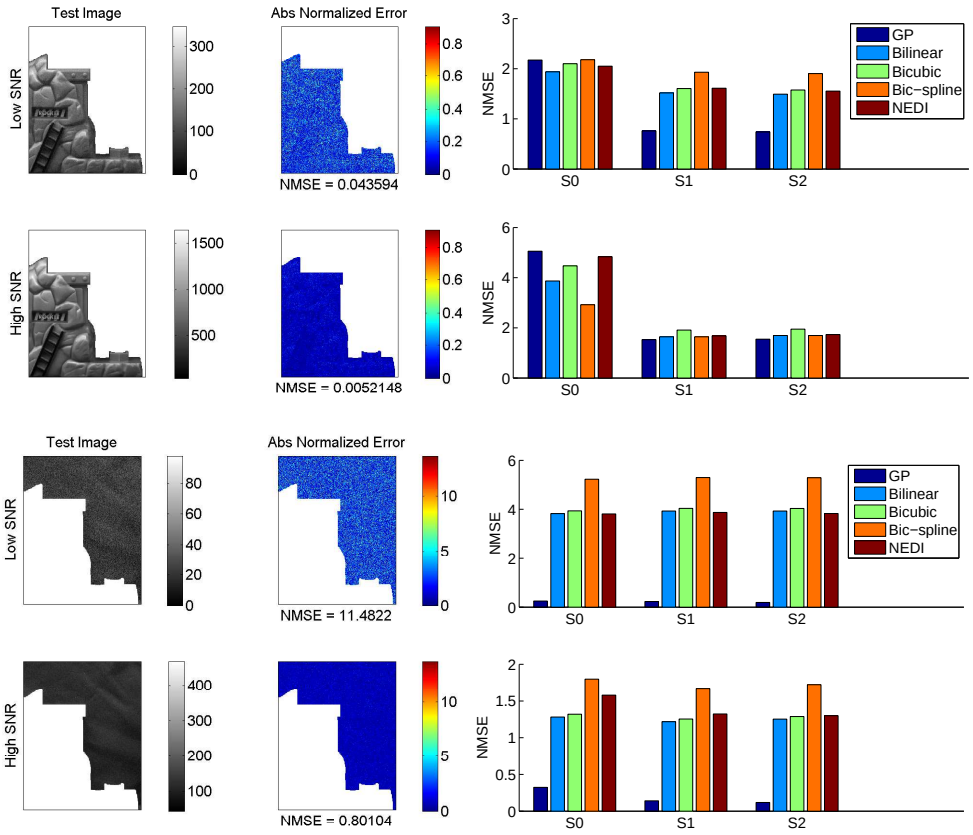


Figure 5.5: Toy Scene. See caption of Fig. 5.3.

Overall, the results show that the GP framework allows for improved results over conventional interpolation methods. GP allows for statistical interpolation that can naturally incorporate the camera noise model, and improvement is most evident for low SNR images where having a good prior can help reduce the effect of the noise.

Another interesting realization that came out of the comparison presented in this paper is that the Bicubic-spline algorithm performance greatly degrades in the presence of noise. This result is different than other papers in the literature where the comparison was done on the averaged images only [49, 50].

GP becomes tractable for image data by using the GP-grid algorithm we introduce here, and it is a convenient technology to naturally incorporate our two performance-critical advances: segmentation (incomplete grids) and a known noise model. As the results show, all of these advances are important in order for GP to be considered a general framework for image data.

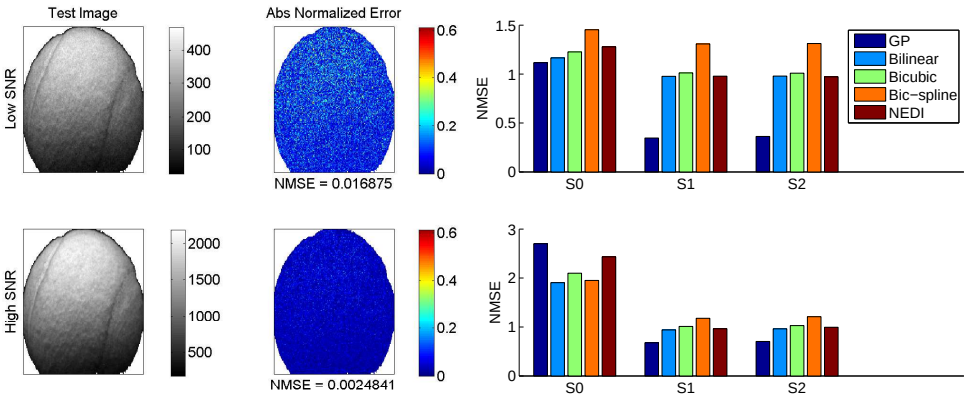
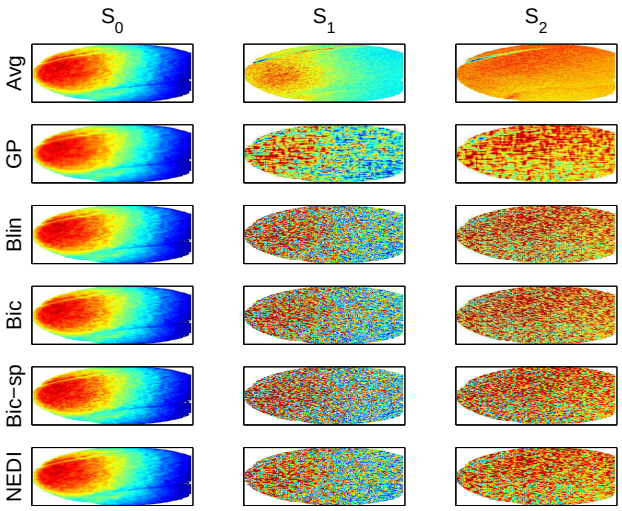
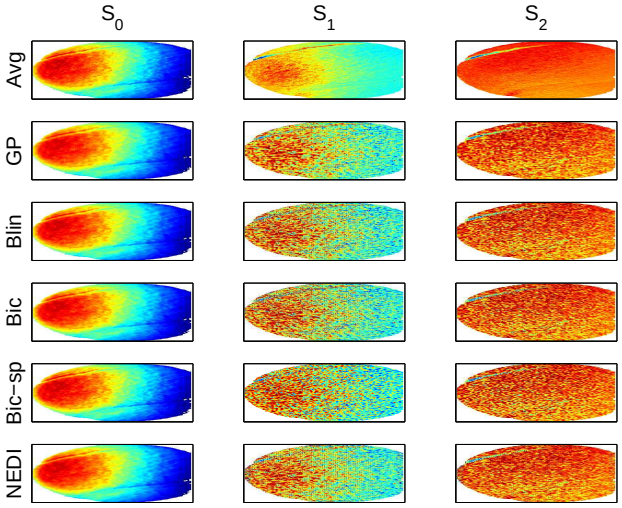


Figure 5.6: Tennis Ball Scene. See caption of Fig. 5.3.

It is common practice in image processing to mix different methods in order to improve the overall results, e.g., alternate methods close to an edge. Integrating GP-grid together with other state-of-the-art interpolation methods to achieve further improvement is an interesting topic for future work.



(a) Low SNR



(b) High SNR

Figure 5.7: Results of the Stokes parameters for the different interpolation methods. The first row shows the Stokes parameters computed using the averaged images which we use as the underline ground truth. The GP interpolation achieves significantly better polarization accuracy in the low SNR case. The improvement is most evident for the S_1 and S_2 parameters since they are more susceptible to noise.

Chapter 6

Gaussian Processes for Denoising fMRI Data

6.1 Abstract

Traditional approaches to denoising fMRI data underutilize the rich information available in the data. A more powerful approach utilizes spatiotemporal models that exploit temporal and spatial correlations. Fitting a full spatiotemporal model to fMRI data is highly challenging. Although there is a growing interest in Bayesian nonparametric frameworks for inference with spatiotemporal models, they are computationally intensive for practical use. As an alternative, we offer a novel methods based on Gaussian processes regression, which recent advances have made attractive for denoising fMRI data. Our GP-based framework learns and adapts to spatial neural activity patterns and applies smoothing based on the noise levels.

6.2 Introduction

The high spatial resolution of functional Magnetic Resonance Imaging (fMRI) allows studying fine-scale activity patterns in the brain. However, fMRI data is corrupted by complicated noise from multiple sources, which degrades the sensitivity of statistical tests to differentiate between different conditions. Commonly, noise is reduced in fMRI by spatially smoothing the data as part of a preprocessing step. This approach is based on the assumption of an intrinsic smoothness of neural activations [25, 93]. Under the Matched Filter Theorem, a spatial smoother, which matches the intrinsic smoothness of the neural data, will optimally

detect the neural signal that is embedded in white noise [62]. However, it is widely accepted in neuroscience that smoothing the data carries an inherent tradeoff between sensitivity and the spatial specificity of statistical tests [15, 43, 75, 160].

Sensitivity and specificity are two commonly used metrics for summarizing statistical classification tests. In the context of fMRI, the sensitivity of a method is the ability to correctly detect neural activity across different experimental conditions. The spatial specificity of a method is the ability to correctly exclude spatial inactive regions. This metric captures whether the structure of the spatial neural activity detected is an accurate reflection of the underlying true neural activity structure. Although sensitivity and spatial specificity are functions of the chosen statistical test, they are implicitly also dependent on two properties of the data, namely, the amplitudes of the signal and noise components. A stronger signal allows for clearer separation between conditions, but stronger noise increases the uncertainty in the measured data and blurs the differences between conditions. In the smoother method context, the sensitivity/spatial-specificity tradeoff is more properly recast as the tradeoff between improving the ratio between the amounts of signal to noise in the data (SNR) and corrupting the signal (fidelity).[15]

The majority of smoothing methods in the literature utilize a fixed-width Gaussian spatial smoother. For fixed-width smoothers, there is an inherent tradeoff between sensitivity and spatial specificity. Under-smoothing allows retaining the fine details of the data but will not reduce the noise. Under-smoothing results in low SNR and high fidelity, but often leads to a loss of sensitivity. Over-smoothing the data reduces the noise but also removes the fine details (high SNR, low fidelity, often results in loss of spatial specificity). Finding the optimal tradeoff is difficult and often requires setting multiple parameters, causing many practitioners to either excessively smooth the data or abandon smoothing altogether.

However, the sensitivity and spatial specificity metrics are distinct, and, in theory, a method could achieve good performance on both metrics. Thus, there is not always a tradeoff between the two. The tradeoff is especially pronounced in fixed-width smoothers because they cannot adapt to spatially varying neural activity patterns. For example, some brain regions may have very smoothly varying neural activity patterns, whereas other regions may have quickly changing (rugged) neural activity patterns. Fixed-width smoothers also cannot

[15] In the rest of the paper we will refer to the tradeoff as the sensitivity / spatial-specificity tradeoff for consistency with previous works.

adapt to spatially varying noise levels. For example, for voxels with low noise levels, strong smoothing corrupts the signal far more than it reduces the noise, while weak smoothing is ineffective for high noise voxels [78].

Here we propose the use of Gaussian process (GP) regression for denoising fMRI data. GP is a nonparametric Bayesian regression method that has received much attention in the machine learning literature [121]. Applying GP to fMRI data offers the two main advantages lacking in fixed-width smoothers. First, it adapts to the spatial structure of the signal across voxels, learning the smoother structure from the data. Second, it learns and adapts to the noise level in the data, varying the amount of smoothing applied for each voxel, depending on the learned voxel's noise level.

Although conceptually attractive, GP use in the neuroscience community has been limited by burdensome scaling properties. Naively solving exact GP inference is limited to datasets with only a few thousands data points. In a standard fMRI experiment the number of data points (voxels) can easily reach hundreds of thousands, if not millions, making GP infeasible. Fortunately, recent advances in GP research have allowed significant scaling reduction for certain structured datasets, where the data lie on a multidimensional grid [51]. The grid structure is implicit in fMRI data, where the inputs (voxels) are indexed on a 4D grid (3D space + time). This makes exact GP inference, for the first time, competitive for fMRI analysis.

The central aim of this paper is to introduce efficient GP-based analysis for denoising fMRI data. It is not our intention to compare the GP method to the numerous other smoothing heuristics, but rather show the benefits of GP as an efficient statistical smoothing method. We will show that our GP-based method allows for several advances: 1) It removes ultra-low temporal frequency fMRI noise (henceforth called the drift) by learning the drifts' properties simultaneously on the entire brain. 2) It jointly learns the localized spatial and temporal correlations and the heteroscedastic voxels' noise. 3) It adaptively varies the smoothing level of the voxel. 4) It shows significant improvement for real fMRI data over fixed-width smoothers, while lessening the sensitivity/specificity tradeoff.

6.3 Statistical Smoothing

FMRI data is smoothed both to improve the voxel signal and to reduce unwanted artifacts (noise) by combining information from neighboring voxels. However, there are numerous ways to perform smoothing. A naive way, for example, is to smooth the data by taking localized averages of neighboring voxels in regions of interest. A more realistic method than the simple average is to use weighted averages, where the weight given to neighboring voxels (representing similarity between voxels) decreases with distance. When we model this weight-distance function as a Gaussian, we are performing the common preprocessing step of smoothing with a fixed-width Gaussian kernel. The fixed-width Gaussian kernel is typically set to have a full width at half maximum (FWHM) of around 3-8 mm [109]. Gaussian smoothing is popular for its simplicity and computational efficiency. However, the main drawback with fixed-width smoothers is their inability to address the variation of spatial activation across the brain, leading to a tradeoff between increased sensitivity and spatial specificity [24, 136, 142, 163].

In the literature, a series of methods have been suggested to address the sensitivity/specificity tradeoff by using more complex adaptive methods [83,84,86,115,132,150,160]. The common thread among these methods is adaptively choosing the input of the smoother (informative neighbors) to improve its accuracy.

Other methods considered a statistical smoother approach, adapting the smoothing level according to the degree of uncertainty in the data (e.g., due to high noise variance in a voxel). However, utilizing a noise dependent smoother is challenging since the true voxel noise is unknown and must be estimated from the data. Since traditionally the smoothing step is separated from the statistical model, and since the noise estimation is only as good as the statistical model used, this can cause cyclic problems. For example, a bad model choice will result in a bad noise estimation that can cause unwarranted smoothing, which will result in an even worse model fit. In order to avoid this cycle, a number of Bayesian spatiotemporal models have been proposed to combine the smoother with the statistical model. Using Bayesian models has been done almost exclusively by extending the popular general linearized model (GLM) with the additional smoothness priors on the regression parameters [20, 54, 61, 109, 163]. Although these works presented an important shift from the traditional concept of smoothing, they proved highly computationally intensive and required

significant approximations for running on the massive data of a standard fMRI study. Here, instead, we propose the use of an alternative statistical smoother method based on GP regression, one that will not only enables incorporating the idea of adaptive noise-based smoothing, but also allows for a spatiotemporal model that learns its structure from the data, and runs in practicable time.

6.4 From GLM to GP

The commonly used generalized linear model (GLM) is a parametric mass univariate statistical model (Eq. (6.1), wherein a finite number of parameters (matrix B) control the contributions of predefined predictors (X) in order to best explain the observed data (Y) [20]. [16] In parametric methods, such as GLM, the data is explained using a specific family of statistical models, where the predictors are predefined based on prior knowledge. Parametric models assume that we have sufficient knowledge of the problem to predefine the form of the predictors that will explain the data. For example, it is standard practice in GLM to use a design matrix based on the known experimental design along with predefined hemodynamic response functions (HRF). However, parametric methods critically depend on the choice of model and the assumptions about the data [78]. For example, common basic GLM assumptions are that voxels are independent, time points are independent, the error variance is the same, and the same design matrix based model (and commonly the same the form of HRF) is appropriate for all the voxels [78]. These unrealistic assumptions might prove too restrictive to allow useful inference. Attempting to improve flexibility by increasing the model complexity (e.g., number of predictors) requires fitting multiple parameters (such as shape, curvature, delay, length, amplitude, etc.) and can quickly become computationally prohibitive or overfit to the noise [85]. Placing a distribution over parametric functions (such as in [20, 54, 61, 109]) is challenging since it is difficult to capture properties such as smoothness and differentiability from priors of parameters of complicated predictors [127].

The limitations of parametric methods suggest that we should explore alternative frameworks. A prime candidate is the stochastic processes framework, which allows a natural way to generalize probability distribution to functions [121]. In this family, the Gaussian processes

[16] Other covariates (H), such as the drift, can be included in the model to allow for a better fit to the data.

(GPs) are commonly used in the machine learning community for Bayesian nonparametric regression. GPs are popular for two reasons. First, their ability to learn the model structure from the data makes the method less susceptible to preconceived (and somewhat arbitrary) notions and assumptions [78]. Second, they provide a closed form posterior distribution over functions, which is rarely feasible with other Bayesian models. The form of GLM and GP models can written as:

$$\mathrm{G}LM: \qquad Y(v) = XB(v) + Hu(v) + e(v) \tag{6.1}$$

$$\mathrm{G}P: \qquad y(v,t) = f(v,t) + h(v,t) + e(v) \tag{6.2}$$

$$f \sim \mathcal{GP}(0, k(\cdot)),$$

where the main difference is that GLM is a mass univariate model that uses predefined predictors, while GP allows for more flexible regression functions. The GLM model only change as a function of space, while the GP the model depends on both space and time. In the GLM model the prior knowledge is used to make the predictor matrices X and H, and perhaps for priors over the parameter matrix B. In GP, prior knowledge is used for choosing the covariance function $k(\cdot)$ [121].

Nonparametric approaches are typically computationally intensive. Exact GP regression suffers from $\mathcal{O}(N^3)$ runtime for data size N, making it intractable for fMRI data, where often N is on the order of millions of voxels. Recently, advances in GP regression have allowed overcoming the GP scaling burden in applications that possess a grid data structure [51]. This condition applies very naturally to fMRI data, which lie on a four dimensional grid (3 spatial dimensions + time).[17] New extensions, such as the ability to process incomplete grid data, account for heteroscedastic noise [51], and utilize expressive kernels [157], make GP, for the first time, an attractive model for fMRI data analysis. In our method section we will introduce two GP-based components for drift removal and statistical smoothing. These components can either present an alternative to conventional fMRI data processing methods or work in conjunction with these methods. We will develop this last point further in the discussion.

[17] By considering the runs as another dimension, the data is a 5D structure.

6.5 Validation of the Proposed Method

6.5.1 Comparison using Simulated Data

We start our analysis by comparing our GP smoother with a traditional fixed-width Gaussian smoother. The purpose of this comparison is to present the rational behind the improvement of using GP as an adaptive noise-based smoother. We considered main properties that effect the results of a smoother, namely smoothness, and noise level of the data. We simulated data corresponding to 30 neighboring voxels along the x axis. For simplicity, in this comparison we only consider single dimensional data. We produced two random true plots with different smoothness properties (shown in blue in Fig. 6.1). For each true data we added two levels (low and high) of additive noise in order to get the observed data. For the low noise variance we used 1/20 of the max signal value, and for the high noise we used 1/4 of max signal value. Figure 2.1 illustrates the four conditions tested. We compared our GP smoother with two fixed-width Gaussian smoothers of size 3mm and 6mm FWHM. We ran the analysis multiple times, each time drawing new observed points from the same true data. We used to criterions as a summarizing statistics. We used the signal to signal plus noise ration (SSNR) as a metric to show the reduction of noise compared to the signal. The SSNR was calculated by running the smoother on the signal and noise separately and then use the energies of post-smoother values to calculate ratio. We used the fidelity metric to measure how much was the signal corrupted by the smoother. To measure fidelity we used the normalized correlation between the true data and the data after smoothing. An ideal smoother will significantly reduce the noise without corrupting the signal. Thus, an ideal smoother will have both SSNR and fidelity reaching 1. We show the results of the three smoothers and compare it also to using the unsmoothed data (Raw). It is clear to see from Fig. 6.1 that all the smoothers improve the SNR, however the fixed-width smoothers typically pay for it by corrupting the signal. On the other hand, GP is able to achieve both high SNR and high fidelity. GP superior performance can best be seen in for the rugged data, where GP is able to adapt based on the noise level. For low noise, GP performs no smoothing, achieving similar results as the Raw. For high noise, GP adapt a structure that is closer to the 3mm fixed-width Gaussian smoother.

6.5.2 Results of Real fMRI Data

Here we analyze real fMRI data of six healthy adults with normal or corrected-to-normal visual acuity (mean age 27 years, range 26-31, 1 female), with no past history of psychiatric or neurological disease. All subjects had extensive experience in psychophysical and fMRI experiments and were paid for their participation. We mapped responses to polar angle (measured from the contralateral horizontal meridian around the center of gaze) and eccentricity (distance from the center-of-gaze) using standard phase-encoded retinotopic stimuli [130]. The stimuli were presented using a wide-field display [112] and consisted of high contrast light/dark colored checks flickering in counterphase at 8 Hz in either a wedge or a ring configuration (polar angle and eccentricity mapping, respectively) extending up to 100 degrees of visual angle (see SI). The eccentricity ring expanded linearly with a uniform velocity 1 degrees/s. The average luminance of the stimuli was 105 cd/m2. The duration of one complete polar angle or eccentricity cycle was 64 s; 8 cycles were presented during each fMRI run. During retinotopic mapping, subjects were required to maintain fixation on a central cross.

Drift analysis

The most significant component in the fMRI signal variation is the long-range (low-frequency) drift caused by instabilities in the electromagnetic field. Removal of drift noise is crucial for any statistical inference method. However, accurate modeling of the drift is challenging due to its complex nonlinearities and heterogeneity. Parametric methods, such as GLM, are very susceptible to biases due to model selection, such as the number and type of basis functions to use. As a nonparametric Bayesian regression framework, GP allows for great flexibility in modeling the drift by learning the model structure from the fMRI data and fitting it to each individual voxel's data. Fig. 6.2a illustrates the GP learned drifts (after mean removal) from nine neighboring voxels. In Fig. 6.2b we illustrate the spectral contents of a single voxel's raw data, learned drift, and the data after drift removal. As can be seen, GP was able to accurately model the ultra-low frequencies, which correspond to the drift, while still preserving important components of the signal (e.g., the 1/64 Hz). Moreover, the spectrum of the raw data shows no clear separation between the drift noise components and the signal

frequency band, making it difficult to model as a simple temporal filter, such as a Butterworth high pass filter. The GP learned drift has spectral components going well into the signal frequency band. The drift standard deviation maps are shown in Fig. 6.2c. From the STD maps we can see that the drifts energy are heterogeneous across the brain and across different runs. The spectral contents of the learned drift for a single voxel in four different runs are shown in Fig. 6.2d. As can be seen, for each run GP adapted the drift spectral properties according to the data.

GP Statistical Smoothing

Following drift removal, GP is used again to learn the structure of the signal and the noise level of each voxel, for the entire block. As opposed to parametric models that attempt to summarize the signal with a small number of basis functions, the GP framework learns the correlation functions for each of the data dimensions. The learned correlations, seen in Fig. 6.3a, support three interesting observations. First, the learned spatial correlations (x, y, z) have FWHM 5-9mm in all the blocks. This range fits well with previous assumptions that the fMRI point-spread functions are consistent throughout the brain and are expected to be around 4-12mm [78]. Second, by using the expressive SMP kernel, it was possible to discern meaningful correlation functions that vary throughout the brain and to show that the expected regions respond to the stimuli. Finally, almost no information was shared between different runs, showing that local averaging within the same run is the most informative. The lack of sharing might be explained by the changing conditions between consecutive runs (tiredness, experimental learning).

The learned noise for each voxel is shown in Fig. 6.3b for a subset of voxels. The initial noise variance of each voxel was set, using a univariate empirical estimate, by calculating the variance of the residuals after removing the task. As can be seen from Fig. 6.3b, the learned noise is frequently lower than the empirical estimate. This finding is not surprising since the univariate model cannot model spatial dependencies, causing an increase in estimated error due to poor modeling. The bottom of Fig. 6.3 illustrates the corresponding time courses of voxels from Fig. 6.3b, pre- and post-GP. Each voxel's plot shows the across-trial statistics of four runs. The trial means of each run (8 trials per run) are illustrated in red, green, blue, and cyan traces, and the mean for all trials is shown in bold blue. The gray envelope around

the mean shows one standard deviation of the across-trials. It can be seen from the pre- and post-GP time courses that the effect of running GP varied significantly among different voxels. A closer examination shows that voxels with low signal and a low noise (plots marked with a blue outline) or voxels with a high signal and a low noise (yellow outline), were not changed much by the GP smoothing. Thus the smoothing did not corrupt voxels with low noise. On the other hand, voxels with high noise (red outline) were significantly smoothed to improve their SNR.

Searchlight Results

Next, we tested whether GP processing improves the results of standard neuroscience multivariate analysis. The first analysis we considered was the searchlight method [73] with a linear discriminant analysis (LDA) classifier, for checking the ability of voxels to differentiate between two polar angles (shown at the bottom of Fig. 6.4a). We compared the results using the original data pre-GP (after removing the drift), the data after GP processing, and the original data using a standard Gaussian convolution kernel with 3mm FWHM. Fig. 6.4a illustrates the classification accuracies for four slices. Comparing the results of the searchlight on the original data to the results using GP processing shows that the shape of significant regions did not change, while there was a clear improvement in accuracy in the GP processed data. As a further comparison, data that was smoothed using a 3mm FWHM Gaussian shows a loss in the separation between the regions. This loss in distinction between the regions is a clear example of the loss in spatial specificity that is a known drawback of standard smoothing methods.

For a more detailed comparison of individual regions, we compared the accuracy probability distribution of the voxels in each region (Fig. 6.4b). This distribution can be understood as the probability of randomly choosing a voxel from the region with specific classification accuracy. In the visual areas, where we conducted the comparison, we expected that at least some areas would accurately differentiate between the two polar angles. For each region, we plotted the accuracy distribution for the searchlight using the original data (red), and the GP processed data (blue). Fig. 6.4b shows that the searchlight accuracies of GP processed data are shifted toward higher accuracy values, meaning that more voxels showed a significant distinction between the two polar angles.

Retinotopic Results

Retinotopic mapping are commonly used to define borders of early visual areas in occipital cortex [130]. In Fig. 6.5, we created retinotopic maps for data before GP smoothing (pre GP) and after GP smoothing (post GP). The colors in the retinotopic maps give a representation of the sensitivity of an neural area to a polar angle stimulus. The maps show only areas that are more significant than a chosen threshold. For all the maps we use the same significance threshold. Similar to previous results, we see here that the retinotopic maps that used the GP smoothed data improved the significance in areas that were not significant otherwise. Furthermore, the spatial information that was presented in the Pro GP case was preserved in the Post GP maps. We also compared the results averaged over two runs. Averaging multiple runs the traditional method for improving the quality of retinotipic maps. Comparing the results of the two runs for the pre GP and post GP we can see that more areas are significant than for a single run. However, it is clear to see that by utilizing GP smoothed data, certain areas become significant that were not significant when averaged over unsmoothed data.

6.6 Discussion

We propose statistical-smoothing-based Gaussian process regression, an alternative to the classical fixed-width Gaussian filter and the statistical-smoothing-based GLM approaches. The core idea is to learn the smoother structure from the fMRI data, and alternate its smoothing intensity per each voxel, depending on the amount of noise present. Our results show that spatial areas with high noise levels were not corrupted, while noisy areas were significantly improved. The adaptive nature of our GP method along with its powerful and convenient modeling ability allow us to avoid the sensitivity/specificity tradeoff and gain insights about activity pattern properties across brain regions.

When can GP statistical smoothing be use?
GP smoothing is a general method that can be applied for any fMRI experiment. In this context the block designs and event-related designs are mostly needed for better estimation of the voxel noise levels to be used as an initial guess values for GP. Thus, a standard experimental design-based GLM can be used as a prior step to GP. This joint method allows

utilizing the knowledge of the experimental design without overly constraining the model to that structure.

For improved performance it is important to have a good initial estimation of the voxel noise. The commonly used GLM method constraints provide a good restriction bias for robust initial estimation of the voxel noise. A combination of the constrained but robust GLM as a starting point for the flexible GP should provide improved denoising results in fMRI. How to best integrate these methods is an interesting topic problem for future work.

6.7 Methodological Details

Our method is a two-stage procedure (see Fig. 6.6). Initially, we applied GP to model the drift. The input for the method was the raw fMRI data after motion and movement correction only. For each run, the drift temporal properties were learned simultaneously on the entire brain. Next, we ordered the multiple runs in a single 5D data structure, where the fifth dimension indexed the runs. We applied GP a second time, jointly learning the correlation functions over all dimensions (spatial, temporal, and runs). We used two common simplifying assumptions in the prior distribution: stationarity (correlations are unaffected by shifts in space or time) and separability (a multiplicative kernel separates correlations in each dimension). These assumptions help with computational tractability by limiting the number of hyperparameters in the model, and they provide a restriction bias that can help with learning [158]. Since we assume that the neural activity is only locally stationary but varies between spatial areas, the correlation functions were learned separately on 189 spatially overlapping parcellated blocks of data. Running the analysis over blocks allowed for both an efficient parallelism of the method and for variation in the correlation functions across the brain, while still holding the assumption of local stationarity within each block. Note that under GP there are no artifacts near the borders of each block, which commonly appear with other filtering methods.

To model the spatial correlations, we used the squared exponential correlation function [121] which relates to the commonly used FWHM Gaussian convolution kernel. To model the temporal correlation, we used the spectral mixture product (SMP) kernel [157]. The expressivity of the SMP kernel allows it to discover complex temporal correlations, which

can depend on the experiment design and vary across the brain. To model the correlation between different runs, we again used a simple squared exponential kernel.

Next, GP uses the data in each block to learn the hyperparameters of the correlation functions and to estimate the voxels' noise variance. The learned correlations and the voxels' noises were used in the adaptive smoother for denoising the block fMRI data. In the final step, all the denoised blocks were joined together, averaging overlapping areas.

Fourier analysis retinotopic data were analyzed using UCSD/UCL FreeSurfer [31, 42] based on standard procedures described in detail in many previous publications (e.g., [58, 112–114, 130, 143]. The first (pre-magnetization steady-state) four volumes were discarded. Motion correction and cross-scan alignment were performed using the AFNI (Analysis of Functional NeuroImages) 3dvolreg (3T data). Phase-encoded retinotopic data were analyzed by voxel-wise Fourier transforming the fMRI time series (after removing constant and linear terms). This Fourier analysis generated real and imaginary components (equivalently, amplitude and phase) at each frequency. To estimate the significance of the BOLD signal modulation at the stimulus frequency (eight cycles per scan), the squared Fourier amplitude was divided by the summed mean squared amplitude (power) at all other frequencies, which included noise. The ratio of two chi-squared variates follows the F-distribution [76], with degrees of freedom equal to the number of time points from which statistical significance can be calculated. The second harmonic of the stimulus frequency and very low frequencies (1 and 2 cycles per scan, residual motion artifacts) were ignored. The response phase at the stimulus frequency was used to map retinotopic coordinates (polar angle or eccentricity). In these maps, hue represents phase, and saturation represents a sigmoid function of the response amplitude. The sigmoid function was arranged so that visibly saturated colors begin to emerge from the gray background at a threshold of $p < 10^{-2}$. The computed significance at the most activated cortical surface loci ranged from $p < 10^{-5}$ to 10^{-10}. Since this analysis does not take into account fMRI time series autocorrelation [164], these p-values are properly regarded as descriptive. Boundaries of retinotopic cortical areas were defined on the cortical surface for each individual on the basis of phase-encoded wide field retinotopy [33, 34, 38, 39, 130] and subsequent calculation of the visual field sign. This latter provides an objective means of drawing borders between areas based on the angle between the gradients (directions of

fastest rate of change) in the polar angle and eccentricity with respect to the cortical surface [130,131]. Each field sign map used here was based on at least four scans (two scans for polar angle and two scans for eccentricity).

Time course analysis For each individual, the AFNI-preprocessed data were coregistered across sessions and then registered (12-parameter affine transform) to Talairach space using an atlas-representative template conforming to the SN method of Lancaster et al. (1995). After composition of transforms, the functional data were resampled in one step to 3 mm isotropic voxels. Polar angle modulations were extracted independently for each time point (32 frames per cycle) using a general linear model (GLM) [44, 105]. The GLM included nuisance regressors representing baseline, linear trend, and low frequency components (¡ 0.009 Hz).

6.8 Conclusions

The advances of using GP are threefold. First, it extends the previous ad-hoc fixed-width Gaussian filter methodology to a more rigorous framework where the optimal filter learns the structure of the smoother from the data. Second, it allows jointly processing space and time, thereby better utilizing spatiotemporal dependence. Third, different from previous methods, where the filter was homogenously applied to all the voxels in the region, GP learns and varies the amount of smoothing for each voxel, depending on the amount of noise present in the voxel. We illustrated the power of these advances using real data, showing improvements over previous state-of-the-art methods. The significant improvement in the signal to noise ratio was accompanied by a significant reduction in trial-to-trial variability. The GP processed data provided more accurate multivariate analysis and more significant retinotopic phase maps.

Improving the signal to noise ratio of fMRI data will allow for shorter scanning time, more comprehensive experimental design, higher resolution, more significant findings, and major savings in both time and money.

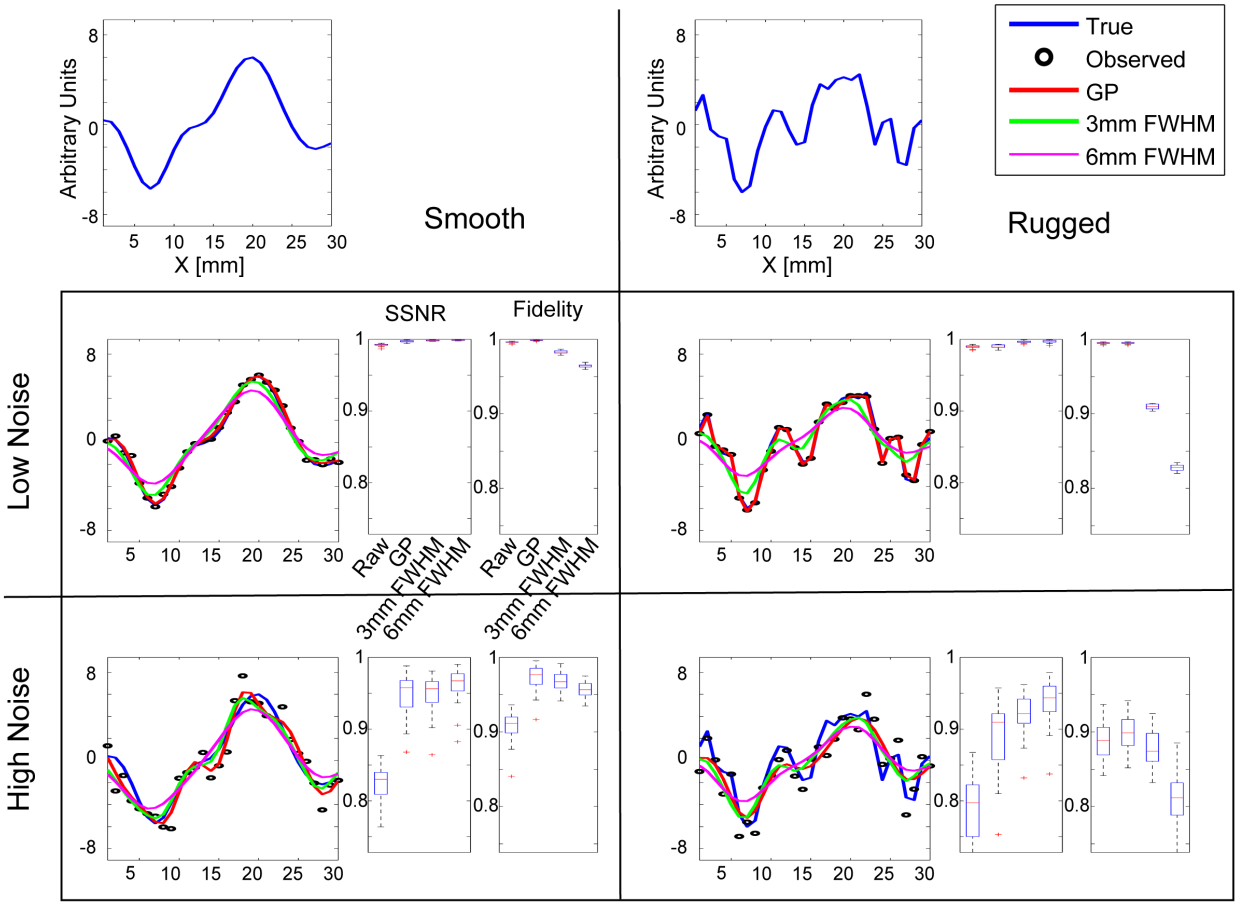


Figure 6.1: Smoothing comparison of simulated data. The figure shows results for combinations of two conditions - smoothness of the true data (top) and the amount of added white noise (left). We compared the GP smoothing results to 3mm and 6mm FWHM Gaussian fixed-width smoothers, and to the not smoothing at all (Raw). For each of the two true patterns we drew multiple noisy observations. On the right of each of the plots we added a summary statistics for each smoother. The metrics used are signal to signal plus noise (SSNR) and fidelity (see text for further details).

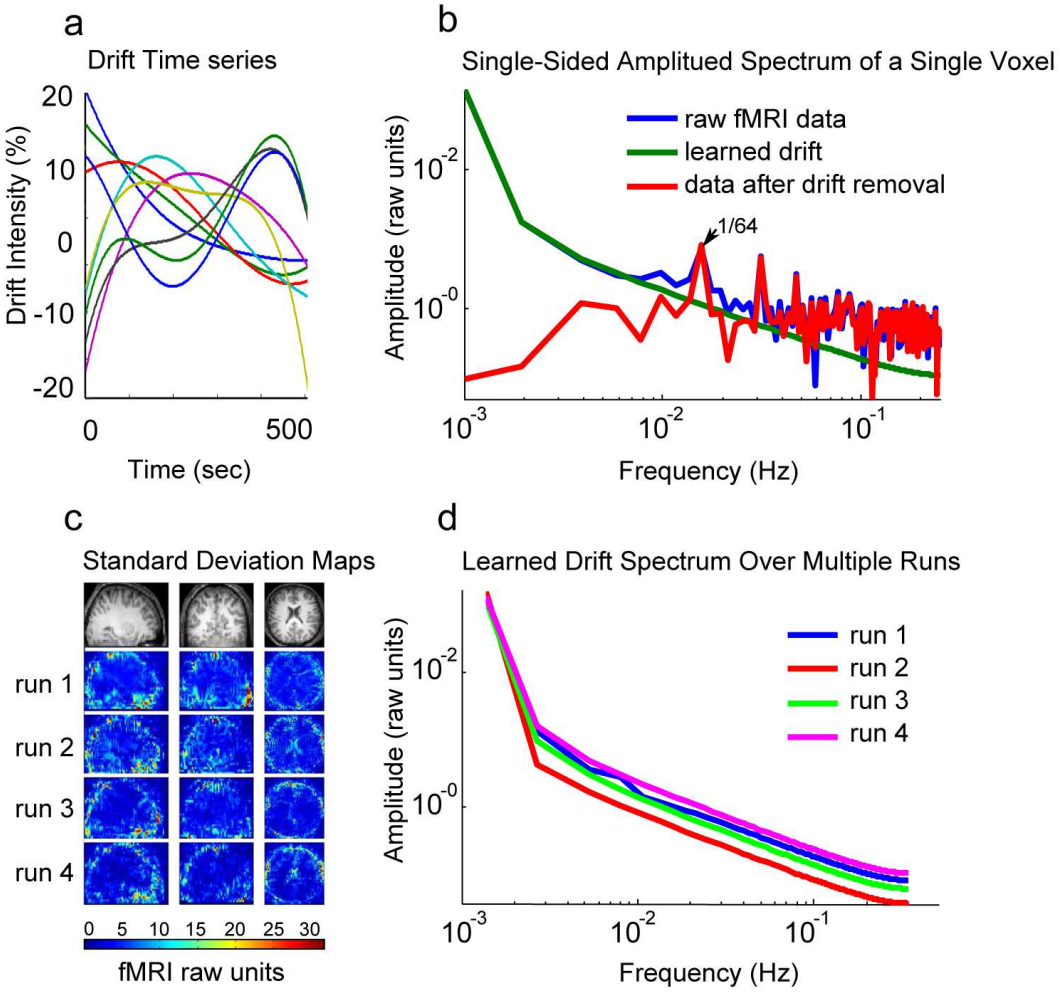


Figure 6.2: This figure illustrates the different behaviors of the fMRI ultra-low temporal frequency noise (drift). Fig. 6.2a shows the GP learned drifts (after mean removal) from nine neighboring voxels. Fig. 6.2b illustrates the spectral contents of a single voxel's raw data, learned drift, and the data after drift removal. Fig. 6.2 shows the standard deviation of the drift in each voxel across a slice of brain. Fig. 6.2d shows the spectral components of the learned drift for the same voxel over four different runs

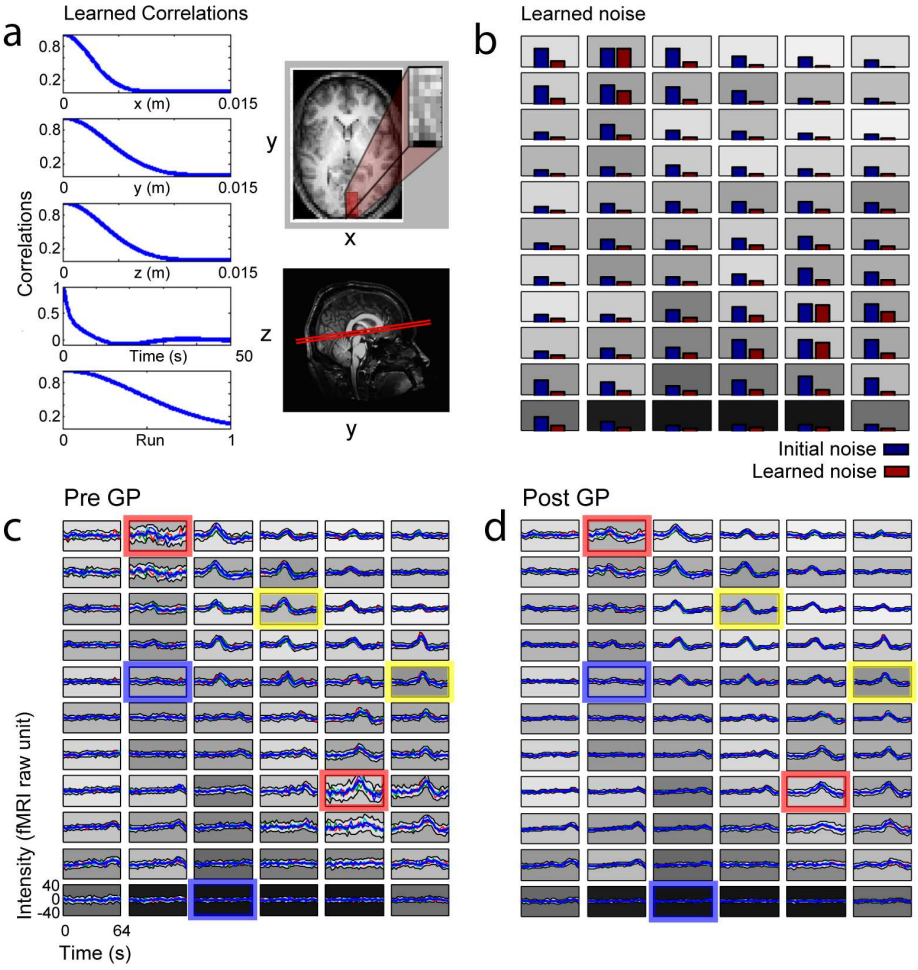


Figure 6.3: Results of the GP statistical smoother for real fMRI data. Figure 6.3a shows the learned correlations for the five dimensions. The figure also show a 2D representation of the block of neural data that was used in the analysis (The block is actually 3D). Fig. 6.3b shows the learned noise for each voxels in the block. The background of the each plot corresponds to the anatomical T1 grayscale from Fig. 6.3a. Fig. 6.3c shows the a per-trial representation of each voxel. The trial mean is shown in bold blue and the trial-to-trial variance is shown by the gray envelope. Fig. 6.3d shows for the same data as in Fig. 6.3c after GP smoothing. The colored outlines represent three types of cases: voxels with high signal low noise (yellow), voxels with low signal low noise (blue), and voxels with high noise (red).

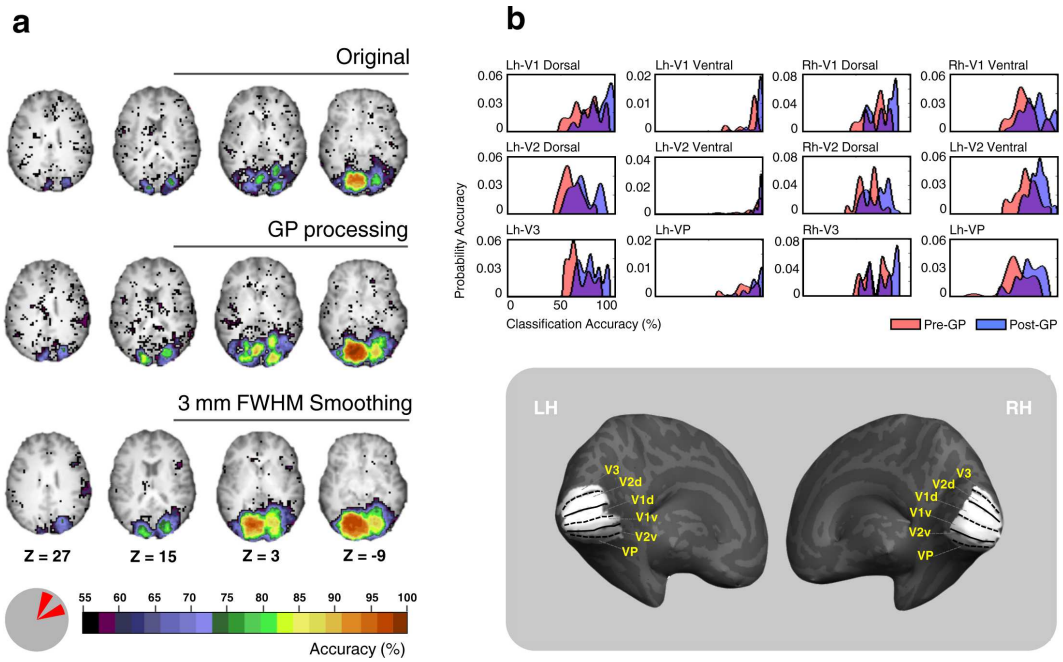


Figure 6.4: Comparison of searchlight results for smoothed and unsmoothed data. Figure 6.4a shows classification accuracy results of four brain slices. The classification problem was to classify between the two polar angles shown in the bottom left. The top row results are for unsmoothed data (after drift removal), the middle and bottom rows show results for GP and 3mm FWHM smoothed data, respectively. Figure 6.4b shows the accuracy distributions for several early visual areas. The anatomical locations of the visual areas are shown in the bottom right.

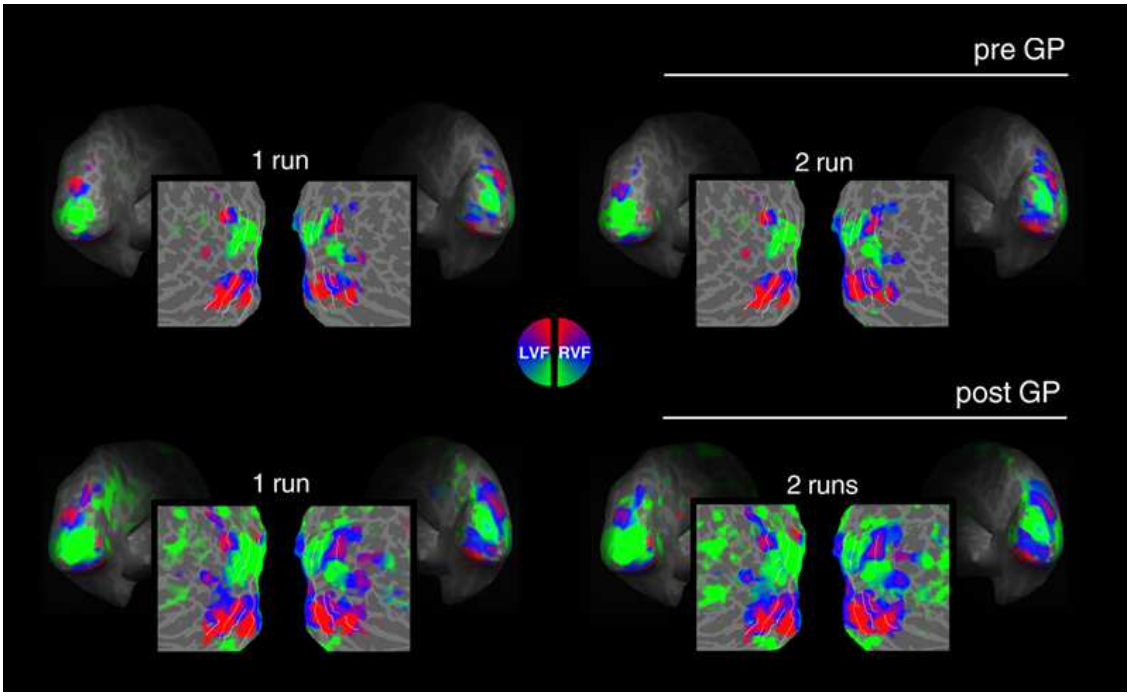


Figure 6.5: Results of retinotopic analysis. This figure shows the retinotopic maps for the data before and after GP smoothing, pre GP and post GP, respectively. On the left we show results for a single run, and on the right we show results for an average of two runs. The white lines show the borders between the retinotopic visual areas. The same significance threshold was used for all maps. We show the retinotopic maps both as a flat and inflated maps.

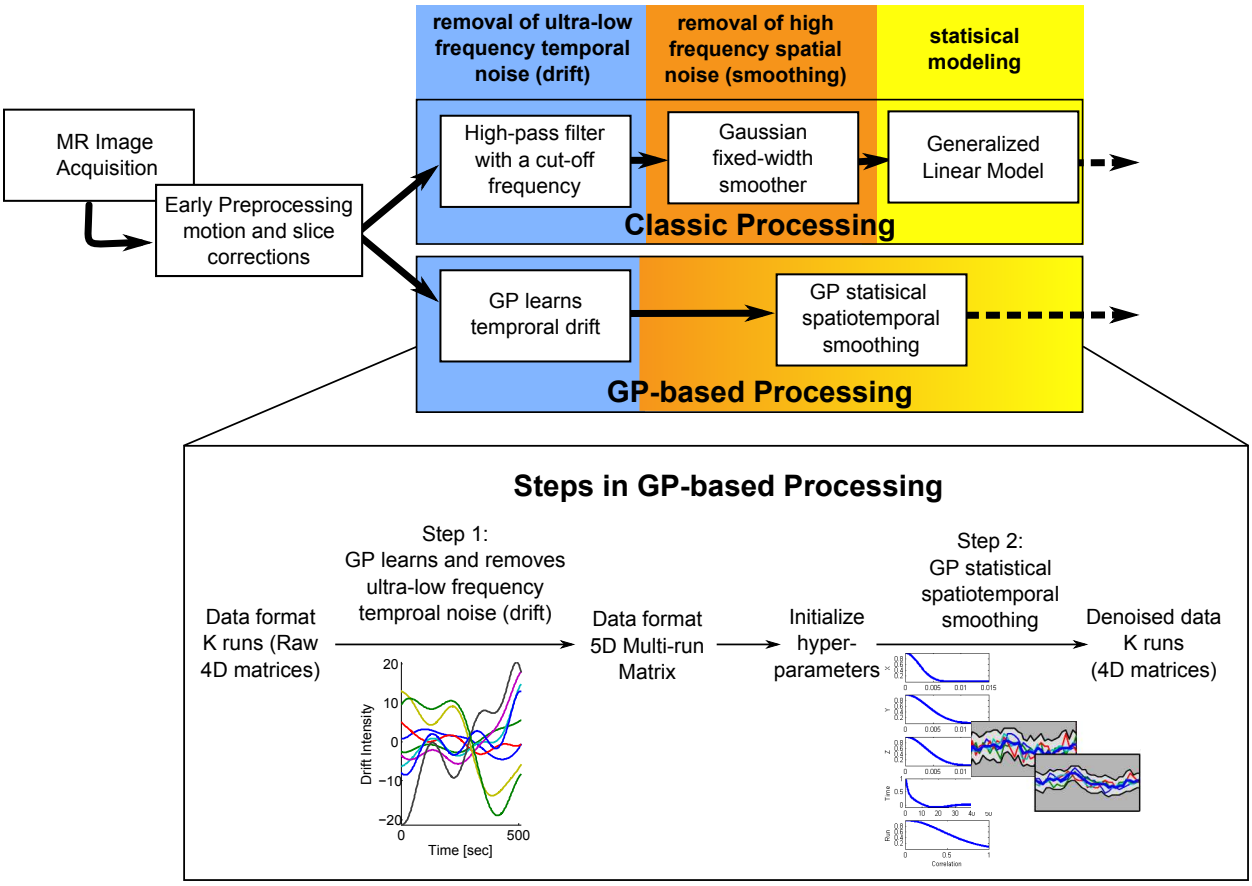


Figure 6.6: Methodological sketch of our GP-based statistical smoothing. Top panel shows a comparison between classical processing of fMRI data using a fixed-width kernel and our new GP-based processing. GP-based processing is composed of two steps, draft removal and statistical smoothing. Differently than in the classical processing, the GP statistical smoother combines the spatial smoother with the statistical spatiotemporal model. After denoising the fMRI data with our GP-based processing, the denoised data can be an input to further statistical modeling (such as GLM) or used directly for testing (such as with multivariate analysis). The bottom panel shows in greater details the steps of the GP-processing.

Chapter 7

Conclusions and Future Work

7.1 Summary and Conclusions

In this dissertation we studied ways to reduce the scaling burden of inference methods for multidimensional structured data. Often powerful inference methods are limited by their burdensome scaling properties. By exploiting the useful structure that commonly exists in engineered systems it was possible to lower the memory and runtime complexity of these problems without affecting the modeling accuracy of the inference methods. We explored two types of structured problems: optimization problems with block separable constraints (Chapters 2-3), and Gaussian Process for multidimensional lattice input (Chapters 4-6). We showed the use of our methods on a wide range of difficult applications that were often intractable for inference methods due to their large dataset size.

Optimization problems with block separable constraints (Chapters 2-3): We proposed a distributed optimization framework where each agent solves a simple local optimization problem. The different subproblems are dependent but only a small subset of global information is shared (such as cost per hour scheduling problems), while private information (such as preferences) is not shared. Coordination between the subproblem was achieved with a regularization term that penalized the changes in the successive iterations with an adaptive regularization coefficient. Our numerical simulations showed that our distributed algorithm converged to similar optimum points such as more complicated joint optimization algorithms where all the information was shared, at a much lower computational time. We next used this algorithm for the complicated problem of solving the inverse problem of estimating the local conductivities of a excitable tissue. To solve this problem we first modeled the excitable tissue using a diffusion-reaction model. We then used a set of spatiotemporal measurements taken with a high resolution microelectrode array in order to solve the inverse

problem. Solving the inverse problem proved highly difficult as it is an ill posed problem and required solving a very high dimensional optimization problem that was very unstable. We lowered the complexity of the optimization by using a single-step approximation employing a parallel block-relaxation optimization method. We analyzed the performance of our method using numerical examples of several electrical conductivity field topologies and noise levels, and discussed its application to real measurements obtained from a smooth cardiac mouse tissue slice.

Gaussian Process for multidimensional lattice input (Chapters 4-6): We made nontrivial advances to extend structured GP techniques for real multidimensional applications. We extended our GP-grid to deal with incomplete grids, heteroscedastic noise, and to utilize expressive SMP kernels, while still maintaining close to linear complexity in both memory and time. These advances allow the use of GP-grid on a wide range of applications. An application that showed good results was for interpolation and denoising of division of focal plane cameras. We showed in Chapter 5 that GP-grid is a useful and elegant tool that enables performance improvements in recovered phase information. The results show that the GP framework allows for improved results over conventional interpolation methods. GP allows for statistical interpolation that can naturally incorporate the camera noise model, and improvement is most evident for low SNR images where having a good prior can help reduce the effect of the noise.

Finally, we used our GP-grid framework for denoising fMRI data. The advances of using GP are threefold. First, it extends the previous ad-hoc fixed-width Gaussian filter methodology to a more adaptive data-driven framework where the smoother structure is learned from the data. Second, it allows jointly processing the data in both space and time, thereby better utilizing spatiotemporal dependence. Third, GP varies the amount of smoothing for each voxel, depending on the amount of learned noise of each voxel. We illustrated the power of these advances using real data, showing improvements over previous state-of-the-art methods. The significant improvement in the signal to noise ratio was accompanied by a significant reduction in trial-to-trial variability. The GP processed data provided more accurate multivariate analysis and more significant retinotopic phase maps.

7.2 Future Directions

In this section, we point out several potential future research directions.

Optimization problems with block separable constraints (Chapter 2): The convergence proof necessitates a switch between states of the λ function (Eq. 2.6). The second state is needed for guaranteeing asymptotic convergence. This requires a difficult choice of defining the switching iteration (which is often done ad-hoc). We believe that there should be a λ that will achieve convergence without needing this change. It is also important to explore other λ functions and understand their effect on the convergence rate. This will allow a rigorous way for choosing the optimal λ for a given dataset.

Estimating Electrical Conductivity Tensors of Biological Tissues Using Microelectrode Arrays (Chapter 3): In the future, it is important to consider optimizing model parameter fitting from the data by employing more advanced learning schemes and better utilizing prior biological information. Further, it will be useful to extend the model to nonhomogeneous reaction dynamics and establish a methodology for fusing conductivity tensor field information from different post-stimulus experiments.

Scaling Multidimensional Inference for Structured Gaussian Processes (Chapter 4): Our GP-grid method opens up an entirely new set of applications for GP, such as image and video processing, or financial engineering applications such as implied volatility surfaces. In the future it will be interesting and explore the benefits of GP inference in these application domains. Our GPatt methods is a small step in the direction of using GPs for automatic pattern discovery on large multidimensional datasets, with scalable and exact inference procedures. We believe that Bayesian nonparametric models are naturally suited to pattern extrapolation on large multidimensional datasets, where extra training instances can provide extra opportunities to learn additional structure in data. Such models could play a role in a new era of machine learning, where models are expressive and scalable, but also interpretable and manageable, with simple exact learning and inference procedures. Understanding how our existing nonparametric models can scale and be used in real data, and how these models connect to other areas of statistics, will increase the utility of machine learning algorithms in general.

Image interpolation and denoising for division of focal plane sensors using Gaussian Processes (Chapter 5) GP becomes tractable for image data by using the GP-grid algorithm, and it is a convenient technology to naturally incorporate our two performance-critical advances: segmentation (incomplete grids) and a known noise model. As the results show, all of these advances are important in order for GP to be considered a general framework for image data. It is common practice in image processing to mix different methods in order to improve the overall results, e.g., alternate methods close to an edge. Integrating GP-grid together with other state-of-the-art interpolation methods to achieve further improvement is an interesting topic for future work.

Gaussian Processes for Denoising fMRI Data (Chapter 6): For improved performance it is important to have a good initial estimation of the voxel noise. The commonly used GLM method constrains provide a good restriction bias for robust initial estimation of the voxel noise. A combination of the constrained but robust GLM as a starting point to the flexible GP should provide improved denoising results in fMRI. How to best integrate these methods is again an important problem for future work.

References

[1] A. Ackleh. Parameter estimation in nonlinear evolution equations. *Numer. Funct. Anal. and Optimiz.*, 19(9-10):933–947, 1998.

[2] S. Aja-Fernandez, R. de Luis Garcia, D. Tao, and X. Li. *Tensors in Image Processing and Computer Vision*. Springer, 2009.

[3] M. Anastasiadou, A. D. Martino, D. Clement, F. Liége, B. Laude-Boulesteix, N. Quang, J. Dreyfuss, B. Huynh, A. Nazac, L. Schwartz, and H. Cohen. Polarimetric imaging for the diagnosis of cervical cancer. *Phys. Status Solidi*, 5:1423–1426, 2008.

[4] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56:411–421, 2006.

[5] Kendall E Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[6] C. T. H. Baker. *The Numerical Treatment of Integral Equation*. Clarendon Press, Oxford, 1977.

[7] A. Barmpoutis, B. C. Vemuri, T. M. Shepherd, and J. R. Forder. Tensor splines for interpolation and approximation of DT-MRI with applications to segmentation of isolated rat hippocampi. *IEEE TMI: Transactions on Medical Imaging*, 26(11):1537–1546, 2007.

[8] P. J. Basser, J. Mattiello, and D. Le Bihan. MR diffusion tensor spectoscopy and imaging. *Biophys*, 66:256–267, 1994.

[9] Peter J. Basser. Inferring microstructural features and the physiological state of tissues from diffusion-weighted images. *NMR IN BIOMEDICINE*, 8:333–344, 1995.

[10] P. G. Batchelor, M. Moakher, D. Atkinson, F. Calamante, and A. Connelly. A rigorous framework for diffusion tensor calculus. *Magnetic Resonance in Medicine*, 53:221–225, 2005.

[11] B.E. Bayer. Color imaging array, Jul 1976.

[12] Maxim Bazhenov, Peter Lonjers, Steven Skorheim, Claude Bedard, and Alain Destexhe. Non-homogeneous extracellular resistivity affects the current-source density profiles of up?down state oscillations. *Philosophical Transactions of the Royal Society*, 369(1952):3802–3819, 2011.

[13] Claude Bedard and Alain Destexhe. A generalized theory for current-source density analysis in brain tissue. *Physical Review E*, 2011.

[14] Luca Berdondini, Kilian Imfeld, Alessandro Maccione, Mariateresa Tedesco, Simon Neukom, Milena Koudelka-Hep, and Sergio Martinoia. Active pixel sensor array for high spatio-temporal resolution electrophysiological recordings from single cell to large scale neuronal networks. *Lab Chip*, 9:2644–2651, 2009.

[15] J.L. Bernal-Rusiel, M. Atienza, and J.L. Cantero. Determining the optimal level of smoothing in cortical thickness analysis: A hierarchical approach based on sequential statistical thresholding. *NeuroImage*, 52:158–171, 2010).

[16] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[17] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[18] James C. Bezdek and Richard J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11:351–368, December 2003.

[19] S Bochner. *Lectures on Fourier Integrals.(AM-42)*, volume 42. Princeton University Press, 1959.

[20] F.D. Bowman, B. Caffob, S.S. Bassettc, and C. Kiltsd. A bayesian hierarchical framework for spatial modeling of fMRI data. *NeuroImage*, 1:146156, 2008).

[21] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[22] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. In *International Conference on Machine Learning (ICML 2011)*, Bellevue, Washington, June 2011.

[23] Paul C. Bressloff. Traveling fronts and wave propagation failure in an inhomogeneous neural network. *Phys. D*, 155:83–100, June 2001.

[24] A. Brezger, L. Fahrmeir, and A. Hennerfeind. Adaptive gaussian markov random fields with applications in human brain mapping. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 56:327345, 2007).

[25] V.D. Calhoun, T. Adali, G.D. Pearlson, and J.J. Pekar. A method for making group inferences from functional MRI data using independent component analysis. *Human Brain Mapping*, 14:140–151, 2001).

[26] Kenneth R. Castleman. *Diginal Image Processing*. Prentice Hall, 1996.

[27] K. Chalupka, C. K. I. Williams, and I. Murray. A framework for evaluating approximation methods for Gaussian process regression. *JMLR*, 2013.

[28] P. Chavali and Arye Nehorai. Managing multi-modal sensor networks using price theory. *Signal Processing, IEEE Transactions on*, 60(9):4874–4887, Sept 2012.

[29] N. Cressie and G. Johannesson. Fixed rank Kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1), 2008.

[30] J. P. Cunningham, K. V. Shenoy, and M. Sahani. Fast Gaussian process methods for point process intensity estimation. In *ICML*, pages 192–199, 2008.

[31] A.M. Dale, B. Fischl, and M.I. Sereno. Cortical surface-based analysis. I. segmentation and surface reconstruction. *Neuroimage*, 9:179–194, 1999).

[32] Jan de Leeuw. Block-relaxation algorithms in statistics. Technical report, Dept. of Statistics, Univ. of California at Los Angeles, 1994.

[33] E.A. DeYoe, P. Bandettini, J. Neitz, D. Miller, and P. Winans. Functional magnetic resonance imaging (FMRI) of the human brain. *J Neurosci Methods*, 54:171–187, 1994).

[34] E.A. DeYoe, G.J Carman, P. Bandettini, S. Glickman, J. Wieser, R. Cox, D. Miller, and J. Neitz. Mapping striate and extrastriate visual areas in human cerebral cortex. *Proc Natl Acad Sci*, 93:2382–2386, 1996).

[35] Socrates Dokos and Nigel H. Lovell. Parameter estimation in cardiac ionic models. *Progress in Biophysics and Molecular Biology*, 85(23):407 – 431, 2004. ¡ce:title¿Modelling Cellular and Tissue Function.

[36] D. Duvenaud, J.R. Lloyd, R. Grosse, J.B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, 2013.

[37] D.K. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive Gaussian processes. In *NIPS*, pages 226–234, 2011.

[38] S.A. Engel, G.H. Glover, and B.A. Wandell. Retinotopic organization in human visual cortex and the spatial precision of functional MRI. *Nature*, page 369:525, 1997).

[39] S.A. Engel, D.E. Rumelhart, B.A. Wandell, A.T. Lee, G.H. Glover, E.J. Chichilnisky, and M.N. Shadlen. Retinotopic organization in human visual cortex and the spatial precision of functional MRI. *Cereb Cortex*, 7:181–192, 1994).

[40] M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4:815–832, 1994.

[41] Miroslav Fiedler. Bounds for the determinant of the sum of hermitian matrices. In *Proceedings of the American Mathematical Society*, volume 30, pages 27 – 31, 1971.

[42] B. Fischl, M.I. Sereno, and A.M. Dale. Cortical surface-based analysis. II: Inflation, flattening, and a surface-based coordinate system. *Neuroimage*, 9:195–207, 1999).

[43] K.J. Friston, A. Holmes, J.B. Poline, C.J. Price, and C.D. Frith. Detecting activations in PET and fMRI: Levels of inference and power. *NeuroImage*, 4:223–235, 1996).

[44] K.J. Friston, A.P. Holmes, J.B. Poline, P.J. Grasby, S.C.R Williams, R.S.J. Frackowiak, and R. Turner. Analysis of fMRI time-series revisited. *Neuroimage*, 2:45–53, 1995.

[45] J. Fritz, I. Neuweiler, and W. Nowak. Application of FFT-based algorithms for large-scale universal Kriging problems. *Mathematical Geosciences*, 41:509–533, 2009.

[46] W. J. Fu. Penalized regressions: The bridge versus the LASSO. *Journal of Comp. and Graphical Statistics*, 7(3):397–416, 1998.

[47] M. Fuentes. Approximate likelihood for large irregularly spaced spatial data. *J. American Statistical Association*, 102, 2007.

[48] R. Furrer, M. G. Genton, and D. Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15:502–523, 2006.

[49] S. Gao and V. Gruev. Bilinear and bicubic interpolation methods for division of focal plane polarimeters. *Opt. Express*, 19:26161–26173, 2011.

[50] Shangui Gao and V. Gruev. Gradient-based interpolation method for division-of-focal-plane polarimeters. *Opt. Express*, 21:1137–1151, 2013.

[51] E. Gilboa, Y. Saatçi, and J. P. Cunningham. Scaling multidimensional Gaussian processes using projected additive approximations. *JMLR: W&CP*, 28, 2013.

[52] E. Gilboa, Y. Saatçi, and J. P. Cunningham. Scaling multidimensional inference for structured Gaussian processes. *IEEE-TPAMI*, 2013.

[53] P. W. Goldberg, C. K.I. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. *NIPS*, 10, 1998.

[54] C. G ossl, D.P. Auer, and L. Fahrmeir. Bayesian spatiotemporal inference in functional magnetic resonance imaging. *Biometrics*, 57(2):1310.5288, 2001).

[55] Leon S. Graham and David Kilpatrick. Estimation of the bidomain conductivity parameters of cardiac tissue from extracellular potential distributions initiated by point stimulation. *Annals of Biomedical Engineering*, 38(12):3630–3648, 2010.

[56] B. Grone, C. Johnson, E. M. De Sa, and H. Wolkowicz. Improving Hadamard's inequality. *Linear and Multilinear Algebra*, 16:305–322, 2010.

[57] V. Gruev, R. Perkins, and T. Yor. Ccd polarization imaging sensor with aluminum nanowire optical filters. *Optics Express*, 18:19087–19094, 2010.

[58] D.J. Jr. Hagler and M.I. Sereno. Spatial maps in frontal and prefrontal cortex. *J Neurosci*, 29:567–577, 2006).

[59] P. Hagmann, L. Jonasson, P. Maeder, J. Thiran, V. Wedeen, and R. Meuli. Understanding diffusion MRI imaging techniques: from scalar diffusion-weighted imaging to diffusion tensor imaging and beyond. *Radiographics*, pages S205–S223, 2006.

[60] R. Hardie, D. LeMaster, and B. Ratliff. Super-resolution for imagery from integrated microgrid polarimeters. *Opt. Express*, 19:12937–12960, 2011.

[61] L.M. Harrison, W. Penny, J. Ashburner, N. Trujillo-Barreto, and K.J. Friston. Diffusion-based spatial priors for imaging. *Neuroimage*, 38:677695, 2007.

[62] Niels Væver Hartvig. *Parametric modelling of functional magnetic resonance imaging data*. PhD thesis, University of Aarhus, 2000.

[63] J Hays and A Efros. Scene completion using millions of photographs. *Communications of the ACM*, 51(10):87–94, 2008.

[64] H. He and W. Siu. Single image super-resolution using Gaussian process regression. In *CVPR, 2011 IEEE Conference on*, pages 449–456. IEEE, 2011.

[65] Yuan He and David Keyes. Reconstructing parameters of the fitzhugh nagumo system from boundary potential measurements. *Journal of Computational Neuroscience*, 23:251–264, 2007. 10.1007/s10827-007-0035-9.

[66] D. Higdon. Space and space–time modeling using process convolutions. *Quantitative methods for current environmental issues*, page 37, 2002.

[67] L Hörmander. *The Analysis of Linear Partial Differential Operators I, Distribution Theory and Fourier Analysis*. Springer-Verlag, 1990.

[68] K. Imfeld, S. Neukom, A. Maccione, Y. Bornat, S. Martinoia, P.A. Farine, M. Koudelka-Hep, and L. Berdondini. Large-scale, high-resolution data acquisition system for extracellular recording of electrophysiological activity. *Biomedical Engineering, IEEE Transactions on*, 55(8):2064 –2073, aug 2008.

[69] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401 – 422, mar 2004.

[70] Jari Kaipio and Erkki Somersalo. *Statistical and Computational Inverse Problems*, volume 60. Springer, 2005. chapter 4, Nonstationary Inverse Problems.

[71] C. Kaufman, M. Schervish, and D. Nychka. Covariance tapering for likelihood-based estimation in large spatial data sets. *J. American Statistical Association*, 103(484), 2008.

[72] N Kostantinos. Gaussian mixtures and their applications to signal processing. *Advanced Signal Processing Handbook: Theory and Implementation for Radar, Sonar, and Medical Imaging Real Time Systems*, 2000.

[73] N. Kriegeskorte, R. Goebel, and P. Bandettini. Information-based functional brain mapping. *PNAS*, 103(10):3863–3868, March 2006).

[74] T. Krishna, C. Creusere, and D. Voelz. Passive polarimetric imagery-based material classification robust to illumination source position and viewpoint. *IEEE Trans. Image Process*, 20:288–292, 2011.

[75] F. Kruggel, D.Y. von Cramon, and X. Descombes. Comparison of filtering methods for fMRI datasets. *NeuroImage*, 10:530543, 1999).

[76] R.J. Larsen and M.L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. New Jersey: Prentice-Hall, 1986.

[77] Mark L. Latash. *Neurophysiological Basis of Movement*. Human Kinetics Publishers, 2008.

[78] N.A. Lazar. *The statistical Analysis of functional MRI Data*. Springer, 2008.

[79] M. Lázaro-Gredilla. *Sparse Gaussian Processes for Large-Scale Machine Learning*. PhD thesis, Universidad Carlos III de Madrid, Madrid, Spain, March 2010.

[80] M. Lázaro-Gredilla, J. Quiñonero-Candela, C.E. Rasmussen, and A.R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[81] Q. Le, T. Sarlos, and A. Smola. Fastfood-computing hilbert space expansions in log-linear time. In *Proceedings of the 30th International Conference on Machine Learning*, pages 244–252, 2013.

[82] X. Li and M. T. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10:1521–1527, 2001.

[83] M. Lindquist and T. Wager. Spatial smoothing in fMRI using prolate spheroidal wave functions. *Human Brain mapping*, 29:12761287, 2008.

[84] M. Lindquist, C. Zhang, G. Glover, L. Shepp, and Q. Yang. A generalization of the two dimensional prolate spheroidal wave function method for non-rectilinear MRI data acquisition methods. *IEEE Transactions in Image Processing*, 15:27922804, 2006.

[85] M.A. Lindquist. The statistical analysis of fMRI data. *Statistical Science*, 23(4):469–464, 2008.

[86] M.A. Lindquist and T.D. Wager. Spatial smoothing in fMRI using prolate spheroidal wave functions. *Human Brain Mapping*, 29:1279–1287, 2008.

[87] P. J. Liu. Using Gaussian process regression to denoise images and remove artifacts from microarray data. Master's thesis, University of Toronto, 2007.

[88] Y. Liua, T. York, W. Akersa, G. Sudlowa, V. Gruev, and S. Achilefua. Complementary fluorescence-polarization microscopy using division-of-focal-plane polarization imaging sensor. *Journal of Biomedical Optics*, 17(11):116001.1–116001.4, 2012.

[89] Yao Lu, Yuzuru Sato, and Shunichi Amari. Traveling bumps and their collisions in a two-dimensional neural field. *Neural Computation*, 23(5):1248–60, May 2011.

[90] Y. Luo and R. Duraiswami. Fast near-GRID Gaussian process regression. *JMLR: W&CP*, 31, 2013.

[91] J. Luttinen and A. Ilin. Efficient Gaussian process inference for short-scale spatio-temporal modeling. In *JMLR: W&CP*, volume 22, pages 741–750, 2012.

[92] D.J.C MacKay. Bayesian nonlinear modeling for the prediction competition. *Ashrae Transactions*, 100(2):1053–1062, 1994.

[93] D. Malonek, U. Dirnagl, U. Lindauer, K. Yamada, I. Kanno, and A. Grinvald. Vascular imprints of neuronal activity: Relationships between the dynamics of cortical blood flow, oxygenation, and volume changes following sensory stimulation. *Neurobiology*, 94:14826–14831, 1997).

[94] Kevin A. McCabe, Mary L. Rigdon, and Vernon L. Smith. Positive reciprocity and intentions in trust games. *Journal of Economic Behavior and Organization*, 52(2):267 – 275, 2003.

[95] P. McCullagh. What is a statistical model? *The Annals of Statistics*, 30(5), 2002.

[96] Raman K. Mehra. Optimal input signals for parameter estimation in dynamic systems-survey and new results. *IEEE Transaction on Automatic Control*, 1974.

[97] W.T. Miller and D.B. Geselowitz. Simulation studies of the electrocardiogram. I. the normal heart. *Circulation Research*, 43(2):301–315, 1978.

[98] W.T. Miller and C.S. Henriquez. Finite element analysis of bioelectric phenomena. *CRC Crit. Rev. Biomed. Eng. (USA)*, 18(3):207–33, 1990.

[99] D. Miyazaki, R. Tan, K. Hara, and K. Ikeuchi. Polarization-based inverse rendering from a single view. In *Ninth IEEE International Conference on Computer Vision*, volume 2, pages 982–987, 2003.

[100] A Naish-Guzman and S Holden. The generalized fitc approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1064, 2007.

[101] Oliver Nelles. *Nonlinear system identification : from classical approaches to neural networks and fuzzy models*. Berlin ; New York : Springer, 2001.

[102] T. Nilssen, K. Karlsen, T. Mannseth, and X.-C. Tai. Identification of diffusion parameters in a nonlinear convection-diffusion equation using the augmented lagrangian method. *Computational Geosciences*, 13:317–329, 2009.

[103] T. K. Nilssen and X. C. Tai. Parameter estimation with the augmented Lagrangian method for a parabolic equation. *journal of optimization theory and applications*, 124(2):435–453, 2005.

[104] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research, 2006.

[105] J.M. Ollinger, M. Corbetta, and G.L. Shulman. Separating processes within a trial in an event-related functional MRI. *Neuroimage*, 13:218–229, 2001).

[106] C. Paciorek. Bayesian smoothing with Gaussian processes using Fourier basis functions in the spectralGP package. *Journal of Statistical software*, 19(2), 2007.

[107] R. Christian Penland, David M. Harrild, and Craig S. Henriquez. Modeling impulse propagation and extracellular potential distributions in anisotropic cardiac tissue using a finite volume element discretization. *Computing and Visualization in Science*, 4:215–226, 2002. 10.1007/s00791-002-0078-4.

[108] X Pennec, P Fillard, and N.A Ayache. Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66:41–66, 2006.

[109] W. Penny, G. Flandin, and N. Trujillo-Barreto. Bayesian comparison of spatially regularised general linear models. *Neuroimage*, 24:350–362, 2005.

[110] R. Perkins and V. Gruev. Signal-to-noise analysis of Stokes parameters in division of focal plane polarimeters. *Optics Express*, 18, 2010.

[111] Theo C. Pilkington. *High performance computing in biomedical research*. CRC Press, 1993.

[112] S. Pitzalis, C. Galletti, R.S. Huang, F. Patria, G. Committeri, G. Galati, P. Fattori, and M.I. Sereno. Wide-field retinotopy defines human cortical visual area. *J Neurosci*, 26:7962–7973, 2006).

[113] S. Pitzalis, M.I. Sereno, G. Committeri, P. Fattori, G. Galati, F. Patria, and C. Galletti. Human v6: the medial motion area. *Cereb Cortex*, 20:411–424, 2010).

[114] S. Pitzalis, M.I. Sereno, G. Committeri, P. Fattori, G. Galati, A. Tosoni, and C. Galletti. The human homologue of macaque area V6A. *Neuroimage*, 82:517–530, 2013).

[115] J.B. Poline and B.M. Mazoyer. Analysis of individual brain activation maps using hierarchical description and multiscale detection. *IEEE transactions on medical imaging*, 13(4):702–710, 1994.

[116] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge university press, 3rd edition, 2007.

[117] John G. Proakis and Dimitris K Manolakis. *Digital Signal Processing*. Pearson Prentice Hall, fourth edition, 2007.

[118] K. Prouskas, A. Patel, J. Pitt, and J. Barria. A multi-agent system for intelligent network load control using a market-based approach. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 231–238, 2000.

[119] J. Quiñonero-Candela and C.E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6, 2005.

[120] A Rahimi and B Recht. Random features for large-scale kernel machines. In *In Neural Information Processing Systems*, 2007.

[121] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[122] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for Machine Learning*. The MIT Press, 2006.

[123] B. Ratliff, C. LaCasse, and S. Tyo. Interpolation strategies for reducing ifov artifacts in microgrid polarimeter imagery. *Opt. Express*, 17:9112–9125, 2009.

[124] J. Rauch and J. Smoller. Qualitative theory of the FitzHugh-Nagumo equations. *Advances in Mathematics*, 27:12–44, 1978.

[125] Patricio S. La Rosa, Hari Eswaran, Hubert Preissl, and Arye Nehorai. Multiscale forward electromagnetic model of uterine contractions during pregnancy. in revision for BMC Medical Physics.

[126] H. Rue and H. Tjelmeland. Fitting Gaussian Markov random fields to Gaussian fields. *Scandinavian Journal of Statistics*, 29(1), 2002.

[127] Y. Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

[128] Y. Y. Schechner and N. Karpel. Recovery of underwater visibility and structure by polarization analysis. *IEEE J. Oceanic Eng*, 30(3):570–587, 2005.

[129] N. Sepulveda, B. Roth, and J. Wikswo. Finite element bidomain calculations. In *IEEE Engineering in Medicine and Biology Society 10th Annual International Conference*, 1988.

[130] M.I. Sereno, A.M. Dale, J.B. Reppas, K.K. Kwong J.W. Belliveau, T.J. Brady, B.R. Rosen, and R.B.H Tootell. Borders of multiple visual areas in humans revealed by functional magnetic resonance. *Science*, 268:889–893, 1995).

[131] M.I. Sereno, C.T. McDonald, and J.M. Allman. Analysis of retinotopic maps in extrastriate cortex. *Cereb Cortex*, 4:601–620, 1994).

[132] K. Shafie, B. Sigal, D. Siegmund, and K. Worsley. Rotation space random fields with an application to fMRI data. *Annals of Statistics*, 31:17321771, 2003).

[133] A Sitz, J Kurths, and H U Voss. Identification of nonlinear spatiotemporal systems via partitioned filtering. *Physical Review*, 2003.

[134] A. Sitz, U. Schwarz, J. Kurths, and H. U. Voss. Estimation of parameters and unobserved components for nonlinear systems from noisy time series. *Phys. Rev. E*, 66:016210, Jul 2002.

[135] Jonas Sjoberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Deylon, Pierreyves Glorennec, Hakan Hjalmarsson, and Anatoli Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1724, 1995.

[136] M.S. Smith and L. Fahrmeir. Spatial bayesian variable selection with application to functional magnetic resonance imaging. *Neuroimage*, 102:417431, 2007).

[137] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, volume 18, page 1257. MIT Press, 2006.

[138] Torsten Soderstrom and Petre Stoica. *System Idntification*. Prentice Hall, 1989.

[139] A. J. Storkey. Truncated covariance matrices and Toeplitz methods in Gaussian processes. In *ICANN*, 1999.

[140] John C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth and Brooks, 1989.

[141] N.Z Sun. *Inverse Problems in Groundwater Modeling*. Kluwer Academic Publishers, Boston, 1994.

[142] K. Tabelow, J. Polzehl, H.U. Voss, and V. Spokoiny. Analyzing fMRI experiments with structural adaptive smoothing procedures. *Neuroimage*, 33:55–62, 2006).

[143] R.B. Tootell, J.D. Mendola, N.K. Hadjikhani, P.J. Ledden, A.K Liu, J.B. Reppas, M.I. Sereno, and A.M. Dale. Functional analysis of V3A and related areas in human visual cortex. *J Neurosci*, 17:7060–7078, 1997).

[144] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[145] John N. Tsitsiklis, Dimitri P. Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, AC-31(9):803–812, 1986.

[146] L. Tung. *A bi-domain model for describing ischemic myocardial D-C potentials*. PhD thesis, MIT, Cambridge, MA, 1978.

[147] J. Tyo, C. LaCasse, and B. Ratliff. Total elimination of sampling errors in polarization imagery obtained with integrated microgrid polarimeters. *Opt. Lett*, 34:3187–3189, 2009.

[148] Pedro A. Valdes-Sosa, Alard Roebroek, Jean Daunizeau, and Karl Friston. Effective connectivity: Influence, causality and biophysical modeling. *NeuroImage*, 2011.

[149] V. K. Vanag and I. R. Epstein. Localised patterns in reaction-diffusion systems. *Chaos*, 17(037110), 2007.

[150] D. Van De Ville, T. Blu, and M. Unser. Surfing the brain: An overview of wavelet-based techniques for fMRI data analysis. *IEEE Engineering in Medicine and Biology Magazine*, 25:6578, 2006).

[151] T. White. *Hadoop: The Definitive Guide*. OReilly Media, 2012.

[152] C. Williams, M. Baran, and E. Bonilla. A note on noise-free Gaussian process prediction with separable covariance functions and grid designs, Dec 2007.

[153] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS 13*, 2001.

[154] Allan R. Willms, Deborah J. Baro, Ronald M. Harris-Warrick, and John Guckenheimer. An improved parameter estimation method for hodgkin-huxley models. *Journal of Computational Neuroscience*, 6:145–168, 1999. 10.1023/A:1008880518515.

[155] A. G. Wilson, D. A. Knowles, and Z. Ghahramani. Gaussian process regression networks. *ICML*, 2012.

[156] A.G. Wilson and R.P. Adams. Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning*, 2013.

[157] A.G. Wilson and R.P. Adams. Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning*, 2013.

[158] A.G. Wilson, E. Gilboa, A. Nehorai, and J.P. Cunningham. GPatt: Fast multidimensional pattern extrapolation with Gaussian processes. *ArXiv*, page 1310.5288, 2014).

[159] C. K. Winkle and N. Cressie. A dimension-reduced approach to space-time Kalman filtering. *Biometrika*, 86, 1999.

[160] K.J. Worsley, S. Marrett, and P. Neelin andA.C. Evans. Searching scale space for activation in PET images. *Human Brain Mapping*, 4:74–90, 1996).

[161] G. Xia and A. E. Gelfand. Stationary process approximation for the analysis of large spatial datasets. Technical report, Duke U, 2006.

[162] Peng Y., P. Chavali, E. Gilboa, and A. Nehorai. Parallel load schedule optimization with renewable distributed generators in smart grids. *Smart Grid, IEEE Transactions on*, 4(3):1431–1441, Sept 2013.

[163] Y. Yue, J.M. Loh, and M.A. Lindquist. Adaptive spatial smoothing of fMRI images. *Statistics and Its Interface*, 3:3–13, 2010.

[164] E. Zarahn, G.K Aguirre, and M. D'Esposito. Empirical analyses of BOLD fMRI statistics. I. spatially unsmoothed data collected under null-hypothesis conditions. *Neuroimage*, 5:179–197, 1997).

[165] Y. Zhang and W.E. Leithead. Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression. *Journal of Statistical Computation and Simulation*, 77:329–348, 2007.

Vita

Elad Gilboa

Degrees

Ph.D. Electrical and System Engineering, Washington University in St. Louis, USA, August 2014
M.S. Electrical and System Engineering, Washington University in St. Louis, USA, May 2011
M.E. Biomedical Engineering, Technion, Israel, December 2009
B.S. Cum Laude, Electrical and Computer Engineering, Technion, Israel, June 2004

Professional Societies

Association for Graduate Engineering Students

Journal Publications

E. Gilboa, Y. Saatci, and J. P. Cunningham. "Scaling multidimensional inference for structured Gaussian processes." *IEEE-TPAMI*. 2013.
P. Yang, P. Chavali, **E. Gilboa**, and A. Nehorai, "Parallel load schedule optimization with renewable distributed generators in smart grids," *IEEE Trans. on Smart Grid*, 2013.
E. Gilboa, P. S. La Rosa, and A. Nehorai. "Estimating electrical conductivity tensors of biological tissues using microelectrode arrays," *Annals of Biomedical Engineering*, 2012

Conference Publications

E. Gilboa, J. P. Cunningham, A. Nehorai, and V. Gruev. "Fast Gaussian processes for image interpolation of division of focal plane polarimeter." *SPIE* 2014
E. Gilboa, Y. Saatci, and J. P. Cunningham. "Scaling Multidimensional Gaussian Processes using Projected Additive Approximations". *ICML* 2013
E. Gilboa, P. Chavali, P. Yang, and A. Nehorai. "Distributed Optimization via Adaptive Regularization for Large Problems with Separable Constraints." *ICCASP* 2013

E. Gilboa, P. S. La Rosa, and A. Nehorai. "Estimating electrical conductivity tensors of biological tissues using microelectrode arrays," *IEEE EMBS*, 2012

C. Holmes, M. Wronkiewicz, T. Somers, J. Liu, D. Kim, D. Bundy, **E. Gilboa**, E. Leuthardt. "IPSIHAND BRAVO: An Improved EEG-Based Brain-Computer Interface for Hand Motor Control Rehabilitation," *IEEE EMBS*, 2012

Working Publications

A. G. Wilson*, **E. Gilboa*** , A. Nehorai, J. P. Cunningham. "GPatt: Fast multidimensional pattern extrapolation with Gaussian Processes." *UAI*, 2014 Submitted

E. Gilboa, J. P. Cunningham, A. Nehorai, and V. Gruev. "Fast Gaussian processes for image interpolation of division of focal plane polarimeter." *Optic Express*, Submitted.

E. Gilboa, F. Strappini, K. Kay, A. Nehorai, A. Snyder. "Gaussian Processes denoising for fMRI data." *PNAS*, In Preparation.

August 2014