Project Thesis
PRO-010

Stochastic Processes for Estimation of Spatiotemporal Environmental Fields

by
Nils Rottmann

Supervisors: Prof. Dr.–Ing. R. Seifried
Dipl.–Ing. Eugen Solowjow

Hamburg University of Technology
Institute of Mechanics and Ocean Engineering
Prof. Dr.–Ing. R. Seifried

Hamburg, May 2017

Contents

List of Figures **iii**

List of Abbreviations **v**

1 Introduction **1**

1.1 Thesis Organization 2

1.2 Mathematical Notation 3

2 Theory **5**

2.1 Gaussian Processes 5

2.1.1 Conditional Distribution 6

2.1.2 Gaussian Processes without Noise 8

2.1.3 Noisy Observations 9

2.1.4 Parameter Adjustment 9

2.2 Markov Models 10

2.2.1 Markov Chains 10

2.2.2 Markov Random Fields 11

2.2.3 Gaussian Markov Random Fields 12

2.3 GP with built-in GMRF 13

2.3.1 Spatial Field 13

2.3.2 Expansion to Noise-free Observations 15

2.3.3 Parameter Adjustment 17

2.4 Variogram 18

3 Toolbox **19**

3.1 Folder Structure 19

3.2 Utilization of the Toolbox 20

3.3 Adding Covariance or Precision Functions 21

4 Simulation Results **25**

4.1 Data Sets 26

4.1.1 Artificial Data 26

4.1.2 Real Data 26

4.2 GP vs GP with built-in GMRF 27

4.3 Optimization 29

4.3.1 Hyperparameters 29

4.3.2 Generating Points: Positions 32

4.3.3 Generating Points: Number 32

4.4 Time Variant Systems 34

4.5 Variogram 36

4.6 Noise-free Observations and Mean Function 38

4.7 Identity Precision Matrix and Sparse Covariance Functions 40

5 Summary and Conclusions **43**

5.1 Summary 43

5.2 Conclusion 44

5.3 Outlook 45

Bibliography 47

Appendix 49

List of Figures

1.1 Autonomous Underwater Vehicle HippoCampus 2
1.2 Oil Slick in the Golf of Mexico, 2010 3

2.1 Multivariate Gaussian Distribution 7
2.2 Gaussian Process 9
2.3 Directed Markov Chain 11
2.4 Markov Graph 12
2.5 Neighbouring System 12
2.6 Data Arrangement 18

3.1 Folder Structure 20

4.1 Artificial Concentration Field 27
4.2 ROMS Model Temperature Field 28
4.3 Comparison GP and GP with built-in GMRF 29
4.4 Computational Time of GP Estimation 30
4.5 Improvement of the Hyperparameters 31
4.6 Optimization of the Generating Points 33
4.7 Number of Generating Points 35
4.8 Comparison Time Variant Methods 36
4.9 Estimation with Sequential Algorithm 37
4.10 Semi-Variograms 38

4.11 Unidimensional-, Multidimensional Covariance Function 39
4.12 Noise-free Estimation 40
4.13 Mean Function Comparison 41
4.14 (a) Identity Matrix as Precision Matrix; (b) Distributed Covariance Matrix 42

5.1 Example Path Finding 45

List of Abbreviations

GP	Gaussian Process
GMRF	Gaussian Markov Random Field
AUV	Autonomous Underwater Vehicle
MSE	Mean Squared Error
AIC	Akaike Information Criterion
CPU	Central Processing Unit
RAM	Random-Access Memory
CFD	Computational Fluid Dynamics

Chapter 1

Introduction

Autonomous mobile devices, such as autonomous underwater vehicles (AUV) in figure 1.1), desire a fast environmental monitoring to full fill given tasks. Such tasks could be the recording of the water temperature in a certain area or the analysis of environmental pollution (figure 1.2). The acquired information can also be used for path planning and control of the autonomous vehicles ([Binney et al., 2010],[Sydney and Paley, 2014],[Binney et al., 2013]). Such a path planning algorithm tries to reduce the estimation uncertainty. Therefore uncertainty estimations have to be identified. Thus, the method used for monitoring should enable persistent sensing, mean and variance estimation. The mean values are the required results of the estimation procedure and the variance the uncertainty measurement which could be used for path planning. Such a model can either be a computational fluid dynamics (CFD) simulation or a probabilistic model. The CFD model is based on physical differential equations to evaluate the field. In order to solve this equations numerically, boundary conditions have to be known. A major disadvantage of such an approach is the huge amount of computational time required to solve the differential equations numerically. In contrast, a probabilistic model does not require any boundary conditions. But this kind of model is not based on physical laws, it is only a probabilistic scheme. A major advantage of such models are the low computational costs to achieve estimation results. Well known probabilistic models are Gaussian Processes ([Rasmussen and Williams, 2006]). A drawback of such GPs is the computational complexity which scales with the power of three for an increasing number of observations. In order to reduce computational effort a planning schedule to get data with the most information ([Ma et al., 2016]) or sparse Gaussian Processes ([Csató and Opper, 2002],[Ranganathan et al., 2011]) could be used. Another approach of environmental monitoring are Gaussian Markov Random Fields ([Rue and Held, 2005]), which are frequently used as computationally efficient models. A further improvement of these models can be achieved

by linkage with Gaussian Random Fields ([Simpson et al., 2012]).
In this thesis the combination of Gaussian Processes and Gaussian Markov Random Fields ([Xu et al., 2015]) is introduced. The advantage of such a linkage is the reduced computational complexity, because the size of the covariance inverse used in this method remains constant by increasing the number of observations. The presented method offers different tuning knobs such as the hyperparameters of the covariance function or the number of so called generating points. In order to find an optimal scheme for environmental monitoring, parameter improvement using the log likelihood are used. The Akaike information criterion ([Akaike, 2011]) can be used to obtain an optimal number of generating points which create the Gaussian Markov Random Field. Furthermore, an analysis if a given field is non-stationary is introduced in this thesis([Cressie, 1993]). The non-stationary behaviour can be taken into account in the spatial models ([Fuglstad et al., 2014],[Fuglstad et al., 2013],[Fuglstad et al., 2015]).

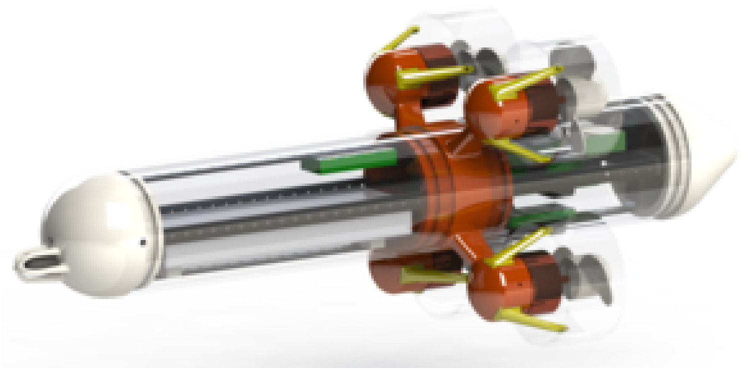


Figure 1.1: The figure shows the AUV HippoCampus developed at the Institute of Mechanics and Ocean Engineering, Hamburg University of Technology.

1.1 Thesis Organization

Chapter 2 introduces the theoretical background used in this thesis. First Gaussian Processes are explained. In the following Markov Models, in the order Markov Chains, Markov Random Fields and Gaussian Markov Random Fields are introduced. At the end of this chapter, the combination of Gaussian Processes with built-in Gaussian Markov Random Fields are derived.
In chapter 3 the toolbox is introduced, which implements the theoretical results and is used for benchmarking estimations.
The toolbox is applied to different data sets representing different problems in chapter 4. Problems which are considered comprise of the comparison between

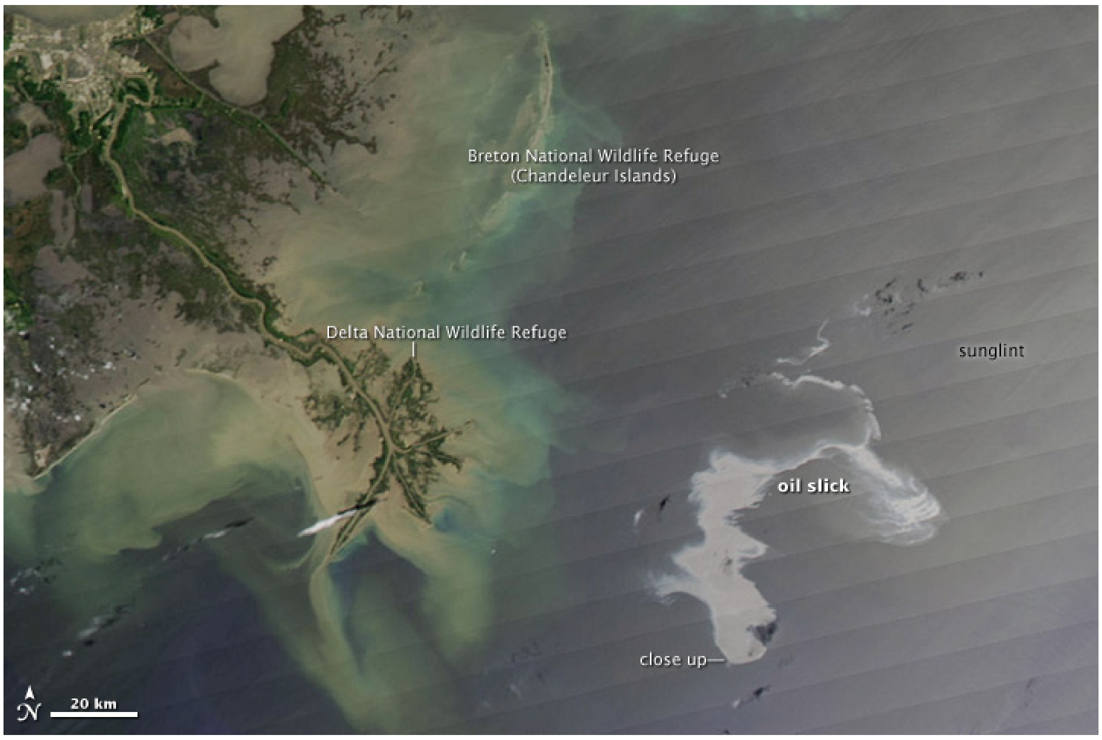

Breton National Wildlife Refuge
(Chandeleur Islands)
Delta National Wildlife Refuge
sunglint
oil slick
close up
20 km

Figure 1.2: The figure shows the oil slick after the catastrophe of the Deepwater Horizon. The offshore pliable platform is located at the position marked with close up. The light grey region is the oil slick.
`http://eoimages.gsfc.nasa.gov/images/imagerecords/43000/43768/gulf_amo_2010115.jpg`

Gaussian Processes and Gaussian Processes with built-in Gaussian Markov Random Fields, the optimization of the hyperparameters in the covariance function and the positions of the generating points, time variant fields, non-stationary estimations and estimations without zero mean function.
Chapter 5 presents a summary of the simulation results and proposes an effective scheme for environmental field estimation. At the end a short outlook is given and, in addition, the importance of the variance for path planning applications are shown and illustrated by a brief example.

1.2 Mathematical Notation

In this section the mathematical notation, which is used throughout this paper, is introduced.

- Scalar values are written as lower-case letters (e.g. a), vectors are depicted as small bold lower-case letters (e.g. $\boldsymbol{a}, \boldsymbol{\mu}$), matrices as bold capital letters (e.g. $\boldsymbol{A}, \boldsymbol{\Sigma}$) and sets as capital letters (e.g. A).

- A single scalar value from a matrix placed in row i and column j is written as $(\boldsymbol{A})_{ij}$.
- The Moore-Penrose pseudoinverse of a Matrix $\boldsymbol{A}$ is depicted as $\boldsymbol{A}^{\dagger}$.
- The trace of a matrix is written as $\mathrm{tr}(\boldsymbol{A})$.
- A function $\mu(x)$ can be shortly written as μ_x.
- The identity matrix is written as $\boldsymbol{I}$ where the size is chosen appropriate to the usage.
- The variance of a variable x is written as $\mathrm{var}(x)$, the covariance between the variables x and y as $\mathrm{cov}(x, y)$ and the expectation value of a variable x as $\mathrm{E}(x)$.

Chapter 2

Theory

In this chapter the theory of Gaussian Processes (GPs) [Rasmussen and Williams, 2006], Markov Random Fields ([Blake et al., 2011], [Li, 2009]) and Gaussian Markov Random Fields (GMRFs, [Rue and Held, 2005]) is described. After introducing GPs and GMRFs they are combined and result in Gaussian Processes with built-in Gaussian Markov Random Fields ([Xu et al., 2015]). In this method a discontinuity for noise free measurements can be shown. A solution to avoid such a discontinuity is explained. As a result, a method using an identity matrix as precision matrix is introduced. In the last section the computation of a variogram [Dutter, 2000] is shown. The variogram is required for the analysis whether a given field is stationary or not. Such a result can lead to an idea which kind of kernel function should be used for the estimation process.

2.1 Gaussian Processes

First, the derivation of the conditional distribution of a Gaussian random vector, given another Gaussian random vector, is shown. Considering this derivation, Gaussian Processes are explained. At the end, a method is presented how to enhance the estimation by improving the hyperparameters of the kernel function. A detailed description of Gaussian Processes is given by [Rasmussen and Williams, 2006].

2.1.1 Conditional Distribution

In order to understand Gaussian Processes first consider a multivariate normal distribution with $\boldsymbol{x}' = \{x'_1, \dots, x'_k\}$

$$f(\boldsymbol{x}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \cdot \exp\left(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}' - \boldsymbol{\mu})\right), \tag{2.1}$$

which can be written as

$$\boldsymbol{x}' \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.2}$$

Figure 2.1 shows such a normal distribution.
By partition the Gaussian random vector $\boldsymbol{x}'$ into $\boldsymbol{x}$ and $\boldsymbol{y}$, where both are jointly Gaussian random vectors, the term (2.2) becomes

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{B} \end{bmatrix}\right). \tag{2.3}$$

The marginal distribution of $\boldsymbol{x}$ is

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{A}) \tag{2.4}$$

and the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ is

$$\boldsymbol{x}|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y), \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top). \tag{2.5}$$

Thus, the conditional expectation and the covariance matrix can be written as

$$\begin{aligned} \mathrm{E}(\boldsymbol{x}|\boldsymbol{y}) &= \boldsymbol{\mu}_x + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y) \\ \mathrm{var}(\boldsymbol{x}|\boldsymbol{y}) &= \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top. \end{aligned} \tag{2.6}$$

Proof. Define $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{M}\boldsymbol{y}$ with $\boldsymbol{M} = -\boldsymbol{C}\boldsymbol{B}^{-1}$. There $\boldsymbol{z}$ and $\boldsymbol{y}$ are uncorrelated and, since they are jointly normal, they are independent. This can be shown by

$$\begin{aligned} \mathrm{cov}(\boldsymbol{z}, \boldsymbol{y}) &= \mathrm{cov}(\boldsymbol{x}, \boldsymbol{y}) + \mathrm{cov}(\boldsymbol{M}\boldsymbol{y}, \boldsymbol{y}) \\ &= \boldsymbol{C} + \boldsymbol{M}\mathrm{var}(\boldsymbol{y}) \\ &= \boldsymbol{C} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{B} \\ &= 0 \end{aligned}$$

The expectation value of $\boldsymbol{x}|\boldsymbol{y}$ is calculated to proof the first part of equation (2.5), using the fact that $\mathrm{E}(\boldsymbol{z}|\boldsymbol{y}) = \mathrm{E}(\boldsymbol{z}) = \boldsymbol{\mu}_x + \boldsymbol{M}\boldsymbol{\mu}_y$.

$$\begin{aligned} \mathrm{E}(\boldsymbol{x}|\boldsymbol{y}) &= \mathrm{E}(\boldsymbol{z} - \boldsymbol{M}\boldsymbol{y}|\boldsymbol{y}) \\ &= \mathrm{E}(\boldsymbol{z}|\boldsymbol{y}) - \mathrm{E}(\boldsymbol{M}\boldsymbol{y}|\boldsymbol{y}) \\ &= \mathrm{E}(\boldsymbol{z}) - \boldsymbol{M}\boldsymbol{y} \\ &= \boldsymbol{\mu}_x + \boldsymbol{M}(\boldsymbol{\mu}_y - \boldsymbol{y}) \\ &= \boldsymbol{\mu}_x + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y) \end{aligned}$$

The covariance matrix can be derived from

$$\begin{aligned}
\mathrm{var}(\boldsymbol{x}|\boldsymbol{y}) &= \mathrm{var}(\boldsymbol{z} - \boldsymbol{M}\boldsymbol{y}|\boldsymbol{y}) \\
&= \mathrm{var}(\boldsymbol{z}|\boldsymbol{y}) - \mathrm{var}(\boldsymbol{M}\boldsymbol{y}|\boldsymbol{y}) - \boldsymbol{M}\mathrm{cov}(\boldsymbol{z}, -\boldsymbol{y}) - \mathrm{cov}(\boldsymbol{z}, -\boldsymbol{y})\boldsymbol{M}^\top \\
&= \mathrm{var}(\boldsymbol{z}|\boldsymbol{y}) \\
&= \mathrm{var}(\boldsymbol{z}) \\
&= \mathrm{var}(\boldsymbol{x} + \boldsymbol{M}\boldsymbol{y}) \\
&= \mathrm{var}(\boldsymbol{x}) + \boldsymbol{M}\mathrm{var}(\boldsymbol{y})\boldsymbol{M}^\top + \boldsymbol{M}\mathrm{cov}(\boldsymbol{x}, \boldsymbol{y}) + \mathrm{cov}(\boldsymbol{y}, \boldsymbol{x})\boldsymbol{M}^\top \\
&= \boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{C}^\top - 2\boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top \\
&= \boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top - 2\boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top \\
&= \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^\top
\end{aligned}$$

$\square$

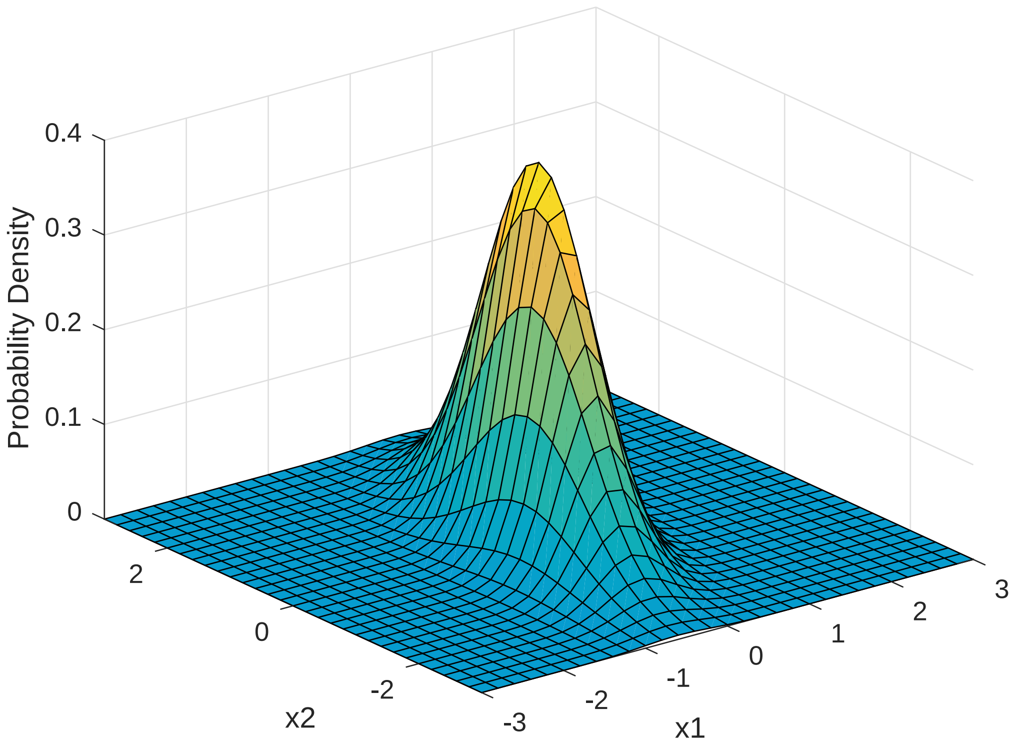


Figure 2.1: The figure shows a two dimensional Gaussian Distribution created using Matlab with $\boldsymbol{\mu} = [0\ 0]$ and $\boldsymbol{\Sigma} = [0.2\ 0.2;\ 0.2\ 1.0]$.

2.1.2 Gaussian Processes without Noise

Gaussian Processes are probabilistic models which can be used to estimate, on the basis of known data, a mean and a variance for an unknown data point. They use the relation given in equation (2.6).

From now on a more convenient nomenclature is used. Consider a training set $T = \{(\boldsymbol{x}_i, y_i)\} = (X, \boldsymbol{y})$ where $\boldsymbol{x}_i$ denotes an input vector of dimension D and y_i denotes a scalar output. Here, X contains all the input data and $\boldsymbol{y}$ all the output (target) data of the training set. In order to determine expectation values for $\boldsymbol{y}_*$, the values $\boldsymbol{y}$ at the states $X = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ have to be known. Rewriting equation (2.3) with the predefined training set yields

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}(X) \\ \boldsymbol{\mu}(X_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(X,X) & \boldsymbol{K}(X,X_*) \\ \boldsymbol{K}(X_*,X) & \boldsymbol{K}(X_*,X_*) \end{bmatrix} \right), \tag{2.7}$$

where $\boldsymbol{\mu}(\cdot)$ are the expectation values given a certain input set and $\boldsymbol{K}(\cdot,\cdot)$ a covariance matrix determined through two input sets. The expectation values and the variance of the unknown values $\boldsymbol{y}_*$ can be derived with equation (2.6), as

$$\begin{aligned} \mathrm{E}(\boldsymbol{y}_*|\boldsymbol{y}, X, X_*) &= \boldsymbol{\mu}_* + \boldsymbol{K}_*^\top \boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \\ \mathrm{var}(\boldsymbol{y}_*|\boldsymbol{y}, X, X_*) &= \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top \boldsymbol{K}^{-1}\boldsymbol{K}_*, \end{aligned} \tag{2.8}$$

where $\boldsymbol{\mu}(X) = \boldsymbol{\mu}$, $\boldsymbol{\mu}(X_*) = \boldsymbol{\mu}_*$, $\boldsymbol{K}(X,X) = \boldsymbol{K}$, $\boldsymbol{K}(X,X_*) = \boldsymbol{K}_*$ and $\boldsymbol{K}(X_*,X_*) = \boldsymbol{K}_{**}$. The mean function $\boldsymbol{\mu}(\cdot)$ can be generated using proper information of the target values y_i given the input $\boldsymbol{x}_i$ or, if non such information is available, can be set to zero. The covariance function (kernel function) defines nearness or similarity and the covariance matrix $\boldsymbol{K}$ has to be positive definite. Possible types of such kernels are

- $k = k(\boldsymbol{x} - \boldsymbol{x}')$ (stationary, invariant to translation in the input space)
- $k = k(||\boldsymbol{x} - \boldsymbol{x}'||)$ (isotropic, invariant to all rigid motions)
- $k = k(\boldsymbol{x} \cdot \boldsymbol{x}')$ (dot product, invariant to rotation)

A commonly used kernel function is the squared-exponential covariance function $k_{\mathrm{se}}(\tau) = \sigma_f^2 \cdot \exp(-\frac{\tau^2}{2l^2})$ with $\tau = ||\boldsymbol{x} - \boldsymbol{x}'||$. The variables σ_f and l are hyperparameters. This kernel is smooth because the function is infinitely differentiable. The properties of a kernel around $\boldsymbol{0}$ determine the smoothness of the stationary process. A one dimensional example of a Gaussian Process is shown in figure 2.2.

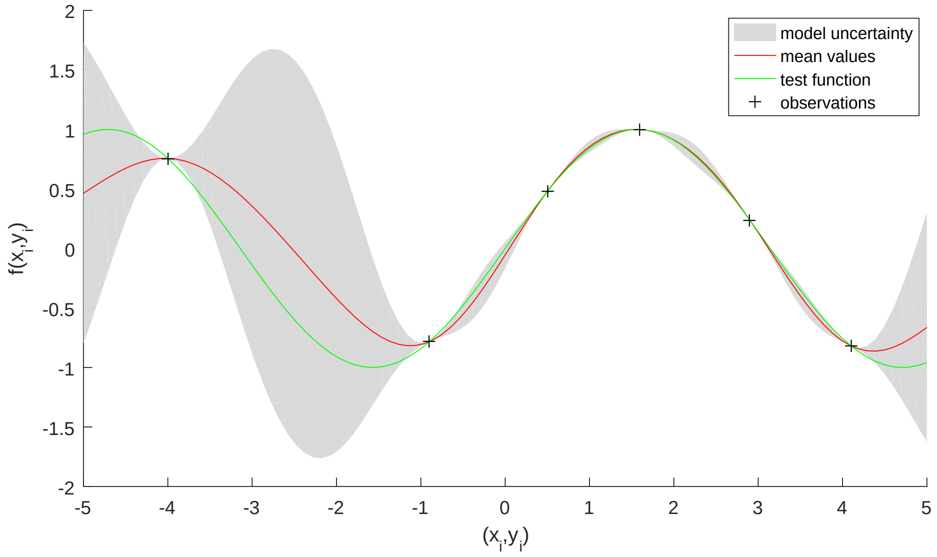


Figure 2.2: Example for a one dimensional Gaussian Process. The test function is a sinus; Six observation points have been used to determine an estimation result (mean values) and the model uncertainty (variance).

2.1.3 Noisy Observations

Are the given data, thus the training set $T = \{(\boldsymbol{x}_i, y_i)\} = (X, \boldsymbol{y})$, corrupted with white noise, e.g. $y_i = z(\boldsymbol{x}_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma_n^2)$, it can be taken into account by adding $\sigma_n^2 \boldsymbol{I}$ to the covariance matrix $\boldsymbol{K}$. The matrix $\boldsymbol{I}$ describes an identity matrix with convenient size. Equation (2.7) becomes

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}(X) \\ \boldsymbol{\mu}(X_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^\top & \boldsymbol{K}_{**} \end{bmatrix} \right) \tag{2.9}$$

with

$$\begin{aligned} \mathrm{E}(\boldsymbol{y}_* | \boldsymbol{y}, X, X_*) &= \boldsymbol{\mu}_* + \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) \\ \mathrm{var}(\boldsymbol{y}_* | \boldsymbol{y}, X, X_*) &= \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{K}_*. \end{aligned} \tag{2.10}$$

2.1.4 Parameter Adjustment

Assuming zero mean ($\boldsymbol{\mu}(\cdot) = 0$), in equation (2.10) only the parameters of the kernel function $\boldsymbol{\theta}$ are unknown. Either one can choose these parameters by trial and error or use Bayesian Optimization to find optimal parameters for a certain problem. The aim of Bayesian Optimization is to find parameters $\boldsymbol{\theta}$ which maximize equation (2.1). The logarithm of the likelihood,

$$\log(f(\boldsymbol{y}|X,\boldsymbol{\theta})) = -\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{1}{2}\log||\boldsymbol{K}|| - \frac{n}{2}\log(2\pi), \tag{2.11}$$

is used for maximization. In order to use optimization methods, such as gradient descent, the derivation of the log likelihood (2.33), defined as

$$\frac{\partial \log(f(\boldsymbol{y}|X,\boldsymbol{\theta}))}{\partial \theta_j} = \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \boldsymbol{K}^{-1})\frac{\partial \boldsymbol{K}}{\partial \theta_j}\right), \tag{2.12}$$

with $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{y}$ and n the number of data points in the training set, is required.

2.2 Markov Models

In this section Gaussian Markov Random Fields are explained. The section starts with Markov Chains, turns than over into Markov Random Fields and finally gets to Gaussian Markov Random Fields. The theory about Markov Random Fields presented in this section can be found in [Blake et al., 2011] and [Li, 2009]. A detailed description of Gaussian Markov Random Fields is given by [Rue and Held, 2005].

2.2.1 Markov Chains

A Markov Chain is a representation of a sequence of variables $(x_1, x_2, \dots)$ which are specified by the conditionals $p(x_i|x_{i-1}, \dots, x_1)$. A common example for a Markov Chain uses the weather which can be either sunny or rainy, such that $x_i \in L = \{\text{sunny}, \text{rainy}\}$. The weather on day i can be influenced by the weather of many foregoing days but, for the simplest case, it would be only directly related to the weather of day $i-1$. Such a Markov chain is shown in figure 2.3. The underlying assumption is called a first-order Markov assumption

$$p(x_i|x_{i-1}, \dots, x_1) = p(x_i|x_{i-1}). \tag{2.13}$$

In the same way a n-th-order Markov assumption can be described as

$$p(x_i|x_{i-1}, \dots, x_1) = p(x_i|x_{i-1}, \dots, x_{i-n}). \tag{2.14}$$

Of course, even in the first-order Markov chain all foregoing variables are linked implicitly with x_i. As for our example, the conditional probabilities can be described as

$$P = \begin{bmatrix} p(\text{sunny}|\text{sunny}) & p(\text{rainy}|\text{sunny}) \\ p(\text{sunny}|\text{rainy}) & p(\text{rainy}|\text{rainy}) \end{bmatrix} = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}, \tag{2.15}$$

which can also be represented in a graph as shown in figure 2.4. Continuing with the example and assuming the weather on day one to be sunny gives $X_1 = [p(\text{sunny}) \quad p(\text{rainy})] = [1 \quad 0]$. Calculating the conditional probabilities for day two leads to

$$X_2 = X_1 P = \begin{bmatrix} 0.9 & 0.1 \end{bmatrix} \tag{2.16}$$

and for day $1 + n$ to

$$X_{1+n} = X_1 P^n. \tag{2.17}$$

2.2.2 Markov Random Fields

A Markov Random Field is described by an undirected graph $S = (V, E)$, where $V = \{1, \ldots, n\}$ are the vertices and E the edges of the graph, in which the vertices and edges are related via a neighbouring system

$$N = \{N_i | \forall i \in V\}. \tag{2.18}$$

Here N_i is a set of vertices neighbouring i. The properties for such a neighbouring system are

(1) $i \notin N_i$

(2) $i \in N_j \leftrightarrow j \in N_i$

For a regular lattice such a neighbouring system could be described as

$$N_i = \{j \in V | d(x_j, x_i) \leq r; j \neq i\}, \tag{2.19}$$

where x_j and x_i are the positions of the vertices j and i and $d(\cdot, \cdot)$ a distance measure. In figure 2.5 examples of such a neighbouring system are shown. Other (irregular) neighbouring systems are possible.

A Markov Random Field can be defined as a family of random variables $F = \{F_i, \ldots, F_n\}$, which are defined on the set of vertices V. Each variable F_i takes a value $f_i \in L$, where L is a label set (e.g. $L = \{\text{sunny}, \text{rainy}\}$) and the probability of F taking the value f_i is

$$p(F_i = f_i) = p(f_i). \tag{2.20}$$

For a Markov Random Field on V the following two properties have to hold:

$$\begin{tikzcd} \boxed{x_1} \arrow[r] & \boxed{\cdots} \arrow[r] & \boxed{x_{i-1}} \arrow[r] & \boxed{x_i} \arrow[r] & \boxed{x_{i+1}} \arrow[r] & \boxed{\cdots} \arrow[r] & \boxed{x_n} \end{tikzcd}$$

Figure 2.3: The figure shows a directed Markov Chain where the foregoing events have an influence on the following events.

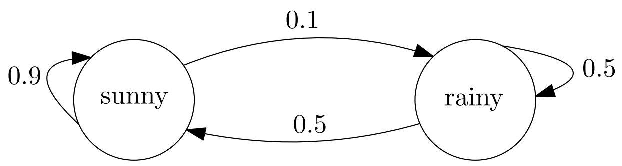
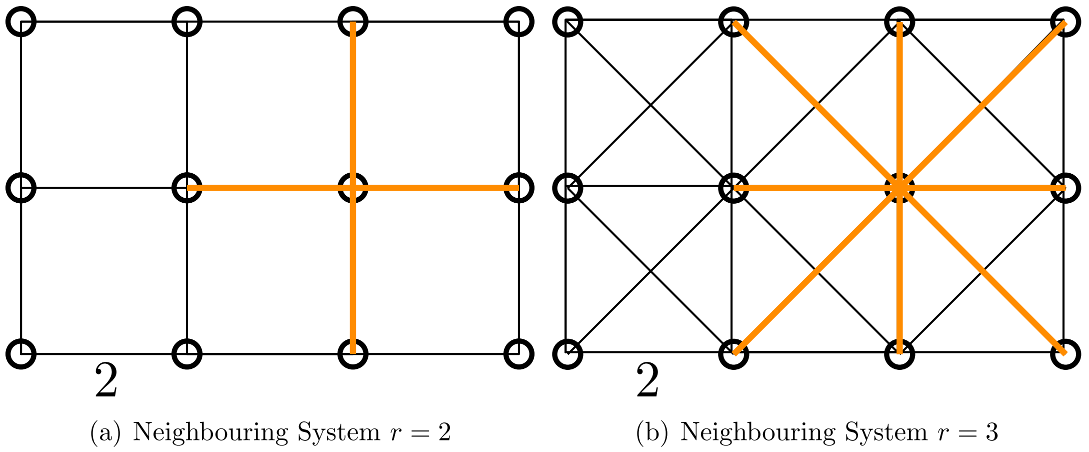


Figure 2.4: A Markov Graph with a first-order Markov assumption. The conditional probabilities are written on the arrows.



(a) Neighbouring System $r = 2$ (b) Neighbouring System $r = 3$

Figure 2.5: Systems with a Regular Lattice and $r = 2$ (left) and $r = 3$ (right). The yellow lines lead to the neighbours of the central point. They represent the edges. The distance between two nearest points, vertical and horizontal, is 2.

(1) $p(f) > 0 \quad \forall f \in F$ (Positivity)

(2) $p(f_i | f_{V-\{i\}}) = p(f_i | N_i)$ (Markovianity)

The positivity is assumed for technical reasons and normally no problem in practice. The Markovianity depicts that, given all neighbours to i, the non-neighbours do not have any influence on the probability of i. In other words, labels without edges in between have to be conditionally independent, thus the equation

$$p(x_i, x_j | \boldsymbol{x}_{-\{ij\}}) = p(x_i | \boldsymbol{x}_{-\{ij\}}) p(x_j | \boldsymbol{x}_{-\{ij\}}) \tag{2.21}$$

has to be fulfilled if x_i and x_j are not neighbouring each other.

2.2.3 Gaussian Markov Random Fields

A Gaussian Markov Random Field has the same structure as a Markov Random Field adding one restriction, such that the probability distribution of

$F = \{F_i, \dots, F_n\}$ is normally distributed with mean $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. The conditional independence between pairwise unconnected vertices, thus the information of the graph S, can be found in the parameters of the normal distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Since the mean does not have any influence on the conditional independence properties of $\boldsymbol{F}$ only the covariance matrix $\boldsymbol{\Sigma}$ remains. Considering the inverse covariance matrix instead of the covariance matrix, the so called precision matrix $\boldsymbol{Q} = \boldsymbol{\Sigma}^{-1}$, it turns out that

$$p(x_i, x_j | \boldsymbol{x}_{-\{ij\}}) = p(x_i | \boldsymbol{x}_{-\{ij\}}) p(x_j | \boldsymbol{x}_{-\{ij\}}) \quad \Longleftrightarrow \quad \boldsymbol{Q}_{ij} = 0. \tag{2.22}$$

A proof of this property can be found in [Rue and Held, 2005, p. 22]. These important property of Gaussian Markov Random Fields makes the precision matrix sparse, which leads to an immense reduction of computational effort.

2.3 GP with built-in GMRF

This section deals with the construction of a Gaussian Process with built-in Gaussian Markov Random Field. This method can lead to an immense reduction in computational effort compared to Gaussian Processes but leads to worse estimation results. Nevertheless, the reduction in computational effort makes this method interesting for the utilization in AUVs. A detailed overview of the theory for Gaussian Processes with built-in Gaussian Markov Random Fields can be found in [Xu et al., 2015].

2.3.1 Spatial Field

First, consider a Gaussian Markov Random Field with respect to a graph $S = (V, E)$ and $F = \{f(\boldsymbol{p}_1), \dots, f(\boldsymbol{p}_m)\}^\top \sim \mathcal{N}(0, \boldsymbol{Q}^{-1})$. As shown $Q_{ij} \neq 0 \Leftrightarrow \{i, j\} \in E$ (equation 2.22) or, in other words, $f(\boldsymbol{p}_i)$ and $f(\boldsymbol{p}_j)$ are not conditionally independent.

A spatial field can be modelled as

$$z(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sum_{j=1}^{m} \lambda(\boldsymbol{x}, \boldsymbol{p}_j) f(\boldsymbol{p}_j), \tag{2.23}$$

which is a Gaussian Process with a built-in Gaussian Markov Random Field with a weighting function $\lambda(\cdot, \cdot)$. The weighting function can be seen as a similarity function (e.g. squared exponential kernel). The vertices of the graph S, $\{\boldsymbol{p}_1, \dots, \boldsymbol{p}_m\}$, are called generating points. The number and the position of the generating points can be determined using a model selection criterion and the log

likelihood method.
The advantage of such an approach is, that there are many tuning knobs, such as a different number of generating points or a different structure of the precision matrix. Thus, the model can be adjusted to different problems using this tuning knobs.

In order to develop an estimation scheme using the given spatial field, assume there are observations $\boldsymbol{y} = (y_1, \dots, y_n)^\top$ at points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ given. The observations are noisy with $y_i = z(\boldsymbol{x}_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma_w^2)$, thus an independent and identically distributed Gaussian white noise. The covariance matrix for $\boldsymbol{y}$ is

$$\boldsymbol{C} = \boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\Lambda}^\top + \sigma_w^2\boldsymbol{I} \tag{2.24}$$

and the covariance between $\boldsymbol{y}$ and $z_0(\boldsymbol{s})$, where $\boldsymbol{s}$ is a certain point of interest, is

$$\boldsymbol{k} = \boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\lambda}. \tag{2.25}$$

Here $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times m}$ is a matrix given by $(\boldsymbol{\Lambda})_{ij} = \lambda(\boldsymbol{x}_i, \boldsymbol{p}_j)$ and $\boldsymbol{\lambda} \in \mathbb{R}^{1\times m}$ is a vector given by $(\boldsymbol{\lambda})_i = \lambda(\boldsymbol{p}_i, \boldsymbol{s})$. A proof can be found in [Xu et al., 2015, p. 79,80].

With the foregoing conclusions, it is possible to make a prediction by using the terms for Gaussian Process Regression as shown in section (2.1). By inserting the covariance matrices from (2.24) and (2.25) into (2.8), where $\boldsymbol{K} = \boldsymbol{C}$ and $\boldsymbol{K}_* = \boldsymbol{k}$,

$$\begin{aligned} \mathrm{E}(z_0|\boldsymbol{y}) &= \mu(\boldsymbol{s}) + (\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\Lambda}^\top + \sigma_w^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}(X)) \\ \mathrm{var}(z_0|\boldsymbol{y}) &= \boldsymbol{\lambda}^\top\boldsymbol{Q}^{-1}\boldsymbol{\lambda} - (\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\Lambda}^\top + \sigma_w^2\boldsymbol{I})^{-1}(\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\lambda}) \end{aligned} \tag{2.26}$$

can be derived. Using the Woodbury Matrix Identity equation 2.26 transforms into

$$\begin{aligned} \mathrm{E}(z_0|\boldsymbol{y}) &= \mu(\boldsymbol{s}) + \boldsymbol{\lambda}^\top\hat{\boldsymbol{Q}}^{-1}\hat{\boldsymbol{y}} \\ \mathrm{var}(z_0|\boldsymbol{y}) &= \boldsymbol{\lambda}^\top\hat{\boldsymbol{Q}}^{-1}\boldsymbol{\lambda} \end{aligned} \tag{2.27}$$

with

$$\begin{aligned} \hat{\boldsymbol{Q}} &= \boldsymbol{Q} + \sigma_w^{-2}\boldsymbol{\Lambda}^\top\boldsymbol{\Lambda} \\ \hat{\boldsymbol{y}} &= \sigma_w^{-2}\boldsymbol{\Lambda}^\top(\boldsymbol{y} - \boldsymbol{\mu}(X)). \end{aligned} \tag{2.28}$$

A proof can be found in [Xu et al., 2015, p. 81,82]. Comparing the estimation method with the Gaussian Processes presented in section 2.1, a difference in the matrices which have to be inverted can be seen. For the Gaussian Process a matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ has to be inverted, whereas for the GP with built-in GMRF a matrix $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ has to be inverted. The computational complexity, without calculating the inverse, grows linearly with the number of observations n. In combination with the computational effort for the matrix inversion it can be shown that Gaussian Processes have a computational complexity of $\mathcal{O}(n^3)$, whereas GP with built-in GMRF have a computational complexity of $\mathcal{O}(nm^2)$. In most cases the number of generating points m will be chosen at the beginning of the process and then remains constant. On the other hand the number of observations n may rise with time. Thus, the computational complexity may grow fast for Gaussian Processes considering the factor n^3. In chapter 4 simulation results related to this behaviour are shown.

2.3.2 Expansion to Noise-free Observations

Taking a closer look at the GP with built-in GMRF, a discontinuity in the method for $\sigma_w \to 0$ can be found. In order to derive an algorithm for noise free observations, $\sigma_w = 0$ has to be inserted in equation 2.26. This yields

$$\begin{aligned} \mathrm{E}(z_0|\boldsymbol{y}) &= \mu(\boldsymbol{s}) + (\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\Lambda}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}(X)) \\ \mathrm{var}(z_0|\boldsymbol{y}) &= \boldsymbol{\lambda}^\top\boldsymbol{Q}^{-1}\boldsymbol{\lambda} - (\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\Lambda}^\top)^{-1}(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\lambda}). \end{aligned} \tag{2.29}$$

Using the Moore-Penrose pseudoinverse equation 2.29 transforms into

$$\begin{aligned} \mathrm{E}(z_0|\boldsymbol{y}) &= \mu(\boldsymbol{s}) + \boldsymbol{\lambda}^\top(\boldsymbol{\Lambda}^\top\boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}^\top(\boldsymbol{y} - \boldsymbol{\mu}(X)) \\ \mathrm{var}(z_0|\boldsymbol{y}) &= 0. \end{aligned} \tag{2.30}$$

Proof. Consider the part $(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\Lambda}^\top)^{-1}$. A transformation using the Moore-Penrose pseudoinverse leads to

$$(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\lambda})^\top(\boldsymbol{\Lambda Q}^{-1}\boldsymbol{\Lambda}^\top)^\dagger = \boldsymbol{\lambda}^\top\boldsymbol{Q}^{-\top}\boldsymbol{\Lambda}^\top\boldsymbol{\Lambda}^{\top\dagger}\boldsymbol{Q}^{-1^\dagger}\boldsymbol{\Lambda}^\dagger.$$

For an invertible matrix the Moore-Penrose pseudoinverse is the same as the conventional inverse, thus $\boldsymbol{Q}^{-1^\dagger} = \boldsymbol{Q}^{-1^{-1}} = \boldsymbol{Q}$. To evaluate $\boldsymbol{\Lambda}^\dagger$ and $\boldsymbol{\Lambda}^{\top\dagger}$ a singular value decomposition is used

$$\begin{aligned} \boldsymbol{\Lambda} &= \boldsymbol{U\Sigma V}^{-1} \\ \boldsymbol{\Lambda}^\top &= \boldsymbol{V}^{-\top}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \\ \boldsymbol{\Lambda}^\dagger &= \boldsymbol{V\Sigma}^\dagger\boldsymbol{U}^{-1} \\ \boldsymbol{\Lambda}^{\top\dagger} &= \boldsymbol{U}^{-\top}\boldsymbol{\Sigma}^{\top\dagger}\boldsymbol{V}^\top. \end{aligned}$$

Hereby $\boldsymbol{U}$ is a $n \times n$ unitary matrix, $\boldsymbol{\Sigma}$ is a $n \times m$ rectangular diagonal matrix with non-negative real numbers on the diagonal

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & & 0 \\ 0 & \cdots & \sigma_{mm} \\ & 0 & \end{bmatrix}$$

(assumption w.l.o.g.: $m < n$) and $\boldsymbol{V}$ is a $m \times m$ unitary matrix. Based on the fact that all columns of $\boldsymbol{\Lambda}$ are linear independent, all singular values are non-zero $\sigma_{ii} \neq 0$ for $i = 1, \dots, m$. The Moore-Penrose pseudoinverse of $\boldsymbol{\Sigma}$ is

$$\boldsymbol{\Sigma}^\dagger = \begin{bmatrix} \frac{1}{\sigma_{11}} & & 0 & \\ 0 & \dots & \frac{1}{\sigma_{mm}} & 0 \end{bmatrix}.$$

It can be shown that $\boldsymbol{\Sigma}^\top \boldsymbol{\Sigma}^{\top\dagger} = \boldsymbol{I}$, where $\boldsymbol{I}$ is a $m \times m$ identity matrix.
Inserting the singular value decomposition yields

$$\begin{aligned}
&\boldsymbol{\lambda}^\top \boldsymbol{Q}^{-1} \boldsymbol{\Lambda}^\top \boldsymbol{\Lambda}^{\top\dagger} \boldsymbol{Q} \boldsymbol{\Lambda}^\dagger \\
=&\boldsymbol{\lambda}^\top \boldsymbol{Q}^{-1} \boldsymbol{V}^{-\top} \boldsymbol{\Sigma}^\top \boldsymbol{U}^\top \boldsymbol{U}^{-\top} \boldsymbol{\Sigma}^{\top\dagger} \boldsymbol{V}^\top \boldsymbol{Q} \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \\
=&\boldsymbol{\lambda}^\top \boldsymbol{Q}^{-1} \boldsymbol{V}^{-\top} \boldsymbol{\Sigma}^\top \boldsymbol{\Sigma}^{\top\dagger} \boldsymbol{V}^\top \boldsymbol{Q} \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \\
=&\boldsymbol{\lambda}^\top \boldsymbol{Q}^{-1} \boldsymbol{V}^{-\top} \boldsymbol{V}^\top \boldsymbol{Q} \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \\
=&\boldsymbol{\lambda}^\top \boldsymbol{Q}^{-1} \boldsymbol{Q} \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \\
=&\boldsymbol{\lambda}^\top \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1}
\end{aligned}$$

With

$$\boldsymbol{\Lambda}^\top \boldsymbol{\Lambda}^{\top\dagger} = \boldsymbol{I}$$

reducing to a $m \times m$ identity matrix, the term $\boldsymbol{\Lambda}^{\top\dagger} \boldsymbol{\Lambda}^\top$ reduces to

$$\boldsymbol{\Lambda}^{\top\dagger} \boldsymbol{\Lambda}^\top = \boldsymbol{U}^{-\top} \boldsymbol{\Sigma}^{\top\dagger} \boldsymbol{V}^\top \boldsymbol{V}^{-\top} \boldsymbol{\Sigma}^\top \boldsymbol{U}^\top = \boldsymbol{I}_* = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $\boldsymbol{I}_*$ is a $n \times n$ matrix with an $m \times m$ identity matrix at the upper left and zeros everywhere else. Multiplying this matrix to $\boldsymbol{\lambda}^\top \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1}$ does not change the outcome since

$$\begin{aligned}
\boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} &= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \\
\boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \boldsymbol{I}_* &= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}.
\end{aligned}$$

Thus, $\boldsymbol{\Lambda}^{\top\dagger} \boldsymbol{\Lambda}^\top$ can be multiplied to the equation which yields

$$\begin{aligned}
&\boldsymbol{\lambda}^\top \boldsymbol{V} \boldsymbol{\Sigma}^\dagger \boldsymbol{U}^{-1} \\
=&\boldsymbol{\lambda}^\top \boldsymbol{\Lambda}^\dagger \\
=&\boldsymbol{\lambda}^\top \boldsymbol{\Lambda}^\dagger \boldsymbol{\Lambda}^{\top\dagger} \boldsymbol{\Lambda}^\top \\
=&\boldsymbol{\lambda}^\top (\boldsymbol{\Lambda}^\top \boldsymbol{\Lambda})^\dagger \boldsymbol{\Lambda}^\top \\
=&\boldsymbol{\lambda}^\top (\boldsymbol{\Lambda}^\top \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^\top
\end{aligned}$$

□

Two properties of equation (2.30) are further investigated. First, there is the zero variance $\mathrm{var}(z_0|\boldsymbol{y}) = 0$, which averts path finding. Here

$$\mathrm{var}(z_0|\boldsymbol{y}) = \boldsymbol{\lambda}^T(\boldsymbol{\Lambda}^T\boldsymbol{\Lambda})^{-1}\boldsymbol{\lambda} \tag{2.31}$$

is proposed, which can be derived if equation (2.27) is multiplied by $\frac{1}{\sigma_w}$. Afterwards the limit can be built setting $\sigma_w \to 0$.
The second property of the equation is that the precision matrix is no longer required. Thus, here an algorithm without precision matrix for the case of noisy measurements is proposed. Therefore, in equation (2.28) only $\boldsymbol{Q} = \boldsymbol{I}$ has to be set. A further analysis considering this proposal can be found in section 4.7.

If parameter adjustment is required for an estimation with noise-free observations,

$$\boldsymbol{C} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + c_0\boldsymbol{I} \quad c_0 \geq 0 \tag{2.32}$$

can be used. The precision matrix is set to be the identity matrix and the additional term $c_0\boldsymbol{I}$ ensures the positive definiteness of the covariance matrix.

2.3.3 Parameter Adjustment

In section 2.1 a method for the adjustment of the hyperparameters from the kernel function has been shown. The same method can be used to adjust the parameters required for the weighting function $\lambda(\cdot,\cdot)$. In addition, the positions of the generating points can be optimized this way. First, consider again the log likelihood and its derivation from equation (2.33) and (2.34). Adapted to the GP with built-in GMRF, assuming $\boldsymbol{\mu}(\cdot) = 0$, equation (2.33) becomes

$$\log(f(\boldsymbol{y}|X,\boldsymbol{\theta})) = -\frac{1}{2}\boldsymbol{y}^\top\boldsymbol{C}^{-1}\boldsymbol{y} - \frac{1}{2}\log||\boldsymbol{C}|| - \frac{n}{2}\log(2\pi) \tag{2.33}$$

and equation 2.34

$$\frac{\partial\log(f(\boldsymbol{y}|X,\boldsymbol{\theta}))}{\partial\theta_j} = \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \boldsymbol{C}^{-1})\frac{\partial\boldsymbol{C}}{\partial\theta_j}\right). \tag{2.34}$$

Again $\boldsymbol{\alpha} = \boldsymbol{C}^{-1}\boldsymbol{y}$ and n the number of data points in the training set. The covariance matrix $\boldsymbol{C}$ is $\boldsymbol{C} = \boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{\Lambda}^\top + \sigma_w^2\boldsymbol{I}$. The parameter vector $\boldsymbol{\theta}$ can contain both the hyperparameters for the weighting function and the positions of the generating points. The log likelihood has to be maximized to find optimal hyperparameters and positions for the generating points.

2.4 Variogram

In this section the computation of the experimental semi-variogram is shown. The variogram is used in spatial statistics to describe the degree of spatial dependence of a spatial field. Thus, in case of an estimation problem, it can give information about which dimensions have a great influence on the estimation process. Such information can give an idea how to set the hyperparameters of a certain covariance function and which type of covariance function should be chosen. Simulation results considering this topic can be found in section 4.5.

For simplicity only the two dimensional field is considered. Assume a vector $\boldsymbol{h}$ which has a length $||\boldsymbol{h}|| = l$ and a direction α (angle). Given n pairs of data separated by $\boldsymbol{h}$, the experimental semi-variogram can be calculated for the distance l and angle α by

$$\gamma(\boldsymbol{h}) = \gamma(l, \alpha) = \frac{1}{2n} \sum_{i=1}^{n} \left(y(\boldsymbol{x}_i + \boldsymbol{h}) - y(\boldsymbol{x}_i)\right)^2 , \tag{2.35}$$

where $y(\boldsymbol{x})$ is the value at position $\boldsymbol{x}$.
This is the expectation for the squared increment of the values between locations $\boldsymbol{x}$ and $\boldsymbol{z}$:

$$\gamma(\boldsymbol{x}, \boldsymbol{z}) = \mathrm{E}\left[(y(\boldsymbol{x}) - y(\boldsymbol{z}))^2\right] . \tag{2.36}$$

Thus, the lower the value of the variogram, the higher is the spatial dependence between points in $\boldsymbol{h}$ direction.

The methods used for the construction of the variogram depend highly on the structure of the given data. In this thesis, only aligned data within a regularly spaced grid are considered. Therefore, four main directions, α_1, α_2, α_3 and α_4, with four main distances, l_1, l_2, l_3 and l_4 are used (figure 2.6).

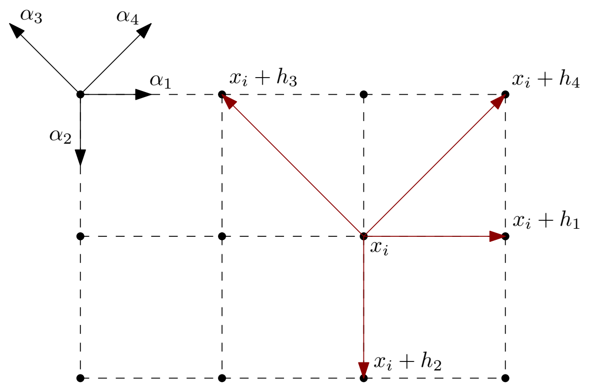


Figure 2.6: The figure shows how the data is arranged by calculating the experimental semi-variogram.

Chapter 3

Toolbox

In this chapter a Matlab toolbox, which has been programmed in this thesis to test and evaluate the methods presented in chapter 2 , is introduced. Section 3.1 explains the general structure in combination with a broad overview over the Matlab files. At the end of the section a recommendation on how to integrate the toolbox into Matlab is given. Section 3.2 contains a short introduction about the utilization of the toolbox. It is explained which data are required for the estimation process and which options are available. Section 3.3 shows how to add customized covariance or precision functions.

3.1 Folder Structure

In figure 3.1 the folder structure of the toolbox is presented. The toolbox has one main folder called "Toolbox" where the main functions can be found. These main functions can be called by the user, such as "GMRF_Spatial.m". The sub-functions, required for the estimation process, can be found in different sub-folders ("covariance_functions", "subfunctions", ...). These sub-functions should not be called directly, but it is possible to add additional sub-functions, e.g. covariance or precision functions. More information regarding this topic can be found in section 3.3.
It is possible to add the path for the toolbox permanently to Matlab, so that the main functions can be called whatever directory is currently chosen. Therefore, the path of the toolbox can be added in a "startup.m"-file. First, a "startup.m"-file has to be created and stored in a folder on the Matlab search path. It is recommended to add the "startup.m"-file in the *userpath* folder. These folder can be found by typing "pwd" into the Matlab Command Window. After creating the file, the command "addpath *pathOfTheToolbox*" can be added. The toolbox is ready for utilization.

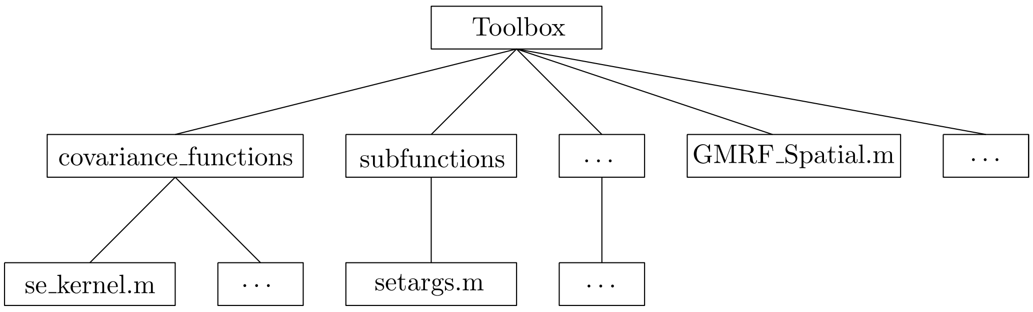


Figure 3.1: Folder structure of the toolbox for the Gaussian Processes with built-in Gaussian Markov Random Field.

If a non permanent way of adding the toolbox is preferred, the “addpath”-command can be added at the beginning of a Matlab script.

3.2 Utilization of the Toolbox

In order to understand the utilization of the toolbox and all its possibilities for estimation, a closer look at the main function “GMRF_Spatial.m” is required. The main framework of the function is

```
[mu, Sigma , time ] = GMRF_Spatial(p ,x ,s ,y)
```

where p, x and s are matrices with the dimension $D \times m$, $D \times n$ and $D \times l$, where m is the number of generating points, n the number of observation points, l the number of estimation points and D the dimension of the states. The last input y is a vector which consists of the measurements at the states given in x and is a vector with dimension $n \times 1$. The outputs of the function are the mean values for the estimation (mu) at the states given in s and the variance (Sigma). Furthermore, the computation time (time) is given as an output.
In order to adjust the estimation scheme to a given problem, various options can be used. Therefore, a more general framework of the function “GMRF_Spatial.m” is given:

```
[mu, Sigma , time , opt_time , opt_p , opt_covParam ] =
  GMRF_Spatial(p ,x ,s ,y , 'propertyname ' , 'propertyvalue ' ,...)
```

In order to use an additional option, the name of the property has to be written as a string. After this, the property value has to be added. For example, if a different noise value as the default value ($\sigma_w = 0.1$) is required, for instance $\sigma_w = 1$, the estimation function has to be called as

```
[mu, Sigma , time ] = GMRF_Spatial(p ,x ,s ,y , 'noise' ,1)
```

In order to see all possible options have a look into the "GMRF_Spatial.m"-file. All properties are listed and explained at the beginning of the function.

Additional outputs can be chosen. Some of them only make sense together with certain properties. For instance, the optimization time (opt_time) is only non zero if an optimization occurs.

Two more main functions can be found in the toolbox folder: "GP.m" which does estimation using Gaussian Process Regression and "GMRF_Sequential.m" which uses a sequential algorithm for a time depending estimation process. Both functions can be handled in the same way as "GMRF_Spatial.m".

3.3 Adding Covariance or Precision Functions

In this section it is explained how to add a custom covariance or precision function to the toolbox. Therefore, both folders ("covariance_functions", "precision_functions") contain example files ("cov_example.m", "pre_example.m"), which can be used to generate custom covariance or precision functions. In these files the places where to add code are marked with three dots. In the following, the example file for the "cov_example.m" is shown.

```
function [K, varargout] = cov_example(X1,X2, varargin)
% ...          % explain your covariance function
%
% equation:
%   k(x,x') = ...
%
% syntax:
%   [K] = se_kernel(X1,X2)
%   [K, optionalOutputs] = se_kernel(X1,X2,
%                               'propertyname', 'propertyvalue',...)
%
% Description:
%   Generates a covariance matrix between
%       the points given in X1 and X2.
%
% Input:
%   X1, X2: Matrices of data points with dimension D x n, D x m
%
% Propertyname/-value pairs:
%   CovParam - array of the covariance function parameters
%         ... x 1 array required:
%         (1): ...
%         (2): ...
%    struct - gives out the hyperparameters, additional
%         output argument required (true or false, default: false)
%         (this is required for the optimization scheme)
%
% Output:
%    K - covariance matrix (n x m)
%
% Date: ...
% Author: ...

% Default values
defaultargs = {'CovParam', [...], 'struct',0};
% add your default values
params = setargs(defaultargs, varargin);

% look if hyperparameters are wanted
if params.struct == true
  varargout{1} = params.CovParam;
end

% error checking
if iscell(params.CovParam)
  error('covariance␣parameters␣are␣in␣a␣cell␣array')
end
if size(params.CovParam) ~= [... 1]
% add the number of hyp.-param.
  error('covariance␣parameters␣have␣not␣the␣right␣size')
end

% Rewriting parameters in variables
...       % here you can rename your hyperparameter if required

% computing the dimension of the kernel matrix
l1 = length(X1(1,:));
l2 = length(X2(1,:));

% providing the kernel matrix for efficient computing
K = zeros(l1,l2);

% computing the kernel for each element of the covariance matrix
```

```
for i=1:1:l1
  for j=1:1:l2
      K(i,j) = ... % add you formula k(x,x') = k(X1(:,i),X2(:,j))
  end
end
 end
```

As shown, only a few things have to be added. Most important is the formula at the end, used for the calculation of the covariance matrix.

Chapter 4

Simulation Results

This chapter deals with the analysis of the estimation methods presented in chapter 2. First, two data sets are introduced in section 4.1. These data sets contain data of two dimensional fields which are used for the estimation process. In the first part, GPs and GPs with built-in GMRF are compared. The aim is to identify a preferred method for further research. In the following, optimization methods are tested (section 4.3) and time variant estimations analysed (section 4.4) considering the preferred method. Furthermore, a variogram is developed to test impacts of non-stationary fields on the estimation results (section 4.5). At the end the effect of different mean functions (section 4.6) and an identity matrix as precision matrix (section 4.7) are analysed.

In order to compare the estimation results presented in this chapter, the mean squared error

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(\hat{y}_i - y_i)^2 \tag{4.1}$$

is used, where k is the number of estimation points, $\hat{y}_i$ the estimation value at point i and y_i the data value at point i.

Whenever a covariance function or weighting function is required and nothing else is stated, the squared-exponential covariance function

$$k_{se}(\tau) = \sigma_f^2 \cdot e^{-\frac{\tau^2}{2l^2}}, \tag{4.2}$$

with $\tau = \boldsymbol{x} - \boldsymbol{x}'$, is used. The hyperparameters are set to $\sigma_f = 3$ and $l = 3$. As function for the generation of the precision matrix, which is required for GPs with built-in GMRF, the function

$$Q_{ij} = \left\{\begin{matrix} ||N(i)|| + c_0 & \text{if } j = i \\ -1 & \text{if } j \text{ element of } N(i) \\ 0 & \text{otherwise} \end{matrix}\right\} \tag{4.3}$$

is used, if nothing else is stated. The neighbouring system for the precision function is $N(i) = \{(i,j)| \; ||p_i - p_j|| \leq r = 3\}$ and $c_0 \geq 0$ is a constant to ensure positive definiteness of the precision matrix $\boldsymbol{Q}$. In order to compare the presented methods as presented in chapter 2, noisy observations with $\sigma_n = 0.1$ are assumed. The mean function is set to $\mu(\cdot) = 0$ for all simulations except in section 4.6.

The simulations are carried out on an Acer Aspire VN7-572G with Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz 2.60 GHz and 8.00 GB RAM with Windows 10 as the 64 bit OS and Matlab R2016a.

4.1 Data Sets

In this section two data sets are introduced. The first data set has been designed using a 2D advection-diffusion equation. The second data set contains real data from the California Coastal Regional Ocean Modelling System. Both data sets are used for the analysis of the methods presented in chapter 2.

4.1.1 Artificial Data

In order to create an artificial field for the first trials, an 2D advection-diffusion equation has been used. A Matlab algorithm computed the solution of this equation with a domain length of 10, using a spatial resolution of 0.01. Thus, 1001 x 1001 spatial data points are available. Furthermore, 70 time steps have been generated such that the complete dataset has a size of 1001 x 1001 x 70 data points. In figure 4.1 the artificial field at a certain time step is shown.

4.1.2 Real Data

A real data set, taken from the California Coastal Regional Ocean Modelling System (ROMS) (`http://www.ngdc.noaa.gov/docucomp/page?xml=NOAA/IOOS/CeNCOOS/iso/xml/CA_FCST.iso.xml&view=getDataView&header=none`) is used to test the introduced methods. In order to get the data the programme OPeNDAP has been used (`http://thredds.axiomdatascience.com/thredds/dodsC/CA_FCST.nc`). The received data starts from the west coast of the USA and extends from just north of the CA-Oregon border south to Mexico. It contains information about the salinity, the temperature and the current in a region of 31.3 up to 43 degrees north latitude and 232.5 up to 243 degrees east longitude with a resolution of 0.03 degrees. The time steps of the data set are from 379,107 up to 379,206 hours counting from the 01.01.1970, 00:00:00, thus 100 time steps

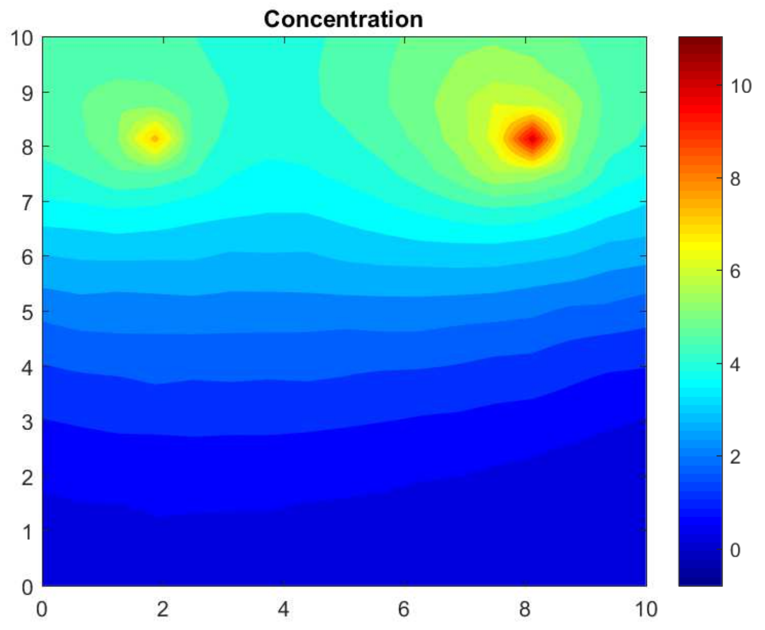


Figure 4.1: The figure above shows an artificial field created using the solution of the 2D advection-diffusion equation. The resolution of the field is 0.01.

are available. There are 13,7241 spatial data points and in total, adding the temporal dimension, 1,3724,100 data points. In this thesis only the temperature data is used. In figure 4.2 the temperature field at a certain time step is shown to illustrate the observed region.

4.2 GP vs GP with built-in GMRF

In this section Gaussian Processes and Gaussian Processes with built-in Gaussian Markov Random Field are compared. Therefore GPs and GPs with built-in GMRF are used to estimate the virtual data field. As evaluation parameters the computational time as well as the MSE are used. The computational time which is measured contains only the time required to execute the algorithms given by the equations ((2.9), (2.27) - (2.28)) introduced in chapter 2. The time which is required to create the covariance or precision matrices are not taken into account.

Both, computational time and mean squared error in relation to the number of observation points n are shown in figure 4.3. In order to generate the estimations, generating points, observations points and estimation points have been ordered in grids. For the estimation process 25 generating points and 900 estimation points

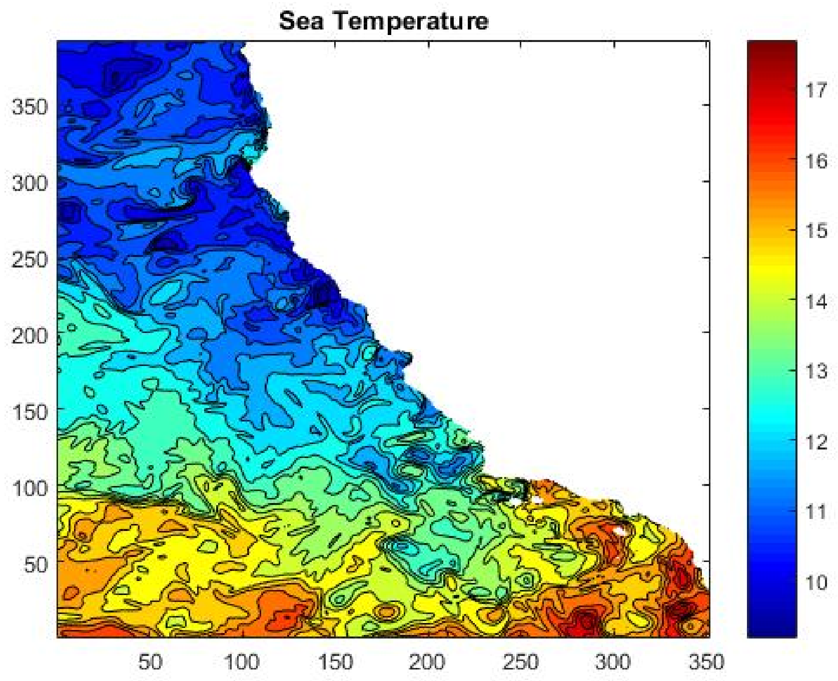


Figure 4.2: Temperature field created using data from the California Coastal Regional Ocean Modeling System (ROMS) model. The data extends from just north of the CA-Oregon border south to Mexico and has a resolution of 0.03 degrees.

have been used. The number of observation points used for the analysis is $n = o^2$ with the steps $o = \{1, 2, \ldots, 50\}$.
The computational time of the GP estimation is increasing much faster as the computational time of the estimation using GP with built-in GMRF. Such a behaviour was expected, since in section 2.3, the computational complexity for GPs has been given as $\mathcal{O}(n^3)$ and for GPs with built-in GMRF as $\mathcal{O}(nm^2)$. In figure 4.4 a smoothing function is shown which illustrates the order of the slope for the GP. The test results in 4.3 show a trade-off between estimation accuracy and computational time. Based on the computational resources a decision has to be taken if GPs or GPs with built-in GMRF should be used.
In this thesis an estimation procedure for autonomous underwater vehicle is explored. In such systems computational power is a limiting resource. Thus, in the next sections GPs with built-in GMRF are further investigated. Therefore, optimization techniques and different ideas to improve the estimation procedure are tested.

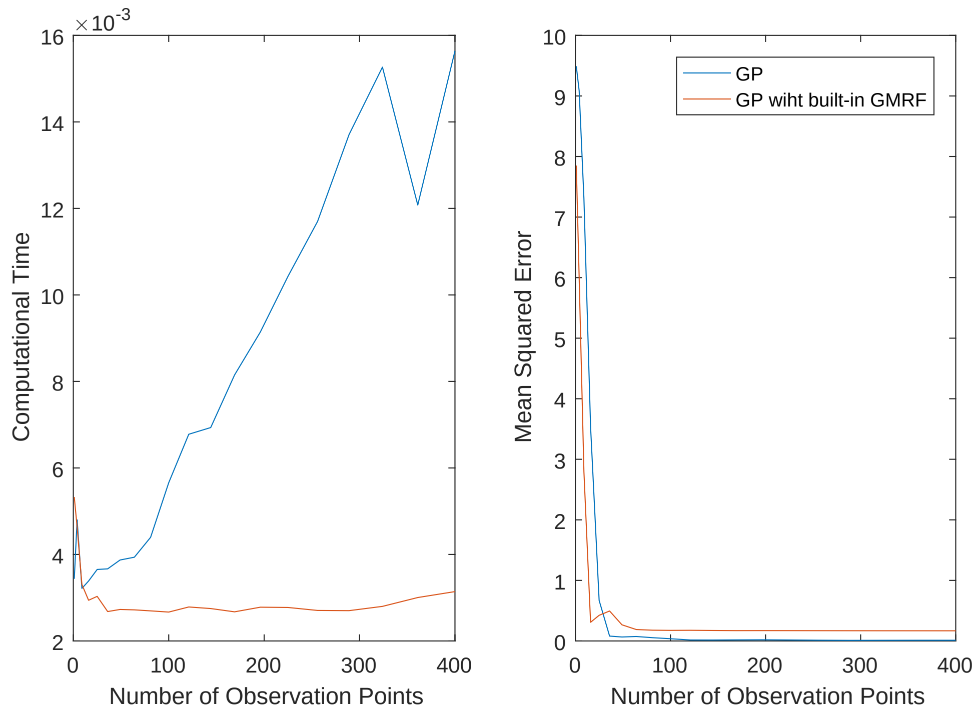


Figure 4.3: Computational time required for the estimation process (left) and mean squared error (right). Both are calculated by estimating the virtual data field.

4.3 Optimization

In order to adjust the parameter of the GP with built-in GMRF remember the log likelihood equations (2.33) and (2.34) with $\boldsymbol{\mu}(\cdot) = \mathbf{0}$. The log likelihood is multiplied by -1 which turns the maximum search into a minimum search. The fminunc function from Matlab, which is based on the interior-reflective Newton method, is used to find a minimum.

4.3.1 Hyperparameters

In order to test the parameter optimization, the squared-exponential covariance function is used. Remember

$$k_{se}(\tau) = \sigma_f^2 \cdot e^{-\frac{\tau^2}{2l^2}} \tag{4.4}$$

with $\tau = \boldsymbol{x} - \boldsymbol{x}'$ and the hyperparameters $\theta_1 = \sigma_f$ and $\theta_2 = l$, which have to be optimized. In figure 4.5 the estimation results using hyperparameter improvement are shown. The top images show the original temperature field and the

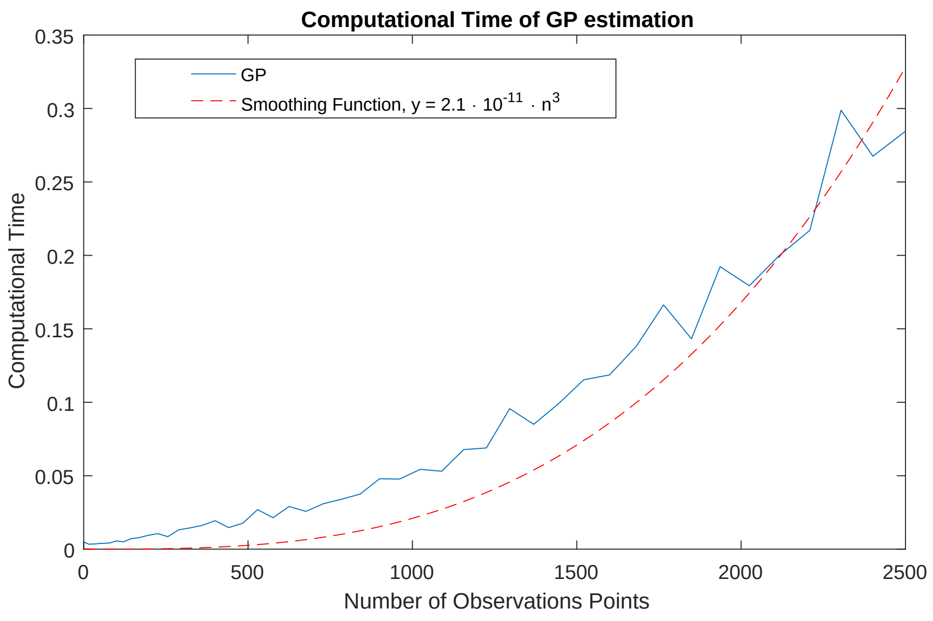


Figure 4.4: The figure shows the increasing computational time for a GP estimation in relation to the number of observation points. A smoothing function has been designed to show the order of the slope.

original concentration field. The black stars illustrate the observation points (30, randomly placed) and the red stars the generating points (36, placed in a grid). The middle images are depicting the estimation results for the fields without hyperparameter improvement. Above the images, the mean squared error is shown. The mean squared error for the temperature field is $MSE = 1.098$ and for the concentration field $MSE = 0.287$. The bottom images represent the estimation results with hyperparameter improvement. The mean squared error can be found above of the images. In comparison to the MSE without parameter optimization the MSE is reduced. The optimization time has been around $t_{opt} = 0.2s$. Hyperparameter optimization can be a good method to improve the estimation result for the GP with built-in GMRF but the optimum search can be complicated.
For the optimum search the choice of the initial parameters is crucial, because the optimization method tends to get stuck in local optima. There is no guarantee, that the estimation result using the likelihood optimization leads to more accurate results than without it, because the optimization procedure can only consider the known observation points to maximize the likelihood. Nevertheless, if no valid information about the hyperparameters is available, the log likelihood optimization can be used to find convenient ones.

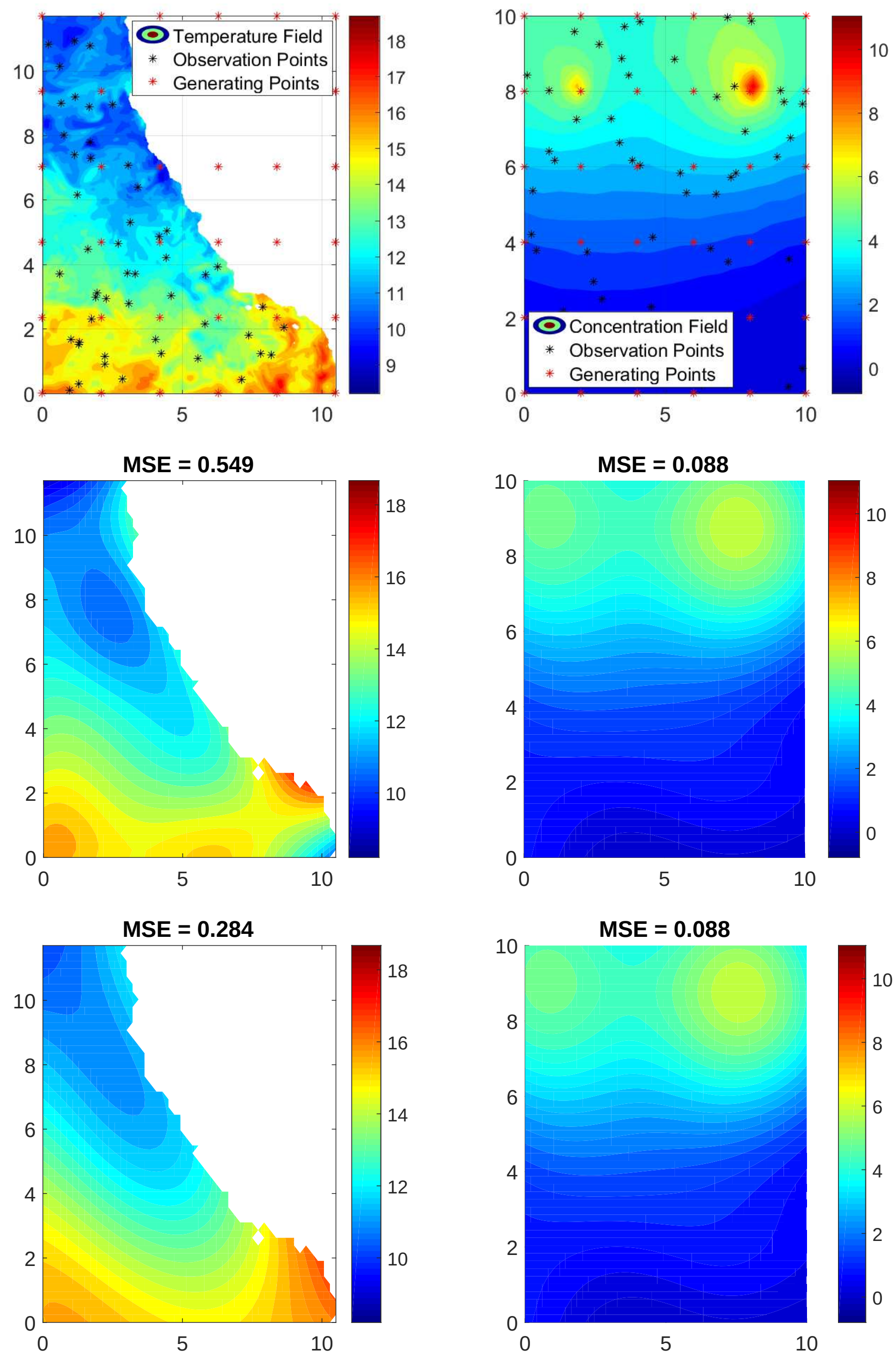


Figure 4.5: The 36 generating points are placed in a grid structure and the 30 observations points are random placed. The top images are showing the original fields. The middle images the estimated fields without optimization of the hyperparameters and the bottom images show the estimated fields with optimization of the hyperparameters.

4.3.2 Generating Points: Positions

In section 4.3.1 the hyperparameters of the covariance function have been optimized. Therefore, the generating points have been ordered in a grid structure. This is more or less an intuitive decision since it leads to equally distributed generating point, but it is not necessarily the best way to distribute generating points over a defined space for a certain estimation problem.
The positions of the generating points, thus the coordinates of these points, can be handled as additional hyperparameters and can be improved in the same way as the hyperparameters in section 4.3.1. A major difference is the number of parameters which have to be optimized. In case of two dimensional states, the number of additional parameters for the optimization is two times the number of generating points. Thus, the computational effort is much higher and the structure of the likelihood function becomes more complex compared to the optimization of only two hyperparameters. In figure 4.6 the estimation results for the temperature field using optimized generating points is shown. In order to compare the estimation results, an estimation using the initial generating points from the optimization is shown in the middle. In the left pictures, the initial positions of the generating points (36) as well as the observation points (30) have been chosen randomly and in the right pictures the generating points have been ordered in a grid structure. The estimation results are getting better using the optimization of the generating points. Thus, generating points optimization can be used for the improvement of the estimation accuracy but it comes with a huge amount of additional computational effort. For instance, the computational time required to improve the given 36 generating points has been around 500 seconds. As mentioned in 4.3.1 it is not guaranteed that an optimization of the estimation accuracy really occurs considering the badly shaped optimization function.

4.3.3 Generating Points: Number

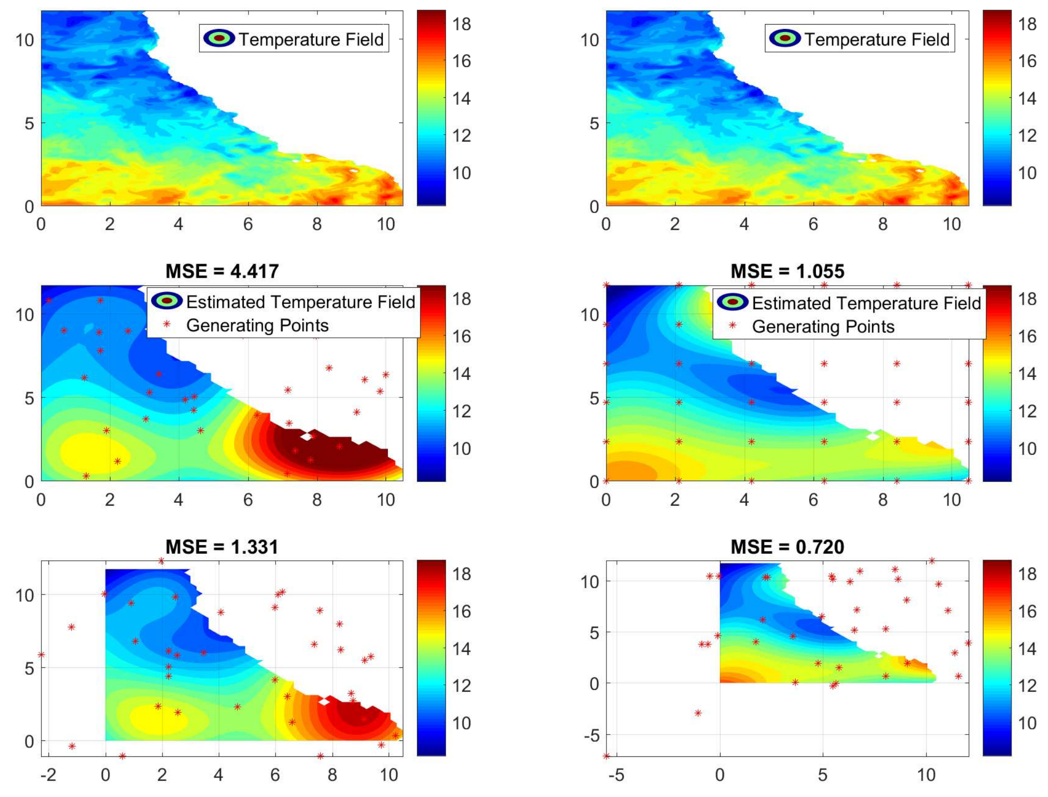
In the previous sections always 36 generating points have been used for the estimation. The number has been found by trial and error. It seems to be a good trade-off between estimation accuracy and computational effort. In this section a more analytical way of finding a proper trade-off between estimation accuracy and computational effort is shown.

First, consider the Akaike Information Criterion

$$AIC = 2k - \log(\mathcal{L}), \tag{4.5}$$

where k is the number of parameters used for an estimation model and $\mathcal{L}$ the maximum value of the likelihood function. The preferred model is the model with the lowest AIC-value. The trade-off between computational effort (model complexity) and model accuracy is considered by the fact that an accurate model leads to a high likelihood $\mathcal{L}$ (low AIC), whereas a complex model will have a high number of parameters k (high AIC).

In order to test the model selection criterion, the generating points have to be somehow distributed over the field of estimation (assuming a rectangular field). Therefore, a distribution is used which tries to generate an equally spaced grid in both, x- and y-direction. The overall idea is to let the distance between the



(a) Random placement (b) Grid placement

Figure 4.6: The optimization of the generating points. Left, randomly placed generating points as initial points have been chosen and right, the initial generating points have been ordered in a grid structure. The bottom pictures are showing the optimized estimation, the mid pictures the unoptimized estimation and the top pictures the original temperature field.

points in x- as well as in y-direction be the same:

$$d_x = \frac{r_x}{k_x} = \frac{r_y}{k_y} = d_y \quad \Rightarrow \quad k_x = \sqrt{\frac{r_x}{r_y}k}; \quad k_y = \sqrt{\frac{r_y}{r_x}k}. \tag{4.6}$$

There k_x and k_y are the number of generating points in x- and in y-direction and $k = k_x k_y$. If k_x and k_y are rounded to integers not all grid distributions are possible. This is a drawback which can not be avoided if using a simple distribution algorithm. Simulations in a range of $k_{min} = 5$ up to $k_{max} = 1000$ have been done. 52 out of 100 times a number of 288 generating points has been the optimum. Other solution have been 110, 182, 195 and 210. Every simulation has been done with a different set of thirty observations points, which have been chosen randomly. In figure 4.7 an estimation result using the in average most optimal number of 288 generating points is presented. The estimation accuracy is higher compared with the estimation using 36 generating points, which has been expected. Nevertheless, considering computational time constraints the number of 36 generating points has been chosen for most of the simulations in this thesis. A weight can be added to equation (4.5), such that computational time can be made more costly or cheaper. Equation 4.5 becomes

$$AIC = ak - log(\mathcal{L}), \tag{4.7}$$

with $a \in \mathbb{R}^+$ as the weighting factor. If a is large, the computational time is costly. Otherwise, if a is small, the computational time is cheap.

4.4 Time Variant Systems

In the previous sections only spatial, time invariant fields have been explored. In this section an analysis about time variant fields is done. Therefore, two different approaches are analysed.

The first approach treats the time variance as an additional dimension, such that a state vector for a two dimensional field can be written as

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ t \end{bmatrix}. \tag{4.8}$$

Such an approach allows to use the GP with built-in GMRF as usual. An additional requirement is the change of the covariance function. The new one should take the different scales and dependencies in the three dimensions into account. Such a covariance function is given by

$$k_{se_{II}}(\tau) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}\tau^\top \Delta \tau\right), \tag{4.9}$$

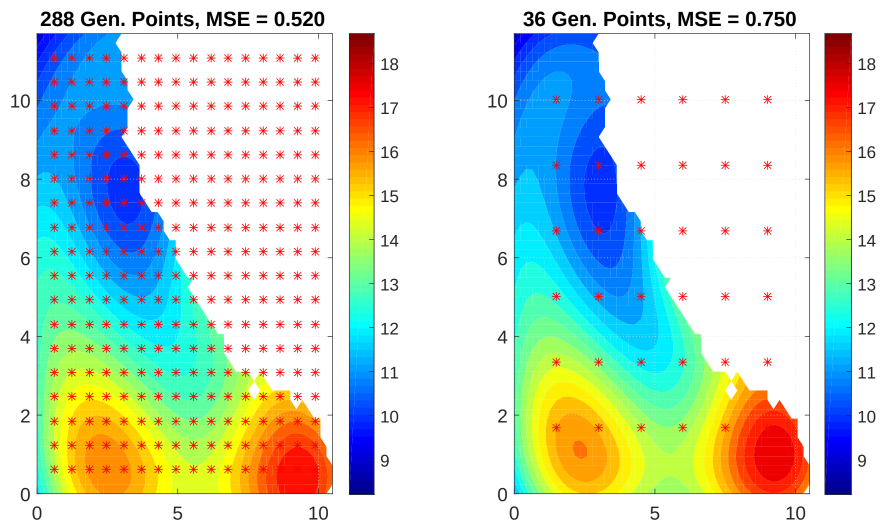


Figure 4.7: The left figure shows an estimation with the in average at most optimal number of generating points ordered in a grid. The right figure shows for comparison an estimation using 36 generating points as used throughout this paper.

where $\tau = \boldsymbol{x} - \boldsymbol{x}'$ and $\Delta = \operatorname{diag}(\frac{1}{l_1^2}\ \frac{1}{l_2^2} \dots)$. This function is similar to the introduced squared exponential function changing only the parameter $\frac{1}{l}$ into a diagonal matrix Δ. Such a covariance function can also be used to improve the estimation accuracy for non-stationary fields (section 4.5).

The second approach leads to a sequential algorithm ([Xu et al., 2015]) where the intermediate states $\hat{\boldsymbol{Q}}$ and $\hat{\boldsymbol{y}}$ have to be updated in every time step:

$$\begin{aligned} \hat{\boldsymbol{Q}}_t &= \hat{\boldsymbol{Q}}_{t-1} + \sigma_w^{-2} \boldsymbol{\Lambda}_t^\top \boldsymbol{\Lambda}_t \quad , \quad \hat{\boldsymbol{Q}}_0 = \boldsymbol{Q} \\ \hat{\boldsymbol{y}}_t &= \hat{\boldsymbol{y}}_{t-1} + \sigma_w^{-2} \boldsymbol{\Lambda}_t^\top (\boldsymbol{y}_t - \boldsymbol{\mu}_t) \quad , \quad \hat{\boldsymbol{y}}_0 = 0. \end{aligned} \tag{4.10}$$

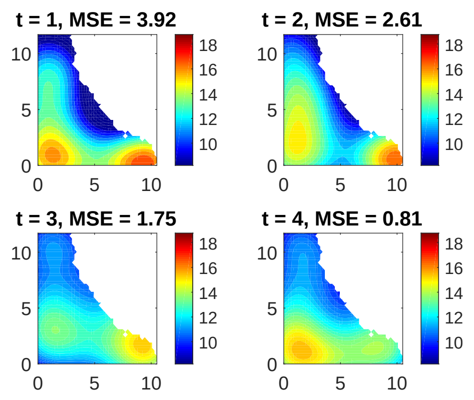
In this approach the state vector remains two dimensional with

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.11}$$

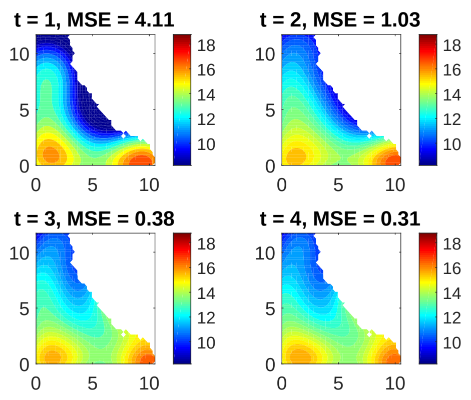
In the case of observations without noise, as discussed in section 2.3.2, the updating rule reduces to

$$\begin{aligned} \hat{\boldsymbol{Q}}_{\mathrm{wn},t} &= \boldsymbol{\Lambda}_{\mathrm{wn},t-1}^\top \boldsymbol{\Lambda}_t \quad , \quad \hat{\boldsymbol{Q}}_{\mathrm{wn},0} = 0 \\ \hat{\boldsymbol{y}}_{\mathrm{wn},t} &= \hat{\boldsymbol{y}}_{\mathrm{wn},t-1} + \boldsymbol{\Lambda}_t^\top (\boldsymbol{y}_t - \boldsymbol{\mu}_t) \quad , \quad \hat{\boldsymbol{y}}_{\mathrm{wn},0} = 0. \end{aligned} \tag{4.12}$$

In figure 4.8 a comparison between both methods is shown. The hyperparameters chosen for the first approach are $\Delta = \operatorname{diag}(\frac{1}{9}, \frac{1}{9}, 1)$ and $\sigma_f = 3$ and for the



(a) Additional Dimension



(b) Sequential Algorithm

Figure 4.8: In the left figure a prediction with the time as an additional dimension is shown. A covariance function with $\Delta = \mathrm{diag}(\frac{1}{9}\ \frac{1}{9}\ 1)$ and $\sigma_f = 3$ has been used. In the right figure a prediction using the sequential algorithm is shown. A covariance function with $\sigma_f = 3$ and $l = 3$ has been used. In every time step 10 observations have been generated.

second approach the hyperparameters are set to $\sigma_f = 3$ and $l = 3$. The sequential algorithm improves its estimation accuracy in every time step whereas the other method first improves the estimation accuracy but then decreases it. Furthermore, the computational time required for executing the sequential algorithm remains constant for every time step, whereas the computational time required for executing the dimension approach increases with every additional time step. This behaviour comes from the additional generating points which are added in every time step. An endless increasing of the number of generating points can be avoided by setting a limit of previous time steps used for the current estimation. Considering the computational efficiency and the estimation accuracy the sequential algorithm is preferable for the implementation in AUVs. In figure 4.9 a long time estimation over 16 time steps using the sequential algorithm is shown. The mean squared error decreases with the estimation steps.

4.5 Variogram

In section 2.4 the equation to calculate a variogram has been introduced. In this section equation (2.35) is used to generate the variogram for the real data set. In figure 4.10 the results are shown. First consider figure (a). In direction α_1 the variogram shows a low value which indicates a high dependency between points in this direction. For comparison, the values in α_2-direction are increasing

rapidly. This shows that pairs of data in this direction get uncorrelated quickly. In (b) the variogram is plotted as a contour plot. Values in between the dashed lines have been interpolated.
These information about the spatial dependencies can be used to improve the

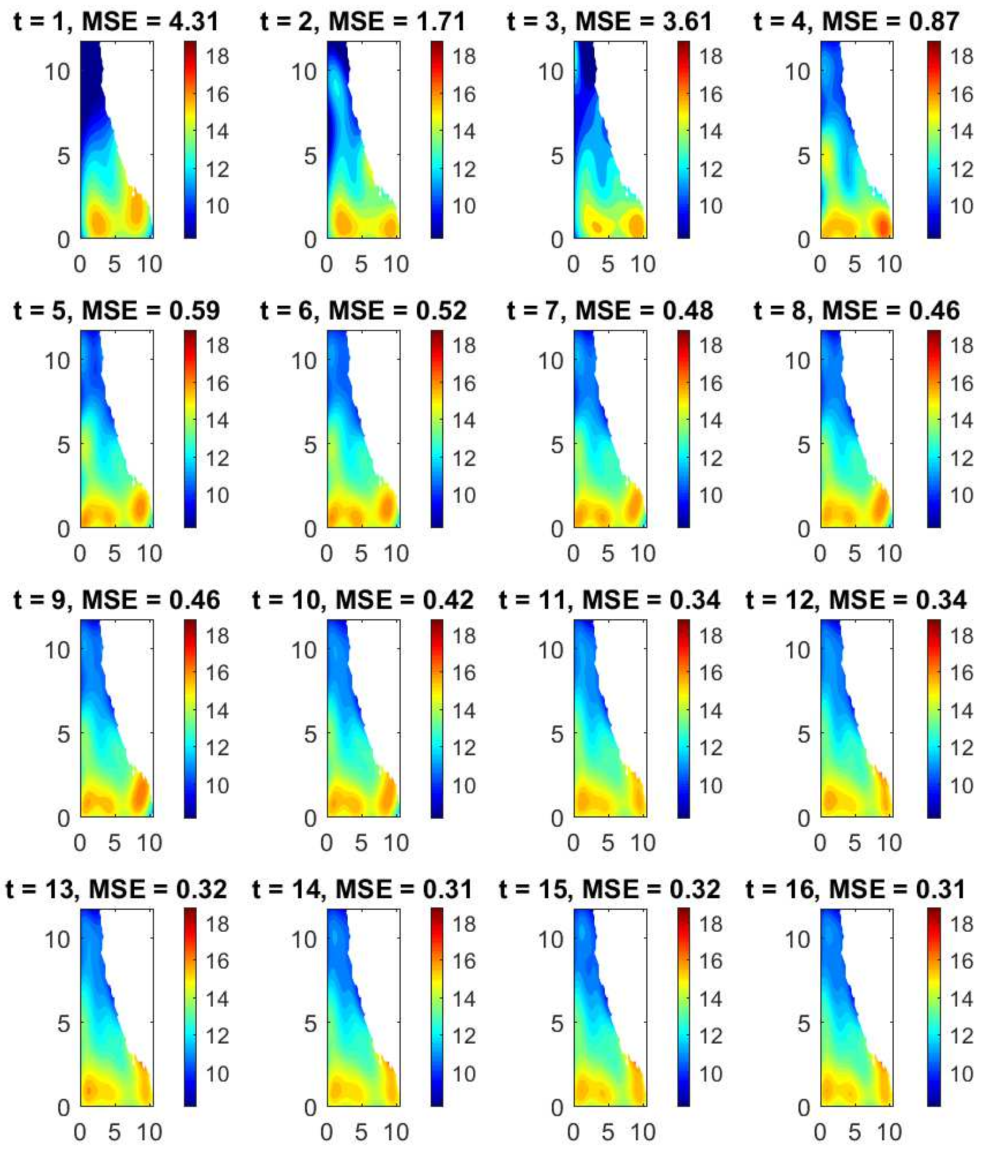
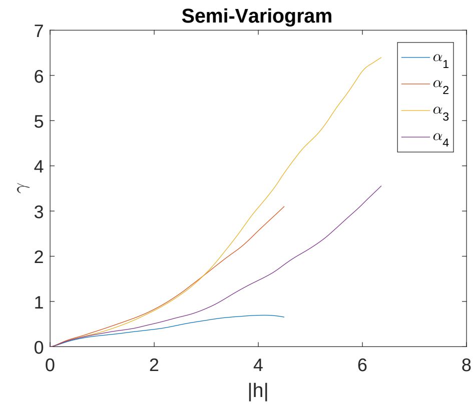
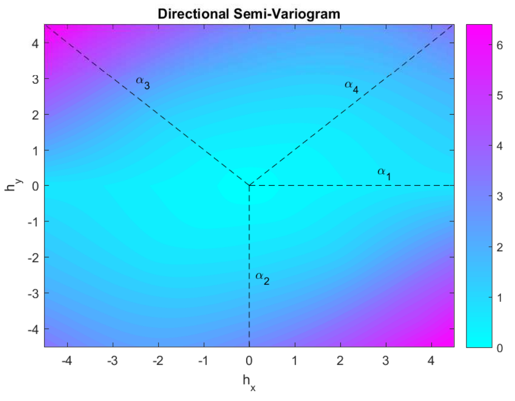


Figure 4.9: Estimations of the temperature field using the sequential algorithm over 16 time steps.



(a) One Dimensional Semi-Variogram



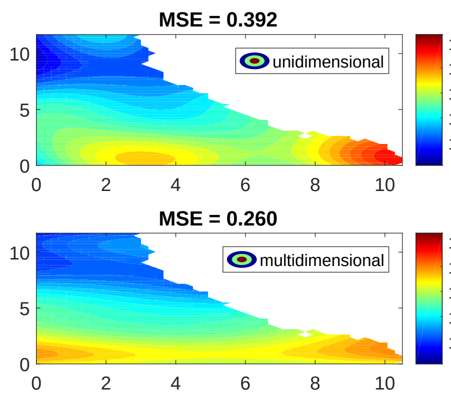
(b) Directional Semi-Variogram

Figure 4.10: (a): Variograms for the four different directions α_1,...,α_4. (b) Variogram values outgoing from the reference point (0,0). The dashed lines are showing the four main directions.

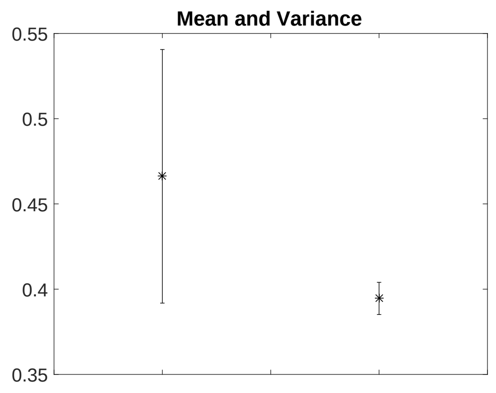
estimation result of the GP with built-in GMRF. Therefore, a different type of covariance function has to be used which takes different correlation behaviour for the different directions into account. In section 4.4 such a covariance function has been introduced (equation 4.9). In figure 4.11 the comparison between the squared exponential covariance function with unidimensional hyperparameters and multidimensional hyperparameters is presented. For both approaches the hyperparameter σ_f is set to $\sigma_f = 3$. Both covariance functions are leading to the same results if, in case of the unidimensional approach, l is set to $l = 3$ and, for the multidimensional approach, l_1 and l_2 are also set to $l_1 = l_2 = l = 3$. Considering the results from the variogram, the hyperparameters for the multidimensional approach have been set to $l_1 = 9$ and $l_2 = 2$. Thus, the correlation between points in α_1-direction is much higher as between points in α_2-direction. This leads to improved estimation results as shown in figure 4.11.

4.6 Noise-free Observations and Mean Function

So far, for every simulation noisy observations have been assumed with $\sigma_n = 0.1$. Now both, an estimation assuming noisy observations and an estimation assuming noise-free observations are done. In figure 4.12 both estimation results are shown. The algorithm for noise-free observations leads to good results. In fact since there is no noise in the given data set, an estimation assuming $\sigma_n = 0$ should lead to better results as an estimation assuming $\sigma_n = 0.1$. This results confirms the derivation which has been given in section 2.3.2.



(a) Estimation Example



(b) Statistical Evaluation

Figure 4.11: (a): The figure shows an estimation example for the same set up using an unidimensional and a multidimensional covariance function. (b):The figure shows the mean and the variance (scaled) between estimations using an unidimensional covariance function (left) and a multidimensional covariance function (right). The estimation has been done 50 times. 50 random observation points and 36 generating points (placed in a grid) have been used for every run.

In all previous simulations a zero mean function $\mu(\boldsymbol{x}) = 0$ has been used. Another possibility for estimation accuracy improvement is the choice of a convenient mean function. By using such a function, eventually chosen using pre knowledge of the given field, a better result can be obtained. In this thesis only the comparison between a zero mean function and a constant mean function $\mu(\boldsymbol{x}) = a$ is presented. The constant a is defined as the mean of the given observations $\boldsymbol{y}$:

$$a = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{4.13}$$

In figure 4.13 an estimation using a constant mean function is shown. The estimation result is slightly better as with a zero mean function. In addition a test run with 100 cycles has been done. Calculating the mean squared error leads to a mean value of 2.17 for the estimation with the zero mean function and a mean value of 1.90 for the estimation with constant mean function. The variances are nearly the same with 230 and 233. Thus, choosing a convenient mean function can lead to an improvement of the estimation results.

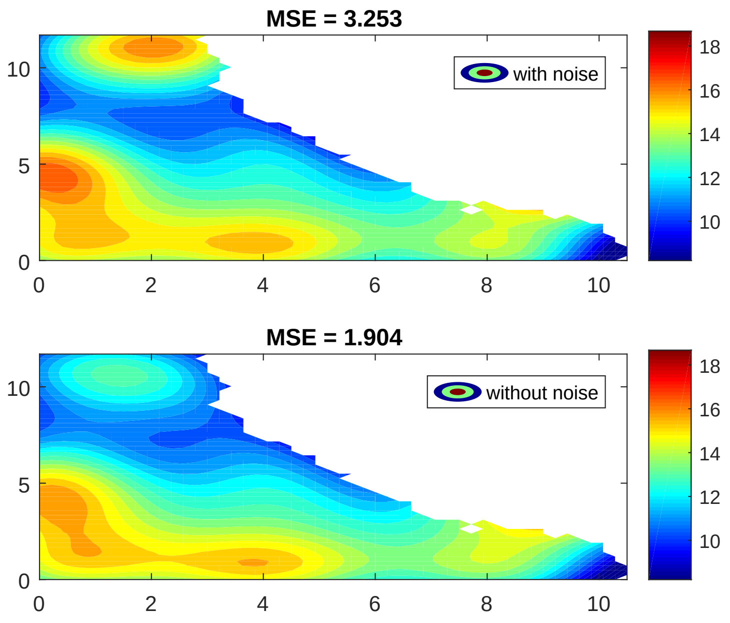


Figure 4.12: In the top figure, an estimation assuming noisy observations with $\sigma_n = 0.1$ is shown and in the bottom figure, an estimation assuming noise-free observations is shown. For both estimations all other parameters are the same.

4.7 Identity Precision Matrix and Sparse Covariance Functions

As proposed in chapter 2, the precision matrix of the Gaussian Process with built-in Gaussian Markov Random Field can be set to an appropriate identity matrix $\boldsymbol{I}$. In figure 4.14 (a) the mean and variance considering 50 estimation procedures are depicted. The proposed method using an identity matrix for the estimation process works even slightly better as the method with a radial precision matrix.

In 4.14 (b) the mean and variance considering 50 estimation procedures are depicted. The left result belongs to estimation using the squared-exponential kernel and the left to estimations using a distributed kernel function (proposed in [Xu et al., 2015])

$$k(h) = \begin{Bmatrix} (1-h)\cos(\pi h) + \frac{1}{\pi}\sin(\pi h), & h \leq 1 \\ 0, & \text{otherwise} \end{Bmatrix}, \tag{4.14}$$

where $h = ||\boldsymbol{x} - \boldsymbol{x}'||/r$ and r is the hyperparameter. Such a covariance function leads to a sparse covariance matrix, which can reduce the computational effort.

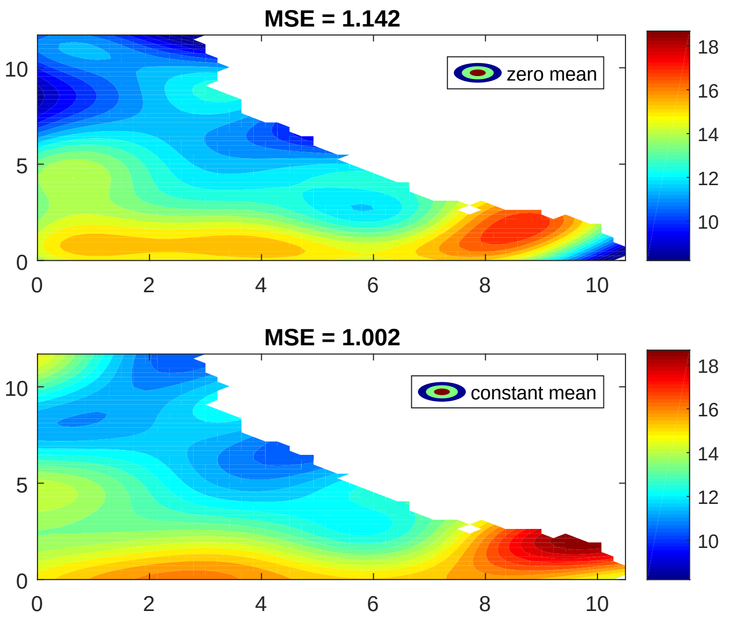
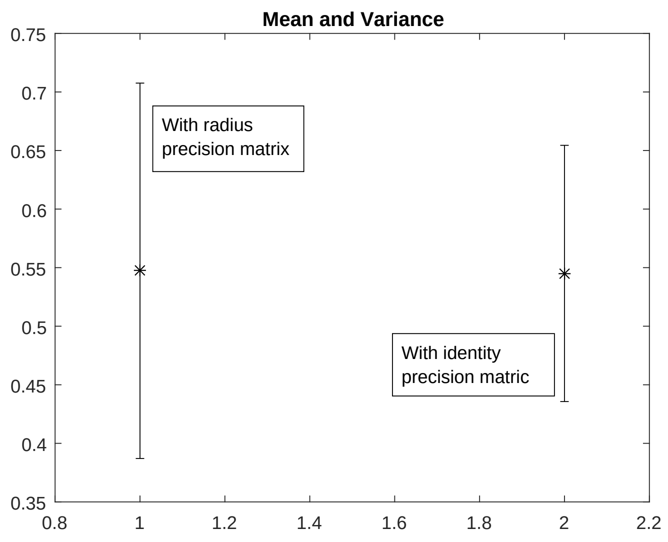
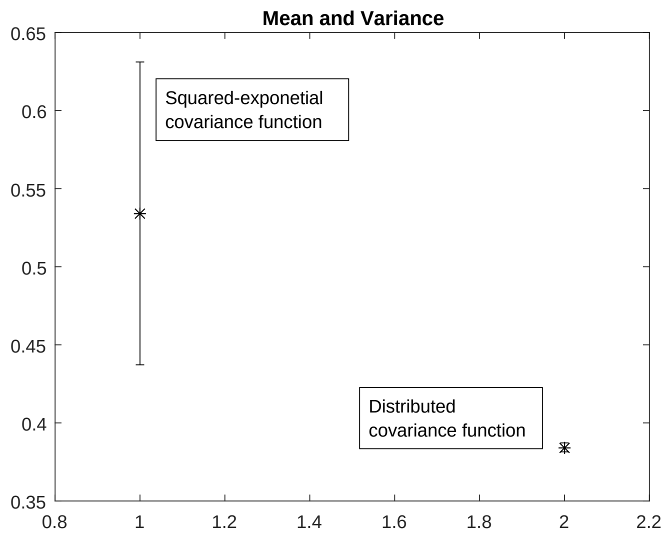


Figure 4.13: The top figure shows an estimation result using a zero mean function and the bottom figure using a constant mean function. For both estimations all other parameters are the same.

The mean calculation time for the estimation with the squared-exponential covariance function has been $t = 0.1141s$ and for the estimation with distributed covariance function $t = 0.1106s$. Thus, the estimation has been speeded up a little bit. In addition, the estimation results get better using the proposed distributed covariance function. Nevertheless, the accuracy of an estimation with GP with built-in GMRF, using a certain covariance function, depends strongly on the structure of the given problem. Thus, to find a good covariance matrix the structure of the problem has to be known.



(a) Identity Matrix as Precision Matrix



(b) Sparse Covariance Matrix

Figure 4.14: (a): Comparison between estimation with radius precision matrix and identity matrix as precision matrix using the mean (0.547; 0.545) and variance (16.02; 10.94) of the MSE. The variance has been scaled. 50 random observation points and 36 generating points have been used. The estimation has been done 50 times with improvement of the hyperparameters. (b): The same adjustments as by (a). Comparison between estimation with squared-exponential covariance function and distributed covariance function using the mean (0.53; 0.38) and variance (9.70; 0.29) of the MSE. The mean calculation time has been $t = 0.1141s$ and $t = 0.1106s$.

Chapter 5

Summary and Conclusions

First, a short summary of all topics treated in this paper is given in section 5.1. Afterwards, in section 5.2, conclusions are presented considering the in chapter 4 tested methods. At the end, in section 5.3, a brief outlook is given.

5.1 Summary

In chapter 1 a short introduction into the topic has been provided. The aim of this thesis has been explained and probabilistic models introduced. In order to get a brief overview of probabilistic models, the state of the art has been presented.
In chapter 2 an introduction to the theory of probabilistic models such as Gaussian Processes has been given. Gaussian Processes in particular have been developed. Markov Chains, Markov Random Fields and Gaussian Markov Random Fields have been explored as well. A Gaussian Process with built-in Gaussian Markov Random Field has been adopted from [Xu et al., 2015]. An inconsistency for the noise σ_w going to zero has been shown. In order to deal with this inconsistency an algorithm has been proposed and a proof has been provided. It turned out that for zero noise the precision matrix required for the GP with built-in GMRF disappears. This has led to the proposal, that an identity matrix can be used as the precision matrix. At the end, parameter adjustment has been introduced.
Based on the theoretical framework of GPs and GPs with built-in GMRF a toolbox has been created throughout the work on this thesis. The utilization of this toolbox has been explained in chapter 3.
In chapter 4 simulations considering questions about performance, parameter adjustment and functionality have been done. Therefore two data sets have been used. Estimating the fields given by these data sets, the performance of GPs

and GPs with built-in GMRF have been compared. Afterwards, the optimization methods for hyperparameter improvement and generating points positioning have been tested. Time variant estimations and the effect of non-stationary fields have been analysed. At the end of this chapter, the effect of setting an identity matrix as the precision matrix and the effect of sparse covariance matrices have been tested.

5.2 Conclusion

In section 4.2 a comparison between Gaussian Processes and its development with built-in Gaussian Markov Random Field has been shown. The GP with built-in GMRF has shown a much lower increase in computational time over the number of observation points than the standard Gaussian Process. Thus, in the case of an autonomous system with restricted computational resources such an estimation model is preferred. Nevertheless, the reduction of computational effort comes with a decrease of estimation accuracy. This trade-off has to be balanced out. The estimation accuracy for the Gaussian Process with built-in Gaussian Markov Random Field is restricted by the number of generating points used for the estimation, whereas the standard Gaussian Process is not restricted in estimation accuracy.
In order to get more accurate estimations, the optimization of the hyperparameters, the positions of the generating points and the number of generating points have been tested. Adjusting the hyperparameters of the covariance function can lead to better estimation results, but such an improvement comes with an increase of computational time and does not lead necessarily to better results. The positioning of the generating points can lead to improved estimation results, but the computational costs optimizing the positions are high. The generating points can be placed in a grid structure to avoid the positioning, which leads to sufficient results as presented in section 4.3.2. In order to optimize the number of generating points an information criterion can be used, which represents a decision for a trade-off between computational complexity (number of generating points) and estimation accuracy.
Two models for time variant estimations have been proposed. Both algorithms provide good estimation results but only the sequential algorithm has a constant computational effort in every estimation step, which makes it optimal for the addressed autonomous systems in this thesis.
The proposed idea working with an identity matrix as the precision matrix leads to good results. Thus, it is possible to avoid generating a precision matrix for the estimation method.
Using a sparse covariance matrix results into a slightly faster computation and finding a convenient mean function leads to an improvement in estimation accu-

racy.

5.3 Outlook

As mentioned in chapter 1, one objective for the proposed methods can be the exploration of an unknown environment in a fast and efficient way. Therefore, path finding algorithms play a major role. These path finding algorithms need an uncertainty criterion. Such an uncertainty criterion is provided by Gaussian Processes, or in general, by methods using a Bayesian framework. However, Gaussian Processes can be too slow for autonomous systems with limited computational power. Thus, a faster way of estimating fields and uncertainty have been provided by introducing Gaussian Processes with built-in Gaussian Markov Random Fields. GPs with built-in GMRF provide a fast uncertainty measure. In figure 5.1 a simple path finding algorithm is demonstrated. In this algorithm the next point of observation is chosen by searching the highest variance of the previous estimation result using a GP with built-in GMRF.

Gaussian Processes are still topic of various recent researches. A major problem, as shown in this thesis, is the scalability onto problems with a massive number of observations. This results from the computational complexity of building an inverse of the covariance matrix. A way to avoid inverting such a huge covariance

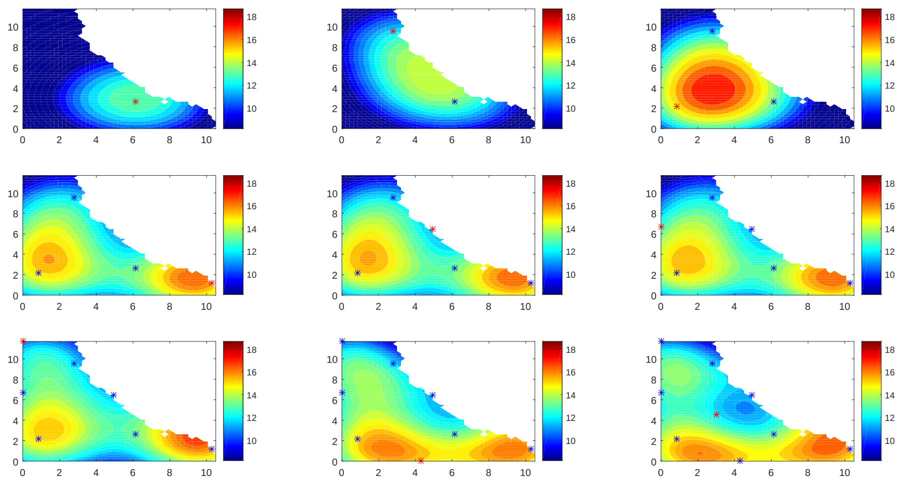


Figure 5.1: Sequential estimation using the marked points as observation points. The blue points are the previous observations whereas the red point is the new observation. The new observation point is chosen by searching the point with the highest variance from the previous estimation.

matrix is the utilization of Gaussian Markov Random Fields. Another interesting approach of enabling Gaussian Processes for huge data sets can be found in [Wilson et al., 2015]. There the KISS-approach from [Wilson and Nickisch, 2015] is extended, which then may lead to a reduction of complexity for Gaussian Processes from $O(n^3)$ to $O(n)$.

Various tuning knobs considering GPs with built-in GMRF can be explored further. Such tuning knobs are, for instance, an optimal structure of the covariance matrix or an optimal way to distribute the generating points.

Bibliography

[Akaike, 2011] Akaike, H. (2011). Akaike's information criterion. In *International Encyclopedia of Statistical Science*, pages 25–25. Springer.

[Binney et al., 2010] Binney, J., Krause, A., and Sukhatme, G. S. (2010). Informative path planning for an autonomous underwater vehicle. In *Robotics and automation (icra), 2010 IEEE international conference*, pages 4791–4796. IEEE.

[Binney et al., 2013] Binney, J., Krause, A., and Sukhatme, G. S. (2013). Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888.

[Blake et al., 2011] Blake, A., Kohli, P., and Rother, C. (2011). *Markov Random Fields for vision and image processing.* MIT Press.

[Cressie, 1993] Cressie, N. (1993). Statistics for spatial data: Wiley series in probability and statistics. *Wiley-Interscience, New York*, 15:105–209.

[Csató and Opper, 2002] Csató, L. and Opper, M. (2002). Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668.

[Dutter, 2000] Dutter, R. (2000). Geostatistics. *Univ. of Technology, Vienna.*

[Fuglstad et al., 2013] Fuglstad, G.-A., Lindgren, F., Simpson, D., and Rue, H. (2013). Exploring a new class of non-stationary spatial gaussian random fields with varying local anisotropy. *arXiv:1304.6949.*

[Fuglstad et al., 2014] Fuglstad, G.-A., Lindgren, F., Simpson, D., and Rue, H. (2014). Do we need non-stationarity in spatial models. *arXiv:1409.0743.*

[Fuglstad et al., 2015] Fuglstad, G.-A., Simpson, D., Lindgren, F., and Rue, H. (2015). Does non-stationary spatial data always require non-stationary random fields? *Spatial Statistics*, 14:505–531.

[Li, 2009] Li, S. Z. (2009). *Markov random field modeling in image analysis.* Springer Science & Business Media.

[Ma et al., 2016] Ma, K.-C., Liu, L., and Sukhatme, G. S. (2016). Informative planning and online learning with sparse gaussian processes. *arXiv:1609.07560.*

[Ranganathan et al., 2011] Ranganathan, A., Yang, M.-H., and Ho, J. (2011). Online sparse gaussian process regression and its applications. *IEEE Transactions on Image Processing*, 20(2):391–404.

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

[Rue and Held, 2005] Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: theory and applications.* CRC Press.

[Simpson et al., 2012] Simpson, D., Lindgren, F., and Rue, H. (2012). Think continuous: Markovian gaussian models in spatial statistics. *Spatial Statistics*, 1:16–29.

[Sydney and Paley, 2014] Sydney, N. and Paley, D. A. (2014). Multivehicle coverage control for a nonstationary spatiotemporal field. *Automatica*, 50(5):1381–1390.

[Wilson et al., 2015] Wilson, A. G., Dann, C., and Nickisch, H. (2015). Thoughts on massively scalable gaussian processes. *arXiv:1511.01870.*

[Wilson and Nickisch, 2015] Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (kiss-gp). *arXiv:1503.01057.*

[Xu et al., 2015] Xu, Y., Choi, J., Dass, S., and Maiti, T. (2015). *Bayesian Prediction and Adaptive Sampling Algorithms for Mobile Sensor Networks: Online Environmental Field Reconstruction in Space and Time.* Springer.

Appendix

There is a folder **PRO_010_Rottmann/** in the archive. The main folder contains the entries

- **PRO_010.pdf**: the pdf-file of the thesis PRO-010.
- **Data/**: a folder with all the relevant data, programs, scripts and simulation environments.
- **Latex/**: a folder with the *.tex documents of the thesis PRO-010 written in Latex and all figures (also in *.svg data format if available).
- **Presentation/**: a folder with the relevant data for the presentation including the presentation itself, figures and videos.