Multi-Agent Exploration under Sparsity Constraints

Manss, C.

DOI
10.4233/uuid:21a01def-a927-4fb1-8d9d-7c973762e62b

Publication date
2024

Document Version
Final published version

Citation (APA)
Manss, C. (2024). *Multi-Agent Exploration under Sparsity Constraints*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:21a01def-a927-4fb1-8d9d-7c973762e62b

MULTI-AGENT EXPLORATION UNDER SPARSITY CONSTRAINTS

Multi-Agent Exploration under Sparsity Constraints

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 6 May 2024 at 12:30 o'clock

by

Christoph Manss

M. Sc. Electrical and Information Engineering,
Christian-Albrechts-Universität zu Kiel, Germany,
born in Eckernförde, Germany.

This dissertation has been approved by the promotor.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. ir. G.J.T. Leus	Delft University of Technology, promotor
Independent members:	
Prof. dr. T. Keviczky	Delft University of Technology
Prof. dr. ing. C. Mecklenbräuker	Technical University Wien, Austria
Prof. dr. ir. A. Bertrand	Katholieke Universiteit Leuven, Belgium
Prof. Z. Tian	George Mason University, Virginia, USA
Prof. dr. ir. A. J. van der Veen	Delft University of Technology, reserve member
Other members:	
Dr. D. Shutin	German Aerospace Center, Germany

Keywords: Spatial Regression, Entropy, Swarm Systems, Bayesian Optimization, Distributed Processing

Casimir PhD series
ISBN: 978-94-6384-576-2
An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Alice: Would you tell me, please, which way I ought to go from here?
The Cheshire Cat: That depends a good deal on where you want to get to.
Alice: I don't much care where.
The Cheshire Cat: Then it doesn't much matter which way you go.

Lewis Carroll, Alice in Wonderland

CONTENTS

Summary **xi**

Samenvatting **xiii**

Notation **xv**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Challenges of Swarm Exploration . . . 2
1.2.1 Distributed Spatial Regression for a Swarm . . . 4
1.2.2 Swarm Exploration . . . 5
1.3 Problem Statement . . . 6
1.4 Contribution and Outline of the Thesis . . . 7

2 Models for Spatial Regression and their Estimation **11**
2.1 Measurement Model . . . 11
2.1.1 Basis Function Regression . . . 11
2.1.2 Sparse Basis Function Regression . . . 13
2.1.3 Basis Functions . . . 17
2.2 Network topologies . . . 19
2.3 Distributed Models . . . 21
2.3.1 Distributed Processing of Basis Function Regression . . . 21
2.4 Alternating Direction Method of Multipliers . . . 24
2.4.1 ADMM for Sparse Regression . . . 24
2.4.2 ADMM for Splitting Over Examples . . . 25
2.4.3 ADMM for Splitting over Features . . . 25
2.4.4 Automatic Parameter Estimation for ADMM . . . 26
2.5 Summary . . . 27

3 Distributed Sparse Bayesian Learning of Static Processes With Type II Optimization **29**
3.1 Fundamentals of Sparse Bayesian learning . . . 30
3.1.1 Sparse Bayesian Learning . . . 30
3.1.2 Distributed Sparse Bayesian Learning in the Literature . . . 31
3.1.3 Fast Marginal Likelihood Maximization . . . 32
3.1.4 Automatic Relevance Determination . . . 34
3.2 Distributed Sparse Bayesian Learning for Homogeneous Splitting . . . 34
3.2.1 Distributed Fast Marginal Likelihood Maximization . . . 35
3.2.2 Distributed Automatic Relevance Determination . . . 38
3.2.3 Simulations for Distributed SBL and Homogeneous Learning with Artificial Data . . . 39

3.2.4 Simulations for Distributed SBL and Homogeneous Learning with Real Data. . . . 44
3.2.5 Discussion of DFMLM and D-R-ARD for Homogeneous Learning . . 45
3.3 Distributed Sparse Bayesian Learning for Heterogeneous Splitting 47
3.3.1 Distributed Automatic Relevance Determination for Heterogeneous Learning 47
3.3.2 Convergence of D-R-ARD for Heterogeneous Learning 49
3.3.3 Simulations for D-R-ARD for SOF 50
3.4 Summary 52

4 Distributed Exploration with Optimal Experiment Design for the Distribution Paradigms SOE and SOF 53
4.1 Optimal Experiment Design. 54
4.2 D-Optimal Experiment Design for Distributed Exploration 55
4.2.1 Related Methods for Exploration 56
4.2.2 Centralized D-Optimal Experiment Design for Exploration under Sparsity Constraints 56
4.2.3 Impact of the L1-Regularized Linear Measurement Model for the Exploration Criterion 62
4.2.4 Distributed D-Optimal Experiment Design for Heterogeneous learning 64
4.2.5 Distributed D-Optimal Experiment Design for Homogeneous Learning 68
4.2.6 Evaluation of D-Optimal Experiment Design for Exploration with Real Data. . . . 71
4.3 Influence of Basis Functions on the D-optimality Criterion for Exploration . 77
4.4 Parallelization of the D-Optimal design criterion for a Swarm System. . . . 78
4.4.1 Coordination Strategies 79
4.4.2 Simulation Results for Coordination Strategies and D-Optimal Experiment Design for Swarm Systems. 80
4.5 Using a Sparse Bayesian Model for Exploration with D-optimality 82
4.5.1 D-Optimal Experiment Design for SBL and Homogeneous Learning 85
4.5.2 D-Optimal Experiment Design for SBL and Heterogeneous Learning 85
4.6 Summary and Discussion 87

5 Experimental Evaluations 89
5.1 Different Coordination Methods for Entropy Driven Exploration 89
5.1.1 Experiment Setup 90
5.1.2 Experimental Results. 93
5.2 Exploration with Multiple Ground Robots under Map Constraints 95
5.2.1 Preparations for the Experiment 95
5.2.2 Experimental Setting. 97
5.2.3 Experimental Results. 98

5.3 Summary and Conclusion 102

6 Conclusion and Outlook **105**
6.1 Summary of the Main Contribution 105
6.2 Discussion and Limitations 109
6.3 Outlook 110

A Crossvalidation for the Magnetic Field in the Holodeck **113**

B D-Optimality for Exploration **117**
B.1 Adding of Basis Functions to the D-optimality Criterion 117
B.2 Fast Evaluation of the D-Optimality 120
B.2.1 Fast Evaluation of the D-Optimality for SOE 121
B.2.2 Fast Evaluation of the D-Optimality for SOF 122
B.3 The Choice of $\widetilde{\gamma}$ for Exploration 122

C Consensus Algorithms **125**

D Collision Avoidance and Path Planning **127**
D.1 Collision Avoidance 127
D.2 A* - Path Planner 129

E Sensor calibration **131**

References **133**
. 133

Abbreviations **147**

Acknowledgements **149**

Curriculum Vitæ **151**

SUMMARY

The operation of robotic systems on extraterrestrial missions involves long distance communications, which have large delays and make the control of any agent complicated. This problem becomes even more dominant if multiple robotic systems are used. One solution to this problem is to increase the autonomy of each robotic system such that each robot can decide what to do according to its' current task and previous measurements. Furthermore, by utilizing cooperation between each robotic system, a task might even be solved more time efficiently.

Hence, this thesis considers the problem of using a spatially distributed robotic system for exploratory tasks such as mapping of an unknown process in the context of space exploration. This thesis therefore presents an information driven approach for exploration – based on the theory of optimal experiment design – to estimate new measurement locations that increase the accuracy of the used model. As many natural processes can be represented in a sparse basis, this thesis additionally assumes that the underlying model can be considered as sparse. Furthermore, the influence of a sparse model on the exploration phase is also examined. In general, this thesis studies how to do the estimation and exploration cooperatively by information exchange.

This thesis looks at two different estimation frameworks – the frequentist framework and the Bayesian framework – combined with two conceptually different distribution paradigms. Multiple distributed versions of a sparse Bayesian learning algorithm are developed for each distributed paradigm. Then, the estimation results are exploited to derive an information metric that is suited to estimate new measurement locations. Here, the D-optimality criterion is utilized and this thesis presents how to estimate the D-optimality criterion for the considered distributed settings and the different frameworks. Furthermore, the influence of the sparsity assumption is analyzed. For the frequentist framework, the sparsity inducing cost-function is altered into a ridge-regression based on the sparse parameter estimates, in order to approximate a Hessian matrix of the nonzero parameter estimates. For the Bayesian methods the covariance of the posterior probability density function (PDF) is used for the D-optimality criterion.

Next, the estimation of the model parameters and the estimation of the new measurement locations are formulated into multiple exploration algorithms for all frameworks and distribution paradigms. The thesis evaluates multiple optimization strategies of the exploration criteria, in order to figure out which are a better fit for a multi-agent system.

After the analysis of the building blocks of this work – the distributed parameter weight estimation and the distributed exploration – experimental validations demonstrate how the proposed system works in reality. The results show that the exploration algorithms are able to work in real-time and they indicate that the estimated covariance based on the Bayesian framework leads to better performances although the Bayesian methods are computationally more complex.

SAMENVATTING

Robotsystemen voor buitenaardse missies brengen langeafstandscommunicatie met zich mee, met grote vertragingen en ingewikkelde regeltechnieken als gevolg. Dit probleem wordt nog dominanter als er meerdere robotsystemen worden gebruikt. Eén oplossing voor dit probleem is het vergroten van de autonomie van elk robotsysteem, zodat elke robot kan beslissen wat hij moet doen op basis van zijn huidige taak en eerdere metingen. Bovendien kan een taak, door gebruik te maken van de samenwerking tussen de robotsystemen, zelfs sneller worden opgelost.

Daarom onderzoekt dit proefschrift het probleem van het gebruik van een ruimtelijk gedistribueerd robotsysteem voor verkennende taken, zoals het in kaart brengen van een onbekend proces in de context van ruimteverkenning. Dit proefschrift presenteert daarom een informatiegestuurde aanpak voor verkenning, gebaseerd op de theorie van optimaal experimentontwerp, om nieuwe meetlocaties te schatten die de nauwkeurigheid van het gebruikte model vergroten. Omdat veel natuurlijke processen op een schaarse basis kunnen worden weergegeven, wordt er in dit proefschrift bovendien van uitgegaan dat het onderliggende model als schaars kan worden beschouwd. Verder wordt ook de invloed van een schaars model op de exploratiefase onderzocht. In het algemeen onderzoekt dit proefschrift hoe de schatting en verkenning gezamenlijk kunnen worden uitgevoerd door middel van informatie-uitwisseling.

Dit proefschrift onderzoekt twee verschillende schattingsraamwerken – het frequentistische raamwerk en het Bayesiaanse raamwerk – gecombineerd met twee conceptueel verschillende distributieparadigma's. Voor elk gedistribueerd paradigma worden meerdere gedistribueerde versies van een spaarzaam Bayesiaans leeralgoritme ontwikkeld. Vervolgens worden de schattingsresultaten benut om een informatiemetriek af te leiden die geschikt is om nieuwe meetlocaties te schatten. Hier wordt het D-optimaliteitscriterium gebruikt en dit proefschrift presenteert hoe het D-optimaliteitscriterium kan worden geschat voor de beschouwde gedistribueerde omgevingen en de verschillende raamwerken. Verder wordt de invloed van de spaarzaamheidsaanname geanalyseerd. Voor het frequentistische raamwerk wordt de spaarzaamheid-inducerende kostenfunctie veranderd in een regressie op basis van de schaarse parameterschattingen, om de Hessiaan van de niet-nul parameterschattingen te benaderen. Voor de Bayesiaanse methoden wordt de covariantie van de posterior PDF gebruikt voor het D-optimaliteitscriterium.

Vervolgens worden de schatting van de modelparameters en de schatting van de nieuwe meetlocaties geformuleerd in meerdere verkenningsalgoritmen voor alle raamwerken en distributieparadigma's. Het proefschrift evalueert meerdere optimalisatiestrategieën van de verkenningscriteria, om erachter te komen welke beter passen bij een multi-agentsysteem.

Na de analyse van de bouwstenen van dit werk – de gedistribueerde parametergewichtschatting en de gedistribueerde verkenning – tonen experimentele validaties aan hoe het voorgestelde systeem in werkelijkheid werkt. De resultaten laten zien dat de

verkenningsalgoritmen in realtime kunnen werken en geven aan dat de geschatte covariantie op basis van het Bayesiaanse raamwerk tot betere prestaties leidt, hoewel de Bayesiaanse methoden computationeel complexer zijn.

NOTATION

$n, m, k, \ldots$	lowercase Latin letters denote scalars and are usually used as index
$N, M, N_k, M_k, \ldots$	uppercase Latin letters denote scalars that refer to the dimension of vectors or matrices; if they have a subscript k they are usually associated with the k-th sub-vector or sub-matrix
$\lambda, \sigma, \rho, \ldots$	lowercase Greek letters usually denote scalars that refer to parameters for algorithms or simulations
$\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\phi} \ldots$	vectors are denoted as bold lowercase letters
$x_i, w_i, \ldots$	elements of vectors are denoted as lowercase letters with subscript i, where i is the index of the element
$\boldsymbol{X}, \boldsymbol{\Phi}, \boldsymbol{X}_i, \boldsymbol{\Phi}_i \ldots$	matrices are denoted as bold uppercase Latin letters; a subscript indicates that they are sub-matrices
$X_{i,j}, \Phi_{i,j}, \ldots$	elements of a matrix are denoted as uppercase letters with two subscripts i, j, where i is the row index and j is the column index
$\boldsymbol{x}_i, \boldsymbol{\phi}_i, \boldsymbol{w}_i, \ldots$	vectors that belong to a row or column of a matrix and sub-vectors of vectors are denoted as bold lowercase letters with a subscript i, where usually i is the row-number, column-number, or the index of the sub-vector.
$\mathbb{N}, \mathbb{R}, \mathbb{C}$	positive integers, real numbers, and complex numbers, respectively
$\mathcal{G}, \mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A}, \mathcal{I}, \mathcal{M}$	are sets
$\mathscr{L}$	denotes a cost-function; often it also has a subscript with Latin letters for differentiation
$\mathcal{O}(\cdot)$	represents the order of magnitude; this is often used to quantify the complexity of an algorithm
$\widehat{\boldsymbol{w}}, \widehat{\boldsymbol{x}}, \widehat{\boldsymbol{\Gamma}}, \ldots$	a hat denotes an estimate

1

INTRODUCTION

1.1. MOTIVATION

Since the year 1975, humans are exploring the Martian surface with probes that landed on Mars. In 1995, the first roving probe, rover for short, ever sent to Mars by NASA was called Sojourner [1]. Sojourner's mission started on Mars on the 4th of July in 1997 and lasted until the 27th of September in the same year. Then in 2003, NASA sent two rovers to Mars — Spirit and Opportunity [2, 3] — in two separate missions. Spirit landed on Mars on the 4th of January 2004, and its mission lasted until the 25th of May in 2011. The mission of Opportunity started on Mars on the 25th January, 2004 and was ongoing until the 10th of June 2018. Today, the rover Curiosity [4] explores the Martian surface since the 6th August 2012, and more missions are planned [5, 6].

Many of these missions aim at finding evidence for previous life on Mars. Their approach is that a rover takes measurements, which deliver evidence if there has been previous life on Mars, or which give insights into the current conditions for life on Mars. The rover could measure, e.g., radiation, temperature, and humidity.

Yet, a mission on the Martian surface is expensive, and each mission imposes high risks. For example, NASA's Mars Exploration Rover (MER) mission of Spirit and Opportunity was initially planned as a single rover mission, and only the high financial price, the high risk of this mission, and the demand of a successful mission lead to a two-rover design choice. During this mission, Spirit got stuck in sand on the Martian surface in 2009. Spirit was used as a static scientific platform, until the active communication between Earth and Spirit finally broke down in May 2010. However, because of the two rover design choice, the overall mission continued with Opportunity until June 2018, and among other things, brought evidence that there is water on Mars [2].

In the example of Spirit and Opportunity, two rovers were sent to Mars and worked there on different locations without cooperation. What if more than only two rovers could be sent to Mars? Also, what if these rovers could work cooperatively with each other? Indeed, multiple rovers or a swarm of rovers — swarm for short — could bring potential benefits.

1. All rovers in the swarm could take measurements simultaneously and, consequently, more measurements could be taken in the same amount of time.

2. The distances between the swarm elements could be either large or small. If the distances between swarm elements are large, the rovers could be placed in a way such that the swarm achieves a large coverage. If the swarm is densely populated, it could form a sensing aperture, for instance to take pictures of one object from different angles, which could create a more profound image. Furthermore, the swarm could take measurements of an area of interest from different locations at the same time. Hence, the swarm is capable of taking measurements of time-varying and spatially distributed processes, which is not always possible for a single rover as it requires special assumptions on the observations and data.

3. As already presented in the example of Spirit and Opportunity, multiple rovers provide more resilience against system failures. Compared to a single rover, a swarm provides redundancy in hardware. In the considered example, it was the rover Spirit that got stuck in sand, and it was the rover Opportunity that continued with the mission. Of course, the swarm has to be designed in a way to deal with a complete failure of a rover in the swarm. Also, a failure of a sensor device on the rover could be considered and compensated for by the swarm if other rovers in the swarm have the same sensor.

4. In a swarm, the computational load can be distributed. Because the computational power on each rover is limited and because the measurements are collected distributively, distributed processing seems natural. This way, not all measurements have to be distributed in the swarm, but only intermediate processing steps. Also, for a large number of rovers in the swarm, the requirements on the communication connectivity between all rovers can be relaxed, i.e., each rover is able to communicate with another over a multi-hop connection.

In summary, swarms are fit to explore areas faster than a single rover, especially for spatially distributed and rapidly changing environments, and a swarm could provide more robustness compared with a single rover. Although the overall cost of the mission could increase by using a swarm, the individual price per rover could be reduced. For example, the cost of the single rover mission with Spirit in 2003 was estimated at 440 million dollar, whereas the mission for two rovers — Spirit and Opportunity — was estimated at 665 million dollar [7, Chap 6]. The cost was not doubled, but only increased by 50%. Thus, considering the cost and the risk of a space mission, the higher cost of a swarm is reasonable and does not scale equally with the number of rovers.

1.2. CHALLENGES OF SWARM EXPLORATION

Although there are many arguments for an exploration with a swarm, there are also many challenges that are still unaddressed in the literature. This section provides insights into the challenges that arise when dealing with swarm exploration.

Beginning with the swarm itself, a swarm is more complex to control and requires more sophisticated software and protocols [8]. Currently, each rover is controlled by a

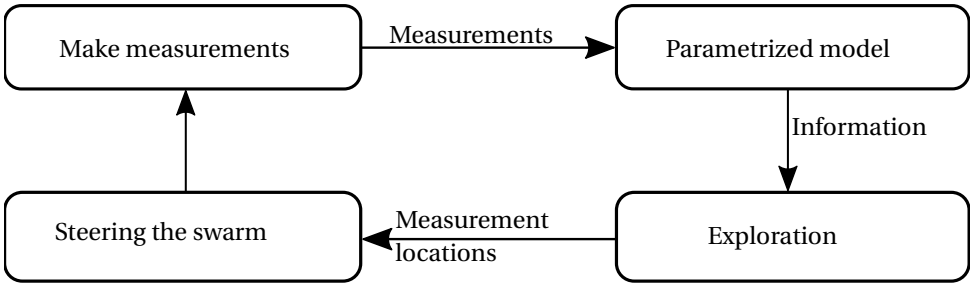


Figure 1.1: The concept of model based exploration. The measurements are used to estimate parameters of a model. Then, the model provides information about the parameter estimates. This information is used in an exploration criterion to estimate the next measurement locations for the swarm. Once the swarm reaches the next measurement locations, it takes new measurements and a new loop commences.

human operator and has a limited capability of autonomy. A swarm of rovers would require more human supervision, where the number of operators increases faster than the number of rovers: one human controls a rover and probably a few more supervisors to coordinate the swarm. The operators are needed to identify new measurement locations and to avoid collisions between the rovers. One approach to decrease the number of operators is to increase the degree of autonomy of each rover, which might involve complex algorithms and protocols. One example that provides more autonomy is path or motion planning. In [9], the authors provide an exhaustive survey for a motion planning algorithms that could provide more autonomy into a robotic system. Other overviews of path planing algorithms are provided in [10, 11], where the authors in [11] focus on path planing for lunar exploration. Although, a sophisticated and intelligent planning algorithm is an additional time investment regarding development, it can also save energy because easier paths are better traversed. Other examples to increase the autonomy could be coordination protocols for multiple robots [12], an increasing perception of the robots' surrounding, e.g., with artificial intelligence [13], or a flexible communication protocol in the swarm [14]. Nonetheless, increasing the autonomy of a swarm is of paramount importance for its practical use.

A promising concept for more autonomy and less human supervision is *model based exploration*. In model based exploration, a parametrized model is exploited to derive a control behavior for a robotic system, e.g., for steering of a rover. In case of a swarm the control behavior would steer the whole swarm as well as an individual robotic system. This concept is presented in Fig. 1.1, where the parametrized model uses the measurements to estimate its parameters and the information about them. This information is handed over to the exploration algorithm, where a criterion is evaluated to estimate new measurement locations, e.g., a criterion which selects measurement locations that posses yet unseen information or that provide a high gain of information. Once a measurement location is estimated, it is handed over to a rover control instance that translates the location's coordinates into steering commands. When the swarm reaches the estimated locations, the swarm takes measurements, and puts the measurements back

into the model.

From Fig. 1.1 it can be seen that model based exploration is a control loop, where the information is used as feedback to navigate each rover in the swarm. However, as the process is measured distributively, the allocated measurements are in the system distributed as well. Therefore, on the one hand, it is challenging to access the joint information and to coordinate the swarm when the model and the data are distributed in the network of the swarm. On the other hand, it is challenging to identify models that can be used for distributed processing, and to develop algorithms that infer the model parameters when distributed processing is used. The first challenge is referred to as *distributed spatial regression for a swarm* and the second challenge is referred to as *swarm exploration*. Both challenges are discussed in the following.

1.2.1. DISTRIBUTED SPATIAL REGRESSION FOR A SWARM

Regression is the analysis of the relation between dependent variables and independent input variables. The relation between these variables is described by a parametrized model. Regression aims at finding model parameters such that the model fits the dependent variables (in the sense that the model is close to the input variables or observations). Here, the measurements are the dependent variables and the location in space, where the model is evaluated, is the input variable. If the measurement data stems from a spatial process, the regression analysis is also referred to as spatial regression. This is considered in this thesis.

For a swarm with spatially distributed rovers, it is reasonable to carry out the spatial regression in a distributed way as well. Therefore, the estimation of the model parameters should be performed distributively. Distributed estimation methods can be roughly categorized into batch approaches and streaming or online approaches. Streaming approaches are used for measurement data that is measured continuously, and only the most recent measurement data is used for the regression analysis [15, 16]. Such approaches assume that there exists a flow of data. Streaming approaches can also be used in applications where computational resources are limited, because less data is held in memory and computations use only the new data. Moreover, streaming approaches are useful in scenarios where a model has to learn from a feedback. Batch approaches, however, require the availability of multiple measurements before the algorithm can commence; all data needs to be in the memory [17, 18]. It is therefore more constrained on the available computation resources, but in favor of more complex models. Because this thesis assumes that the measurement data is collected by multiple rovers at multiple measurement locations and that all data is relevant for the exploration, the batch approach is chosen. Overviews of batch approaches can be found in [19, 20]; nearly all considered methods rely on a consensus for distributing data in a network, which is referred to as the swarm in this thesis.

A popular method for distributed regression analysis that uses consensus is the alternating direction method of multipliers (ADMM) [18]. ADMM empowers implementations of regression analysis in a distributed fashion. Additionally, ADMM provides flexibility if the regression analysis requires extra constraints. These extra constraints comprise, e.g., regularization, which is explained for spatial regression in [21]. Known regularization functions for spatial regression are the ℓ_2-norm [22], the ℓ_1-norm [23], or mix-

tures [24]. The ℓ_1-norm regularization leads to a sparse parameter estimate, i.e., most of the parameter values become zero. This property can be become very useful for feature selection algorithms as parameters with zero value can be neglected. Also, the sparsity potentially leads to a reduction in the parameter space, the communication load, and the computational load. This could lead to a better computational efficiency. Moreover, models with fewer parameters or with only few non-zero parameters are easier to interpret. Lastly, the ℓ_1-norm is more robust against outliers in the data compared with the ℓ_2-norm, because it is less sensitive to large values in the data. Therefore, in this thesis, the ℓ_1-norm regularization is of particular interest. Furthermore, the distribution of the information in the swarm when the ℓ_1-norm is applied and the effect of the ℓ_1-norm on the exploration are not well studied in the literature.

Another methodology, where a sparse parameter estimate is inherent, is sparse Bayesian learning (SBL). SBL is part of empirical Bayes parameter estimation. Instead of an ℓ_1-norm, SBL uses a hierarchical prior on the parameter weights [25] to achieve a sparse parameter estimate. Distributed methods of SBL exist but either use loopy belief propagation [26], which is not guaranteed to converge or use an expectation maximization (EM) algorithm [27], which converges slowly. Therefore, effective SBL methods for distributed spatial regression in a swarm remain an unaddressed issue in the literature.

1.2.2. SWARM EXPLORATION

In model based exploration the learned model, which was used for spatial regression, is exploited for estimating the next measurement locations. This approach computes the information of potential measurement locations in a distributed manner by using the current model estimate. Thus, model based exploration creates a feedback loop, where the statistics of the parameter estimates give feedback to find new measurement locations (see Fig 1.1). Because this approach is based on the information in the model, it is also called information gathering or — due to the tight relation between information and entropy — entropy driven exploration. If model based exploration is applied to a swarm, it is in this thesis referred to as swarm exploration.

The calculation of the information in a model is presented in [28], but the authors assumed non-distributed data. Likewise, the authors in [29] quantified the information of a model to identify measurement locations for a sensor network. Yet, the authors did not consider that data might be distributed and that there might be feedback from the model estimates. In [30] the authors showed that a rover can be navigated based on the information provided by its estimated model. The authors further presented that if the robot chooses its next locations this way, it yields better results compared with other navigation methods — it leads to fewer crashes with the environment and better navigation paths are found. Other criteria that are related to an information metric are optimal experiment design criteria [31]. Although these criteria aim at minimizing the variance of the estimated model parameters, these criteria are mostly applied to sensor placement and not for swarm exploration. Latter considers that the position of the sensors can change, whereas the former assumes a static and final placement in a limited space. Thus, it is an open topic how these criteria can be utilized for exploration with multiple rovers in a distributed manner.

Approaches that involve more than a single robotic system have been looked at re-

cently in [32–38]. For example in [35], the authors evaluated how two rovers can benefit from each other when an information metric about their actions was used. Because the coordination and control of a swarm is already challenging, the authors in [36] looked into the coordination of a swarm for achieving coverage of a map in simulations. Other approaches that focus more on the coordination of a swarm while aiming for map coverage are studied in [39–45]. Nevertheless, how a swarm can cooperatively compute the information of the distributed data together with autonomous coordination, has not been considered yet.

Also, when working with a swarm of rovers, the experimental validation of swarm exploration is inevitable because otherwise the practical use cannot be quantified. The authors in [46] show a system of two robots, which cooperatively map structures by using noninvasive wireless channel measurements, but the robots are not guided with an information approach. The authors in [47] present an experiment, which was a simulated radiation leakage, with a swarm of up to nine robots, which have to explore the leakage. For the considered scenario, this paper observes a performance increase when adding more robots. In [48], the authors present a swarm of unmanned aerial vehicles (UAVs) for grass mapping. The authors highlight the importance of collision avoidance and self-awareness of the swarm, where self-awareness means that the UAVs know the position of each other. Also, the type of sensor plays an important role in the experimental validation. For example, the authors in [49] use cameras as sensors and utilize the obtained images for constructing a 3D image of a farmland while using a swarm. Their gathering of measurements is constrained by the energy consumption of the UAVs, the speed of the UAVs, and the quality of the obtained images. Yet, the swarm aims at covering the area, and uses no model driven exploration approach or an approach based on an information metric.

In summary, the experimental validation requires addressing self-awareness, sensor selection, collision avoidance, energy consumption, and many more aspects. All of these fields add further challenges to swarm exploration. If then regression analysis is applied and the information is used to control the swarm, research fields such as information theory, optimization theory, control theory, and robotics have to be combined. The combination of these fields imposes even more challenges on swarm exploration.

1.3. PROBLEM STATEMENT

Not all of the aforementioned challenges can be addressed in this work. The experimental validation requires a collision avoidance, the navigation within a map, and the map estimation itself. This thesis, does not study these topics but uses state-of-the-art solutions.

The focus of this thesis lies on the distribution of the regression analysis, and how the information of a distributed model can be accessed by the whole swarm. It is investigated how to navigate a swarm to new measurement locations by exploiting the statistics of the parameter weights. Furthermore, it is looked at how and if a sparse regularization leads to a more efficient exploration due to the aforementioned benefits of the ℓ_1-norm. In addition to that, efficient computations have to be developed to make swarm exploration working in real-time for an experimental validation. Hence, the following research questions are addressed in this thesis:

Q1 Can model based exploration be applied to a swarm with a distributed structure in real-time when using the entropy of a model's parameters?

Q1.1 If optimal design strategies are used for entropy driven exploration, what is the effect of the regularization on the exploration?

Q1.2 What are the benefits of a sparse regularization for the exploration with a swarm?

Q2 How should a model be distributed to be most suitable for model based exploration in a swarm, and how well does the distributed model approximate the measurements?

Q2.1 How do Bayesian methods perform compared with non-Bayesian methods when applied to distributed spatial regression for swarm exploration?

Q2.2 When using a distributed model, what data is required to be communicated for exploration and distributed spatial regression?

1.4. CONTRIBUTION AND OUTLINE OF THE THESIS

This section lists the outline of each chapter in this thesis. Along with the outline, the conducted research for this thesis is highlighted. As a summary, Fig. 1.2 displays the dependencies and links between the sections of the thesis.

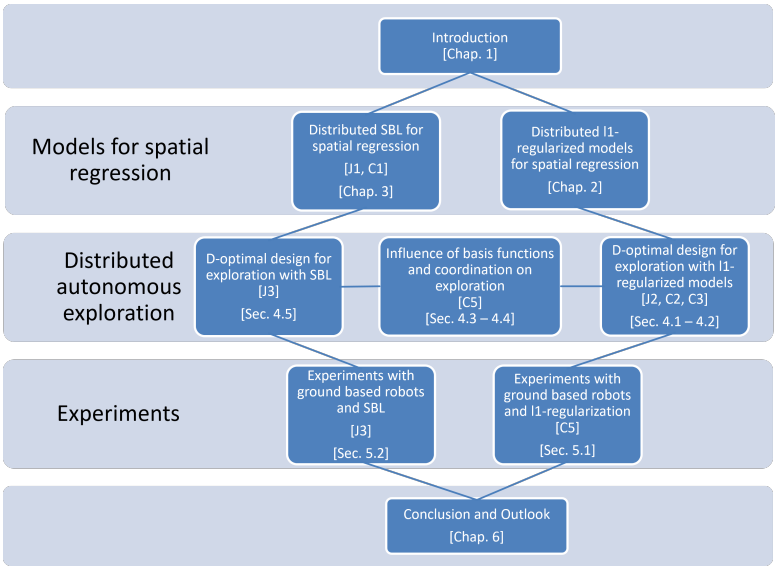


Figure 1.2: Overview of the thesis.

- Chapter 2 introduces the foundations of this thesis' theory. Because this work aims at distributed spatial regression, this chapter starts by introducing two distribution paradigms, which are applied to the used model. In addition, different network

topologies are shown and their applicability for swarms is discussed. Finally, this chapter shows how the parameters of the distributed spatial regression can be estimated.

- Chapter 3 introduces Bayesian methods, which provide uncertainties of the unknown parameters. Here, Bayesian methods are used in a distributed framework. This chapter provides the analysis and derivation of multiple distributed SBL algorithms for regression. As a result, a distributed SBL algorithm that is suited for the challenges of swarm exploration is chosen. The research regarding this chapter has been published in

  J1 C. Manss, D. Shutin, und G. Leus, *Consensus Based Distributed Sparse Bayesian Learning By Fast Marginal Likelihood Maximization*, IEEE Signal Processing Letters, S. 11, 2020

  C1 C. Manss, D. Shutin, and G. Leus, *Distributed Splitting-Over-Features Sparse Bayesian Learning with Alternating Direction Method of Multipliers*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, S. 3654-3658

- Chapter 4 focuses on swarm exploration. The methods from the previous two chapters are discussed for their applicability to swarm exploration. Depending on each regression method and each distribution paradigm, a criterion for model-based exploration is derived. This criterion has to be estimated in a distributed manner and in real-time. After the derivation, this criterion is tested in simulations. This chapter concludes that the distributed computation of the information is possible with consensus algorithms. Furthermore, computationally efficient performance metrics are derived. The research of this chapter has been carried out and published in

  J2 C. Manss and D. Shutin, *Global-Entropy Driven Exploration with Distributed Models under Sparsity Constraints*, Applied Sciences, Bd. 8, Nr. 10, S. 21, 2018

  C2 C. Manss, D. Shutin, A. V. Ruiz, T. Wiedemann, and J. Mueller, *Exploration under sparsity constraints*, in 2015 European Conference on Mobile Robots (ECMR), 2015, S. 1-6

  C3 C. Manss, D. Shutin, T. Wiedemann, A. Viseras, and J. Mueller, *Decentralized multi-agent entropy-driven exploration under sparsity constraints*, in 2016 4$^{\text{th}}$ International Workshop on Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing (CoSeRa), 2016, S. 143-147

- Chapter 5 examines swarm exploration in experiments. The developed techniques for distributed spatial regression are tested and evaluated. This chapter contains a discussion of the experiments and their analysis. The main result is that multiple rovers can navigate in a map and use the information criterion for exploration. Compared with other exploration and coverage algorithms, the entropy driven exploration requires fewer measurements to achieve an equal performance. The research that leads to this chapter is published in

J3 C. Manss, I. Kuehner, und D. Shutin, *Experimental Validation of Entropy-Driven Swarm Exploration under Sparsity Constraints with Sparse Bayesian Learning*, Entropy, Bd. 24, Nr. 5, Art. Nr. 5, Mai 2022

C4 C. Manss, T. Wiedemann, and D. Shutin, *Entropy Driven Height Profile Estimation with Multiple UAVs under Sparsity Constraints*, in 2017 IEEE Globecom Workshops (GC Wkshps), Singapore, Singapore, 2017, S. 1-6

C5 C. Manss, D. Shutin, and G. Leus, *Coordination methods for entropy-based multi-agent exploration under sparsity constraints*, in CAMSAP 2019, Le Gosier, 2019.

- The last chapter, Chapter 6, concludes the overall thesis. It summarizes the benefits of distributed spatial regression and the results of the experiments and simulations. After the summary, the research questions from the first chapter are answered. This chapter ends with an outlook on possible future research topics.

Other research that has been carried out besides this thesis Apart from the aforementioned papers, the author also did collaborate with other authors in the following papers:

- T. Wiedemann, C. Manss, D. Shutin, A. J. Lilienthal, V. Karolj, und A. Viseras, *Probabilistic modeling of gas diffusion with partial differential equations for multi-robot exploration and gas source localization*, in 2017 European Conference on Mobile Robots (ECMR), Sep. 2017, S. 17.

- T. Wiedemann, C. Manss, und D. Shutin, *Multi-agent exploration of spatial dynamical processes under sparsity constraints*, Auton Agent Multi-Agent Syst, Bd. 32, Nr. 1, S. 134162, Jan. 2018.

- J. Müller, A. V. Ruiz, C. Manss, und T. Wiedemann, C-ABT: A continuous control layer for inter-agent collision avoidance based on asynchronous backtracking, in 2015 IEEE Conference on Control Applications (CCA), Sep. 2015, S. 539544.

- E. Staudinger, D. Shutin, C. Manss, A. Viseras, und S. Zhang, *Swarm Technologies For Future Space Exploration Missions*, 14$^{\text{th}}$ International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-sairas), Madrid, Spain, Juni 2018.

- A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, L. Merino, *Decentralized multi-agent exploration with online-learning of Gaussian processes*, in 2016 IEEE International Conference on Robotics and Automation (ICRA), Mai 2016, S. 42224229.

2

Models for Spatial Regression and their Estimation

The current chapter presents the theoretical basics that are required throughout the thesis. The thesis' approach for swarm exploration is to use distributed models to interpret measurements, which are collected by a swarm. Therefore, each element in the swarm collects measurements of an observed spatially distributed physical process, and introduces new measurements into the model as time progresses. The new measurements are introduced by locally forming a batch of data. Consequently, at the beginning of the exploration, the model is empty and it is filled adaptively with more measurements while the swarm explores. Figure 2.1 presents the aforementioned approach. Each agent collects measurements along its way such that the model is updated with time.

This chapter, thus, begins by introducing a model that is suited for the chosen approach. In particular, basis function regression with sparsity constraints is utilized. Afterward, different network topologies are defined and two distribution paradigms for the utilized model — heterogeneous learning and homogeneous learning — are introduced. Both distribution paradigms inherit opposing properties, which are weighed against each other. Subsequently, Chapter 2 introduces how to estimate the model parameters with the ADMM algorithm, because it can be used for both paradigms together with sparsity constraints. A summary closes this chapter.

2.1. Measurement Model

2.1.1. Basis Function Regression

From now on, this thesis assumes that all vector and matrix operations are done in a Euclidean vector space. Basis function regression [60] is the analysis of the relation between, e.g., a process dependent variable $y \in \mathbb{R}$ and a process independent input variable $x \in \mathbb{R}$ by using functions $\boldsymbol{\phi}(x) = [\phi_1(x), \ldots, \phi_N(x)]$, with $\phi_n(x) \in \mathbb{R}$ for $n = 1, \ldots, N$ being a basis function. The weights are also referred to as model parameters $\boldsymbol{w} = [w_1, \ldots, w_N]^T \in \mathbb{R}^N$. Both $\boldsymbol{w}$ and $\boldsymbol{\phi}(x)$ form a model $f(\boldsymbol{w}, x)$. If f is linear in $\boldsymbol{w}$, i.e., $f(\boldsymbol{w}, x) = \phi_1(x)w_1 +$

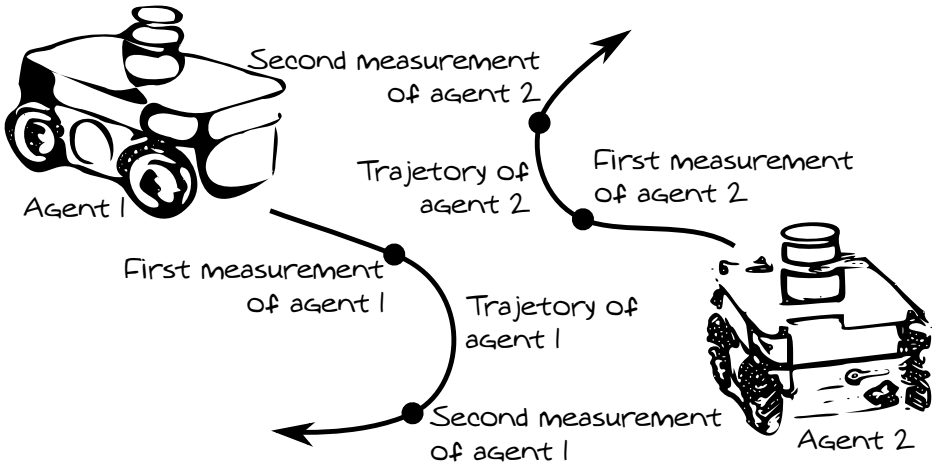


Figure 2.1: The approach of the thesis for exploration with multiple agents.

$\phi_2(x)w_2+\cdots=\boldsymbol{\phi}(x)\boldsymbol{w}$, then, f is called a *generalized linear model* [61]. This way, the relationship between y and x becomes non-linear, but the model parameters remain linear. Although the model complexity, i.e., the dimensionality of the model, can be large, basis function regression permits analytical deductions and computations because of its linear design [60].

In this thesis, a spatial process is considered, where it is assumed that the process is smooth, continuous, and static, where static means that it does not change over time. Furthermore, this process can be measured at a two-dimensional measurement position $\boldsymbol{x}\in\mathbb{R}^2$, yielding a scalar measurement $y(\boldsymbol{x})\in\mathbb{R}$. Please note that $\boldsymbol{x}$ can be easily extended to $\mathbb{R}^3$, but it is not considered here as the processes of interest are measured in two-dimensions. The measurements can be represented by a pair $\{y,\boldsymbol{x}\}$, which is denoted as the measurement tuple. This thesis uses a generalized linear measurement model $f(\boldsymbol{w},\boldsymbol{x})$ with N functions $\phi_n(\boldsymbol{x}),\ n=1,\ldots,N$ as

$$f(\boldsymbol{w},\boldsymbol{x})=\sum_{n=1}^{N}\phi_n(\boldsymbol{x})w_n=\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{w}. \tag{2.1}$$

Basis function regression aims now at finding the model parameters such that $f(\boldsymbol{w},\boldsymbol{x})$ approximates $y(\boldsymbol{x})$. This thesis assumes that this approximation is prone to noise, which can be modeled as additive white Gaussian noise (AWGN) such that

$$y(\boldsymbol{x})=f(\boldsymbol{w},\boldsymbol{x})+e(\boldsymbol{x}), \tag{2.2}$$

where $e(\boldsymbol{x})\sim\mathcal{N}(0,\lambda^{-1})$ is identically and independent distributed (i.i.d.) AWGN with zero mean and precision λ. The difference between the estimated model and $y(\boldsymbol{x})$ is defined as the error $\eta(\widehat{\boldsymbol{w}})=|y(\boldsymbol{x})-f(\widehat{\boldsymbol{w}},\boldsymbol{x})|$. If $f(\boldsymbol{w},\boldsymbol{x})$ approximates $y(\boldsymbol{x})$ well, the error is small.

When considering multiple measurements, matrix vector notation is more convenient. Define, $\boldsymbol{y}(\boldsymbol{X}) \triangleq \left[y(\boldsymbol{x}_1),\dots,y(\boldsymbol{x}_M)\right]^T \in \mathbb{R}^M$ as the measurement vector at $\boldsymbol{X}$, where $\boldsymbol{X} \triangleq [\boldsymbol{x}_1,\dots,\boldsymbol{x}_M]^T \in \mathbb{R}^{M\times 2}$ are all the two-dimensional measurement positions. The noise $e(\boldsymbol{x})$ is similarly defined. Further, all basis functions for all measurement locations are grouped to obtain $\boldsymbol{\phi}_n(\boldsymbol{X}) \triangleq \left[\phi_n(\boldsymbol{x}_1),\dots,\ \phi_n(\boldsymbol{x}_M)\right]^T \in \mathbb{R}^M$. Then, all basis functions together are represented by the matrix $\boldsymbol{\Phi}(\boldsymbol{X}) = \left[\boldsymbol{\phi}_1(\boldsymbol{X}),\dots,\ \boldsymbol{\phi}_N(\boldsymbol{X})\right] \in \mathbb{R}^{M\times N}$. The matrix $\boldsymbol{\Phi}(\boldsymbol{X})$ is also referred to as dictionary, feature matrix, or design matrix. For avoiding clutter in the notation, the dependency of the measurement locations $\boldsymbol{X}$ in $\boldsymbol{\Phi}(\boldsymbol{X})$, $\boldsymbol{e}(\boldsymbol{X})$, and $\boldsymbol{y}(\boldsymbol{X})$ is dropped and remains implicit unless it is needed in the context of the analysis. Hence, for now $\boldsymbol{\Phi}(\boldsymbol{X}) \equiv \boldsymbol{\Phi}$, $\boldsymbol{e}(\boldsymbol{X}) \equiv \boldsymbol{e}$, and $\boldsymbol{y}(\boldsymbol{X}) \equiv \boldsymbol{y}$. Hence, the measurement model for multiple measurements is given by

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{e}. \tag{2.3}$$

Likewise the definition of the error of the model changes to $\eta(\hat{\boldsymbol{w}}) = \|\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{w}}\|_2$, where $\|\cdot\|_2$ is the ℓ_2-norm. Similar to this error, the following cost function is defined

$$\mathscr{L}_{\text{LS}}(\boldsymbol{w}) = \frac{1}{2}\left\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\right\|_2^2. \tag{2.4}$$

The parameter $\hat{\boldsymbol{w}}$ that minimizes (2.4) can be found by a least squares solution as

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}, \tag{2.5}$$

if $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}$ exists. If $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is not invertible, $\hat{\boldsymbol{w}}$ can be computed by replacing $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$ with the Moore-Penrose pseudo inverse [62].

2.1.2. SPARSE BASIS FUNCTION REGRESSION

The model (2.3) can generally be used for estimating a spatial process. The process is then approximated by a number of basis functions according to the obtained M measurements $\{\boldsymbol{y},\boldsymbol{X}\}$. Typically, this process is unknown before the exploration starts, and $N > M$ basis functions are used to provide a good flexibility in the model. This thesis assumes that the observed process is sparse, which means, if an appropriate basis is chosen, it can be represented by few basis functions. Thus, the model can represent the unknown process in a compact manner. To estimate a sparse representation, the model can be regularized in a way that the model becomes compact. This could be achieved by setting most of the parameter weights to zero and, thus, making the corresponding basis functions irrelevant to the model. Regularization by sparsity is a way to make the model more compact, which will be explained in the following.

Definition 1. *A vector $\boldsymbol{a} \in \mathbb{C}^P$ or a matrix $\boldsymbol{A} \in \mathbb{C}^{P\times K}$ is considered to be sparse, if most of its entries are zero. Let S be the number of elements in either $\boldsymbol{a}$ or $\boldsymbol{A}$ that are equal to zero. Then $\boldsymbol{a}$ or $\boldsymbol{A}$ are considered to be S-sparse [63]. S is also denoting the degree of sparsity.*

Sparsity can be enforced by constraining or regularizing the problem (2.4) [64]. One regularizer could be the ℓ_0 "norm" or rather pseudo norm. For simplicity, however, this thesis from now on writes it as ℓ_0-norm.

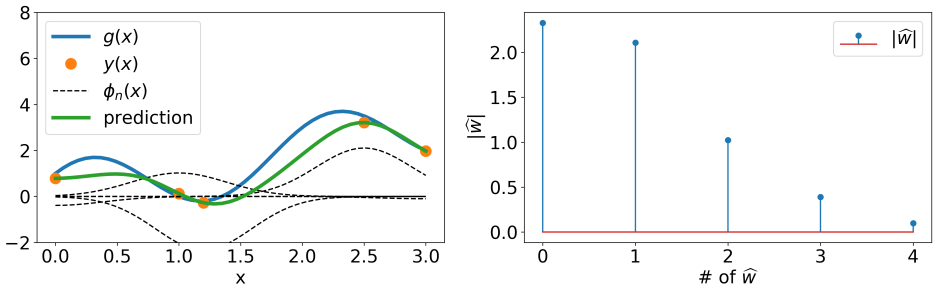


Figure 2.2: On the left, an example function in blue, which has been sampled at five points (orange dots). The basis functions are represented in dashed lines and the resulting prediction with the estimated weights is shown as a green line. On the right, the sorted absolute values of the parameter weights.

The ℓ_0-norm counts the number of non-zero elements of a vector. In the noiseless case, the ℓ_0-norm has to be minimized constrained by the model in the absence of noise. This leads to

$$\begin{aligned} &\min_{\boldsymbol{w}} \|\boldsymbol{w}\|_0 \\ &\text{subject to } \boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w}, \end{aligned} \tag{2.6}$$

where $\|\cdot\|_0$ is the ℓ_0-norm. The minimization of this problem leads to parameter weights $\hat{\boldsymbol{w}}$ with few non-zero entries. These non-zero entries correspond to basis functions that approximate the measurements well. The difference between regular and sparse estimation is exemplified in Ex. 1.

Example 1. *Figure 2.2 displays on the left a function $g(x)$, which is measured at $M = 5$ locations, indicated by $y(x)$. When these measurements are used to estimate the $N = 5$ parameter weights for the basis functions $\boldsymbol{\phi}_n(x), n = 1,\ldots,N$, the model results in the green line, denoted as* prediction. *The right of Fig. 2.2 shows the absolute values of the parameter weights. All of them are non-zero, because a least squares solution* (2.5) *is used, which does not lead to a sparse result. If the model is constrained by the ℓ_0-norm, a sparse solution might be found. Figure 2.3 displays the same scenario as in Fig. 2.2, but regularized by the ℓ_0-norm. Then, only two basis functions have a corresponding parameter weight that is non-zero. Although fewer basis functions are selected, the prediction is close to $g(x)$ and almost equal to the prediction in Fig. 2.2.*

Sparsity has also another advantage for exploration. At the beginning of the exploration the observed process is unknown and thus the number of measurements is small $M < N$. This means that the model is not uniquely solvable, and multiple basis functions could be equally relevant. A sparse regularization would remove irrelevant basis functions by setting their parameter weights to zero. This way, a sparse regularization might lead to a solvable model and to a parameter vector with many zeros.

The disadvantage of the ℓ_0-norm is that it is not smooth, not convex, in general not analytically solvable, and NP-hard [65, Sec. 3.4]. Therefore, an alternative is needed.

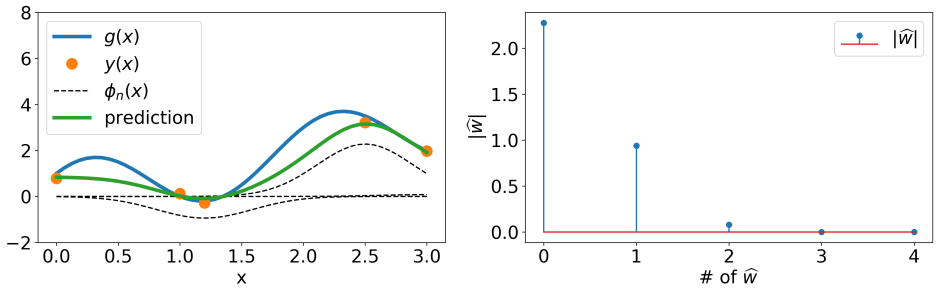


Figure 2.3: On the left, an example function in blue, which has been sampled at five points (orange dots). The basis functions are represented in dashed lines and the resulting prediction with the estimated weights is shown as a green line. On the right, the sorted absolute values of the parameter weights. The ℓ_0-regularization leads to a sparse result.

SPARSE REGRESSION WITH BASIS FUNCTIONS AND THE ℓ_1-NORM

Luckily, the ℓ_0-norm can be approximated under certain conditions by the ℓ_1-norm [63, 66, 67]. The ℓ_1-norm has gained a lot of interest in the last decades [67–69]. Similar to (2.6), a problem formulation with the ℓ_1-norm is

$$\begin{aligned} &\min_{\boldsymbol{w}} \|\boldsymbol{w}\|_1 \\ &\text{subject to } \boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w}. \end{aligned} \tag{2.7}$$

This norm is not smooth, but convex such that the theory of convex optimization can be applied [70] to solve (2.7) under certain conditions [63, 71].

Figures 2.4(a) and 2.4(b) give a graphical explanation for the ℓ_0-norm, the ℓ_1-norm, and the ℓ_2, if the model fit allows for an error ϵ. Then, the equality constraint in (2.6) and (2.7) is replaced by $\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2 \leq \epsilon$. In each figure, the uncertainty ellipsoid of the model fit, shown in gray, touches the objective, shown in blue. The optimal solution is highlighted by the dashed circle. Figure 2.4(b) shows the capability of the ℓ_1-norm to estimate the same solution as the ℓ_0-norm in Fig. 2.4(a). For completeness, Fig. 2.4(c) displays if the ℓ_2-norm of the parameter weights is minimized, subject to the measurement model. In this case, w_1 and w_2 would both be non-zero, and $\boldsymbol{w}$ would not be sparse.

THE LEAST ABSOLUTE SHRINKAGE AND SELECTION OPERATOR

The constrained form in (2.7) can also be formulated in a penalized form assuming noise. Accordingly, the ℓ_1-norm is added to (2.4) as a regularization term, which yields the objective

$$\mathscr{L}(\boldsymbol{w}) = \frac{1}{2} \left\| \boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y} \right\|_2^2 + \delta \|\boldsymbol{w}\|_1 , \tag{2.8}$$

where $\delta > 0$ is a regularization parameter. The choice of δ can be critical, as it plays a vital role in achieving the right amount of sparsity and reducing the error, and has to be determined before the estimation. Thus, this parameter is determined by cross

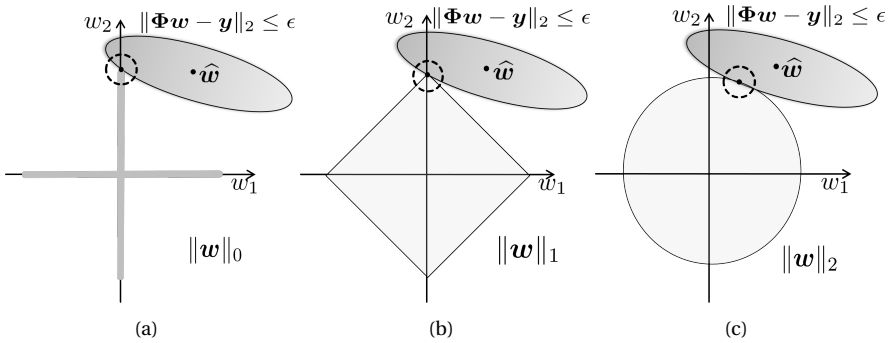


Figure 2.4: In all plots the dashed circle shows the intercept of both, the objective and the constraints, which is the optimal solution. (a) Graphical explanation of (2.6). Here only w_2 is non-zero. (b) Graphical explanation of (2.7). The ℓ_1-norm yields the same result as the ℓ_0-norm. (c) If the ℓ_2 norm is minimized subject to the constraints, both parameter weights will be non-zero.

validation and the research about the choice of δ is still ongoing [72, 73]. In [74, Section 5.9], the authors propose to choose $\delta = \lambda^{-1}\sqrt{2\log M}$, which indicates an influence of the number of measurements and the signal to noise ratio (SNR) on δ. However, this requires to know the precision λ^{-1} of the signal, which is often unknown for real data.

Minimizing (2.8) with respect to $\boldsymbol{w}$ is called the least absolute shrinkage and selection operator (LASSO) [74, Chap 3]

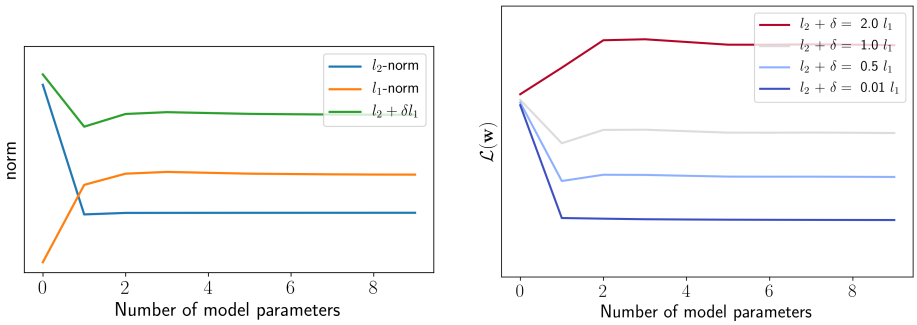
$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \mathscr{L}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\left\|\boldsymbol{\Phi w} - \boldsymbol{y}\right\|_2^2 + \delta\,\|\boldsymbol{w}\|_1\,. \tag{2.9}$$

Example 2 shows the influence of δ for Ex. 1.

Example 2. *Continue from Ex. 1, but for the LASSO and for an increasing number of basis functions. Both norms in* (2.9) *are shown as separate plots in Fig. 2.5(a). The LASSO estimator will use the addition of both norms, as shown in* (2.9)*. Increasing the number of basis functions leads to a lower ℓ_2-norm, whereas the ℓ_1-norm increases. As a consequence, both norms have a sweet spot, which is the minimum of the sum of both norms.*

This optimum depends on the choice of δ, which is shown in Fig. 2.5(b). If δ is small, the LASSO puts more weight on the ℓ_2-norm, and if δ is large the LASSO puts more weight on the ℓ_1-norm. The choice of δ is data dependent. In this example a good choice would be $\delta = 1$ because there is a global minimum where both norms balance each other out.

Many first order methods, which utilize the gradient of a function for optimization, exist to estimate $\widehat{\boldsymbol{w}}$, e.g. basis pursuit [68], iterative soft-thresholding algorithm (ISTA) [75–77], fast ISTA (FISTA) [77, 78], ADMM [18], etc. Often the motivation of these methods is that there is no analytical solution to the original problem. However, through an iterative optimization approach a solution of the problem can be estimated. Gradient methods are simple, yet known to be slow, but they can be improved through other methods. For example, the ISTA algorithm was improved by the FISTA algorithm [78], which uses a Nesterov method [79] to accelerate the convergence. Second order methods are computationally more expensive and require that the cost function is two times



(a) (b)

Figure 2.5: (a) A plot of the ℓ_1-norm, the ℓ_2-norm, and the combined norm (LASSO) when minimizing (2.8). With an increasing number of basis functions, the ℓ_1-norm increases and the ℓ_2-norm decreases. (b) This plot shows the influence of the penalty parameter δ on the LASSO.

differentiable. Thus such methods are applied less often. Here, because the ℓ_1-norm introduces a non-smooth term, it is not two times differentiable. Therefore, the second derivative of (2.8) is not defined, but it can be approximated with the theory of sub-differentials. Applying sub-differentiability, leads to a semi-smooth-Newton method [80, 81].

2.1.3. BASIS FUNCTIONS

Depending on the function, the concept of basis function regression can be related to other methods, such as kernel methods [60, 82–84] or Gaussian process (GP) models [85]. This thesis, however, is not restricted to a specific type of functions and is more general. The following introduces the types of functions, which are used in this thesis. In this thesis, these functions are classified into local and global basis functions, because of the following reasons. Local basis functions have non-zero values close to a defined center and are zero or close to zero everywhere else. They can describe abrupt changes in the exploration space that are not periodic. This makes them very flexible. Global basis functions on the other side have mostly non-zero values in the whole exploration space. They can be used to describe a more general behavior in the exploration space that can also be periodic or symmetric.

RADIAL BASIS FUNCTIONS

The radial basis functions (RBFs) are real valued functions that are symmetric around their center. They evaluate an input variable by the distance to some center. The center can be chosen arbitrarily, e.g. if the center is zero $\phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x}-\mathbf{0}\|)$ with $\|\cdot\|$ being a norm in the corresponding vector space.

An often applied basis function is the Gaussian function, which is defined as

$$\phi_n(\boldsymbol{x}) = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_n\|^2}{2\sigma_n^2}\right\}, \tag{2.10}$$

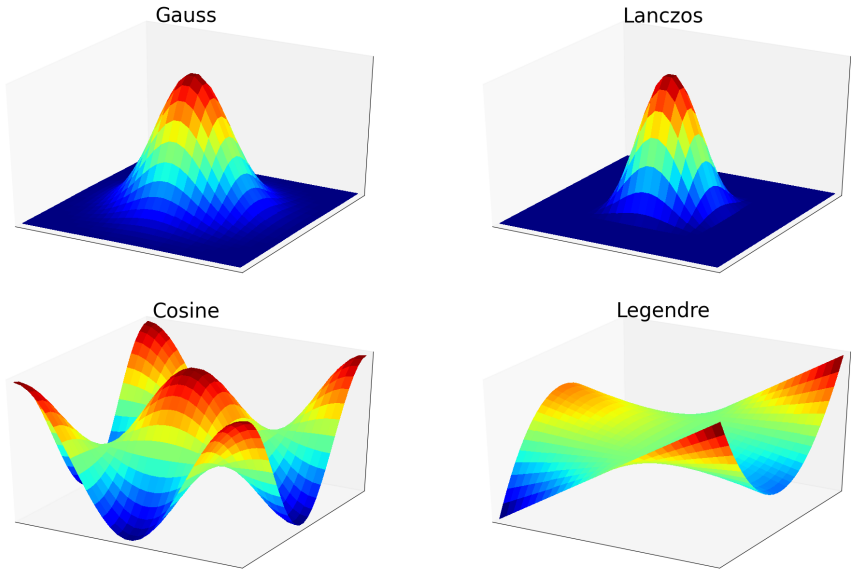


Figure 2.6: This figure exemplifies the four introduced basis functions. The Gaussian and Lanczos functions are spatially well localized, whereas the cosine function and the Legendre polynomial are defined everywhere.

where $\boldsymbol{\mu}_n$ is its center and σ_n describes the width of the Gaussian function. The Gaussian function is exemplified in Fig. 2.6. For distances larger than $3\sigma_n$ from its center $\boldsymbol{\mu}_n$, the Gaussian RBF becomes close to zero. Thus, it can be considered as a local basis function.

LANCZOS FUNCTION

The Lanczos function is often used in computer graphics for a smooth interpolation between digital values [86, Chapter 10]. Basically, it is a truncated product of two sinc-functions, which are evaluated for each dimension as

$$\phi_n(\boldsymbol{x}) = \begin{cases} \prod_{i=1}^{2} \operatorname{sinc}(x_i - \mu_{n,i}) \operatorname{sinc}\left(\frac{x_i - \mu_{n,i}}{\sigma_n}\right), & -\sigma_n < \|\boldsymbol{x} - \boldsymbol{\mu}_n\|_2 < \sigma_n \,,\ \sigma_n \neq 0 \\ 0 & \text{else,} \end{cases} \tag{2.11}$$

where $\sigma_n > 0$ is the width of the sinc-window. In (2.11) the sinc is calculated element-wise. Compared to the Gaussian function, the Lanczos function is narrower and sets all values larger than σ_n to zero. An example of the Lanczos function is shown in Fig. 2.6. Lanczos functions count as local basis functions.

POLYNOMIAL FUNCTIONS

Another example of basis functions are polynomials. Because of their orthogonality, i.e. the inner product of any two polynomials is zero, and their ability to represent smooth processes [87], Legendre polynomials are utilized in this thesis. They can be expressed

in a compact form with Rodrigues' formula [88] as

$$\phi_n(\boldsymbol{x}) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}\boldsymbol{x}^n} (\boldsymbol{x}^T \boldsymbol{x} - 1)^n, \tag{2.12}$$

where n is the degree of the polynomial. If $\boldsymbol{x}$ is normalized, Legendre polynomials are also orthonormal. According to the terminology in this thesis, Legendre polynomials are classified as global functions. An example is shown in Fig. 2.6. This classification is however not true for all polynomial functions. For example, *spline* functions, which are spatially constrained and often used in image processing [74, Chap. 5], are classified as local functions.

COSINE FUNCTION

In image compression, cosine functions of different frequencies form a discrete cosine transformation (DCT) [89]. Each function in the DCT is orthogonal to each other. In this thesis, it is assumed that the underlying spatial process can be separated into multiple cosine function, just as an image. Defining the frequencies $\boldsymbol{\kappa}_n = [\kappa_{n,1}, \kappa_{n,2}]^T \in \mathbb{R}^2$ the cosine function for a two-dimensional field is

$$\phi_n(\boldsymbol{x}) = \cos(\pi x_1 \kappa_{n,1}) \cos(\pi x_2 \kappa_{n,2}). \tag{2.13}$$

Fig. 2.6 shows an example of the cosine function. Cosine functions are defined everywhere and are therefore global functions.

2.2. NETWORK TOPOLOGIES

All agents in the swarm form a network, which can have different topologies. To describe these network topologies, (undirected) graphs can be utilized.

Definition 2. *Consider a set of sets $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_K\}$ is a set of vertices, and $\mathcal{E} = \{e_{i,j} = (v_i, v_j) : v_i, v_j \in \mathcal{V}; i < j\}$ is a set of edges. Then, $\mathcal{G}$ is called a graph [20].*

If every vertex is connected to another one over one or multiple edges, the graph is called connected.

Definition 3. *A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is connected if $\mathcal{V} \neq \emptyset$ and if there is a (possibly multihop) path between every pair of vertices $v_i, v_j \in \mathcal{V}$.*

If all vertices have a direct edge to all other vertices in the network, the graph is called fully connected. With Def. 2 and Def. 3, a swarm can be defined as follows.

Definition 4. *Consider a connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where each vertex $v_i \in \mathcal{V}$ is a robotic unit with an associated position $\boldsymbol{x}_i$ with at least one communication connection $e_{i,j} \in \mathcal{E}$ to a neighboring vertex v_j, then $\mathcal{G}$ is considered a swarm and each vertex is an agent.*

In a swarm, three different network topologies can be considered, which are shown in the following [19, 20, 90, 91].

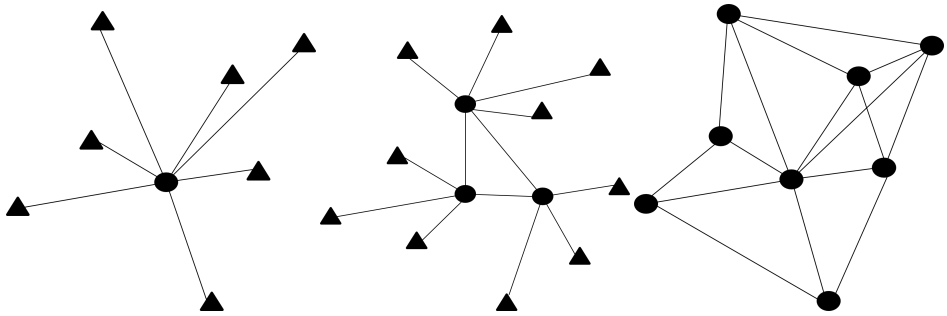


Figure 2.7: Different swarm topologies that are considered in this work. Red triangles represent an agent that can only measure. Red circles represent computing agents; they can measure and compute a regression given the measurements. (a) a centralized structure, where multiple agents transmit their measurements to a central computing agent. (b) a decentralized structure, where multiple computing agents cooperatively estimate a regression given the measurements. Not all agents are computing agents. (c) a fully distributed swarm, where all agents can measure and compute.

1. The swarm can be organized with multiple measuring agents, which communicate their measurements to a single agent with a computing unit. Then the swarm consists of two different types of robots, which makes the swarm heterogeneous. This network topology only addresses the *decentralized measurement allocation*, and is exemplified in Fig. 2.7(a), where a circle represents the computing unit and a triangle represents a measurement unit.

2. If the number of agents with computing units is increased, the swarm can be considered as a *distributed swarm*. Then, the measurements are gathered in a distributed fashion, and the computation is distributed over the specified computing units. However, not every computing agent has access to the full data, i.e. measurements or model parameters, in the swarm. Hence, the computing agents have to cooperate by means of communicating with each other. Figure 2.7(b) shows this topology.

3. If every agent in the swarm is a measuring and computing unit, the swarm is *fully distributed*. The fully distributed approach is shown in Fig. 2.7(c). Then, the swarm is most likely homogeneous, i.e. all agents are identical. Because each agent has now only partial information about the environment, the agents have to cooperate.

In summary, each network topology, shown in Fig. 2.7, potentially influences the distributed processing of the measurements. This thesis focuses on a fully distributed system. The following describes how the model (2.3) can be distributed, by considering two conceptually different distribution paradigms.

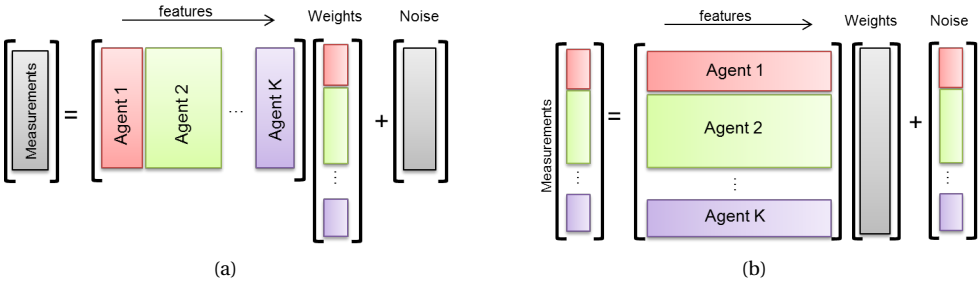


Figure 2.8: The colored areas represent an individual agent and gray areas are shared by all agents. (a) SOF paradigm: The measurement vector $\boldsymbol{y}$ and noise vector $\boldsymbol{e}$ are shared by all agents. (b) SOE paradigm: The weight vector $\boldsymbol{w}$ is shared by all agents.

2.3. DISTRIBUTED MODELS

This section extends now the introduced models to a distributed setting in a network. For a distributed setting two things are considered: the network topology and the distribution of the data. The network topology heavily impacts the communication, and it influences the data flow in the network. Whereas the data distribution defines what information is available at which agent. Thus it defines the distribution of the measurements, of the basis functions, and of the parameter weights.

2.3.1. DISTRIBUTED PROCESSING OF BASIS FUNCTION REGRESSION

For distributed processing of basis function regression, two distribution paradigms can be considered [92]. The first paradigm is called heterogeneous splitting. It is also known as heterogeneous learning, attribute distribution, or splitting-over-features (SOF). In SOF, all measurements or observable data are distributed among the computing units, and the agents exchange their intermediate estimates. However, each agent has its own local model. This paradigm is also presented in Fig. 2.8(a). The second paradigm is known as homogeneous splitting, homogeneous learning, splitting-over-examples (SOE), or instance distribution. In SOE, each agent has its own measurements or observed data, and contributes to the computation by exchanging local estimates. It is displayed in Fig. 2.8(b). This section presents both paradigms and how they are applied to basis function regression [92–95]. Mixtures of these two paradigms can also be considered [96, 97], but are not studied in this thesis.

HETEROGENEOUS SPLITTING

Heterogeneous splitting or SOF uses a basis function regression on each computing unit. Each k-th, where $k = 1,\dots,K$, computing unit in the swarm utilizes N_k basis functions in its model, with $N = \sum_{k=1}^{K} N_k$. The dictionary is then split column-wise as $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1(\boldsymbol{X}),\dots,\boldsymbol{\Phi}_K(\boldsymbol{X})]$, $\boldsymbol{\Phi}_k(\boldsymbol{X}) \in \mathbb{R}^{M\times N_k}$, $k = 1,\dots,K$. Likewise, the parameter weights can be re-formulated as $\boldsymbol{w} = \left[\boldsymbol{w}_1^T,\dots,\boldsymbol{w}_K^T\right]^T$, $\boldsymbol{w}_k \in \mathbb{R}^{N_k}$, $k = 1,\dots,K$. For linear combinations of basis functions, SOF splits $\boldsymbol{\Phi}(\boldsymbol{X})$ into different columns and $\boldsymbol{w}$ into rows.

Applying the new definitions to (2.3), the distributed model for SOF is

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\Phi}_1(\boldsymbol{X}) & \dots & \boldsymbol{\Phi}_K(\boldsymbol{X}) \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_K \end{bmatrix} + \boldsymbol{e} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k(\boldsymbol{X}) \boldsymbol{w}_k + \boldsymbol{e}, \tag{2.14}$$

as depicted in Fig. 2.8(a). In (2.14), all agents need access to the whole set of M measurements $\boldsymbol{y}$, but each agent has its own model $\boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k$, which adds up to the joint model. Thus, each agent only computes a model of size $N_k \ll N$ and every agent has its own design matrix $\boldsymbol{\Phi}_k(\boldsymbol{X})$. As this design matrix $\boldsymbol{\Phi}_k(\boldsymbol{X})$ is only known to the k-th agent, agent k can also define it as needed. For example, agent k can use different basis functions in $\boldsymbol{\Phi}_k(\boldsymbol{X})$ compared with the other agents, agent k can add or delete basis functions, and agent k can change parameters of basis functions without any further communication required. Hence, SOF provides more flexibility locally, as the local agent has full control over its used model. However, all measurements have to be distributed within the swarm, and all local models have to be gathered to compute the joint model.

Inserting (2.14) into the objective (2.8), it is redefined as the distributed objective for SOF and can be written as

$$\mathscr{L}(\boldsymbol{w}) = \frac{1}{2} \left\| \sum_{k=1}^{K} \boldsymbol{\Phi}_k(\boldsymbol{X}) \boldsymbol{w}_k - \boldsymbol{y} \right\|_2^2 + \delta \sum_{k=1}^{K} \|\boldsymbol{w}_k\|_1 . \tag{2.15}$$

The resulting objective contains now a cost-of-sums, which can optimized in a distributed manner with respect to the parameter weights [18].

HOMOGENEOUS SPLITTING

In homogeneous splitting or SOE every computing unit has the same model parameters $\boldsymbol{w}$, but each unit learns these parameters from its own measurements $\boldsymbol{y}_k(\boldsymbol{X}_k) \in \mathbb{R}^{M_k}$ and measurement locations $\boldsymbol{X}_k \in \mathbb{R}^{M_k \times 2}$, where $M = \sum_{k=1}^{K} M_k$. For brevity agent k's measurements can also be expressed as $\boldsymbol{y}_k(\boldsymbol{X}_k) = \boldsymbol{y}_k$ such that the measurement tuple for SOE is defined as $\{\boldsymbol{y}_k, \boldsymbol{X}_k\}$. In SOE, the measurement positions of the agents are different, and, thus, also the design matrices $\boldsymbol{\Phi}_k(\boldsymbol{X}_k) \in \mathbb{R}^{M_k \times N}$ where $\boldsymbol{X} = [\boldsymbol{X}_1^T, \dots, \boldsymbol{X}_K^T]^T$ and $\boldsymbol{\Phi}(\boldsymbol{X}) = [\boldsymbol{\Phi}_1^T(\boldsymbol{X}_1), \dots, \boldsymbol{\Phi}_K^T(\boldsymbol{X}_K)]^T$. The design matrix is split row-wise. Because each agent has its own measurements, each agent has an individual measurement noise $\boldsymbol{e}_k \in \mathbb{R}^{M_k}$.

These definitions are now inserted into (2.3) yielding the SOE model as

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_K \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_1(\boldsymbol{X}_1) \\ \vdots \\ \boldsymbol{\Phi}_K(\boldsymbol{X}_K) \end{bmatrix} \boldsymbol{w} + \begin{bmatrix} \boldsymbol{e}_1 \\ \vdots \\ \boldsymbol{e}_K \end{bmatrix}. \tag{2.16}$$

The distributed SOE model is also shown in Fig. 2.8(b).

	SOF	SOE
Shared	$\{\boldsymbol{y}, \boldsymbol{X}\}$	$\boldsymbol{w}$
Private	$\boldsymbol{\Phi}_k(\boldsymbol{X})\,\boldsymbol{w}_k,\ k=1,\dots,K$	$\{\boldsymbol{y}_k, \boldsymbol{X}_k\},\ k=1,\dots,K$
Complexity	is dominated by M	is dominated by N
Objective	$\frac{1}{2}\left\|\sum_{k=1}^{K}\boldsymbol{\Phi}_k(\boldsymbol{X})\,\boldsymbol{w}_k-\boldsymbol{y}\right\|_2^2 + \delta\sum_{k=1}^{K}\|\boldsymbol{w}_k\|_1$	$\frac{1}{2}\sum_{k=1}^{K}\|\boldsymbol{\Phi}_k(\boldsymbol{X}_k)\,\boldsymbol{w}-\boldsymbol{y}_k\|_2^2+\delta\,\|\boldsymbol{w}\|_1$

Table 2.1: Differences between SOF and SOE.

Applying this distribution paradigm to basis function regression, (2.16) is combined with (2.8) as

$$\mathscr{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{K}\left\|\boldsymbol{\Phi}_k(\boldsymbol{X}_k)\,\boldsymbol{w}-\boldsymbol{y}_k\right\|_2^2+\delta\,\|\boldsymbol{w}\|_1\,. \tag{2.17}$$

Compared to (2.15), SOE leads to a sum-of-costs. This cost-function can then be solved with multiple distributed algorithms [18, 98].

REMARKS ON THE DISTRIBUTION PARADIGMS

Both distribution paradigms have advantages and disadvantages. For distributed basis function regression, the agents cooperate by communicating measurements, measurement locations, and parameter estimates. Each paradigm has an effect on the communication load. For SOF the communication complexity is dominated by the number of measurements M. Hence, this paradigm is of advantage if the number of model parameters is higher than the number of measurements, i.e. $M \ll N$. SOE, on the other hand, is advantageous if the number of measurements is larger than the model parameters, i.e. $M \gg N$, because the number of model parameters mostly dominates the communication load and computational complexity.

The SOE paradigm aligns with the concept of a fully distributed swarm as explained in Sec. 2.2. All agents obtain measurements, and later exchange only their local estimates. Because the agents do not distribute their measurements and measurement locations, they have their *own* data. For SOF, the agents have to distribute their measurements $\boldsymbol{y}, \boldsymbol{X}$. This sharing of all data might seem counter-intuitive for a distributed system with a distributed data structure as it could be seen as less independent. However, it could also be seen as more cooperative, because the overall computational complexity could be large. Thus, SOF reduces this complexity. In [18, Sec. 8], the authors discuss the advantages and disadvantages for both paradigms. Yet, the influence of both distribution paradigms on the exploration is not studied and, thus, will be focused on in a later chapter in this thesis. Table 2.1 summarizes the differences of both distribution paradigms.

In a distributed setting, both distribution paradigms can be solved by means of an ADMM algorithm, which is presented in the following section.

2.4. ALTERNATING DIRECTION METHOD OF MULTIPLIERS

One approach to solve (2.8), (2.15), or (2.17) for $\boldsymbol{w}$ is the ADMM algorithm [18]. This method originates from the dual ascent method and the method of multipliers. The dual ascent method has the benefit that it can lead to a distributed optimization, while requiring strong assumptions on the objective function. On the other side, the method of multipliers uses an augmented Lagrangian on the objective, which leads to better convergence properties under milder conditions on the objective. However, the method of multipliers is not necessarily suited for distributed processing.

The ADMM is intended to take the benefits of both algorithms: the distributed processing of the dual ascent and the convergence properties of the method of multipliers. The ADMM assumes problems, which can be separated into two independent problems, of the form

$$\begin{aligned} &\text{minimize } h(\boldsymbol{w}) + l(\boldsymbol{z}) \\ &\text{subject to } \boldsymbol{A}\boldsymbol{w} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned} \tag{2.18}$$

where $h(\cdot)$ and $l(\cdot)$ are assumed to be convex functions and $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, $\boldsymbol{B} \in \mathbb{R}^{M\times N}$, $\boldsymbol{z} \in \mathbb{R}^N$, and $\boldsymbol{c} \in \mathbb{R}^M$. In the context of this thesis, ADMM provides multiple benefits: First the considered problems can be directly formulated into the ADMM formulation. Second, ADMM can be formulated in a distributed manner. Third, this algorithm tends to be robust and converge comparably fast [18]. In the following, ADMM will be analyzed for the distributed regression problem with sparsity constrains. However, for a better understanding, it is more useful to have a look at the centralized version first.

2.4.1. ADMM FOR SPARSE REGRESSION

The problem definition of ADMM (2.18) can be reformulated to fit the LASSO problem in (2.8). To this end, the following is used as a substitution

$$h(\boldsymbol{w}) \triangleq \frac{1}{2}\left\|\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{y}\right\|_2^2, \qquad l(\boldsymbol{z}) \triangleq \delta\left\|\boldsymbol{z}\right\|_1. \tag{2.19}$$

The constraint is then modified such that $\boldsymbol{z} = \boldsymbol{w}$. The problem in (2.8) is reformulated as

$$\begin{aligned} &\underset{\boldsymbol{w},\boldsymbol{z}}{\text{minimize}}\ \frac{1}{2}\left\|\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{y}\right\|_2^2 + \delta\left\|\boldsymbol{z}\right\|_1, \\ &\text{s.t. } \boldsymbol{z} - \boldsymbol{w} = 0. \end{aligned} \tag{2.20}$$

To formulate an objective function out of (2.20), the constraints are introduced with a Lagrangian. In addition, likewise to the method of multipliers, strong convexity is ensured by introducing an augmented Lagrangian as well. The resulting objective function is

$$\mathscr{L}(\boldsymbol{w},\boldsymbol{z},\boldsymbol{u}') = h(\boldsymbol{w}) + l(\boldsymbol{z}) + \boldsymbol{u}'^T(\boldsymbol{z}-\boldsymbol{w}) + \frac{1}{2}\rho\left\|\boldsymbol{z}-\boldsymbol{w}\right\|_2^2, \tag{2.21}$$

with $\boldsymbol{u}' \in \mathbb{R}^N$ being the dual variable, and $\rho > 0$ as a penalty parameter on the augmented Lagrangian.

For the analysis, ADMM is usually written in a more convenient form, where the linear and quadratic terms are combined

$$\mathscr{L}(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{u}) = h(\boldsymbol{w}) + l(\boldsymbol{z}) + \frac{1}{2}\rho \left\| \boldsymbol{z} - \boldsymbol{w} + \boldsymbol{u} \right\|_2^2, \tag{2.22}$$

with $\boldsymbol{u} = \boldsymbol{u}'/\rho$ as the scaled dual variable. The cost (2.22) is also known as the *scaled* form of ADMM.

2.4.2. ADMM FOR SPLITTING OVER EXAMPLES

Compared to the centralized problem, the distributed problems follow a similar approach for constructing the ADMM. In (2.17) every agent k has the same parameter weights $\boldsymbol{w}$. To make things distributed, it is assumed that every agent now has its own parameter weights $\boldsymbol{w}_k \in \mathbb{R}^N$, and that they are the same for all agents, i.e. $\boldsymbol{w}_k = \boldsymbol{w}_l,\ k, l = 1, \ldots, K, k \neq l$. Introducing this constraint in (2.17) and, then, comparing (2.17) with (2.18) yields

$$h(\boldsymbol{w}_k) = \frac{1}{2}\left\| \boldsymbol{\Phi}_k(\boldsymbol{X}_k)\boldsymbol{w}_k - \boldsymbol{y}_k \right\|_2^2, \qquad l(\boldsymbol{z}) = \delta \left\| \boldsymbol{z} \right\|_1. \tag{2.23}$$

The problem definition of (2.17) for ADMM is therefore

$$\begin{aligned} \text{minimize } & \sum_{k=1}^{K} \frac{1}{2}\left\| \boldsymbol{\Phi}_k(\boldsymbol{X}_k)\boldsymbol{w}_k - \boldsymbol{y}_k \right\|_2^2 + \delta \left\| \boldsymbol{z} \right\|_1 \\ \text{subject to } & \boldsymbol{w}_k - \boldsymbol{w}_l = 0,\ k, l = 1, \ldots, K, k \neq l \\ & \boldsymbol{w}_k - \boldsymbol{z} = 0,\ k = 1, \ldots, K. \end{aligned} \tag{2.24}$$

This problem is then solved locally at agent k. Implementations of ADMM for distributed linear sparse regression can be found in [90, 98–101]. Algorithm 1 shows the pseudo code of the ADMM version in [18] for agent k. There, at the beginning of the $i+1$-th iteration agent k computes a local estimate of $\widehat{\boldsymbol{w}}_k^{[i+1]}$. All other agents do likewise in parallel. Then, all agents distribute their estimates such that each agent k can compute an average of all estimates $\overline{\boldsymbol{w}}^{[i+1]} = \frac{1}{K}\sum_{k=1}^{K} \widehat{\boldsymbol{w}}_k^{[i+1]}$. Here, consensus algorithms can be applied to compute this efficiently, see App C. This average $\overline{\boldsymbol{w}}^{[i+1]}$ is then used to update $\boldsymbol{z}_k$ and $\boldsymbol{u}_k$. One possible convergence criterion could be that the difference $|\widehat{\boldsymbol{u}}_k^{[i+1]} - \boldsymbol{u}_k^{[i]}|$ drops below a pre-defined threshold or if the number of iterations reaches a pre-defined limit.

2.4.3. ADMM FOR SPLITTING OVER FEATURES

As shown already before, the objective for SOF includes the sum $\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k$ inside the norm. Thus, every agent approximates all measurements $\boldsymbol{y}$ partially with its model $\boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k$. For solving (2.15) distributively, the local basis functions and weights

Algorithm 1 ADMM for SOE at agent k

$\boldsymbol{z}_k^{[0]} = 0$ and $\boldsymbol{u}_k^{[0]} = 0$
for $i = 1, \ldots,$ **do**
  $\widehat{\boldsymbol{w}}_k^{[i+1]} = \arg\min_{\boldsymbol{w}_k} \frac{1}{2} \left\| \boldsymbol{\Phi}_k(\boldsymbol{X}_k)\boldsymbol{w}_k - \boldsymbol{y}_k \right\|_2^2 + \frac{1}{2}\rho \left\| \boldsymbol{w}_k - \boldsymbol{z}_k^{[i]} + \boldsymbol{u}_k^{[i]} \right\|_2^2$
  Distribute $\widehat{\boldsymbol{w}}_k^{[i+1]}$ to the neighbors to calculate $\overline{\boldsymbol{w}}^{[i+1]} = \frac{1}{K}\sum_{k=1}^{K} \widehat{\boldsymbol{w}}_k^{[i+1]}$
  $\widehat{\boldsymbol{z}}_k^{[i+1]} = \arg\min_{\boldsymbol{z}_k} \delta \left\| \boldsymbol{z}_k \right\|_1 + \frac{K\rho}{2} \left\| \boldsymbol{z}_k - \overline{\boldsymbol{w}}^{[i+1]} - \boldsymbol{u}_k^{[i]} \right\|_2^2$
  $\widehat{\boldsymbol{u}}_k^{[i+1]} = \boldsymbol{u}_k^{[i]} + \overline{\boldsymbol{w}}^{[i+1]} - \widehat{\boldsymbol{z}}_k^{[i+1]}$
  Check convergence
  $\boldsymbol{z}_k^{[i+1]} \leftarrow \widehat{\boldsymbol{z}}_k^{[i+1]}$, and $\boldsymbol{u}_k^{[i+1]} \leftarrow \widehat{\boldsymbol{u}}_k^{[i+1]}$,
Construct $\widehat{\boldsymbol{w}} = \widehat{\boldsymbol{w}}_k^{[i+1]}$
return $\widehat{\boldsymbol{w}}$

are substituted with $\boldsymbol{z}_k = \boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k$, with $\boldsymbol{z}_k \in \mathrm{R}^M$ such that the problem becomes

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{z}}{\text{minimize}}\ & \frac{1}{2} \left\| \sum_{k=1}^{K} \boldsymbol{z}_k - \boldsymbol{y} \right\|_2^2 + \delta \sum_{k=1}^{K} \left\| \boldsymbol{w}_k \right\|_1 \\ \text{s.t. } & \boldsymbol{z}_k - \boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k = 0,\ \forall k = 1, \ldots, K. \end{aligned} \tag{2.25}$$

In [18, Chap. 8], the authors provide a solution for solving (2.25) distributively. In [102], the authors suggest instead to optimize the dual function of (2.15). It is claimed that the dual function converts into a sum-of-costs, which is straightforward to optimize distributively. The downside of this approach is that it requires to compute two dual functions of the separated problem. For reasons of comparison, the distributed ADMM algorithm in [18, Chap. 8] is chosen for this thesis.

Algorithm 2 presents the pseudo code of an ADMM algorithm for the SOF paradigm as introduced in [18]. There, Algorithm 2 assumes that all measurements $\boldsymbol{y}$ have been distributed to all K agents prior the algorithm commences. Then, for each $i+1$-th iteration, each agent k locally estimates $\widehat{\boldsymbol{w}}_k^{[i+1]}$. Then, the locally estimated model $\boldsymbol{\Phi}_k(\boldsymbol{X})\widehat{\boldsymbol{w}}_k^{[i+1]}$ is distributed to all agents such that an average over all models can be computed locally as $\overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\Phi}_k(\boldsymbol{X})\widehat{\boldsymbol{w}}_k^{[i+1]}$. Here, consensus algorithms can be applied to compute this efficiently, see App C. Based on the computed average over all models, $\boldsymbol{z}_k^{[i+1]}$ and $\boldsymbol{u}_k^{[i+1]}$ can be computed. As before in Algorithm 1, the difference $|\widehat{\boldsymbol{u}}_k^{[i+1]} - \boldsymbol{u}_k^{[i]}|$ can be utilized as a convergence criterion, if it drops below a pre-defined threshold.

2.4.4. AUTOMATIC PARAMETER ESTIMATION FOR ADMM

ADMM introduces a penalty parameter ρ for the augmented Lagrangian, which is data dependent and has to be validated beforehand. The adaptive selection of this penalty parameter has been discussed in the literature [103, 104]. Particularly in [104], the au-

Algorithm 2 ADMM for SOF at agent k

Require: $\boldsymbol{X}$ and $\boldsymbol{y}$ are known locally

1: $\boldsymbol{z}_k^{[0]} = 0$ and $\boldsymbol{u}_k^{[0]} = 0$
2: $\overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[0]} = 0$
3: **for** $i = 1, \ldots,$ **do**
4: $\quad \widehat{\boldsymbol{w}}_k^{[i+1]} = \arg\min_{\boldsymbol{w}_k} \delta \|\boldsymbol{w}_k\|_1 + \frac{\rho}{2} \left\| \boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k - \boldsymbol{\Phi}_k(\boldsymbol{X})\boldsymbol{w}_k^{[i]} - \boldsymbol{z}_k^{[i]} + \overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i]} + \boldsymbol{u}_k^{[i]} \right\|_2^2$
5: $\quad$ Distribute $\boldsymbol{\Phi}_k(\boldsymbol{X})\widehat{\boldsymbol{w}}_k^{[i+1]}$ to the neighbors
6: $\quad \overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\Phi}_k(\boldsymbol{X})\widehat{\boldsymbol{w}}_k^{[i+1]}$
7: $\quad \widehat{\boldsymbol{z}}_k^{[i+1]} = (K+\rho)^{-1}\left(\boldsymbol{y} + \rho(\overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} + \boldsymbol{u}_k^{[i]})\right)$
8: $\quad \widehat{\boldsymbol{u}}_k^{[i+1]} = \boldsymbol{u}_k^{[i]} + \overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} - \widehat{\boldsymbol{z}}_k^{[i+1]}$
9: $\quad$ Check convergence
10: $\quad \boldsymbol{z}_k^{[i+1]} \leftarrow \widehat{\boldsymbol{z}}_k^{[i+1]}$ and $\boldsymbol{u}_k^{[i+1]} \leftarrow \widehat{\boldsymbol{u}}_k^{[i+1]}$,
11: Communicate $\widehat{\boldsymbol{w}}_k = \widehat{\boldsymbol{w}}_k^{[i+1]}$ to construct $\widehat{\boldsymbol{w}} = \left[\widehat{\boldsymbol{w}}_1^T, \ldots, \widehat{\boldsymbol{w}}_K^T\right]^T$
12: **return** $\widehat{\boldsymbol{w}}$

thors define the primal residual and the dual residual respectively as

$$r^{[i]} = \|\boldsymbol{w}^{[i]} - \boldsymbol{z}^{[i]}\|_2, \qquad s^{[i]} = \|\boldsymbol{z}^{[i]} - \boldsymbol{z}^{[i-1]}\|_2, \tag{2.26}$$

where $\boldsymbol{z}^{[i]}$ is the variable $\boldsymbol{z}$ at the i-th iteration when ADMM is applied. Then, the proportion of the primal residual and the dual residual are observed to derive a method to determine ρ during the estimation. This method is called residual balancing and is applied as follows. At each iteration of ADMM the update rule of the ADMM penalty parameter is

$$\rho^{[i+1]} = \begin{cases} \tau^{\text{incr}}\rho^{[i]}, & \text{if } r^{[i]} > \mu s^{[i]} \\ \rho^{[i]}/\tau^{\text{decr}}, & \text{if } s^{[i]} > \mu r^{[i]} \\ \rho^{[i]}, & \text{else,} \end{cases} \tag{2.27}$$

where $\mu > 1, \tau^{\text{incr}} > 1$, and $\tau^{\text{decr}} > 1$. According to [18, Sec. 3.4], typical choices are $\mu = 10$ and $\tau^{\text{incr}} = \tau^{\text{decr}} = 2$. If the scaled form of ADMM is applied, the scaled dual variable $\boldsymbol{u} = \boldsymbol{u}'/\rho$ has to be updated with the new ρ as well.

2.5. SUMMARY

This chapter laid down the foundation for the methods used in this thesis. The model, which is introduced in Sec. 2.1.1, will be used in Chapter 3 for a Bayesian framework. The network topologies and the introduced ADMM algorithm will be used as well in the following chapter. The current chapter also introduced different distribution paradigms — SOF and SOE. Each of these paradigms will be used in the Bayesian framework for the estimation of the parameter weights. Furthermore, as will be shown, each paradigm influences the exploration. This is investigated in Chapter 4.

3

Distributed Sparse Bayesian Learning of Static Processes With Type II Optimization

The last chapter introduced a method for a sparse estimate of the parameter weights, the LASSO and its distributed version realized with an ADMM approach. Despite the fact that the LASSO algorithm finds a sparse parameter weight estimate for the problem (2.9), it has some drawbacks. First, the LASSO requires a penalty parameter, which controls the degree of sparsity. This parameter is data dependent and has to be set beforehand. Second, the second derivative of the LASSO is not defined such that mostly first order methods exist, which have a slower convergence compared with second order methods. Thus, it is hard to develop faster methods with regard to their convergence. However, there are accelerated methods [77, 78, 105], which put additional requirements or assumptions on the problem. Third, the exploration, which will be introduced in the next chapter, aims at using the second order information as an exploration criterion. For LASSO methods, this second order information could only be approximated, as will be shown in Chapter 4, because the second derivative of the problem (2.9) is not defined. Bayesian methods however provide a covariance matrix of the parameter estimates, which can be used as an alternative to the frequentist approach.

A Bayesian interpretation of the model (2.3) would treat the parameter weights as random variables. As the parameter weights are assumed to be sparse, methods of sparse Bayesian learning (SBL) are applied in this thesis. SBL belongs to the family of empirical Bayes techniques, where the sparsity of the parameter weights $\boldsymbol{w}$ is enforced by choosing an appropriate prior, e.g. a gamma-Gaussian prior [25]. This prior ensures that the probability mass is concentrated on the axes of the parameter space. Furthermore, the SBL methods decide the degree of sparsity based on the data and, compared with the LASSO methods, do not require a penalty parameter. Thus, SBL methods are more data dependent. Algorithms that use SBL for solving (2.9) can be found in [106–108]. How-

ever, these algorithms are only centralized, and are in this form unsuited for distributed processing. Yet, distributed processing is crucial for the work considered in this thesis.

The current chapter presents the derivation of distributed Bayesian methods, which fit for swarm exploration as applied in this thesis. This chapter starts by explaining SBL in general and presenting two methods: the fast marginalized likelihood maximization (FMLM) algorithm and the reformulated automatic relevance determination (ARD) (R-ARD) algorithm. These methods are analyzed regarding their applicability to distributed processing. Then, these algorithms are considered for the distribution paradigms SOE and SOF, which are introduced in Sec. 2.3.1 such that new distributed versions of these algorithms are developed. Each of the derived distributed algorithms is then evaluated regarding its performance, convergence, and the amount of data that needs to be communicated. This chapter ends with a discussion and a summary of the presented results.

3.1. FUNDAMENTALS OF SPARSE BAYESIAN LEARNING

3.1.1. SPARSE BAYESIAN LEARNING

In an SBL framework typically two approaches for algorithm development are distinguished: *Type I* and *Type II*. Both types use a hierarchical prior [109, Chap. 5]. However, algorithms of Type I are based on maximum a-posteriori (MAP) estimation, whereas algorithms of Type II are based on evidence maximization of the hyper-parameter likelihood [25, 110].

In [25], the authors present empirical but comprehensive results that show the superiority of Type II algorithms. Among other things, they conclude that Type II algorithms perform consistently better compared with Type I algorithms, and Type II algorithms are more robust against the choice of hierarchically higher priors. For these reasons, this thesis considers Type II SBL algorithms.

In SBL or hierarchical models the parameter weights $\boldsymbol{w}$ depend on a hyper-parameter $\boldsymbol{\gamma} \in \mathbb{R}_+^N$ such that the prior PDF can be expressed as

$$p(\boldsymbol{w}|\boldsymbol{\gamma}) = \prod_{n=1}^{N} p(w_n|\gamma_n) = \prod_{n=1}^{N} \mathcal{N}(w_n, \gamma_n). \tag{3.1}$$

The prior of the hyper-parameter $p(\boldsymbol{\gamma})$ — the hyper-prior — can be defined by a power exponential scale mixture distribution [25]. In this thesis, the hyper-prior is chosen to be flat (deterministic), because this leads to efficient inference algorithms [107, 108]. Also, a flat hyper-prior performs better or equally well compared with other hyper-priors as pointed out in [25].

Assuming the error $\boldsymbol{e}$ is normal distributed, the measurements $\boldsymbol{y}$ in the model (2.3) can be expressed in terms of a Gaussian likelihood as

$$p(\boldsymbol{y}|\boldsymbol{w}) \propto \exp\left\{-\frac{1}{2}\left\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\right\|_{\boldsymbol{\Lambda}}^2\right\}, \tag{3.2}$$

with $\|\cdot\|_{\boldsymbol{\Lambda}}^2 = \cdot^T \boldsymbol{\Lambda} \cdot$ being the weighted ℓ_2-norm, $\boldsymbol{\Lambda} = \lambda \boldsymbol{I} \in \mathbb{R}^{M\times M}$, and $\boldsymbol{I} \in \mathbb{R}^{M\times M}$ being the identity matrix.

Then, in SBL the marginal likelihood is used to estimate $\boldsymbol{\gamma}$. The marginal likelihood is formulated as [60, 107, 108]

$$p(\boldsymbol{y}|\boldsymbol{\gamma}) = \int_{-\infty}^{\infty} p(\boldsymbol{y}|\boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{\gamma}) \mathrm{d}\boldsymbol{w} = |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\boldsymbol{y}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{y}\right\}, \tag{3.3}$$

where $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1} + \boldsymbol{\Phi}\boldsymbol{\Gamma}\boldsymbol{\Phi}^T$ and $\boldsymbol{\Gamma} = \mathrm{diag}\{\boldsymbol{\gamma}\}$. The logarithm of (3.3) yields the objective function

$$\mathscr{L}(\boldsymbol{\gamma}) = \log p(\boldsymbol{y}|\boldsymbol{\gamma}) = -\frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{y}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{y}. \tag{3.4}$$

To find the most likely solution, the objective function (3.4) is then maximized with respect to $\boldsymbol{\gamma}$:

$$\widehat{\boldsymbol{\gamma}} = \arg\max_{\boldsymbol{\gamma}} \mathscr{L}(\boldsymbol{\gamma}). \tag{3.5}$$

The result in (3.5) can then be used to approximate the posterior PDF of the parameter weights $p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{\gamma}) \propto \mathcal{N}(\widehat{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$. More specifically, given $\widehat{\boldsymbol{\gamma}}$, the posterior of the parameter weights is [106]

$$\widehat{\boldsymbol{w}} = \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{y}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\Phi} + \widehat{\boldsymbol{\Gamma}}^{-1}\right)^{-1}, \tag{3.6}$$

with $\widehat{\boldsymbol{\Gamma}} = \mathrm{diag}\{\widehat{\boldsymbol{\gamma}}\}$.

Because each parameter weight estimate $\widehat{w}_n$ is controlled by a corresponding $\widehat{\gamma}_n$, the estimate $\widehat{\gamma}_n$ influences whether $\widehat{w}_n$ is zero or not. This mechanism can be seen in (3.6), where $\widehat{\boldsymbol{\Gamma}}^{-1}$ is added to the diagonal of $\boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\Phi}$ and then $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ is used for calculating $\widehat{\boldsymbol{w}}$. If, on the one hand, $\widehat{\gamma}_n \to 0$, $\widehat{\gamma}_n^{-1}$ becomes very large. By adding the flat prior $\widehat{\gamma}_n^{-1}$ to the corresponding row in $\boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\Phi}$ in $\boldsymbol{\Sigma}_{\boldsymbol{w}}$, the calculation of $\widehat{w}_n$ leads then to $\widehat{w}_n \to 0$. On the other hand, if $\widehat{\gamma}_n \to \infty$, the corresponding n-th basis function in $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ becomes more relevant in the model. This leads to a non-zero $\widehat{w}_n$. Thus, large hyper-parameters indicate highly relevant basis functions. To this end, if many $\widehat{w}_n$ are estimated to be zero, the estimate $\widehat{\boldsymbol{w}}$ becomes sparse. Likewise, basis functions, which correspond to a zero weight $\widehat{w}_n$, can be excluded from the model. Additionally, by excluding basis functions from the estimation, the computational complexity of the estimation is reduced.

It is worth noting that the mechanism above leads also to a covariance matrix of the parameter weights $\boldsymbol{\Sigma}_{\boldsymbol{w}}$, which is later used in the thesis for exploration.

3.1.2. DISTRIBUTED SPARSE BAYESIAN LEARNING IN THE LITERATURE

This thesis uses SBL methods for a swarm, which is exploring an unknown spatially distributed physical process. Because the swarm consists of multiple agents, a distributed processing of the SBL algorithms is useful. There have been previous works, which showed distributed implementations of SBL methods. In [26] the authors presented a distributed solution of an SBL algorithm. The authors solved the problem for the parameter weights by using variational inference [60, Chap. 10], which was distributed by message passing and loopy belief propagation. However, if the message passing algorithms contain loops, which might happen for a connected swarm, such algorithms are

only guaranteed to converge under some conditions [111, 112]. Thus, this distribution scheme is not considered in this thesis.

For Type II approaches, the authors in [113] used joint sparsity models (JSMs), which assume that all distributed hyper-parameters in a network share the same support. The support of the hyper-parameter is furthermore assumed to be sparse. Using these assumptions, the authors proposed an EM algorithm to solve for the hyper-parameters. The M-Step in the EM algorithm can then be solved by a distributed ADMM algorithm. The benefit of this approach is that the authors showed in a following work [27] that the communication can be highly compressed, which reduces the communication load in the network. Another Type II approach, which uses JSM for distributed processing, is presented in [114]. There, the authors use variational Bayes techniques to estimate the hyper-parameter. The authors further look into quantization errors, which might occur due to the communication. Yet, the approaches in [113, 114] use an approximation to estimate the hyper-parameters from the distributed support. As this thesis is interested in the exact values, the approaches in [113, 114] are not considered here.

Because this thesis assumes Gaussian distributions for the parameter weights, it is possible to analytically solve for the likelihood and perform distributed processing. Especially, the FMLM algorithm presented in [106], which involves the maximization of the marginal likelihood, can be distributed. Also [108], which introduced the R-ARD algorithm, can be formulated in a distributed fashion. The following sections look into both algorithms.

3.1.3. FAST MARGINAL LIKELIHOOD MAXIMIZATION

For the derivation of the FMLM the objective function (3.4) is used. The FMLM decomposes $\boldsymbol{\gamma}$ into its components such that the estimation of the optimal hyper-parameter can be done component-wise. To prepare (3.4) for a component-wise optimization, the matrix inversion lemma [115, 116] is applied to $\boldsymbol{\Sigma}^{-1}$ in (3.4) to isolate the influence of the n-th basis function on $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}_{\overline{n}}^{-1} + \frac{\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}}{\gamma_n^{-1} + \boldsymbol{\phi}_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{\phi}_n}, \tag{3.7}$$

where $\boldsymbol{\Sigma}_{\overline{n}}^{-1} = (\boldsymbol{\Lambda}^{-1} + \sum_{i\neq n}\gamma_i\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T)^{-1}$ [107] is the covariance of the marginal likelihood with the n-th basis function removed.

Inserting (3.7) into (3.4) yields

$$\mathscr{L}(\boldsymbol{\gamma}) = \log\boldsymbol{\Sigma}_{\overline{n}}^{-1} - \boldsymbol{y}^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{y} - \log\gamma_n - \log(\gamma_n^{-1} + \boldsymbol{\phi}_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{\phi}_n) + \frac{(\boldsymbol{\phi}_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{y})^2}{\gamma_n^{-1} + \boldsymbol{\phi}_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{\phi}_n}, \tag{3.8}$$

which can be simplified by using the definitions

$$s_n = \phi_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{\phi}_n, \qquad q_n = \phi_n^T\boldsymbol{\Sigma}_{\overline{n}}^{-1}\boldsymbol{y} \tag{3.9}$$

such that

$$\begin{aligned}\mathscr{L}(\boldsymbol{\gamma}) &= \mathscr{L}(\boldsymbol{\gamma}_{\overline{n}}) - \log\gamma_n - \log(\gamma_n^{-1} + s_n) + \frac{q_n^2}{\gamma_n^{-1} + s_n} \\ &= \mathscr{L}(\boldsymbol{\gamma}_{\overline{n}}) + l(\gamma_n).\end{aligned} \tag{3.10}$$

Algorithm 3 FMLM

$\gamma_n = 0,\ \forall n = 1,\dots,N$
Choose any $n = 1,\dots,N$
$\gamma_n = \frac{\|\boldsymbol{\phi}_n^T \mathbf{y}\| / \|\boldsymbol{\phi}_n\|^2 - \lambda^{-1}}{\|\boldsymbol{\phi}_n\|^2}$
$I = 1$ and only the chosen γ_n is in the model
while not converged **do**
  for $n \in \{1,\dots,N\}$ **do**
    $s_n, q_n \leftarrow$ (3.9)
    if $q_n^2 > s_n$ and γ_n is in the model **then** ▷ Update
      $\gamma_n = \frac{q_n^2 - s_n}{s_n^2}$
    else if $q_n^2 > s_n$ and γ_n is not in the model **then** ▷ Add
      Add n-th basis function to the model
      $\gamma_n = \frac{q_n^2 - s_n}{s_n^2}$
      $I = I + 1$
    else if $q_n^2 \leq s_n$ and γ_n is in the model **then** ▷ Remove
      $\gamma_n = 0$
      remove n-th basis function from the model
      $I = I - 1$
    $\boldsymbol{\Sigma} \leftarrow \left(\boldsymbol{\Lambda}^{-1} + \sum_{i=1}^{I} \gamma_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T\right)^{-1}$
  Check for convergence
$\widehat{\boldsymbol{w}}, \boldsymbol{\Sigma}_w \leftarrow$ (3.6)

With γ_n separated, the FMLM algorithm computes now the hyper-parameter γ_n iteratively by maximizing the marginalized likelihood (3.10) with respect to γ_n. It can be shown that the marginalized likelihood (3.3) is maximized with respect to γ_n at

$$\widehat{\gamma}_n = \begin{cases} \frac{q_n^2 - s_n}{s_n^2}, & \text{if } q_n^2 > s_n, \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

As mentioned before, if $\widehat{\gamma}_n = 0$, the corresponding n-th basis function is excluded from the model, c.f. Alg. 3.

Before the algorithm commences, any γ_n is initialized as

$$\gamma_n = \frac{\|\boldsymbol{\phi}_n^T \mathbf{y}\| / \|\boldsymbol{\phi}_n\|^2 - \lambda^{-1}}{\|\boldsymbol{\phi}_n\|^2}, \tag{3.12}$$

and the rest is set to zero. Then, using (3.9) and (3.11), the algorithm iteratively builds up an estimate $\widehat{\boldsymbol{\gamma}}$. It starts with only a single hyper-parameter and chooses to add or delete basis functions during the estimation. The algorithm is summarized in Alg. 3, with $I \in \mathbb{N}$ representing the number of currently used basis functions in the model.

3.1.4. Automatic Relevance Determination

In [108], the authors introduced the R-ARD by using an auxiliary function that upper bounds the objective (3.4). The auxiliary function permits then to use algorithms for convex optimization, which makes the optimization easier. Here, the auxiliary function also enables for a distributed optimization, which will be derived later on.

The authors in [70, Chap 3] show that the log-determinant in (3.4) is concave in $\boldsymbol{\gamma}$. The authors in [108] use the concavity of (3.4) to construct an upper-bounding hyperplane with the help of dual functions.

The dual function of a concave function is a convex function, and, hence, the dual of the log-determinant in (3.4) is a convex function as well. More specifically,

$$\log|\boldsymbol{\Sigma}| = \min_{\boldsymbol{z}} \boldsymbol{z}^T\boldsymbol{\gamma} - h^*(\boldsymbol{z}), \tag{3.13}$$

with $\boldsymbol{z} \in \mathbb{R}^N$ being the dual variable and $h^*(\boldsymbol{z})$ being the Fenchel conjugate of $\log|\boldsymbol{\Sigma}|$. The dual function does not need to be calculated in this derivation, as will be shown shortly, however for more knowledge about the duality of functions, the reader is referred to [70, Chap. 5] or [117].

With (3.13) an upper bounding function of (3.4) is defined as

$$\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{z}) \triangleq \boldsymbol{z}^T\boldsymbol{\gamma} - h^*(\boldsymbol{z}) + \boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y} \geq \mathscr{L}(\boldsymbol{\gamma}). \tag{3.14}$$

This upper bounding function becomes tight, for any fixed $\boldsymbol{\gamma}$, if minimized over $\boldsymbol{z}$.

In [108], the authors introduced the R-ARD algorithm, which iteratively estimates $\boldsymbol{z}$ and $\boldsymbol{\gamma}$. As already said, $h^*(\boldsymbol{z})$ does not need to be explicitly known for the optimization of $\boldsymbol{z}$ because its optimal $\boldsymbol{z}$ is obtained by the slope of $\log|\boldsymbol{\Sigma}|$ for a fixed $\boldsymbol{\gamma} = \widehat{\boldsymbol{\gamma}}$ [70, 108]. This can be computed as

$$\widehat{\boldsymbol{z}} = \frac{\partial}{\partial \boldsymbol{\gamma}} \log|\boldsymbol{\Sigma}| = \operatorname{diag}\left\{\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\right\}. \tag{3.15}$$

Therefore, $\widehat{\boldsymbol{z}}$ can be calculated by (3.15) and assumed to be fixed for the following step. Next, the optimal $\boldsymbol{\gamma}$ has to be found by minimizing (3.14) with respect to $\boldsymbol{\gamma}$ for $\boldsymbol{z} = \widehat{\boldsymbol{z}}$, which is formulated as

$$\widehat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} \mathscr{L}(\boldsymbol{\gamma}, \widehat{\boldsymbol{z}}) = \arg\min_{\boldsymbol{\gamma}} \widehat{\boldsymbol{z}}^T\boldsymbol{\gamma} + \boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y}, \tag{3.16}$$

where $h^*(\boldsymbol{z})$ has been neglected as it does not depend on $\boldsymbol{\gamma}$. The algorithm is summarized in Alg. 4.

3.2. Distributed Sparse Bayesian Learning for Homogeneous Splitting

This section presents the derivation of distributed SBL algorithms for homogeneous learning, which is also referred to as SOE, where through cooperation, each agent gets the same estimate. The two algorithms FMLM [106] and R-ARD [108] are now exploited to develop a distributed version of them for SOE. Then, these two algorithms are tested for the scenario of swarm exploration. The derivations are published in [118] and [54], respectively.

Algorithm 4 R-ARD

1: $\widehat{z}_n \leftarrow 1,\ \forall n = 1,\dots,N$
2: **while** not converged **do**
3: $\quad \widehat{\boldsymbol{\gamma}} \leftarrow \arg\min_{\boldsymbol{\gamma}} \mathscr{L}(\boldsymbol{\gamma}, \widehat{\boldsymbol{z}})$ ▷ Bear in mind that actually $\widehat{\boldsymbol{z}}(\boldsymbol{\gamma})$, but it's fixed here
4: $\quad \widehat{\boldsymbol{z}} \leftarrow$ (3.15)
5: $\quad$ Check for convergence
6: $\widehat{\boldsymbol{w}} \leftarrow \boldsymbol{\Sigma}_w \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{y}$

3.2.1. DISTRIBUTED FAST MARGINAL LIKELIHOOD MAXIMIZATION

This section presents the derivation of the distributed FMLM (DFMLM). Essentially the DFMLM is a reformulation of the FMLM into a distributed setting.

When looking back to the centralized FMLM, it can be seen that the estimation of $\widehat{\gamma}_n$ depends on the values of s_n and q_n. Yet, these two values are difficult to calculate because they require to create $\boldsymbol{\Sigma}_n^{-1}$ for all $n = 1,\dots,N$ in each iteration. The authors in [106] redefined s_n and q_n as

$$s_n = \frac{\gamma_n^{-1} S_n}{\gamma_n^{-1} - S_n}, \qquad q_n = \frac{\gamma_n^{-1} Q_n}{\gamma_n^{-1} - S_n}, \tag{3.17}$$

with the new terms S_n and Q_n, which are defined as

$$S_n = \boldsymbol{\phi}_n^T \boldsymbol{\Lambda} \boldsymbol{\phi}_n - \boldsymbol{\phi}_n^T \boldsymbol{\Lambda} \boldsymbol{\Phi} \boldsymbol{\Sigma}_w \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\phi}_n, \tag{3.18}$$

$$Q_n = \boldsymbol{\phi}_n^T \boldsymbol{\Lambda} \boldsymbol{y} - \boldsymbol{\phi}_n^T \boldsymbol{\Lambda} \boldsymbol{\Phi} \boldsymbol{\Sigma}_w \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{y}. \tag{3.19}$$

The terms S_n and Q_n can be seen as an intermediate result to calculate (3.17). This thesis exploits now S_n and Q_n to derive a distributed version of the FMLM. For SOE, every agent k knows locally its design matrix $\boldsymbol{\Phi}_k$ and its measurements $\boldsymbol{y}_k$. Thus, for a distributed computation, this information needs to be distributed in the network.

Therefore, the following is defined

$$\boldsymbol{D} \triangleq \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\Phi} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{\Phi}_k, \tag{3.20}$$

$$\boldsymbol{c} \triangleq \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{y} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{y}_k. \tag{3.21}$$

Both, (3.20) and (3.21), can be computed by an averaged consensus algorithm [15, 19, 20, 119]. By defining $\boldsymbol{\epsilon}_n \triangleq [\dots,0,1,0,\dots]^T \in \mathbb{R}^N$ as a vector with only zeros except on the n-th position $\epsilon_n = 1$, (3.18) and (3.19) can be reformulated according to the newly defined consensus terms as

$$S_n = \boldsymbol{\epsilon}_n^T (\boldsymbol{D} - \boldsymbol{D} \boldsymbol{\Sigma}_w \boldsymbol{D}) \boldsymbol{\epsilon}_n, \tag{3.22}$$

$$Q_n = \boldsymbol{\epsilon}_n^T (\boldsymbol{c} - \boldsymbol{D} \boldsymbol{\Sigma}_w \boldsymbol{c}). \tag{3.23}$$

The matrix $\boldsymbol{D}$ can also be used to calculate $\boldsymbol{\Sigma}_w$ as

$$\boldsymbol{\Sigma}_w = (\boldsymbol{D} + \boldsymbol{\Gamma}^{-1})^{-1}. \tag{3.24}$$

Algorithm 5 DFMLM

1: Distribute $\boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{\Phi}_k$ and $\boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{y}_k$ to the neighbors to compute consensus variables.
2: $\boldsymbol{D} \leftarrow$ (3.20)
3: $\boldsymbol{c} \leftarrow$ (3.21)
4: $\gamma_n \leftarrow$ (3.25), $\forall n = 1, \ldots, N$
5: Set all $\gamma_n = 0$ except the largest.
6: **while** not converged **do**
7: **for** $n \in \{1, \ldots, N\}$ **do**
8: $S_n \leftarrow$ (3.22)
9: $Q_n \leftarrow$ (3.23)
10: $s_n, q_n \leftarrow$ (3.17)
11: **if** $q_n^2 > s_n$ and γ_n is in the model **then** ▷ Update
12: $\gamma_n = \frac{q_n^2 - s_n}{s_n^2}$
13: **else if** $q_n^2 > s_n$ and γ_n is not in the model **then** ▷ Add
14: Add n-th basis function to the model
15: $\gamma_n = \frac{q_n^2 - s_n}{s_n^2}$
16: **else if** $q_n^2 \leq s_n$ and γ_n is in the model **then** ▷ Remove
17: $\gamma_n = 0$
18: remove n-th basis function from the model
19: $\boldsymbol{\Sigma}_w \leftarrow$ (3.24)
20: Check for convergence
21: $\widehat{\boldsymbol{w}} \leftarrow \boldsymbol{\Sigma}_w \boldsymbol{c}$

The hyper-parameters $\boldsymbol{\Gamma}^{-1}$ is the same for all agents because it can be initialized with (3.20) and (3.21), as shown in the following paragraph. The algorithm computes locally the variables $\boldsymbol{\Sigma}_w$ and S_n and Q_n for $n = 1, \ldots, N$, followed by s_n and q_n for $n = 1, \ldots, N$. It then iteratively goes through all components of $\boldsymbol{\gamma}$ and checks if the component is updated, added, or removed. For the k-th agent, the DFMLM is summarized in Alg. 5, where the lines 6 until 20 basically implement the procedure of the FMLM algorithm, see Alg. 3, but all calculations depend on the consensus values.

INITIALIZATION OF DFMLM

The initialization of γ_n as introduced in [106] can be reformulated such that it can be computed with $\boldsymbol{D}$ and $\boldsymbol{c}$ as

$$\begin{aligned} \gamma_n &= \frac{\|\boldsymbol{\phi}_n^T \boldsymbol{y}\| / \|\boldsymbol{\phi}_n\|^2 - \lambda^{-1}}{\|\boldsymbol{\phi}_n\|^2} = \lambda \frac{|\lambda \boldsymbol{\phi}_n^T \boldsymbol{y}| / (\lambda \boldsymbol{\phi}_n^T \boldsymbol{\phi}_n) - \lambda^{-1}}{\lambda \boldsymbol{\phi}_n^T \boldsymbol{\phi}_n} \\ &= \lambda \frac{|c_n| / (D_{n,n}) - \lambda^{-1}}{D_{n,n}}, \end{aligned} \tag{3.25}$$

where $D_{n,n} = \boldsymbol{\epsilon}_n^T \boldsymbol{D} \boldsymbol{\epsilon}_n$ and $c_n = \boldsymbol{\epsilon}_n^T \boldsymbol{c}$. This way all values of $\boldsymbol{\gamma}$ can be initialized by means of the consensus terms (3.20) and (3.21). Once γ_n is computed for all $n = 1, \ldots, N$, each agent chooses the highest γ_n because this γ_n contributes the most. The other values are then set to zero. This ensures that all agents have initially the same $\boldsymbol{\gamma}$.

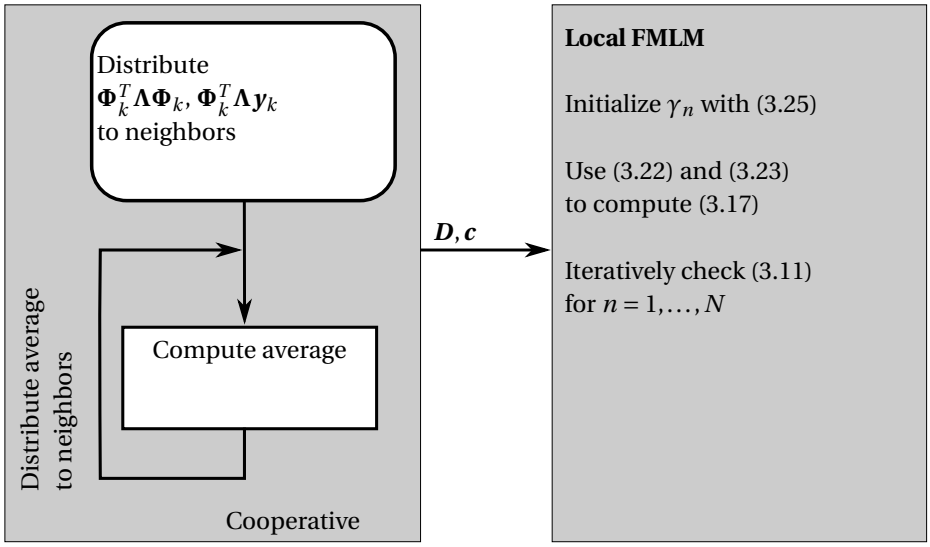


Figure 3.1: The DFMLM requires only an initial communication step. Afterward, every agent k basically computes a local FMLM with the consensus values defined in (3.20) and (3.21).

CONVERGENCE CRITERION OF DFMLM

If between two iterations (in Alg. 5 an iteration starts in line 7 and ends in line 19) $\boldsymbol{\gamma}$ is not changing significantly within some tolerance, then the algorithm is converged. Because every agent locally computes an FMLM, but with $\boldsymbol{D}$ and $\boldsymbol{c}$, the DFMLM algorithm enjoys the same convergence properties as the FMLM. In [120], the authors showed that the optimization in the FMLM is equivalent to a coordinate-wise minimization of the convex upper bound (3.14) of the original cost function (3.4). This ensures the convergence to the minimizer.

COMMUNICATION LOAD FOR DFMLM

Figure 3.1 shows the DFMLM algorithm as a flowchart. Due to (3.20) and (3.21), each agent basically computes a local FMLM. Looking at Fig. 3.1, DFMLM has only one consensus step at its beginning. The agents accumulate the information in the network for estimating $\widehat{\boldsymbol{\gamma}}$. Regarding the communication load, each agent transmits N^2 floats for $\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k$ and N floats for $\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{y}_k$. The matrix $\boldsymbol{D}$ however is symmetric, which can be exploited for communication such that only $\frac{N(N+1)}{2}$ values have to be transmitted per consensus step. Hence, in each communication step, $\frac{N(N+1)}{2} + N$ values are communicated per agent per consensus step. It is worth noting that distributing $\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k$ and $\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{y}_k$ is equivalent to sharing all data, albeit in an indirect fashion. Later in this section, the communication load of the DFMLM and distributed version of R-ARD is compared with each other for different network connectivities.

3.2.2. Distributed Automatic Relevance Determination

This section shows how SOE can be applied for SBL with R-ARD and Type II optimization. This thesis makes use of the research conducted in [108]. Currently, R-ARD is only applicable to a centralized system; all the data has to be available at a single entity. For a distributed formulation, [108, Lemma 2] is useful, which shows the convexity of (3.14) in $\boldsymbol{\gamma}$ for a fixed $\boldsymbol{z}$, which results in (3.16) to estimate $\widehat{\boldsymbol{\gamma}}$.

In [108], the authors show how to express $\boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y}$ as a minimization over $\boldsymbol{w}$

$$\boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y} = \min_{\boldsymbol{w}} \lambda\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \sum_{n=1}^{N}\frac{w_n^2}{\gamma_n}. \tag{3.26}$$

This re-expression leads to an upper-bounding auxiliary function, when (3.26) is inserted into (3.16)

$$\mathscr{L}(\boldsymbol{\gamma},\boldsymbol{w},\widehat{\boldsymbol{z}}) \triangleq \lambda\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \sum_{n=1}^{N}\widehat{z}_n\gamma_n + \frac{w_n^2}{\gamma_n} \geq \mathscr{L}(\boldsymbol{\gamma},\widehat{\boldsymbol{z}}). \tag{3.27}$$

Now, for any fixed $\boldsymbol{w}$ and a fixed $\widehat{\boldsymbol{z}}$, the estimate for $\boldsymbol{\gamma}$ is $\widehat{\gamma}_n = |\widehat{w}_n|/\sqrt{\widehat{z}_n},\ \forall n = 1,\ldots,N$. Inserting this estimate into (3.27), leads to only a function dependent on $\boldsymbol{w}$ for a fixed $\widehat{\boldsymbol{z}}$. It can be optimized such that

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{2}{\lambda}\sum_{n=1}^{N}\sqrt{\widehat{z}_n}|w_n|. \tag{3.28}$$

Now, the function (3.27) can be minimized first over $\boldsymbol{\gamma}$ and $\boldsymbol{w}$ and then $\widehat{\boldsymbol{z}}$ is updated according to (3.15). The question now is how to reformulate (3.28) and (3.15) such that they suit a distributed computation for $\widehat{\boldsymbol{w}}$ and $\widehat{\boldsymbol{z}}$.

1. Expression (3.28) can be distributed similar to (2.17). The difference between (2.17) and (3.28) is that each parameter weight in the ℓ_1-norm is individually weighted. Thus,

$$\mathscr{L}(\boldsymbol{w},\widehat{\boldsymbol{z}}) = \sum_{k=1}^{K}\left\|\boldsymbol{\Phi}_k(\boldsymbol{X}_k)\boldsymbol{w}-\boldsymbol{y}_k\right\|_2^2 + \frac{2}{\lambda}\sum_{n=1}^{N}\sqrt{\widehat{z}_n}|w_n|. \tag{3.29}$$

   This cost function can be distributively minimized with respect to $\boldsymbol{w}$ by means of an ADMM algorithm, see Sec 2.4.2.

2. For a distributed computation of $\widehat{\boldsymbol{z}}$, the Woodbury identity is applied to $\boldsymbol{\Sigma}^{-1}$ as

$$\boldsymbol{\Sigma}^{-1} = \left(\boldsymbol{\Lambda}^{-1}+\boldsymbol{\Phi}\boldsymbol{\Gamma}\boldsymbol{\Phi}^T\right)^{-1} = \boldsymbol{\Lambda}-\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}, \tag{3.30}$$

   with $\boldsymbol{\Sigma}_w = (\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}+\boldsymbol{\Gamma}^{-1})^{-1}$. Inserting then (3.30) and (3.20) into (3.15), leads to

$$\widehat{\boldsymbol{z}} = \operatorname{diag}\left\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}-\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\right\} = \operatorname{diag}\{\boldsymbol{D}-\boldsymbol{D}\boldsymbol{\Sigma}_w\boldsymbol{D}\}, \tag{3.31}$$

   where $\boldsymbol{\Sigma}_w = (\boldsymbol{D}+\boldsymbol{\Gamma}^{-1})^{-1}$ and $\boldsymbol{D}$ is computed distributively by consensus.

3. The hyper-parameter $\widehat{\boldsymbol{\gamma}}$ follows from the distributively estimated $\widehat{\boldsymbol{z}}$ and $\widehat{\boldsymbol{w}}$ as $\widehat{\gamma}_n = |\widehat{w}_n|/\sqrt{\widehat{z}_n},\ \forall n = 1,\ldots,N$.

Now, all quantities can be calculated distributively, and the resulting distributed reformulated ARD (D-R-ARD) algorithm for SOE is shown in Alg. 6.

Algorithm 6 D-R-ARD for SOE

$\widehat{z}_n \leftarrow 1,\ \forall n = 1,\dots,N$
Distribute $\boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{\Phi}_k$ to all neighbors
$\boldsymbol{D} \leftarrow$ (3.20)
while not converged **do**
  $\widehat{\boldsymbol{w}} \leftarrow \arg\min_{\boldsymbol{w}} \mathscr{L}(\boldsymbol{w}, \widehat{\boldsymbol{z}})$ ▷ Can be solved distributively with ADMM (2.24)
  $\widehat{\boldsymbol{\gamma}} \leftarrow \frac{|\widehat{w}_n|}{\sqrt{\widehat{z}_n}},\ \forall n = 1,\dots,N$
  $\boldsymbol{\Sigma}_w \leftarrow (\boldsymbol{D} + \widehat{\boldsymbol{\Gamma}}^{-1})^{-1}$
  $\widehat{\boldsymbol{z}} \leftarrow$ (3.31)
  Check for convergence
$\widehat{\boldsymbol{w}} \leftarrow \boldsymbol{\Sigma}_w \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{y}$

CONVERGENCE OF D-R-ARD

The D-R-ARD algorithm is optimizing the convex upper bound $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \boldsymbol{z}) \geq \mathscr{L}(\boldsymbol{\gamma})$. It is therefore reasonable that the $\boldsymbol{\gamma}$ between two consecutive iterations is used as a stopping criterion. Thus, if the difference of these two $\boldsymbol{\gamma}$ values is below some threshold, the algorithm stops and is considered here to be converged. The convergence of the R-ARD is discussed in [108], and the convergence of the ADMM in [18].

COMMUNICATION LOAD OF D-R-ARD

In the D-R-ARD, first $\boldsymbol{D}$ has to be calculated with a consensus algorithm. Therefore, each agent transmits N^2 floats for $\boldsymbol{\Phi}_k^T \boldsymbol{\Lambda} \boldsymbol{\Phi}_k$ per consensus step. Exploiting the symmetry of $\boldsymbol{D}$ the transmitted floats can be reduced further to $N(N+1)/2$ per consensus step. However, $\boldsymbol{D}$ does not change during the estimation and has to be calculated only once. Second, for the distributed estimation of $\widehat{\boldsymbol{w}}$ with an ADMM algorithm, it is required to distribute the intermediate parameter estimates for $\boldsymbol{w}$. This leads to $N i_{\text{ADMM}}$ additional floats per D-R-ARD iteration, where $i_{\text{ADMM}} \in \mathbb{N}$ is the number of ADMM iterations. Thus, assuming $i_{\text{DRARD}} \in \mathbb{N}$ iterations for the D-R-ARD algorithm, $N i_{\text{ADMM}} i_{\text{DRARD}}$ floats have to be transmitted as well. So far, no network topology is assumed. The communication load for different network topologies is discussed in the following section.

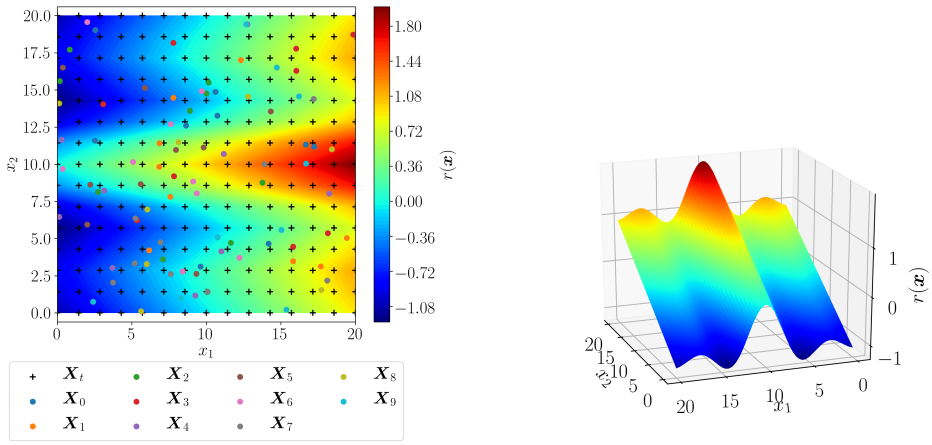
3.2.3. SIMULATIONS FOR DISTRIBUTED SBL AND HOMOGENEOUS LEARNING WITH ARTIFICIAL DATA

The distributed algorithms, mentioned above, are now evaluated against each other and against their centralized versions. This section evaluates two aspects of the algorithms: their performance in terms of convergence, robustness, and sparsity and their communication complexity. For the first evaluation a test function is used, as in [106]. The two dimensional test function is defined as

$$r(\boldsymbol{x}) \triangleq \operatorname{sinc}(x_1) + 0.1 x_2, \tag{3.32}$$

which is shown in Fig. 3.2.

The test function (3.32) is then sampled at M positions $\boldsymbol{X} \in \mathbb{R}^{M \times 2}$. These samples are then distributed to $K = 10$ computing nodes, where each computing node uses M_k



(a) (b)

Figure 3.2: Test function as described in [106]. (a) This figure exemplifies how the measurement positions are assigned to $K = 10$ agents ($\boldsymbol{X}_k,\ k = 1,\dots,K$). A black plus sign indicates a center position in the model $\boldsymbol{\mu}$. The same positions are used as test positions $\boldsymbol{X}_t$ for the evaluation. (b) 3D representation of the data for better visualization.

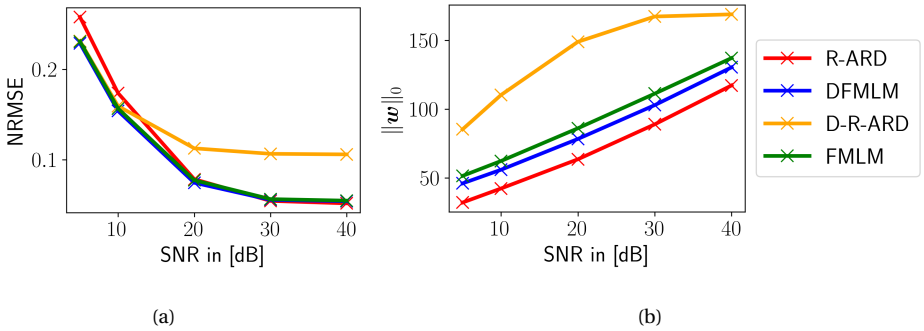
samples at $\boldsymbol{X}_k \in \mathbb{R}^{M_k \times 2}$ positions and thus $\boldsymbol{X} = [\boldsymbol{X}_1^T,\dots,\boldsymbol{X}_K^T]^T$. Effectively, any node k has $M_k = 9$ measurements, a design matrix $\boldsymbol{\Phi}_k(\boldsymbol{X}_k)$ with $N = 225$ Gaussian basis functions ($\sigma_n = 1.5$, a regular grid of center points $\boldsymbol{\mu}_n, n = 1,\dots,N$), and N parameter values, see (2.16). The distribution of the sampling positions reflects the fact that agents move, make measurements, and estimate before they continue to explore, see Fig. 2.1. The parameter ρ in the distributed LASSO (DLASSO) of the D-R-ARD is set to $\rho = 2.5$. It is also assumed that the connection between all agents is synchronous and stable.

Performance of DFMLM and D-R-ARD

For evaluating the performance, it is assumed that the initial consensus for both algorithms has converged. Therefore, for D-R-ARD, every node has an estimate of $\boldsymbol{D}$ and, for DFMLM, every node has an estimate of $\boldsymbol{D}$ and $\boldsymbol{c}$. As evaluation metric, the normalized root mean square error (NRMSE) is used. Therefore, defining $\boldsymbol{r} = [r(\boldsymbol{x}_1),\dots,r(\boldsymbol{x}_{M_t})]^T \in \mathbb{R}^{M_t}$ as a vector which contains M_t values from the observed function $r(\boldsymbol{x})$, the NRMSE is defined as

$$e_{\text{NRMSE}} = \frac{\|\boldsymbol{r} - \boldsymbol{\Phi}(\boldsymbol{X}_t)\widehat{\boldsymbol{w}}\|}{\|\boldsymbol{r}\|}, \tag{3.33}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{M_t \times 2}$ are test positions where the estimated model $\boldsymbol{\Phi}(\boldsymbol{X}_t)\widehat{\boldsymbol{w}}$ is evaluated at. Here $N = M_t = 225$ and the test positions equal the center positions $\boldsymbol{\mu}_n,\ n = 1,\dots,N$, see Fig. 3.2(a). The NRMSE is used because of two properties. First, at the beginning of the iteration, when the estimated model is zero everywhere, the NRMSE results in an error of one. From there on, the NRMSE goes towards zero. If the estimated model is an exact representation of the underlying function, the NRMSE is zero.



(a) (b)

Figure 3.3: (a) shows the NRMSE with respect to the SNR for the test function (3.32). (b) displays the sparsity of each algorithm for approximating (3.32).

Figure 3.3(a) displays the NRMSE for the defined setup, for different SNR values, and averaged over 100 Monte Carlo runs. This way the robustness against noise is tested for the algorithms. Figure 3.3(a) shows that the performance of all algorithms is almost equal. Yet, for D-R-ARD, the performance is worse compared to the other algorithms because of the distributed reweighted ℓ_1 optimization. The optimization in D-R-ARD is solved with a distributed ADMM — DLASSO [98] in particular — which has a penalty parameter ρ for the involved augmented Lagrangian. The penalty parameter is data and SNR dependent, and for this simulations it was fixed. The influence of ρ and how to choose it is discussed in [121–123].

As the goal of SBL is to yield also a sparse result, the sparsity of each algorithm is evaluated in Fig. 3.3(b). The algorithms that are based on the marginal likelihood maximization have a high sparsity, which is almost equal. The difference in sparsity comes from different noise realizations. The D-R-ARD has the most non-zero components. This difference stems again from the DLASSO and ρ, see line 5 in Alg. 1, which is used here to estimate $\widehat{\boldsymbol{w}}$. Depending on the algorithm to solve the reweighted ℓ_1 optimization for estimating $\widehat{\boldsymbol{w}}$, the result might be sparser. In case of the D-R-ARD, the augmented Lagrangian introduces a smoothness term into the optimization, which is controlled by the augmented Lagrangian penalty parameter. This smoothness term leads to less sparsity compared to FMLM and DFMLM. The R-ARD solves the re-weighted ℓ_1-optimization with a centralized FISTA algorithm, which yields the highest sparsity.

The sparsity and the estimate of each algorithm are also shown in Fig. 3.4. There, it can be seen that the D-R-ARD estimates parameter weights, which correspond to basis functions with spatially close center positions. This might result from the ADMM parameter ρ, which controls the influence of the augmented Lagrangian. For the FMLM and the DFMLM the center positions of the basis functions are spatially more separated, which is desired for the SBL algorithm. Consequently the DFMLM leads to sparser results compared with the D-R-ARD.

	communication load before the estimation per consensus step	communication load per ADMM iteration
DFMLM	$\frac{N(N+1)}{2} + N$	-
D-R-ARD	$\frac{N(N+1)}{2}$	$i_{\text{ADMM}} N$

Table 3.1: Communication complexity per agent for the distributed algorithms D-R-ARD and DFMLM.

COMMUNICATION LOAD FOR DFMLM AND D-R-ARD

Communication complexity is certainly an important aspect for a distributed algorithm. The communication complexity of the algorithms explained above is summarized in Tab. 3.1. In a *strongly connected* network the consensus converges within a single iteration because each node is connected with each other. Therefore, this can be seen as a lower bound on the amount of data that needs to be transmitted for the particular problems considered here. If the network structure is only *connected* (see Def. 3), the consensus algorithm needs additional iterations to converge. The convergence of different consensus algorithms is discussed in [20, 119]. To evaluate the communication load, this thesis assumes three different types of connected networks — low, intermediate, and high. In these considered networks each node can only communicate with nodes within a certain range $r > 0\,\text{m}$. For an area, which is 20 m by 20 m large, the connectivity is considered low for $r = 4\,\text{m}$, intermediate for $r = 8\,\text{m}$, and high for $r = 13\,\text{m}$. All nodes within the range r build a neighborhood. Figs. 3.5(a), 3.5(b), and 3.5(c) present examples of such networks for $K = 30$. For this analysis, this thesis assumes that each network structure cannot change over time, i.e. any node cannot move itself such that it could increase the number of nodes in its neighborhood. There are also works that optimize the network structure to yield a better connectivity [124]. However, this is here not considered. So, it is assumed that the agents collect measurements prior to the initialization of the estimation. For homogeneous splitting the number of measurements M is irrelevant for the communication load, but it was always set to $M = 90$ and the measurements were evenly distributed among the agents, i.e., $M_k = M/K$. The number of parameter weights is the same as before namely $N = 225$.

The following simulations are averaged over 10 runs for each number of nodes $K = \{10, 15, 30, 45\}$ and communication ranges $r = \{4, 8, 13\}$. One run is the sum of all transmitted floats for achieving consensus in an estimation. Figure 3.6 shows the averaged communication load for each connectivity value and for the DFMLM and the D-R-ARD algorithm with varying values for K. There, it can be observed that the DFMLM generally requires fewer floats to be transmitted compared with the D-R-ARD. Also, if the connectivity increases, which corresponds to a more connected network, the overall communication load is reduced for both algorithms. Indeed, the distribution of $\boldsymbol{D}$ dominates the communication load for both algorithms. Thus, the curves appear almost parallel.

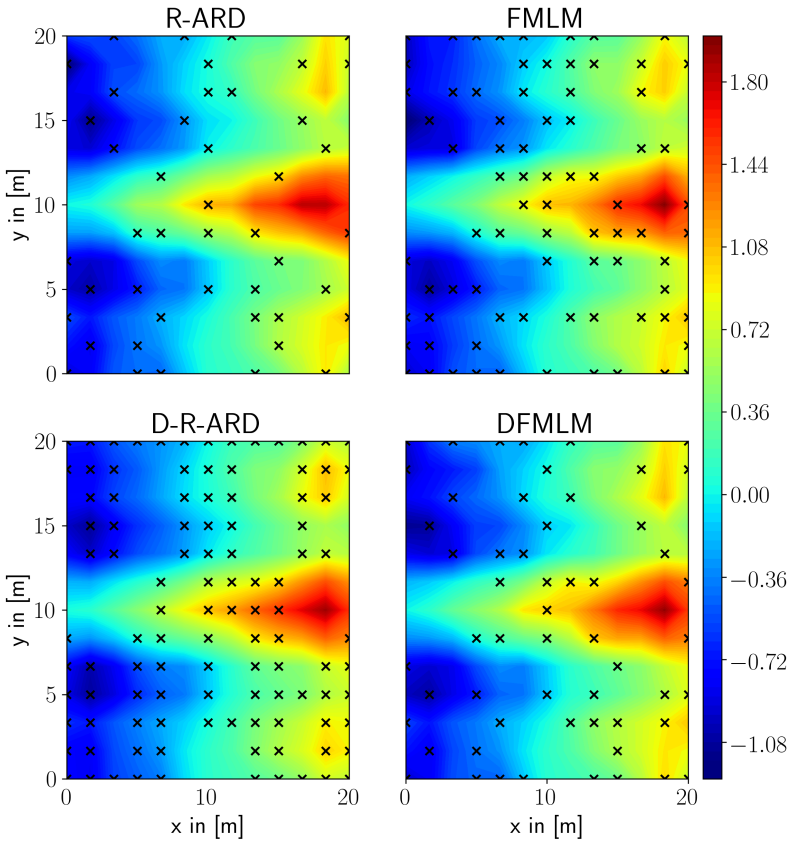
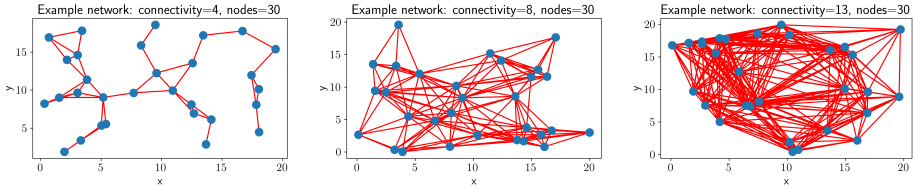


Figure 3.4: Estimates of a particular run with an SNR of 10 dB. The black crosses indicate the center positions of the selected Gaussian basis functions, which correspond to a non-zero weight.



(a) (b) (c)

Figure 3.5: Different network topologies. (a) network topology for $r = 4\,\text{m}, K = 30$. (b) network topology for $r = 8\,\text{m}, K = 30$. (c) network topology for $r = 13\,\text{m}, K = 30$. All topologies are created within a constrained area, which is 20 m by 20 m large.

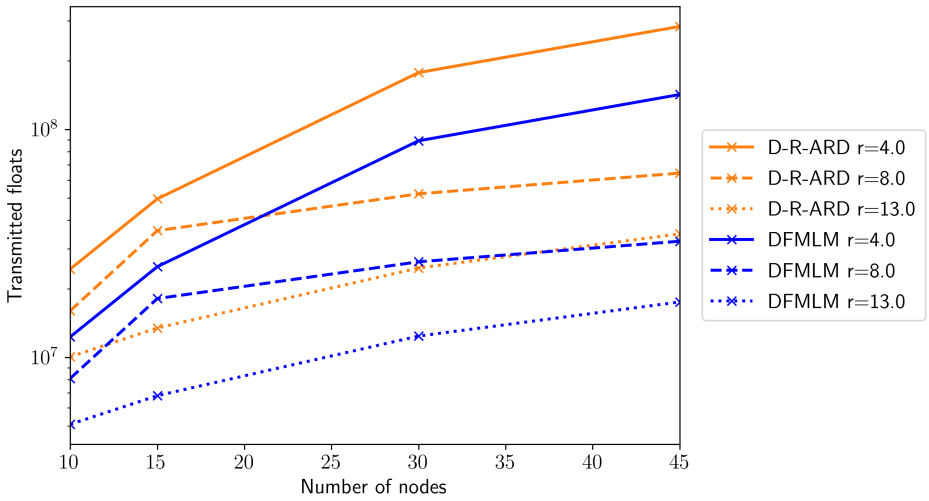


Figure 3.6: Communication load analysis of DFMLM and D-R-ARD for a varying number of nodes K and communication ranges r. The communication load is represented as the number of floats that are transmitted per node.

3.2.4. Simulations for Distributed SBL and Homogeneous Learning with Real Data

The current section analyzes the DFMLM and D-R-ARD for real data with added AWGN with different SNR. In particular, the magnetic field intensity of the Holodeck laboratory in the German Aerospace Center (DLR) is used. The magnetic field is exemplified in Fig. 3.7. For this simulation, $K = 10$ agents are considered. The $M = 240$ measurements are sampled uniformly at random and are then randomly distributed to the agents such that each has $M_k = 24$ measurements. The number of basis functions and parameter weights is set to $N = 560$. As basis functions Gaussians are considered with a width set to $\sigma_n = 0.25$.

Likewise to the previous simulation, the NRMSE is used as a performance metric with $M_t = 560$ test positions. The performance of each algorithm is shown in Figs. 3.8(a) and 3.8(b). Figure 3.8(a) displays the NRMSE for the proposed algorithms and benchmark algorithms. The R-ARD performs worse for lower SNR values, whereas the FMLM and DFMLM perform equally better.

The sparsity of each algorithm for this particular dataset is shown in Fig. 3.8(b). Again, the R-ARD yields the lowest sparsity, which might again result from the centralized FISTA algorithm. The sparsity for the FMLM and DFMLM algorithms is the same, as expected. The D-R-ARD algorithm uses for higher SNR all possible basis functions to approximate the magnetic field. This can again result from the parameter ρ which influences the solution of the parameter weights and which does not promote sparsity.

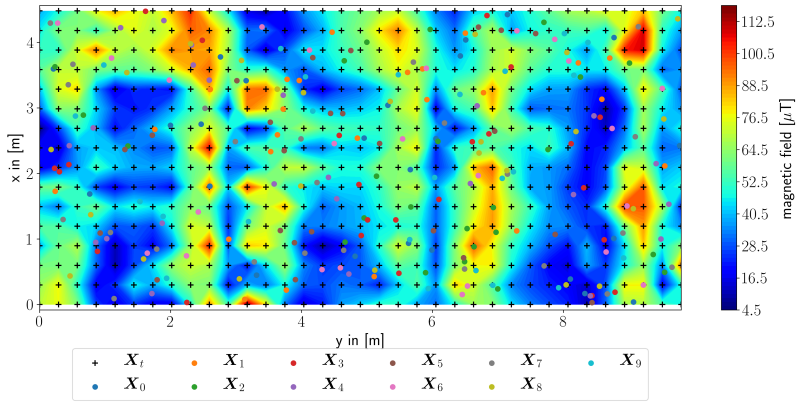


Figure 3.7: The magnetic field in the Holodeck laboratory at the German Aerospace Center (DLR) color coded. The measurement positions are assigned to $K = 10$ agents ($\boldsymbol{X}_k$, $k = 0,\dots,K-1$). A black plus sign indicates a center position in the model $\boldsymbol{\mu}$. The same positions are used as test positions $\boldsymbol{X}_t$ for the evaluation.

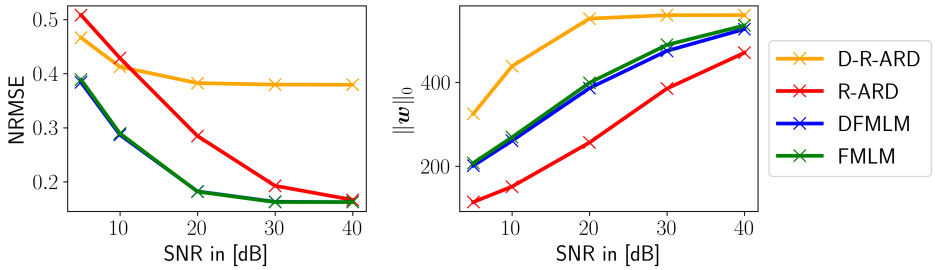
3.2.5. DISCUSSION OF DFMLM AND D-R-ARD FOR HOMOGENEOUS LEARNING

Looking at the simulations of the previous sections, it can be summarized that the methods based on the marginalized likelihood have a lower NRMSE, especially for lower SNR values. For the centralized versions of the algorithms a similar evaluation is presented in [120].

For the distributed setting, the DFMLM is — according to the simulations presented here — the better choice regarding NRMSE and sparsity. DFMLM has only a single consensus step at the beginning of the estimation, which also leads to less transmitted float values. The consensus can then be optimized according to the current network structure [19], e.g., by estimating weighting parameters based on the network (see also App. C). The estimation itself is comparable to the centralized estimation because the DFMLM executes a local FMLM with the obtained consensus values.

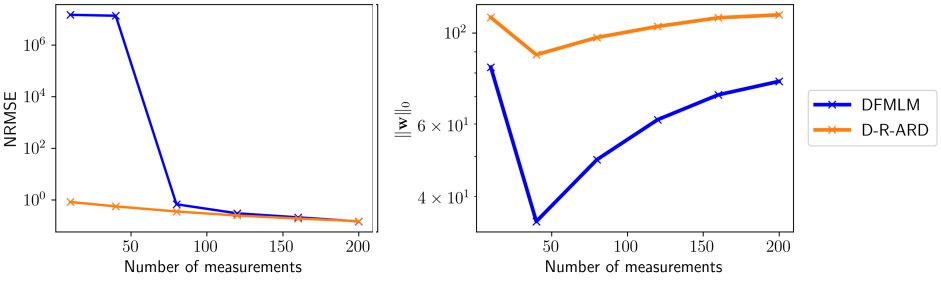
The benefit of the D-R-ARD algorithm is that it can be solved using well known convex optimization algorithms. However, it demands more communication between the nodes in the network, and communication is usually the bottleneck in distributed optimization.

When applying these algorithms to exploration (discussed in detail in the next chapter), the algorithms also have to deal with a small number of measurements, especially at the beginning of the exploration. For example, at the beginning of the exploration, there are K measurements in total (one per agent) and the algorithm has to yield an estimation such that the next measurement location can be determined with the estimated covariance matrix. Therefore, Fig. 3.9(a) shows the performance of the algorithms for a varying number of measurements with artificial data for SNR=10 dB and $\rho = 0.005$. The D-R-ARD algorithm is more robust for a small number of measurements. This might result from the R-ARD algorithm and the minimization of the new upper bounding function $\mathscr{L}(\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{z})$ as it makes the problem convex. The DFMLM on the other side cannot



(a) (b)

Figure 3.8: (a) shows the NRMSE with respect to the SNR for the real data set-up (3.32). (b) displays the sparsity of each algorithm for the real data set-up.



(a) (b)

Figure 3.9: (a) shows the NRMSE with respect to the number of measurements with the test function (3.32). (b) displays the sparsity of each algorithm for approximating (3.32). For both plots, the two first values of the DFMLM are only displayed for explanatory reasons as the algorithm did not converge on these simulation parameters.

handle very few measurements. This might result from the discretization chosen for the center positions of the Gaussian basis functions. If the measurement position does not align with the center position of the Gaussian, the parameter weights become too large. This can result in instabilities. Thus, the DFMLM is not converging and the first NRMSE values (for $M < 50$) in Fig. 3.9(a) are only shown for explanatory reasons. Figure 3.9(b) shows the sparsity for D-R-ARD and DFMLM, where the DFMLM has roughly M non-zero parameter weights. This coincides with the explanation from above as the algorithm tries to reflect all measurements, which might result in very large parameter weight estimates. Again, it is worth noting that the two first sparsity values of the DFMLM in Fig. 3.9(b) are only shown for explanatory reasons. The D-R-ARD algorithm has a lower degree of sparsity, which might come from the combination of the choice of ρ and the number of measurements. The DFMLM yields a better performance for more measurements, but it is not providing reasonable estimates for a smaller number of mea-

surements, which is critical at the beginning of the exploration. Because the D-R-ARD is yielding reasonable estimates independently of the number of measurements, it is considered in the next chapter as the algorithm for exploration, although the estimated $\widehat{\boldsymbol{w}}$ is less sparse compared to DFMLM.

3.3. DISTRIBUTED SPARSE BAYESIAN LEARNING FOR HETEROGENEOUS SPLITTING

In heterogeneous learning, the set of basis functions that are used in the swarm are split among all agents. Therefore, the model complexity is reduced locally at each agent, which alleviates the computation especially when the total number of basis functions dominates the computational complexity. This section presents heterogeneous learning for SBL, and in particular R-ARD as introduced in Sec. 3.1.4 because this method has more robustness for fewer measurements.

3.3.1. DISTRIBUTED AUTOMATIC RELEVANCE DETERMINATION FOR HETEROGENEOUS LEARNING

As shown in Sec. 3.2.2, the authors in [108] use an auxiliary function (3.14) to express an upper bound on the original objective (3.4) to make the computation easier. This chapter follows the same approach, but for heterogeneous learning by using the model (2.15). The following presents the derivation of the D-R-ARD algorithm for SOF.

Consider the upper bound in (3.27), which is presented here again for convenience

$$\mathscr{L}(\boldsymbol{w},\widehat{\boldsymbol{z}}) = \|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{1}{\lambda}\sum_{n=1}^{N}\sqrt{\widehat{z}_n}|w_n|. \tag{3.34}$$

To this end, for SOF, the measurements $\boldsymbol{y}$ are known to all agents and $\boldsymbol{\Phi}$ and $\boldsymbol{w}$ are distributed in the network. The problem that now arises is the distributed computation of $\widehat{\boldsymbol{z}}$ considering the distributed SOF paradigm. The computation of $\widehat{\boldsymbol{z}}$ is not that simple as in the SOE case, but it can be approximated. The reason for this is that for the SOF case $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ can not be split into sums with respect to the measurements.

The optimal value of $\widehat{\boldsymbol{z}}$ is obtained centrally in closed form [108] as

$$\widehat{\boldsymbol{z}} = \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\}. \tag{3.35}$$

In order to get a formulation that is suited for a distributed estimation, the matrix inversion lemma [115, 116] is applied to $\boldsymbol{\Sigma}^{-1}$ resulting into

$$\begin{aligned}\boldsymbol{\Sigma}^{-1} = (\boldsymbol{\Lambda}^{-1}+\boldsymbol{\Phi}\boldsymbol{\Gamma}\boldsymbol{\Phi}^T)^{-1} &= \boldsymbol{\Lambda}-\boldsymbol{\Lambda}\boldsymbol{\Phi}(\boldsymbol{\Gamma}^{-1}+\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\\ &= \boldsymbol{\Lambda}-\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\end{aligned} \tag{3.36}$$

Inserting (3.36) into (3.35) leads to

$$\widehat{\boldsymbol{z}} = \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\} - \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}. \tag{3.37}$$

To distribute this estimate within the SOF paradigm, it is assumed that $\widehat{\boldsymbol{z}}$ can be distributed among the agents as $\widehat{\boldsymbol{z}} = [\widehat{\boldsymbol{z}}_1^T,\ldots,\widehat{\boldsymbol{z}}_K^T]^T$, where $\widehat{\boldsymbol{z}}_k \in \mathbb{R}^{N_k}$, $k = 1,\ldots,K$. The following considers now each term in (3.37) separately to isolate the contribution of agent k to $\widehat{\boldsymbol{z}}$.

For the first diagonal term in (3.37), the contribution of agent k is computed as

$$\operatorname{diag}\left\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\right\}\big|_k = \operatorname{diag}\left\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\right\}, \tag{3.38}$$

where $\cdot|_k$ means in this context that only the contribution of the k-th agent is considered. Here, $\cdot|_k$ also reduces the size of a vector, for SOF, from N to N_k.

For the second term, however, the k-th agent's contribution is not trivial and it will turn out that for SOF the contribution of agent k cannot be isolated completely. With an appropriate permutation matrix, $\boldsymbol{\Sigma}_w = (\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi} + \boldsymbol{\Gamma}^{-1})^{-1}$ can always be brought into the block structure

$$\boldsymbol{\Sigma}_w = \begin{bmatrix} \boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k + \boldsymbol{\Gamma}_k^{-1} & \boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}} \\ \boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k & \boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}} + \boldsymbol{\Gamma}_{\bar{k}}^{-1} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{w},k}^{-1} & \boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}} \\ \boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k & \boldsymbol{\Sigma}_{\boldsymbol{w},\bar{k}}^{-1} \end{bmatrix}^{-1}, \tag{3.39}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{w},k} = (\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k + \boldsymbol{\Gamma}_k^{-1})^{-1}$ and $\boldsymbol{\Sigma}_{\boldsymbol{w},\bar{k}} = (\boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}} + \boldsymbol{\Gamma}_{\bar{k}}^{-1})^{-1}$. Then, (3.39) is inserted into $\operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}$ in (3.37) such that the largest share of the k-th agent can then be isolated by reformulating the term as

$$\begin{aligned} \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}|_k = \operatorname{diag}\big\{&\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k \\ &+(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})^T\boldsymbol{\Omega}_k(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})\big\}, \end{aligned} \tag{3.40}$$

where $\boldsymbol{\Omega}_k = \boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}}(\boldsymbol{\Sigma}_{\boldsymbol{w},\bar{k}}^{-1} - \boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_{\bar{k}})^{-1}\boldsymbol{\Phi}_{\bar{k}}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k$ and the term in the inverse is the Schur complement of (3.39). This thesis now considers only the two contributions to $\widehat{\boldsymbol{z}}$ of agent k, which are combined into

$$\widetilde{\boldsymbol{z}}_k = \operatorname{diag}\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\} - \operatorname{diag}\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\}, \tag{3.41}$$

with $\widetilde{\boldsymbol{z}}_k \in \mathbb{R}^{N_k}$ such that $\widehat{\boldsymbol{z}}_k$ can be reformulated as

$$\widehat{\boldsymbol{z}}_k = \widetilde{\boldsymbol{z}}_k - \operatorname{diag}\left\{(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})^T\boldsymbol{\Omega}_k(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})\right\}. \tag{3.42}$$

The idea is now that $\widetilde{\boldsymbol{z}}_k$ can be used as an approximation to $\widehat{\boldsymbol{z}}_k$. The same can be done for all other agents such that $\widetilde{\boldsymbol{z}} = [\widetilde{\boldsymbol{z}}_1^T, \ldots, \widetilde{\boldsymbol{z}}_K^T]^T \in \mathbb{R}^N$. If then $\widetilde{\boldsymbol{z}}$ is used in (3.27), it leads to the objective function

$$\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widetilde{\boldsymbol{z}}) \triangleq \sum_{n=1}^{N} \widetilde{z}_n\gamma_n + \frac{w_n^2}{\gamma_n} + \lambda\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2. \tag{3.43}$$

It can be shown that the objective in (3.43) upper bounds the objective (3.27), i.e., $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widetilde{\boldsymbol{z}}) \geq \mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widehat{\boldsymbol{z}})$ since

$$\widehat{\boldsymbol{z}}|_k = \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}|_k - \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}|_k \geq 0. \tag{3.44}$$

Thus,

$$\operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}\big|_k \geq \operatorname{diag}\{\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}\}\big|_k. \tag{3.45}$$

Then using (3.40) in the right hand-side results in

$$\begin{aligned}\operatorname{diag}\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\} \geq \operatorname{diag}\left\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\right\} \\ + \operatorname{diag}\left\{(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})^T\boldsymbol{\Omega}_k(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})\right\}. \end{aligned} \tag{3.46}$$

To obtain $\widetilde{\boldsymbol{z}}_k$ as in (3.41), $\operatorname{diag}\{\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\}$ is substracted such that

$$\widetilde{\boldsymbol{z}}_k \geq \operatorname{diag}\{(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})^T\boldsymbol{\Omega}_k(\boldsymbol{\Sigma}_{\boldsymbol{w},k}\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k - \boldsymbol{I})\}. \tag{3.47}$$

With (3.47) it follows that $\widetilde{\boldsymbol{z}}_k \geq \widehat{\boldsymbol{z}}_k$, and, therefore, $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widetilde{\boldsymbol{z}}) \geq \mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widehat{\boldsymbol{z}})$. Furthermore, if there is no correlation between the basis functions of all agents, i.e. they are orthogonal, i.e. $\boldsymbol{\Phi}_k\boldsymbol{\Lambda}\boldsymbol{\Phi}_{k'} = \boldsymbol{0}$ for $k \neq k'$, then this bound becomes tight.

Substituting now $\widehat{\boldsymbol{z}}$ with $\widetilde{\boldsymbol{z}}$ in (3.34), the R-ARD problem is then reformulated in a distributed fashion for SOF as

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\| \boldsymbol{y} - \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{w}_k \right\|_2^2 + \frac{2}{\lambda} \sum_{k=1}^{K} \sum_{n=1}^{N_k} \sqrt{\widetilde{z}_{k,n}} |w_{k,n}|, \tag{3.48}$$

which can be distributively minimized over $\boldsymbol{w}$ with e.g. an ADMM algorithm [18, Sec. 8.3]. It is worth noting that when applying ADMM to solve for $\boldsymbol{w}$ in (3.48) each agent k only requires its own $\widetilde{\boldsymbol{z}}_k$ in this estimation. Thus, the other $\widetilde{\boldsymbol{z}}_{\bar{k}}$ do not need to be known locally.

Likewise, the computation of $\widehat{\boldsymbol{\gamma}}_k$ can be realized locally with $\widetilde{\boldsymbol{z}}_k$ as

$$\widehat{\gamma}_{k,n} = \frac{|\widehat{w}_{k,n}|}{\sqrt{\widetilde{z}_{k,n}}}, \ \forall n = 1, \ldots, N_k; \ k = 1, \ldots, K. \tag{3.49}$$

All steps of the D-R-ARD for SOF are summarized in Alg. 7.

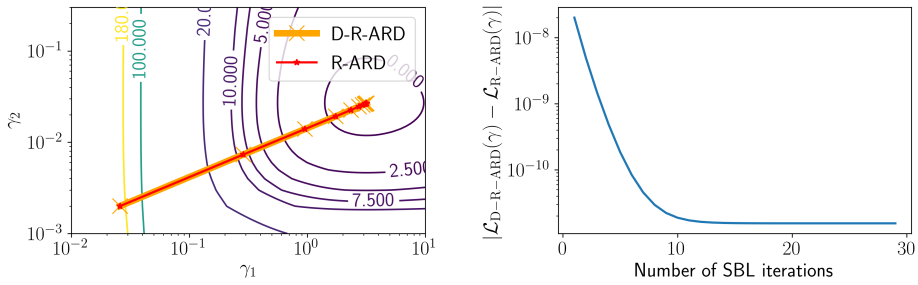
Algorithm 7 D-R-ARD for SOF

1: Initialize $\widetilde{\boldsymbol{z}}_k \leftarrow \operatorname{diag}\left(\boldsymbol{\Phi}_k^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_k\right)$
2: **while** $j <$ max_Iteration$_1$ **do**
3: $\quad \boldsymbol{w}_k \leftarrow$ (3.48) ▷ Solve e.g. with ADMM
4: $\quad \boldsymbol{\gamma} \leftarrow$ (3.49)
5: $\quad$ Check for convergence
6: $\quad \widetilde{\boldsymbol{z}}_k \leftarrow$ (3.41)

3.3.2. CONVERGENCE OF D-R-ARD FOR HETEROGENEOUS LEARNING

As introduced in the previous section, the D-R-ARD for SOF optimizes an upper bounding function $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widetilde{\boldsymbol{z}})$. The current section shows empirically that the minimization of $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \widetilde{\boldsymbol{z}})$ converges to an approximate solution compared to the original cost-function $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \boldsymbol{z})$ of the R-ARD in [108]. Here the R-ARD estimates $\boldsymbol{w}$ with a centralized ADMM with the same ρ as the D-R-ARD for SOF.

For this empirical proof, an artificial process is generated by first sampling the known weights from the prior $p(\boldsymbol{w}|\boldsymbol{\gamma})$, with $\boldsymbol{\gamma} = [\gamma_1, \gamma_2]^T$ and $\boldsymbol{w} \in \mathbb{R}^2$. Then, the weights $\boldsymbol{w}$ are



(a) (b)

Figure 3.10: (a) Two dimensional contour-plot of (3.4) for only two dependent hyper-parameters, as well as the convergence of D-R-ARD for SOF and the algorithm shown in [108]. Please note that the lines in the plot overlap. (b) Absolute difference of (3.4) between D-R-ARD for SOF and R-ARD.

used to generate a process with the model (2.1). For the design matrix $\boldsymbol{\Phi}$ Gaussian basis functions are used. Therefore, the true $\boldsymbol{\gamma}$ is known together with the objective function $\mathscr{L}(\boldsymbol{\gamma})$, which is then optimized by the D-R-ARD.

The contour lines in Fig. 3.10(a) represent the cost-function (3.4). The cost-function (3.4) is the original SBL cost-function, which is upper-bounded by $\mathscr{L}(\boldsymbol{\gamma}, \boldsymbol{w}, \boldsymbol{z})$ for R-ARD. The convergence paths of D-R-ARD for SOF and R-ARD are shown in Fig. 3.10(a) as well. By showing that the convergence paths of both algorithms are the same, it can be assumed that D-R-ARD converges to the same solution for such processes.

An absolute difference between both convergence paths is shown in Fig. 3.10(b), where $\mathscr{L}_{\text{D-R-ARD}}(\boldsymbol{\gamma})$ is (3.4) evaluated for D-R-ARD and $\mathscr{L}_{\text{R-ARD}}(\boldsymbol{\gamma})$ is (3.4) evaluated for R-ARD. Clearly, Fig. 3.10(b) shows a difference between D-R-ARD for SOF and R-ARD. Yet, the difference is minimized with more iterations and stabilizes after few SBL iterations.

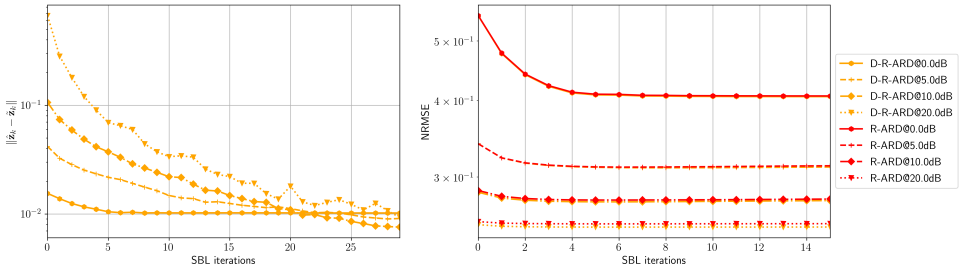
3.3.3. SIMULATIONS FOR D-R-ARD FOR SOF

After showing that D-R-ARD converges to the same solution as the R-ARD algorithm, this section evaluates the robustness and the convergence speed of the algorithm for different SNR values. Therefore, the magnetic field data as shown in Fig. 3.7 is used. From this data, $M = 500$ measurements $\boldsymbol{y}$ are uniformly sampled and disturbed by AWGN as shown in (2.3). The precision of the noise is assumed to be constant for each run, i.e. $\boldsymbol{\Lambda} = \lambda \boldsymbol{I}$. The precision is calculated from the SNR as

$$\lambda = \frac{1}{\varsigma} 10^{0.1 \text{SNR}_{\text{dB}}}, \tag{3.50}$$

where ς is the signal power of the magnetic field and SNR_{dB} is the SNR in dB. Then, it is assumed that this $\boldsymbol{y}$ is distributed over $K = 5$ agents, and that for each agent $\boldsymbol{\Phi}_k(\boldsymbol{X})$ is generated as shown in (2.14). The estimates are averaged over 50 runs per SNR.

The average of the convergence speed for D-R-ARD and R-ARD is displayed in Fig. 3.11(b), where the NRMSE, defined in (3.33), is the used metric. For both algorithms, the



(a) (b)

Figure 3.11: (a) Absolute difference between exact $\widehat{\boldsymbol{z}}$ and approximated $\widetilde{\boldsymbol{z}}$. (b) The NRMSE versus the number of iterations of D-R-ARD for different SNR values. Note that some curves are overlapping.

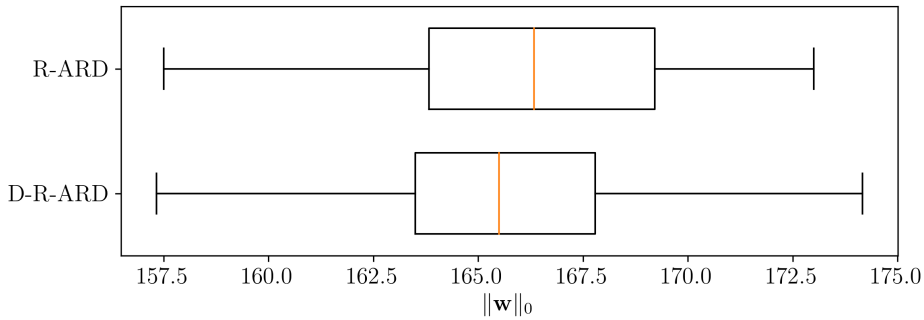


Figure 3.12: The amount of non-zeros in $\widehat{\boldsymbol{w}}$ for D-R-ARD for SOF and R-ARD. The box-plot separates each quartile of the runs into regions and shows the mean value of the runs as an orange line. D-R-ARD for SOF estimates $\widehat{\boldsymbol{w}}$ with less non-zero values.

NRMSE increases with lower SNR with equal performances for both algorithms. Thus, it can be assumed that both algorithms are equally robust for the same noise level. However, if the SNR becomes higher, the difference between both NRMSE values slightly increases, as seen for 20 dB. An explanation could be that for high SNR more parameter weights are non-zero, and due to the approximation $\widetilde{\boldsymbol{z}}$ some information is neglected in the estimation. Therefore, this might result in a slightly higher NRMSE for D-R-ARD for SOF compared to R-ARD.

In addition to the NRMSE, the absolute difference between $\widehat{\boldsymbol{z}}$ and $\widetilde{\boldsymbol{z}}$ is evaluated, which can be seen in Fig. 3.11(a). For low SNR, the difference between the approximated and the exact solution of $\widehat{\boldsymbol{z}}$ is converging fast and not changing with higher number of iterations. If the SNR is higher, $\|\widetilde{\boldsymbol{z}} - \widehat{\boldsymbol{z}}\|$ is still decreasing with later iterations, although the algorithm converges as observed in Fig. 3.11(b). This could imply that for higher SNR the D-R-ARD needs more iterations to converge to the same solution as the R-ARD. Yet, the corresponding NRMSE in Fig. 3.11(b) seems converged.

Because D-R-ARD uses an approximation $\widetilde{\boldsymbol{z}}$ for optimization, it calculates a stronger

penalty on the sparsity. In (3.49) it can be seen that $\tilde{\boldsymbol{z}}$ controls the sparsity of $\hat{\boldsymbol{w}}$. Now that $\tilde{\boldsymbol{z}} \geq \hat{\boldsymbol{z}}$ as shown in (3.42) and the equations that follow, the sparsity should be higher for D-R-ARD for SOF. In other words, using D-R-ARD for SOF $\hat{\boldsymbol{w}}$ has fewer non-zero values. This is shown in Fig. 3.12, where the number of non-zero values is averaged over the number of agents and shown for each run as a box-plot. It can be observed that the D-R-ARD for SOF has on average fewer non-zero values in $\hat{\boldsymbol{w}}$ compared with the R-ARD.

3.4. SUMMARY

This chapter introduced Bayesian methods for a distributed setting. In particular, the methods of SBL were utilized to achieve a sparse solution of the parameter weights. Compared to classical methods, e.g. LASSO, SBL methods do not require a threshold parameter and thus SBL methods need less parameter tuning.

In particular this chapter presented distributed algorithms, which are based on SBL. In total three different algorithms were presented: DFMLM, D-R-ARD for SOE, and D-R-ARD for SOF. The algorithms DFMLM and D-R-ARD for SOE use consensus strategies to distribute the data in the network. The derivations showed that DFMLM only needs a single consensus step and therefore communicates less data to achieve the same result as the D-R-ARD for SOE. The D-R-ARD for SOE on the other side was able to use popular convex optimization schemes. Furthermore, for few measurements this algorithm yields a lower error compared with the DFMLM. This happens at the beginning of the exploration and therefore D-R-ARD is the better choice for swarm exploration. After the evaluation of DFMLM for SOE, it was decided against to derive a DFMLM for SOF out of the following reason: A distributed version of this algorithm with the SOF paradigm would become complicated and would potentially lead to approximations, which in turn could lead to further instabilities.

The third algorithm — D-R-ARD for SOF — permitted the use of a locally model, which provides more flexibility in selecting the basis functions. Yet, the objective had to be reformulated to be suited for this distribution paradigm. The simulations empirically showed that D-R-ARD for SOF converged to an approximate solution of its centralized counterpart. Also the convergence speed of D-R-ARD for SOF was equal to R-ARD.

It can also be concluded that the developed distributed SBL algorithms have no or only minor cuts in performance compared with their centralized versions. For all the D-R-ARD algorithms it is important to choose the parameter ρ carefully as it influences the convergence speed [121, 125], helps to stabilize the estimation if there are only few measurements, and, if the number of iterations is fixed, also the error and the degree of sparsity. However, the simulations indicate that if the parameter is chosen for a low error, it does not necessarily lead to a high degree of sparsity. This might suggest to vary this parameter with increasing iterations. Yet, this has not been looked at and might potentially influence the overall convergence speed of the algorithm.

4

Distributed Exploration with Optimal Experiment Design for the Distribution Paradigms SOE and SOF

The previous chapters introduced methods to estimate model parameters based on measurements but what is the next optimal measurement location? This question can be answered with exploration. Exploration exploits all so far obtained measurement values, measurement locations, and parameter estimates to determine a next measurement location. This is depicted in Fig. 4.1. Ideally, the next measurement location should be most informative such that the confidence about the parameter estimates is increased. One approach to exploration that increases the confidence about the parameters is optimal experiment design (OED) [31]. OED alters the information matrix that depends on the measurement locations. Also, through various criteria, OED can change the type of exploration. It is however not researched how to calculate OED-criteria in a distributed fashion and how the criteria are affected if the parameter estimates are sparse. These two aspects are analyzed in this chapter.

The current chapter therefore starts by explaining two potentially useful OED-criteria for exploration. Afterward, the distribution paradigms introduced in Sec. 2.3.1 are ap-

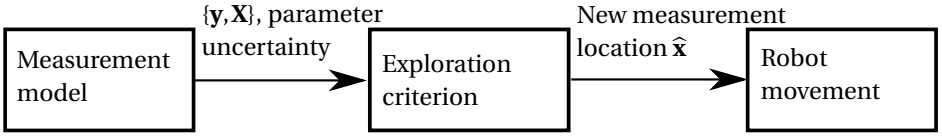


Figure 4.1: The approach of exploration in this thesis.

plied to the D-optimality criterion as it is the potentially most useful criterion for swarm exploration. The resulting distributed computation of the D-optimality criterion based on the distributed LASSO algorithm is then verified in simulations. Additionally, the effects of different basis functions and approaches to parallelize the exploration are analyzed. The parallelization is realized by constraining the search space for measurement locations and coordinate the agents more effectively. Thus, the methods for parallelization are here also referred to as coordination methods. Furthermore, the D-optimality criterion based on the distributed SBL is derived and compared with the D-optimality criterion based on the distributed LASSO. At the end, a summary is given as well as some first concluding remarks on distributed exploration.

4.1. Optimal Experiment Design

OED seeks to minimize the variance of an estimator with respect to a statistical criterion. The latter usually involves the covariance matrix of a parameter vector. In this thesis, criteria are considered that increase the confidence of this parameter vector. For the model (2.3), the parameter vector $\boldsymbol{w}$ can be estimated through the well known least squares method [117, Chap. 3.8] as

$$\widehat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}, \tag{4.1}$$

which corresponds to the maximum likelihood estimate, if Gaussian noise on the measurements is assumed. The confidence of the estimate $\widehat{\boldsymbol{w}}$ is then proportional to

$$\mathrm{var}(\widehat{\boldsymbol{w}}) \propto (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1} = \boldsymbol{C}', \tag{4.2}$$

with $\boldsymbol{C}' \in \mathbb{R}^{N\times N}$, which is known as the covariance matrix of the parameter estimates. Please note that $\mathrm{var}(\widehat{\boldsymbol{w}})$ also depends on the noise variance of the measurements, which is neglected here as the real noise is often unknown. Thus, $\mathrm{var}(\widehat{\boldsymbol{w}})$ is considered to be proportional to $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}$.

For the considered model, its inverse is also known as the Fisher information matrix [126]. Obviously, the covariance matrix depends on the design matrix $\boldsymbol{\Phi}$ and how it is constructed. Depending on the choice or construction of $\boldsymbol{\Phi}$, the variance of the estimate $\widehat{\boldsymbol{w}}$ can be reduced. In the following, two OED-criteria based on the covariance matrix are introduced to increase the confidence about $\widehat{\boldsymbol{w}}$: the A-optimality and the D-optimality criterion.

The A-optimality criterion minimizes the trace of the inverse of the information matrix of the parameter weight estimates

$$\min \mathrm{tr}\{(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\} = \min \mathrm{tr}\{\boldsymbol{C}'\}. \tag{4.3}$$

Because of the fact that $\mathrm{tr}\{\boldsymbol{C}'\} = \sum_{n=1}^{N} \lambda_{C',n}$ with $\lambda_{C',n}, n = 1,\dots,N$, being the eigenvalues of $\boldsymbol{C}'$, the A-optimality criterion is equivalent to minimizing the sum of the eigenvalues of the covariance matrix. The A-optimality can therefore be interpreted as the minimization of the average variance of the parameter weight estimates. It is therefore also called the average variance criterion. The A-optimality is used for exploration in [127] for regression designs, in [128, 129] as an uncertainty prior for map exploration, in [130] for designing calibration measurements, and in [29, 131] for greedy exploration designs.

The D-optimality criterion on the other hand minimizes the determinant of the inverse of the parameter weight estimates' information matrix

$$\min \det\{(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}\} = \min \det\{\boldsymbol{C}'\}. \tag{4.4}$$

The D-optimality criterion can also be related to the minimization of the differential Shannon entropy [31, 132]. Due to this relation, D-optimality can be seen as the minimization of the entropy of the underlying statistical model. The terms D-optimality and entropy might be used interchangeably in this thesis. Because $\det\{\boldsymbol{C}'\} = \prod_{n=1}^{N} \lambda_{C',n}$, this criterion can also be seen as minimizing the product of the eigenvalues of the covariance matrix, which corresponds to the volume of the confidence ellipsoid. In comparison with the A-optimality criterion, the D-optimality criterion also considers the model paramater covariances, where the A-optimality only reduces the variance of the parameter estimates [133–135]. Furthermore, it has been shown that the D-optimality criterion works better in constrained regions [136]. In [137] it has also been shown that D-optimality has desirable effects when it comes to exploration for simultaneous localization and mapping (SLAM) such as a better quantification of the uncertainty compared with A-optimality. Thus, D-optimality is the better choice because of the aforementioned property, and from now on only the D-optimality criterion is considered.

4.2. D-Optimal Experiment Design for Distributed Exploration

optimal experiment design is defined for a centralized planning: a centralized computing unit evaluates the OED criterion and yields a new measurement positions. This section analyzes how the D-optimality criterion (4.4) can be formulated in a distributed fashion for two conceptually different distribution paradigms: the homogeneous learning and the heterogeneous learning, see Sec 2.3.1. Furthermore, this section presents the analysis of the D-optimality criterion for sparse parameter weights when having a sparse covariance matrix. This section shows that these sparsity constraints lead to a D-optimality criterion that is tightly coupled with the estimated parameter weights from the underlying estimator. Due to the sparsity constraints, every new measurement might change the result of the estimation as well as the structure of the covariance matrix. It turns out that in this case the D-optimality criterion has to be evaluated after each new measurement. Consequently, from a theoretical perspective, the D-optimality criterion cannot be planned beforehand and that it has to evaluated after each single measurement. This implies that it needs to be evaluated sequentially. The results from this analysis are new to the literature and have been published in the course of this thesis in [53].

This section starts by presenting the state of the art in exploration methods. Then, the centralized D-optimality criterion but under sparsity constraints is revisited and followed by the derivation of the D-optimality criterion for the two proposed distribution paradigms. Both methods are then compared in simulations, where multiple agents explore a magnetic field.

4.2.1. Related Methods for Exploration

In [28], the authors minimize the determinant of the covariance matrix for an optimal camera placement for a 3D-image. The authors name it *exploration driven by uncertainty* and they motivate their work with the Shannon-entropy criterion. They also create a link to OED as proposed in [138] but the problem they look at is not sparse. In [29], the authors compare the D-optimality criterion and other OED criteria with the mutual information for GP regression and sensor placement. The result of this work is a greedy algorithm that uses the mutual information for finding optimal sensor placements. In their work, the authors do not assume that the considered model is sparse and they make the sensor placements beforehand. Thus, previous sensor measurements are not influential to future sensor measurements. An extension of [29] for multiple agents and a decentralized estimation of the mutual information is presented in [139, 140]. In [139], the authors consider multiple agents that should solve a traveling salesman problem by applying the methods mentioned in [29]. The agents are furthermore partitioned and they estimate the traveling salesman problem in a decentralized fashion. In [140], the authors consider UAVs and they aim at maximizing the information for a traversed path.

Another exploration approach for multiple agents is shown in [45]. There, the authors use a probabilistic approach to steer cameras, which are mounted on multiple UAVs. The authors define an information metric that is based on the information per pixel. However, the authors do not look into a distributed computation of the problem.

4.2.2. Centralized D-Optimal Experiment Design for Exploration under Sparsity Constraints

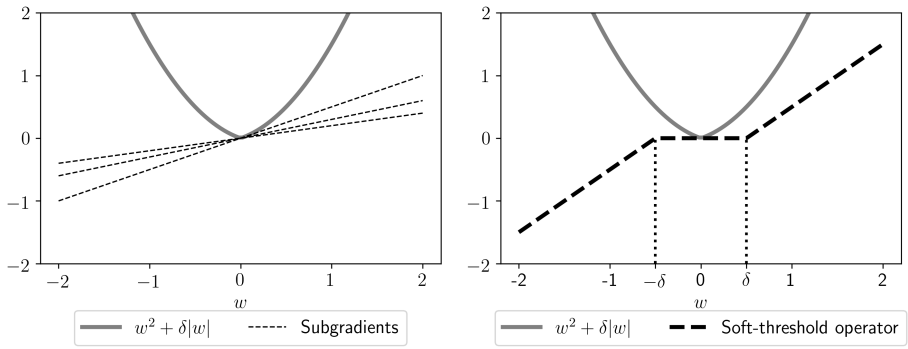
As mentioned before, OED aims at optimizing the variance of an estimator. In the context of exploration, a criterion is optimized by selecting measurement locations $\boldsymbol{X}$, which directly influence the design matrix $\boldsymbol{\Phi}(\boldsymbol{X})$ of the model (2.5) and the covariance matrix of the parameter weights. The inverse of the covariance matrix is also known as the Fisher information matrix [126, 141], which can be approximated by using the Hessian of the cost-function of the estimator [126].

From now on the LASSO estimator in (2.9) is considered. The first derivative of the LASSO cost function (2.8) cannot be calculated because of the ℓ_1-norm, which is non-smooth in 0 and, therefore, the gradient is not defined at 0. The ℓ_1-norm is depicted in Fig. 4.2(a). Subsequently, no second-order derivative is defined. However, it is possible to define a minimizer of (2.9) by using the soft-thresholding operator [23, 142]. The soft-thresholding operator is defined as

$$\mathscr{S}_\delta(\boldsymbol{u}) \triangleq \max(0, |\boldsymbol{u}| - \delta)\,\mathrm{sign}\,(\boldsymbol{u}), \tag{4.5}$$

with sign $(\cdot)$ returning the sign of the vector's components and $\max(0,\cdot)$ returning the components if they are larger than zero and zero otherwise. The soft-thresholding operator results in an approximated derivative, which is displayed in Fig. 4.2(b).

For solving (2.9), the ISTA algorithm [75, 78] can be used together with the soft-thresholding operator. Basically, the ISTA algorithm iteratively applies a gradient step on the considered variable, followed by a shrinkage step. Thus, the minimizer $\widehat{\boldsymbol{w}}$ for (2.9)



(a) (b)

Figure 4.2: (a) Sparse regularized quadratic cost function and possible subgradients at zero. (b) Sparse regularized cost function and the soft-thresholding operator, which acts as a derivative.

can be iteratively calculated by applying

$$\widehat{\boldsymbol{w}} = \mathscr{S}_\delta\left(\boldsymbol{w} - \boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})\right), \tag{4.6}$$

multiple times until $\widehat{\boldsymbol{w}}$ converges.

Due to the max-operator the soft-thresholding operator is also not differentiable. To circumvent this non-differentiability, this thesis proposes to utilize an active set approach, which is introduced in [105] for slantly Newton differentiable functions such as (2.8). In Def. 5 the active and inactive sets are defined.

Definition 5. *Consider a matrix $\boldsymbol{A} \in \mathbb{C}^{P\times Q}$ and a vector $\boldsymbol{b} \in \mathbb{R}^Q$ with entries b_q that is S-sparse. Then, for the product $\boldsymbol{Ab}$, the set $\mathscr{A} = \{q : b_q \neq 0, \forall\, q = 1\ldots, Q\}$ forms the set of the column indexes of $\boldsymbol{A}$ that are weighted by a non-zero component of $\boldsymbol{b}$. Thus, $\mathscr{A}$ is called the active set. Likewise, $\mathscr{I} = \{1\ldots, Q\} \setminus \mathscr{A}$ is the inactive set.*

The active and inactive sets are then defined as

$$\mathscr{A} = \{n \in \mathbb{N} : \widehat{w}_n \neq 0\}, \tag{4.7}$$

$$\mathscr{I} = \{n \in \mathbb{N} : \widehat{w}_n = 0\}. \tag{4.8}$$

To use the D-optimality criterion for an estimator such as (2.9), the Hessian can be approximated as shown in the following steps.

1. Let $\delta_n \triangleq 2\delta/|w_n|$, $n = 1,\ldots,N$, such that (2.8) is rewritten as

$$\mathscr{L}(\boldsymbol{w}) = \frac{1}{2}\left\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\right\|^2 + \frac{1}{2}\sum_{n=1}^{N}\delta_n|w_n|^2. \tag{4.9}$$

This way the resulting objective function becomes smooth in $\boldsymbol{w}$ for any fixed δ_n.

2. Then the second-order derivative of (4.9) is approximated as

$$\frac{\mathrm{d}^2\mathscr{L}(\boldsymbol{w})}{\mathrm{d}\boldsymbol{w}^2} \approx \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{\Delta}\right)^{-1}, \tag{4.10}$$

where $\boldsymbol{\Delta} = \mathrm{diag}\{\delta_1,\ldots,\delta_N\}$.

3. Next, the estimates $\widehat{\boldsymbol{w}}$ have to be inserted to calculate the covariance of the parameter estimates. For the components of the inactive set, where $\widehat{w}_n = 0, n \in \mathscr{I}$, the covariance becomes zero as well because

$$\lim_{\widehat{w}_n\to 0}\widehat{\delta}_n = \lim_{\widehat{w}_n\to 0}\frac{2\delta}{|\widehat{w}_n|} = \infty, \tag{4.11}$$

and due to the inversion in (4.10), the rows and columns that belong to $n \in \mathscr{I}$ become zero as well. Thus, the parameter estimates that belong to the inactive set can be neglected and components of the active set — see Def. 5 — are utilized to compute the covariance matrix.

4. Generate a selection matrix $\boldsymbol{P} \in \mathbb{R}^{N\times|\mathscr{A}|}$ to exclude the inactive set, where $\boldsymbol{P}$ consist of column vectors that have a 1 on the n-th row, where $n \in \mathscr{A}$ such that $\boldsymbol{\Phi}_{\mathscr{A}} = \boldsymbol{\Phi}\boldsymbol{P}$ and $\boldsymbol{\Phi}_{\mathscr{A}} \in \mathbb{R}^{M\times|\mathscr{A}|}$.

5. Then, the Hessian of (2.8) is approximated as

$$\frac{\mathrm{d}^2\mathscr{L}(\boldsymbol{w})}{\mathrm{d}\boldsymbol{w}^2} \approx \left(\boldsymbol{\Phi}_{\mathscr{A}}^T\boldsymbol{\Phi}_{\mathscr{A}}+\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}\right)^{-1} = \boldsymbol{C}, \tag{4.12}$$

where $\widehat{\boldsymbol{\Delta}}_{\mathscr{A}} \in \mathbb{R}^{|\mathscr{A}|\times|\mathscr{A}|}$ and $\widehat{\boldsymbol{\Delta}}_{\mathscr{A}} = \mathrm{diag}\{\widehat{\delta}_1,\ldots,\widehat{\delta}_N\}\ \forall n \in \mathscr{A}$.

It is worth noting that the inactive set has no influence on (4.10), because of the same reasons as explained in Sec. 3.1.1. Parallel to Sec. 3.1.1, δ_n takes over the functionality of the hyper-parameter γ_n^{-1} in (3.6). If $\widehat{w}_n = 0$ then $\widehat{\delta}_n$ is not defined and the corresponding n-th basis function can be excluded from the model.

The approximated Hessian (4.12) is then used as a covariance matrix for the parameter weights. Beside its dependence on the parameter estimates, it is also dependent on the type of basis functions and for which $\boldsymbol{x}$ and $\boldsymbol{\mu}$ they are evaluated. The following considers Gaussian basis functions, which are introduced in (2.10). For convenience the definition of the Gaussian basis function is shown here again

$$\phi(\boldsymbol{x},\boldsymbol{\mu}) = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|^2}{2\sigma^2}\right\}. \tag{4.13}$$

They are centered at a position $\boldsymbol{\mu}$, and are evaluated at a position $\boldsymbol{x}$. Here, the width of these functions is defined by $\sigma > 0$, which is equal for all basis functions.

Consider now M measurement positions such that $\boldsymbol{X} = [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M]^T \in \mathbb{R}^{M\times 2}$. Then, $\boldsymbol{\phi}(\boldsymbol{X},\boldsymbol{\mu}_n) = \left[\phi(\boldsymbol{x}_1,\boldsymbol{\mu}_n),\ldots,\phi(\boldsymbol{x}_M,\boldsymbol{\mu}_n)\right]^T \in \mathbb{R}^M$ is a Gaussian basis function evaluated at all measurement positions in $\boldsymbol{X}$. For N center positions $\boldsymbol{M} = \left[\boldsymbol{\mu}_1^T,\ldots,\boldsymbol{\mu}_N^T\right]^T \in \mathbb{R}^{N\times 2}$, the design matrix is constructed as $\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M}) = \left[\boldsymbol{\phi}(\boldsymbol{X},\boldsymbol{\mu}_1),\ldots,\boldsymbol{\phi}(\boldsymbol{X},\boldsymbol{\mu}_N)\right] \in \mathbb{R}^{M\times N}$. Subsequently,

$\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{M}) = \left[\phi(\boldsymbol{x},\boldsymbol{\mu}_1),\dots\phi(\boldsymbol{x},\boldsymbol{\mu}_N)\right]^T \in \mathbb{R}^N$ represents the row for the measurement $\boldsymbol{x}$ in $\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M})$. As will be shown in the following, the dependency of $\boldsymbol{X}$ and $\boldsymbol{M}$ in $\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M})$ is important for the exploration.

As written above, let $\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M}) = \boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M})\boldsymbol{P} \in \mathbb{R}^{M\times|\mathscr{A}|}$ be the design matrix with column indices from the active set. Writing (4.12) with $\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M})$ highlights the dependency of the approximated covariance on the measurement positions and the center positions of the basis functions as

$$\boldsymbol{C}(\boldsymbol{X},\boldsymbol{M}) = \left(\boldsymbol{\Phi}_{\mathscr{A}}^T(\boldsymbol{X},\boldsymbol{M})\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M}) + \widehat{\boldsymbol{\Delta}}_{\mathscr{A}}\right)^{-1}. \tag{4.14}$$

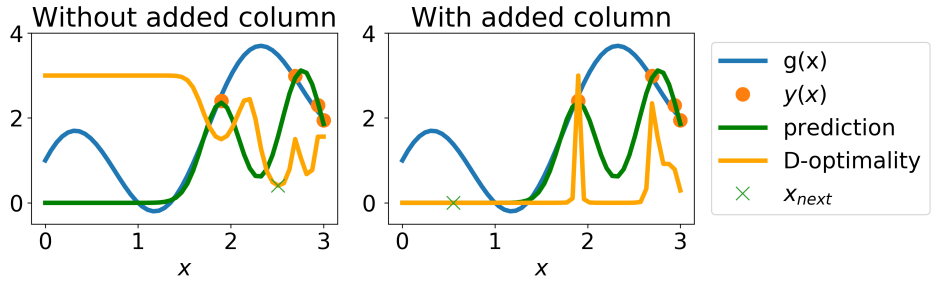
For a hypothetical measurement position $\tilde{\boldsymbol{x}} \in \mathbb{R}^2$ and potential center position of a basis function $\check{\boldsymbol{\mu}} \in \mathbb{R}^2$ the D-optimality criterion can be evaluated by using (4.14). For the evaluation, a row depending on $\tilde{\boldsymbol{x}}$ and a column depending on $\check{\boldsymbol{\mu}}$ is added to $\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M})$ as

$$\widetilde{\boldsymbol{\Phi}}_{\mathscr{A}}([\boldsymbol{X}^T,\tilde{\boldsymbol{x}}]^T,[\boldsymbol{M}^T,\check{\boldsymbol{\mu}}]^T) = \begin{bmatrix} \boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M}) & \boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}) \\ \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) \end{bmatrix}, \tag{4.15}$$

where $\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) = \left(\boldsymbol{\phi}^T(\tilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{P}\right)^T \in \mathbb{R}^{|\mathscr{A}|}$ such that $\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) = \left[\phi(\tilde{\boldsymbol{x}},\boldsymbol{\mu}_1),\dots,\phi(\tilde{\boldsymbol{x}},\boldsymbol{\mu}_n),\dots,\right.$ $\left.\phi(\tilde{\boldsymbol{x}},\boldsymbol{\mu}_N)\right]^T \forall n \in \mathscr{A}$; the subscript $\mathscr{A}$ denotes that only center positions $\boldsymbol{\mu}_n$ are considered where $n \in \mathscr{A}$. The reason why it is important to add also a column is that basis functions that have been canceled out cannot be reconsidered. Otherwise, only active basis functions would be taken into account. This applies as well for global basis functions. For example, a new measurement location might excite currently inactive frequency components, if DCTs are used. A theoretical analysis is presented in Appendix B.1. Example 3 shows the effects, with and without an added column to $\widetilde{\boldsymbol{\Phi}}_{\mathscr{A}}$.

Example 3. *Consider the function $g(x)$ of Example 1. The signal model considered in the current example sets the number of basis functions to $N = 20$ with $\boldsymbol{\phi}_n(x)$, $n = 1,\dots,N$ being Gaussian basis functions. The basis functions are uniformly distributed in $\boldsymbol{x}$. For simplicity, the center positions of the basis functions are set equal to the potential measurement positions, i.e. $\check{\mu} = \tilde{x}$ (Why this is a reasonable choice, is explained later in Sec. 4.2.3). Due to the sparsity constraints, some basis functions belong to the inactive set and are therefore not contributing to the prediction. Even if $N >> M$, e.g. the exploration space can be densely populated with basis functions, the sparsity constraints would control the estimates such that the active set becomes small. As a result, if no column is added to $\boldsymbol{\Phi}$, the D-optimality criterion considers only active basis functions, as shown in Fig. 4.3(a). Accordingly, at locations that have no basis function, the covariance is zero. A low variance is the same as a high precision and subsequently the model is confident about this position. The D-optimality criterion then yields a high value and positions far away would not be chosen as measurement positions.*

In Fig. 4.3(b) a column is added to $\boldsymbol{\Phi}$ as shown in (4.15) *such that the D-optimality can consider an activated basis function at this position. If a column is added, the model estimates that there is no information about the corresponding parameter weight, which leads to a high covariance. Hence, the D-optimality criterion yields a lower value such that measurement positions further away would be considered.*



(a) (b)

Figure 4.3: The observed function is $g(x)$ and $y(x)$ are obtained measurements of $g(x)$. The prediction of the model based on the measurements $y(x)$ is shown in green. The left figure presents the D-optimality criterion, if no column is added to (4.15) in orange. The right figure shows the D-optimality criterion, if a column is added to (4.15) also in orange. The minimum of the D-optimality criterion is highlighted in both figures as a green cross, which would result in the next measurement location x_{next}. This location has not been measured yet.

As presented in Example 3 and Appendix B.1, it is necessary to add a row and a column to $\boldsymbol{\Phi}_{\mathscr{A}}$ to evaluate the D-optimality for potential next measurement locations. This is mainly due to the resulting sparsity, because otherwise the model can not yield predictions that require basis functions that belong to inactive parameter weights. Therefore, $\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}})$ and $\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}})$ are added to $\boldsymbol{\Phi}_{\mathscr{A}}$ as introduced in (4.15). Local basis functions require a center location $\check{\boldsymbol{\mu}}$ and are evaluated at a hypothetical measurement location $\tilde{\boldsymbol{x}}$. Here it is assumed that a new measurement location can only *activate* one basis function, i.e. changing a parameter weight from zero to non-zero. However, the center position $\check{\boldsymbol{\mu}}$ and the hypothetical measurement location $\tilde{\boldsymbol{x}}$ do not necessarily have to be the same. With (4.15) and $\tilde{\boldsymbol{\Phi}}_{\mathscr{A}} \triangleq \tilde{\boldsymbol{\Phi}}_{\mathscr{A}}\left([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T\right)$, the covariance of all measurements and a hypothetical measurement location is given as

$$\tilde{\boldsymbol{C}}\left([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T\right) = \left(\tilde{\boldsymbol{\Phi}}_{\mathscr{A}}^T \tilde{\boldsymbol{\Phi}}_{\mathscr{A}} + \operatorname{diag}\{\hat{\boldsymbol{\Delta}}_{\mathscr{A}}, 0\}\right)^{-1}, \tag{4.16}$$

where setting the additional diagonal term to zero does not regularize the potentially activated basis function; it is therefore in the model. It is worth noting that, if the regularization of the column corresponding to the hypothetical center position would be too high, the covariance might favor already measured positions. With a gradually increasing regularization of the column corresponding to the hypothetical center position, the D-optimality criterion would look like as shown in Fig. 4.3(a).

Inserting (4.16) into the D-optimality criterion leads to the problem formulation

$$\hat{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}} \in \mathscr{X}, \check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}}{\arg\min} \log \left| \tilde{\boldsymbol{\Phi}}_{\mathscr{A}}^T \tilde{\boldsymbol{\Phi}}_{\mathscr{A}} + \begin{bmatrix} \hat{\boldsymbol{\Delta}}_{\mathscr{A}} & \\ & 0 \end{bmatrix} \right|^{-1}, \tag{4.17}$$

where $\mathscr{X}$ are all possible measurement locations, $\mathscr{M}$ are all possible center locations, and $\mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}$ excludes the center locations that belong to the active set $\mathscr{A}$ from $\mathscr{M}$.

For localized basis functions, the minimization of (4.17) involves two parameters, the potential measurement location $\tilde{\boldsymbol{x}} \in \mathscr{X}$ and a potential center position $\check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}$. If global basis functions are utilized, they would not optimize (4.17) for the center position but for another parameter, e.g., the frequency component as in (2.13).

The problem at this point is that (4.17) is not suited for a distributed optimization, and (4.17) requires the calculation of the determinant of the covariance $\widetilde{\boldsymbol{C}}$ for all hypothetical measurement and center positions, which is computationally expensive. Therefore, in the following, (4.17) is algebraically altered to prepare this criterion for a distributed optimization and to reduce the computational complexity.

First, (4.15) is inserted into $\widetilde{\boldsymbol{C}}$ and the logarithm is applied to it

$$\begin{aligned}\log\det\widetilde{\boldsymbol{C}}([\boldsymbol{X}^T,\tilde{\boldsymbol{x}}]^T,[\boldsymbol{M}^T,\check{\boldsymbol{\mu}}]^T) = -\log\Bigg|&\begin{bmatrix}\boldsymbol{\Phi}_{\mathscr{A}}^T\boldsymbol{\Phi}_{\mathscr{A}}+\widehat{\boldsymbol{\Delta}}_{\mathscr{A}} & \boldsymbol{\Phi}_{\mathscr{A}}^T\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})\\ \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Phi}_{\mathscr{A}} & \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})\end{bmatrix}\\ &+\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M})\\ \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\Bigg|,\end{aligned} \tag{4.18}$$

where the argument of the log-determinant is already split into two terms, the first dealing with the measurements $\boldsymbol{X}$ and the second dealing with the hypothetical measurement $\tilde{\boldsymbol{x}}$. To simplify the notation $\log\det\widetilde{\boldsymbol{C}} \triangleq \log\det\widetilde{\boldsymbol{C}}\left([\boldsymbol{X}^T,\tilde{\boldsymbol{x}}]^T,[\boldsymbol{M}^T,\check{\boldsymbol{\mu}}]^T\right)$ is defined. Additionally, define

$$\boldsymbol{A} \triangleq \boldsymbol{\Phi}_{\mathscr{A}}^T\boldsymbol{\Phi}_{\mathscr{A}}+\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}, \quad \boldsymbol{c}(\check{\boldsymbol{\mu}}) \triangleq \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Phi}_{\mathscr{A}}, \quad b(\check{\boldsymbol{\mu}}) \triangleq \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}). \tag{4.19}$$

Then, (4.18) can be written as

$$\log\det\widetilde{\boldsymbol{C}} = -\log\left|\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}})\\ \boldsymbol{c}^T(\check{\boldsymbol{\mu}}) & b(\check{\boldsymbol{\mu}})\end{bmatrix}+\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M})\\ \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\right|. \tag{4.20}$$

Using the matrix determinant lemma [143] on (4.20) yields

$$\begin{aligned}\log\det\widetilde{\boldsymbol{C}} = &-\log\left|\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}})\\ \boldsymbol{c}(\check{\boldsymbol{\mu}})^T & b(\check{\boldsymbol{\mu}})\end{bmatrix}\right|\\ &-\log\left|1+\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}})\\ \boldsymbol{c}(\check{\boldsymbol{\mu}})^T & b(\check{\boldsymbol{\mu}})\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M})\\ \phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\right|.\end{aligned} \tag{4.21}$$

Now, the first term in (4.21) depends only on the potential center position $\check{\boldsymbol{\mu}}$ and the second term depends on the potential measurement location $\tilde{\boldsymbol{x}}$ and $\check{\boldsymbol{\mu}}$. For the first term, it can be shown that

$$\log\left|\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}})\\ \boldsymbol{c}(\check{\boldsymbol{\mu}})^T & b(\check{\boldsymbol{\mu}})\end{bmatrix}\right| = \log|\boldsymbol{A}|+\log q(\check{\boldsymbol{\mu}}), \tag{4.22}$$

where $q(\check{\boldsymbol{\mu}}) = b(\check{\boldsymbol{\mu}}) - \boldsymbol{c}^T(\check{\boldsymbol{\mu}})\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}})$ is the Schur complement [144] of the corresponding matrix. By using the matrix inversion lemma [116] on the inverse in (4.21), it can be shown that

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}}) \\ \boldsymbol{c}(\check{\boldsymbol{\mu}})^T & b(\check{\boldsymbol{\mu}}) \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}})q(\check{\boldsymbol{\mu}})^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}})/q(\check{\boldsymbol{\mu}}) \\ -\boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1}/q(\check{\boldsymbol{\mu}}) & 1/q(\check{\boldsymbol{\mu}}) \end{bmatrix}. \tag{4.23}$$

Inserting (4.23) into the second log-term in (4.21) followed by expanding the product leads to

$$\begin{aligned} &\log\left|1 + \begin{bmatrix} \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{\mu}}) \\ \boldsymbol{c}(\check{\boldsymbol{\mu}})^T & b(\check{\boldsymbol{\mu}}) \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) \\ \phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) \end{bmatrix}\right| \\ &= \log\left(1 + \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M})\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) + \left(\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) - \boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M})\right)^2 / q(\check{\boldsymbol{\mu}})\right). \end{aligned} \tag{4.24}$$

Then (4.22) and (4.24) are inserted into (4.21) such that

$$\begin{aligned} \log\det\widetilde{\boldsymbol{C}}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) = -\log|\boldsymbol{A}| - \log\Big(&q(\check{\boldsymbol{\mu}}) + q(\check{\boldsymbol{\mu}})\boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M})\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) \\ &+ \left(\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) - \boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M})\right)^2\Big). \end{aligned} \tag{4.25}$$

The function (4.25) represents the objective of the D-optimality criterion in the log domain. For the optimization over all potential next measurement and center locations, this formulation only needs to compute $\boldsymbol{A}^{-1}$ once at the beginning and can be reused for the next iterations. The first log term is independent of $\tilde{\boldsymbol{x}}$ and $\check{\boldsymbol{\mu}}$ and can be neglected in the optimization. The calculation of (4.25) does not require the computation of a determinant of a large matrix as in (4.17) but only matrix vector operations. It is therefore computationally less expensive. This criterion is then inserted into (4.17) to formulate the optimization problem for the exploration as

$$\begin{aligned} \hat{\boldsymbol{x}} &= \underset{\substack{\tilde{\boldsymbol{x}}\in\mathscr{X}, \\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}|_{\mathscr{A}}}}{\arg\min}\ \log\det\widetilde{\boldsymbol{C}}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) \\ &= \underset{\substack{\tilde{\boldsymbol{x}}\in\mathscr{X}, \\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}|_{\mathscr{A}}}}{\arg\max}\ \log\Big(q(\check{\boldsymbol{\mu}})\left(1 + \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M})\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M})\right) + \left(\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) - \boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}}, \boldsymbol{M})\right)^2\Big), \end{aligned} \tag{4.26}$$

where the arg min is changed into arg max because of the minus sign in (4.25) and $\log|\boldsymbol{A}|$ is neglected as it is independent of the next measurement location.

4.2.3. IMPACT OF THE L1-REGULARIZED LINEAR MEASUREMENT MODEL FOR THE EXPLORATION CRITERION

In Sec. 4.2.2 the cost function (2.9) is approximated by (4.9). This approximation involves the parameter δ for the sparse regularization. The influence of this parameter on the D-optimality criterion is not evaluated. Therefore, the following evaluates the D-optimality criterion for different values of δ in (4.9).

The influence of the regularization parameter is analyzed by considering a one-dimensional scenario with two measurements at $x[1] = -4$ and $x[2] = 4$ and two one-dimensional Gaussian basis functions as introduced in (2.10) with center positions $\mu[1] = x[1] = -4$ and $\mu[2] = x[2] = 4$. The width of the Gaussian basis functions is fixed at $\sigma = 0.3$ and equal for both considered basis functions. The weights $\widehat{\boldsymbol{w}}$ of this scenario are estimated centrally by (2.9).

Fig. 4.4 presents the D-optimality criterion for $\delta = \{0.001, \ldots, 500\}$ and $\widehat{\delta}_i = 2\delta/|\widehat{w}_i|$. In Fig. 4.4(a), $\log\det \widetilde{\boldsymbol{C}}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}})$ is optimized over a next possible measurement location $\tilde{\boldsymbol{x}}$ and over the next center position $\check{\boldsymbol{\mu}}$ of a new basis function (for every $\tilde{\boldsymbol{x}}$ the best possible $\check{\boldsymbol{\mu}}$ is considered). However, when looking at the definitions of the Gaussian basis function (2.10), its maximum is at the center position and, thus, $\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}})$ in (4.26) has the highest value when $\tilde{\boldsymbol{x}}$ and $\check{\boldsymbol{\mu}}$ coincide. The optimization with $\tilde{\boldsymbol{x}} = \check{\boldsymbol{\mu}}$ for Gaussian basis functions is presented in Fig. 4.4(b).

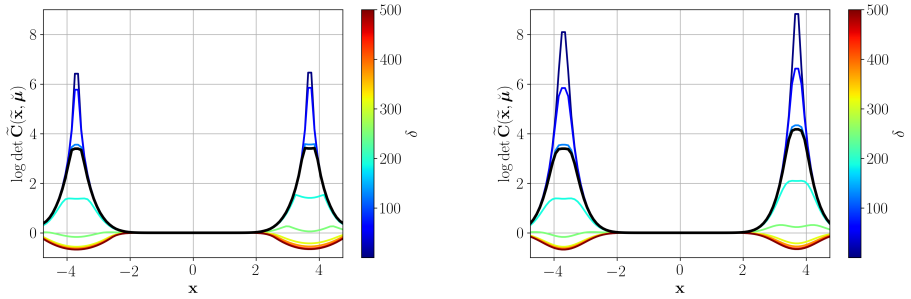


Figure 4.4: Variation of the regularization parameter δ and its influence on the D-optimality criterion under sparsity constraints. The black curve shows the case for $\delta = \lambda^{-\frac{1}{2}}\sqrt{2\log(M)}$ [74, Section 5.9], where $M = 2$ and λ is the precision of the measurement noise. (a) presents the optimization over $\tilde{\boldsymbol{x}}$ and $\check{\boldsymbol{\mu}}$ (for every $\tilde{\boldsymbol{x}}$ the best possible $\check{\boldsymbol{\mu}}$ is considered). (b) presents the same optimization with the additional constraint $\tilde{\boldsymbol{x}} = \check{\boldsymbol{\mu}}$. The smallest value of the D-optimality would be chosen as a next measurement position.

According to (2.9), if $\delta \to 0$ no regularization is used, and the LASSO objective function reduces to the standard least squares problem. In this case, the exploration criterion becomes highly peaked at the existing measurement locations, and the criterion would favor points that are further away from the measurement locations. If $\delta \to \infty$, there is only regularization in (2.9), and consequently most of the parameter weights converge towards zero. As can be seen in Fig. 4.4, for large values of δ the D-optimality criterion favors measurement locations that have already been measured. This behavior results from the diagonal loading in (4.16), which reflects the sparsity constraints in (2.9). A high value of δ leads to a high sparsity regularization and to a high diagonal loading value as $\widehat{\delta}_i = 2\delta/|\widehat{w}_i|$. This diagonal loading mechanism has the effect that it stabilizes the inversion of the covariance matrix by increasing the diagonal. Likewise, the variance of the corresponding parameter weights increases as well. As the D-optimality criterion aims at minimizing the covariance matrix (4.16), the criterion wants to minimize the

high variance of the parameter weights by measuring again at the same position. As a result, the agents do not explore but measure again at already measured locations to decrease the variance. In other words, if δ is large, a non-zero parameter weight somehow contradicts the D-optimality criterion under sparsity constraints and, consequently, the knowledge about this parameter weight has to be increased. If δ is fixed, this effect is controlled by the values of the active parameter weights, i.e., through the definition of $\delta_n \triangleq 2\delta/|w_n|,\ n = 1,\dots,N$.

In [74, Section 5.9], the authors suggest $\delta = \lambda^{-\frac{1}{2}}\sqrt{2\log(M)}$ as a penalty parameter for the regularization in (2.9), where λ is the precision of the measurement noise and M is the number of measurements. The corresponding value for this *adaptive* δ is highlighted as a black line in Fig. 4.4. In each figure it can be seen that this choice is a reasonable compromise, as it is in between a large δ and a small δ. Additionally, this choice makes δ adaptive to the number of obtained measurements.

4.2.4. Distributed D-Optimal Experiment Design for Heterogeneous Learning

In a distributed setting, the D-optimality criterion is not easily computable and requires cooperation between all agents. This section presents how to distribute the D-optimality criterion for the heterogeneous learning paradigm, which is introduced in Sec. 2.3.1.

In heterogeneous learning, every agent k has its own set of N_k basis functions, and every agent has access to all measurements $\boldsymbol{y}$ and all measurement locations $\boldsymbol{X}$. This is represented by splitting the design matrix into $\boldsymbol{\Phi}_{\mathcal{A}} = \left[\boldsymbol{\Phi}_{\mathcal{A},1},\dots,\boldsymbol{\Phi}_{\mathcal{A},K}\right]$ and the regularization term into $\widehat{\boldsymbol{\Delta}}_{\mathcal{A}} \triangleq \operatorname{diag}\left\{\widehat{\boldsymbol{\Delta}}_{\mathcal{A},1},\dots,\widehat{\boldsymbol{\Delta}}_{\mathcal{A},K}\right\}$. Every agent k has now only the local information $\boldsymbol{\Phi}_{\mathcal{A},k}$ and $\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}$ available, but it is also important to obtain knowledge about the other agents' measurements. One potential method to achieve insight about the other agents' measurements is the consensus algorithm [19, 20]. For applying the consensus algorithm on the D-optimality criterion the following is defined

$$\boldsymbol{H} \triangleq \boldsymbol{\Phi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Phi}_{\mathcal{A}}^{T} = \sum_{k=1}^{K}\boldsymbol{\Phi}_{\mathcal{A},k}\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\Phi}_{\mathcal{A},k}^{T}, \tag{4.27}$$

$$\boldsymbol{d}(\widetilde{\boldsymbol{x}}) \triangleq \boldsymbol{\Phi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) = \sum_{k=1}^{K}\boldsymbol{\Phi}_{\mathcal{A},k}\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}_k), \tag{4.28}$$

$$v(\widetilde{\boldsymbol{x}}) \triangleq \boldsymbol{\phi}_{\mathcal{A}}^{T}(\widetilde{\boldsymbol{x}},\boldsymbol{M})\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) = \sum_{k=1}^{K}\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}_k)^{T}\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}_k), \tag{4.29}$$

where $\boldsymbol{\phi}_{\mathcal{A}}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) = \left[\boldsymbol{\phi}_{\mathcal{A}}^{T}(\widetilde{\boldsymbol{x}},\boldsymbol{M}_1),\dots,\boldsymbol{\phi}_{\mathcal{A}}^{T}(\widetilde{\boldsymbol{x}},\boldsymbol{M}_K)\right]^{T}$. The definitions (4.27), (4.28), and (4.29) are suited for a distributed computation by means of an averaged consensus for heterogeneous learning [53]. The next paragraphs show how to express the D-optimality criterion with these definitions for heterogeneous learning.

First, $\boldsymbol{A}^{-1}$ in (4.25) can be computed by applying the Woodbury identity [116] as

$$\boldsymbol{A}^{-1} = \left(\boldsymbol{\Phi}_{\mathcal{A}}^{T}\boldsymbol{\Phi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}\right)^{-1} = \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1} - \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Phi}_{\mathcal{A}}^{T}\left(\boldsymbol{I} + \boldsymbol{\Phi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Phi}_{\mathcal{A}}^{T}\right)^{-1}\boldsymbol{\Phi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}. \tag{4.30}$$

Then, inserting (4.30) into $q(\check{\boldsymbol{\mu}})$ yields

$$\begin{aligned} q(\check{\boldsymbol{\mu}}) &= b(\check{\boldsymbol{\mu}}) - \boldsymbol{c}^T(\check{\boldsymbol{\mu}})\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}}) \\ &= \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\phi}(\boldsymbol{X},\boldsymbol{x}) - \boldsymbol{\phi}^T(\boldsymbol{X},\boldsymbol{x})\boldsymbol{\Phi}_{\mathscr{A}} \\ &\qquad \left(\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1} - \widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\Phi}_{\mathscr{A}}^T(\boldsymbol{I} + \boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\Phi}_{\mathscr{A}}^T)^{-1}\boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\right)\boldsymbol{\Phi}_{\mathscr{A}}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}) \\ &= \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}), \end{aligned} \tag{4.31}$$

where again the Woodbury identity is used. The other summands in the second logarithm of (4.25) can as well be expanded using the definition in (4.30). This results in

$$\begin{aligned} \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) &= \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M})\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) \\ &\quad - \boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M})\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\Phi}_{\mathscr{A}}^T\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) \\ &= v(\tilde{\boldsymbol{x}}) - \boldsymbol{d}(\tilde{\boldsymbol{x}})^T\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{d}(\tilde{\boldsymbol{x}}), \end{aligned} \tag{4.32}$$

and

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) =& \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) \\ &- \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\Phi}_{\mathscr{A}}^T\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) \\ =& \boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})^T\boldsymbol{d}(\tilde{\boldsymbol{x}}) - \boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})^T\boldsymbol{H}\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{d}(\tilde{\boldsymbol{x}}) \\ =& \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\left(\boldsymbol{I} - \boldsymbol{H}\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\right)\boldsymbol{d}(\tilde{\boldsymbol{x}}). \end{aligned} \tag{4.33}$$

All equations (4.31), (4.32), and (4.33) can be calculated in a distributed fashion by using (4.27), (4.28), and (4.29). To calculate the D-optimality criterion distributively for heterogeneous learning, the last step is to insert (4.31), (4.32), and (4.33) into (4.25), which leads to

$$\begin{aligned} \log\det\widetilde{\boldsymbol{C}}(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) = -\log|\boldsymbol{A}| - \log\Big(&\boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}) \\ &+ (\boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}))(v(\tilde{\boldsymbol{x}}) - \boldsymbol{d}(\tilde{\boldsymbol{x}})^T\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\boldsymbol{d}(\tilde{\boldsymbol{x}})) \\ &+ \left(\phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \left(\boldsymbol{I} - \boldsymbol{H}\left(\boldsymbol{I} + \boldsymbol{H}\right)^{-1}\right)\boldsymbol{d}(\tilde{\boldsymbol{x}})\right)^2\Big), \end{aligned} \tag{4.34}$$

where the terms $\boldsymbol{H}$, $\boldsymbol{d}(\tilde{\boldsymbol{x}})$, and $v(\tilde{\boldsymbol{x}})$ are computed by a consensus with (4.27), (4.28), and (4.29), respectively. Because $\boldsymbol{d}(\tilde{\boldsymbol{x}})$ and $v(\tilde{\boldsymbol{x}})$ depend on $\tilde{\boldsymbol{x}}$, it is worth noting that if (4.34) is computed locally for agent k, the hypothetical measurement location $\tilde{\boldsymbol{x}}$ has to be distributed in the network before the consensus can commence. Unlike SOE, SOF does not require to distribute a potential new center position $\check{\boldsymbol{\mu}}$ because by virtue of SOF each agent can choose this independently; the definitions (4.27), (4.28), and (4.29) require only previously known center positions.

The next measurement location is then found by minimizing (4.34) over all possible

Algorithm 8 distributed exploration with SOF for LASSO (DE-SOF-LASSO)

$\boldsymbol{H} \leftarrow$(4.27) ▷ Consensus
for $\widetilde{\boldsymbol{x}} \in \mathscr{X}$ **do**
  Send $\widetilde{\boldsymbol{x}}$ to all neighbors
  $\boldsymbol{d}(\widetilde{\boldsymbol{x}}) \leftarrow$(4.28) ▷ Consensus
  $v(\widetilde{\boldsymbol{x}}) \leftarrow$(4.29) ▷ Consensus
$\widehat{\boldsymbol{x}}_k \leftarrow$ (4.35)

next measurement locations and center positions as shown in (4.26) such that

$$\begin{aligned}\widehat{\boldsymbol{x}}_k &= \underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}_k}}{\arg\min}\ \log\det\widetilde{\boldsymbol{C}}(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) \\ &= \underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}_k}}{\arg\max}\ \log\Big(\boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\,(\boldsymbol{I}+\boldsymbol{H})^{-1}\,\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}) \\ &\qquad + (\boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\,(\boldsymbol{I}+\boldsymbol{H})^{-1}\,\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}))(v(\widetilde{\boldsymbol{x}}) - \boldsymbol{d}(\widetilde{\boldsymbol{x}})^T(\boldsymbol{I}+\boldsymbol{H})^{-1}\boldsymbol{d}(\widetilde{\boldsymbol{x}})) \\ &\qquad + \left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \left(\boldsymbol{I}-\boldsymbol{H}\,(\boldsymbol{I}+\boldsymbol{H})^{-1}\right)\boldsymbol{d}(\widetilde{\boldsymbol{x}})\right)^2\Big). \end{aligned} \tag{4.35}$$

To find $\widehat{\boldsymbol{x}}$, an agent k distributes only the potential next measurement location $\widetilde{\boldsymbol{x}}$ to its neighbors to compute $\boldsymbol{d}(\widetilde{\boldsymbol{x}})$ and $v(\widetilde{\boldsymbol{x}})$ distributively. The advantage of heterogeneous learning is that agent k does not need to know the other potential candidate basis functions of its neighbors, i.e. $\boldsymbol{\phi}$, because each agent chooses the center positions independently. Consequently, for local basis functions, the center positions have to be known only locally (global basis functions have no center positions but are characterized by another parameter). This feature of the heterogeneous learning opens up the possibility of further distribution schemes with *specialized* agents, which focus only on parts of the observed process, i.e. one agent for high frequency components and another for low frequency components of the observed process. Yet, the analysis of this is not within the scope of this thesis.

The distributed exploration with SOF for LASSO (DE-SOF-LASSO) for estimating new measurement locations $\widehat{\boldsymbol{x}}$ is shown in Alg. 8 as pseudocode for agent k. As can be seen in Alg. 8, three consensus iterations are required to estimate the next optimal measurement location $\widehat{\boldsymbol{x}}_k$. The communication load in the network is depending also on the network topology. However for each consensus iteration to compute $\boldsymbol{H}$, agent k has to transmit $M(M-1)/2$ floats to its neighbors as $\boldsymbol{\Phi}_{\mathscr{A},k}\widehat{\boldsymbol{\Delta}}^{-1}_{\mathscr{A},k}\boldsymbol{\Phi}^T_{\mathscr{A},k}$ is a symmetric matrix. Likewise, to compute $\boldsymbol{d}(\widetilde{\boldsymbol{x}})$ and $v(\widetilde{\boldsymbol{x}})$ each consensus iteration requires $M+1$ floats to be transmitted for each $\widetilde{\boldsymbol{x}}$, respectively. Furthermore, $\widetilde{\boldsymbol{x}}$ has to be transmitted beforehand. Let $I_\mathrm{c} \in \mathbb{N}$ be the number of consensus iterations in a network and $|\mathscr{X}| \in \mathbb{N}$ the number of hypothetical measurement positions $\widetilde{\boldsymbol{x}} \in \mathbb{R}^2$. Then, DE-SOF-LASSO requires

$$I_\mathrm{c}\left(\frac{M(M-1)}{2} + |\mathscr{X}|(M+1)\right) + 2|\mathscr{X}| = I_\mathrm{c}\left(\frac{M^2}{2} + \frac{|\mathscr{X}|-1}{2}M + |\mathscr{X}|\right) + 2|\mathscr{X}| \tag{4.36}$$

floats to be transmitted for agent k. This consensus is dominated by M^2 and therefore the communication load depends on $\mathscr{O}(M^2)$.

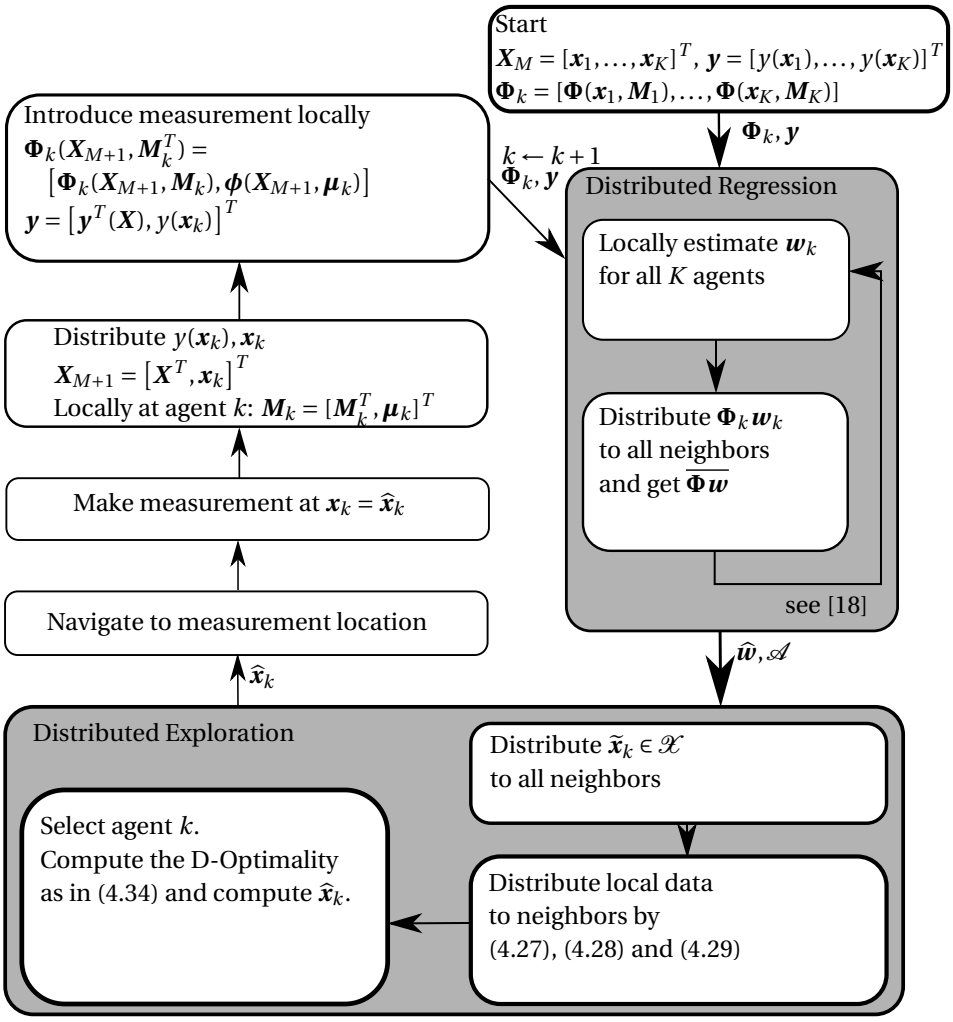


Figure 4.5: This flow-chart summarizes the whole swarm exploration for heterogeneous learning. At start every agent took a measurement and, then, the k-th agent starts with the exploration process. After the agent k has reached its next measurement location and each agent has incorporated the new measurement, agent $k+1$ starts with the exploration.

Figure 4.5 summarizes the whole swarm exploration for heterogeneous learning in a block diagram. At the beginning, every agent takes a measurement at it current location, which is then distributed and introduced into the model of each agent as $\boldsymbol{X}_M = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]^T$, $\boldsymbol{y} = \left[\boldsymbol{y}(\boldsymbol{x}_1), y(\boldsymbol{x}_K)\right]^T$, and locally $\boldsymbol{\Phi}(\boldsymbol{X}_M, \boldsymbol{M}_k)$. Afterward, the distributed regression is applied to infer the new model parameter weights. This is followed by the distributed exploration, which estimates new measurement locations according to (4.35).

Once $\widehat{\boldsymbol{x}}_k$ is estimated, agent k has to measure at this location before the next agent $k+1$ can continue. Thus, agent k navigates to its next measurement location and measures such that the new measurement can be incorporated locally and distributed to the other agents. The new estimated model is then used to estimate the next measurement location for agent $k+1$. Figure 4.5 depicts also that heterogeneous learning involves multiple stages where data has to be distributed: the new measurements and measurement locations, during the regression, and two times during the evaluation of the D-optimality criterion. As for Gaussian basis functions, it is reasonable to select $\boldsymbol{x}_k$ also as the center position, i.e. $\boldsymbol{\mu}_k = \boldsymbol{x}_k$, because the Gaussian basis function is maximal at its center.

4.2.5. Distributed D-Optimal Experiment Design for Homogeneous Learning

Compared with heterogeneous learning, in homogeneous learning all K agents have the same parameter weights $\boldsymbol{w}$. Also, every agent k has its own measurements $\boldsymbol{y}_k$ and its own design matrix $\boldsymbol{\Phi}_k(\boldsymbol{X}_k)$, with $\boldsymbol{y} = \left[\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_K^T\right]^T$, $\boldsymbol{\Phi}(\boldsymbol{X}) = \left[\boldsymbol{\Phi}_1^T(\boldsymbol{X}_1), \ldots, \boldsymbol{\Phi}_K^T(\boldsymbol{X}_K)\right]^T$, and $\boldsymbol{X} = \left[\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_K^T\right]^T$. The number of each agent's measurements does not have to be necessarily equal for all agents, but the number N of basis functions is the same for all agents. When applying homogeneous learning, the measurements are treated in a distributed fashion and this distribution paradigm does not require further communication for the measurements because the measurements stay local. The measurement model – the basis functions and their number, i.e., the model size – is known a-priori for all agents.

This section shows how to distribute the D-optimality presented in (4.25) for homogeneous learning. Each agent k computes the vector $\boldsymbol{\phi}_{\mathscr{A}}(\widetilde{\boldsymbol{x}}, \boldsymbol{M})$ locally because the N basis functions are known to all agents. If localized basis functions are used, also the center positions $\boldsymbol{M}$ are known to all agents by design. Likewise, $\phi(\widetilde{\boldsymbol{x}}, \breve{\boldsymbol{\mu}})$ as well as $\widehat{\boldsymbol{\Delta}}_{\mathscr{A}}$ are locally known quantities. The D-optimality criterion (4.25) is defined by variables that are calculated by products of these locally known quantities due to the cooperative estimation. It is, therefore, possible to make their computation distributed in the network for homogeneous learning without further reformulation. Thus, the variables in (4.19) can be redefined for a consensus as

$$\boldsymbol{A} \triangleq \boldsymbol{\Phi}_{\mathscr{A}}^T \boldsymbol{\Phi}_{\mathscr{A}} + \widehat{\boldsymbol{\Delta}}_{\mathscr{A}} = \widehat{\boldsymbol{\Delta}}_{\mathscr{A}} + \sum_{k=1}^{K} \boldsymbol{\Phi}_{\mathscr{A}}^T(\boldsymbol{X}_k) \boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X}_k), \tag{4.37}$$

$$\boldsymbol{c}(\breve{\boldsymbol{\mu}}) \triangleq \boldsymbol{\Phi}_{\mathscr{A}}^T \boldsymbol{\phi}(\boldsymbol{X}, \breve{\boldsymbol{\mu}}) = \sum_{k=1}^{K} \boldsymbol{\Phi}_{\mathscr{A}}^T(\boldsymbol{X}_k) \boldsymbol{\phi}(\boldsymbol{X}_k, \breve{\boldsymbol{\mu}}), \tag{4.38}$$

$$b(\breve{\boldsymbol{\mu}}) \triangleq \boldsymbol{\phi}^T(\boldsymbol{X}, \breve{\boldsymbol{\mu}}) \boldsymbol{\phi}(\boldsymbol{X}, \breve{\boldsymbol{\mu}}) = \sum_{k=1}^{K} \boldsymbol{\phi}^T(\boldsymbol{X}_k, \breve{\boldsymbol{\mu}}) \boldsymbol{\phi}(\boldsymbol{X}_k, \breve{\boldsymbol{\mu}}). \tag{4.39}$$

In contrast to heterogeneous learning, the above defined consensus terms depend on the center positions of potential basis functions, if localized basis functions are used in the model. If global basis functions are used, e.g. DCTs, the center positions would be substituted by e.g. frequency components. It is worth stressing that N is defined to be fixed here and all possible basis functions are known to all agents. As a consequence,

Algorithm 9 distributed exploration with SOE for LASSO (DE-SOE-LASSO) for agent k

$\boldsymbol{A} \leftarrow$(4.37) ▷ Consensus
for all $\check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}$ **do**
  $\boldsymbol{c}(\check{\boldsymbol{\mu}}) \leftarrow$(4.38) ▷ Consensus
  $b(\check{\boldsymbol{\mu}}) \leftarrow$(4.39) ▷ Consensus
$\widehat{\boldsymbol{x}}_k \leftarrow$ (4.26)

for agent k the position of a potential basis function does not need to be transmitted to agent k's neighbors; all basis functions and their order of appearance in the model is defined beforehand. Thus, if agent k estimates the D-optimality criterion in homogeneous learning, it is only necessary to evaluate the center positions $\check{\boldsymbol{\mu}}$ that are currently not in $\boldsymbol{M}|_{\mathscr{A}}$. Furthermore, to make the computation of $\boldsymbol{c}(\check{\boldsymbol{\mu}})$ and $b(\check{\boldsymbol{\mu}})$ faster, they can be computed in a single consensus for all $\check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}$. By virtue of the homogeneous learning, each consensus to calculate $\boldsymbol{A}$, $\boldsymbol{c}(\check{\boldsymbol{\mu}})$, and $b(\check{\boldsymbol{\mu}})$ is independent on the hypothetical measurement position $\tilde{\boldsymbol{x}}$. This is different to heterogeneous learning, where the potential measurement locations have to be communicated.

In Alg. 9 the distributed exploration with SOE for LASSO (DE-SOE-LASSO) is presented. As can be seen in Alg. 9, the DE-SOE-LASSO requires one initial consensus and two consensus steps for each potential center position. As mentioned earlier the latter two consensus steps can be computed in bulk as here all agents know all possible center positions $\check{\boldsymbol{\mu}}$ and their order of appearance by virtue of the distribution paradigm. Because of this, each agent can locally compute $\boldsymbol{c}(\check{\boldsymbol{\mu}})$ and $b(\check{\boldsymbol{\mu}})$ without further communication and re-use $\boldsymbol{A}$ from the parameter estimation. The communication load for each consensus step depends on the network topology. To calculate the communication load per agent k, let I_c define the number of consensus iterations and $N_{\mathscr{A}} = |\mathscr{A}|$ is the number of active basis functions in the model. Then, the consensus step for $\boldsymbol{A}$ requires $I_c N_{\mathscr{A}}(N_{\mathscr{A}}-1)/2$ floats to be sent per consensus for agent k, where already the symmetry of $\boldsymbol{A}$ is exploited. For the computation of $\boldsymbol{c}(\check{\boldsymbol{\mu}})$ and $b(\check{\boldsymbol{\mu}})$, $I_c N_{\mathscr{A}}$ floats and I_c floats are transmitted per agent k. Therefore, agent k communicates

$$I_c\left(\frac{N_{\mathscr{A}}(N_{\mathscr{A}}-1)}{2} + N_{\mathscr{A}} + 1\right) = I_c\left(\frac{N_{\mathscr{A}}^2}{2} + \frac{N_{\mathscr{A}}}{2} + 1\right) \tag{4.40}$$

floats to calculate the D-optimality criterion. This is dominated by $N_{\mathscr{A}}^2$.

Figure 4.6 summarizes the distributed exploration in more detail from the point-of-view of agent k. The exploration starts for each agent k by making a measurement $y(\boldsymbol{x}_k)$ at the location $\boldsymbol{x}_k$. This measurement is then introduced locally in the model by $\boldsymbol{X}_{M_k+1} = \left[\boldsymbol{X}_k^T, \boldsymbol{x}_k\right]^T$, $\boldsymbol{y}_k = \left[\boldsymbol{y}_k^T, y(\boldsymbol{x}_k)\right]^T$, and $\boldsymbol{\Phi}_k(\boldsymbol{X}_{M_k+1}, \boldsymbol{M}) = \left[\boldsymbol{\Phi}_k^T(\boldsymbol{X}, \boldsymbol{M}), \boldsymbol{\phi}(\boldsymbol{x}_k, \boldsymbol{M})\right]^T$. Then, the parameter weights have to be estimated with the new measurements. The distributed regression can be implemented as suggested by the literature, e.g. [18, 98]. Once the weights $\widehat{\boldsymbol{w}}$ are estimated, the model is passed to the exploration. For the exploration, the aforementioned consensus terms ((4.37), (4.38), and (4.39)) and the evaluation of the D-optimality criterion (4.25) are used to estimate the next measurement location for the agent k. This agent then navigates to its next measurement position and takes a mea-

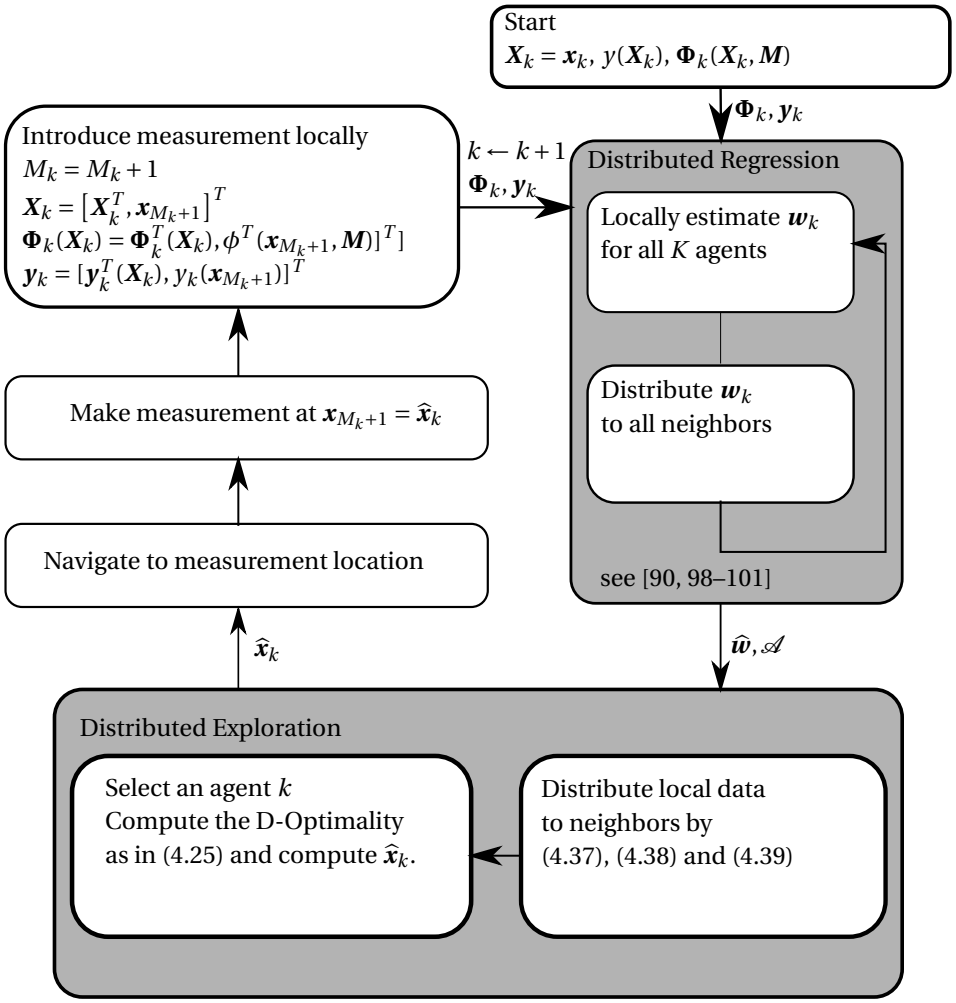


Figure 4.6: This flow-chart summarizes the whole swarm exploration for homogeneous learning. At the beginning each agent makes a measurement at its current position. Then the exploration process starts for agent k by first estimating the parameter weights and afterward estimating a next measurement location. Agent k then navigates to the measurement location and introduces this measurement into its local model. Subsequently, agent $k+1$ continues with the exploration.

surement there. Then again every agent incorporates the single new measurement and the next agent $k+1$ would start exploring. It is worth noting that, to this end, the exploration is applied sequentially; agent k explores and the other agents wait until agent k reached its measurement point, then agent $k+1$ explores, etc. The figure also highlights the data that is passed to each part of the swarm exploration.

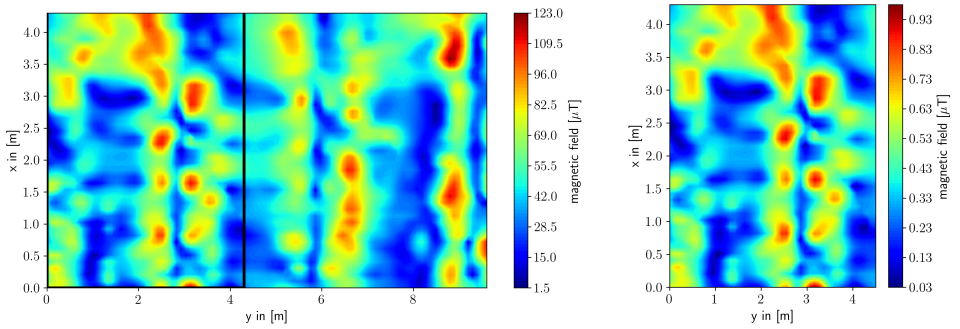
	SOE	SOF
Estimation	Distribute $\boldsymbol{w} \in \mathbb{R}^N$	Distribute $\boldsymbol{\Phi}_k \boldsymbol{w}_k \in \mathbb{R}^M$ and $\{\boldsymbol{x}_m, y_m\}$ for each new measurement
	Fixed number of basis functions N	More flexibility for selecting basis functions
Exploration	Estimate $\boldsymbol{A} \in \mathbb{R}^{N_\mathscr{A} \times N_\mathscr{A}}$, $\boldsymbol{c}(\check{\boldsymbol{\mu}}) \in \mathbb{R}^{N_\mathscr{A}}$ and $b(\check{\boldsymbol{\mu}}) \in \mathbb{R}$ by means of a consensus, where $N_\mathscr{A}$ is the number of all active basis functions. Each agent communicates its part. $\frac{N_\mathscr{A}^2}{2} + \frac{N_\mathscr{A}}{2} + 1$ values have to be communicated per agent and consensus step.	Estimate $\boldsymbol{H} \in \mathbb{R}^{M \times M}$, $\boldsymbol{d}(\tilde{\boldsymbol{x}}) \in \mathbb{R}^M$ and $v(\tilde{\boldsymbol{x}}) \in \mathbb{R}$ by means of a consensus. $\frac{M^2}{2} + \frac{\lvert\mathscr{X}\rvert - 1}{2} M + \lvert\mathscr{X}\rvert$ have to be transmitted per agent and consensus iteration.
	$\check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}\rvert_\mathscr{A}$ does not need to be communicated, because every agent knows all basis functions.	The center positions $\check{\boldsymbol{\mu}}$ are only locally needed. Thus, no communication is required to optimize these.
	Measurements are known locally. Thus $\tilde{\boldsymbol{x}} \in \mathscr{X}$ does not need to be communicated.	All measurements need to be communicated as well as the proposed positions $\tilde{\boldsymbol{x}} \in \mathscr{X}$.
	The selection of new basis functions requires communication with all agents because every agent uses the same model.	Basis functions are optimized and set individually by each agent.

Table 4.1: Differences between the distribution paradigms for estimation and exploration.

Table 4.1 summarizes the different aspects for both distribution paradigms for estimation and exploration. Table 4.1 summarizes also the values that need to be transmitted per proposed measurement location. For SOE, this amount depends on the number of used basis functions N and for SOF this amount depends on the number of measurements M. If the number of measurements is low, e.g. at the beginning of the exploration, SOF has an advantage in terms of communication load. On the other hand, SOE is advantageous if the number of basis functions N is larger than M.

4.2.6. EVALUATION OF D-OPTIMAL EXPERIMENT DESIGN FOR EXPLORATION WITH REAL DATA

This section tests the proposed criterion for exploration and for different distribution paradigms. The unknown process that is explored in this section is the magnetic field



(a) (b)

Figure 4.7: (a) Magnetic field of the Holodeck at DLR premises. This plot displays the magnetic field intensity in color coding. A black rectangle highlights the cut out of the magnetic field, which is shown again in (b) and is used for the simulations.

measured on the floor of the laboratory, which is called *Holodeck*. The magnetic field was measured by a Xsens MTx[1] and was also used in previous literature [58]. The magnetic field is shown in Fig. 4.7(a). The data was sampled on a regular grid $\mathscr{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{|\mathscr{X}|}\}$ with $|\mathscr{X}| = 4128$.

The design matrix in this particular test is built with Gaussian basis functions, c.f. (2.10). The width of each basis function is $\sigma = 30\,\text{cm}$. The number of agents is $K = 10$, and the starting position of each agent is drawn at random uniformly from $\mathscr{X}$. For the homogeneous learning, it is important to define the center positions for the used basis functions in the design matrix. Therefore, for SOE, each point in $\mathscr{X}$ is used as a center point for the basis functions. For heterogeneous learning the construction of the design matrix $\boldsymbol{\Phi}_k(\boldsymbol{X}, \boldsymbol{M})$ of each agent k can be arbitrary. Therefore, for SOF, in this experiment and also in the following of this thesis, the center positions of the basis functions are set equal to the measurement positions. It is assumed that the agents can move to the next position instantly and thus no collision avoidance is considered. For the estimation of the parameter weights a distributed ADMM is employed [18, Section 3.4], with $\rho = 1.0$ for the homogeneous learning. This parameter has been evaluated in cross validations and is data dependent. For heterogeneous learning, the ADMM penalty parameter is learned by residual balancing, c.f. Sec. 2.4.4, and it is initialized with $\rho = 1.0$. Likewise to Sec. 4.2.3, the LASSO penalty parameter is chosen as $\delta = \lambda^{-\frac{1}{2}}\sqrt{2\log(M)}$. A cross validation of δ and the width of the Gaussian basis functions σ is presented in the Appendix A.

The proposed methods are benchmarked against a systematic exploration and a random exploration, i.e. meander and random walk respectively. The benchmarks are therefore explained in more detail in the following.

Meander: For this exploration, all agents start at an individual predefined position, which is at $\boldsymbol{x}_k[0] = \left[1.5, \frac{4.3}{k}\right]^T$ in this simulation, see Fig. 4.7(a). The agents move in a back

[1] https://www.xsens.com/

and forth pattern, starting from the left border to the right. Their step size is related to the Gaussian basis functions to create an overlap between neighboring basis functions. Hence, the step size is set to 30 cm.

Random walk: The agents start at a random position, and their next measurement location is defined by selecting a point random uniformly from $\mathscr{X}$.

Note that both benchmarks are non-cooperative; the agents do not exchange any data or information to do the exploration. The proposed methods are used as described in Alg. 8 and Alg. 9 for heterogeneous learning and homogeneous learning respectively.

As evaluation metric, the NRMSE is used for this simulation as

$$\epsilon_{\text{SOE}} = \frac{\|\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M})\,\boldsymbol{w} - \boldsymbol{f}\|^2}{\|\boldsymbol{f}\|^2}, \tag{4.41}$$

$$\epsilon_{\text{SOF}} = \frac{\left\| \sum_{k=1}^{K} \boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M}_k)\,\boldsymbol{w}_k - \boldsymbol{f} \right\|^2}{\|\boldsymbol{f}\|^2}, \tag{4.42}$$

for SOE and SOF, respectively, where $\boldsymbol{f} = \left[f(\boldsymbol{x}_0),\ldots,f(\boldsymbol{x}_{|\mathscr{X}|})\right]^T \in \mathbb{R}^{|\mathscr{X}|}, \boldsymbol{x} \in \mathscr{X}$ is the true magnetic field discretized at the locations in $\mathscr{X}$.

A prediction of the magnetic field for all exploration strategies and distribution paradigms is displayed in Fig. 4.8, after the estimation of $\widehat{\boldsymbol{w}}$. The left side in Fig. 4.8 shows the estimations for homogeneous learning, and the right side are the estimations for heterogeneous learning. Each row in Fig. 4.8 corresponds to an exploration strategy, as noted by the axis label. The colored circles represent the measurement locations of each agent. The estimated center position of the basis functions, which correspond to a non-zero weight, are shown by black crosses.

Both of the proposed strategies tend to make measurements at the border of the environments, as indicated by the colored circles. This effect is of numerical nature and has also been observed in [29]. Yet, the proposed methods yield a good coverage of the area as the measurement locations are well distributed. For the meander strategy this coverage depends obviously on the choice of the step size. In this scenario, either the step size should be increased or more measurements would have been required to achieve better coverage. However, the step size here is chosen according to the width of the local basis function and changing the step size might lead to smaller overlap between the basis functions and might therefore lead to a lower sparsity or a higher error. Also, increasing the number of measurements solely for the meander might be an unfair comparison. The random uniform sampling from $\mathscr{X}$ of the random walk does not consider already measured positions. Furthermore, as only a finite number of measurements is considered here, it might happen that parts of the area are not sampled at all.

The movement direction and the step size of the meander influence how the magnetic field is measured and subsequently how each measurement reduces the NRMSE. For the meander approach this is even independent of the distribution paradigm as shown in Fig. 4.9. A similar behaviour, independent on the distribution paradigm, can be seen for the random walk in Fig. 4.9 but less pronounced. This figure also shows that

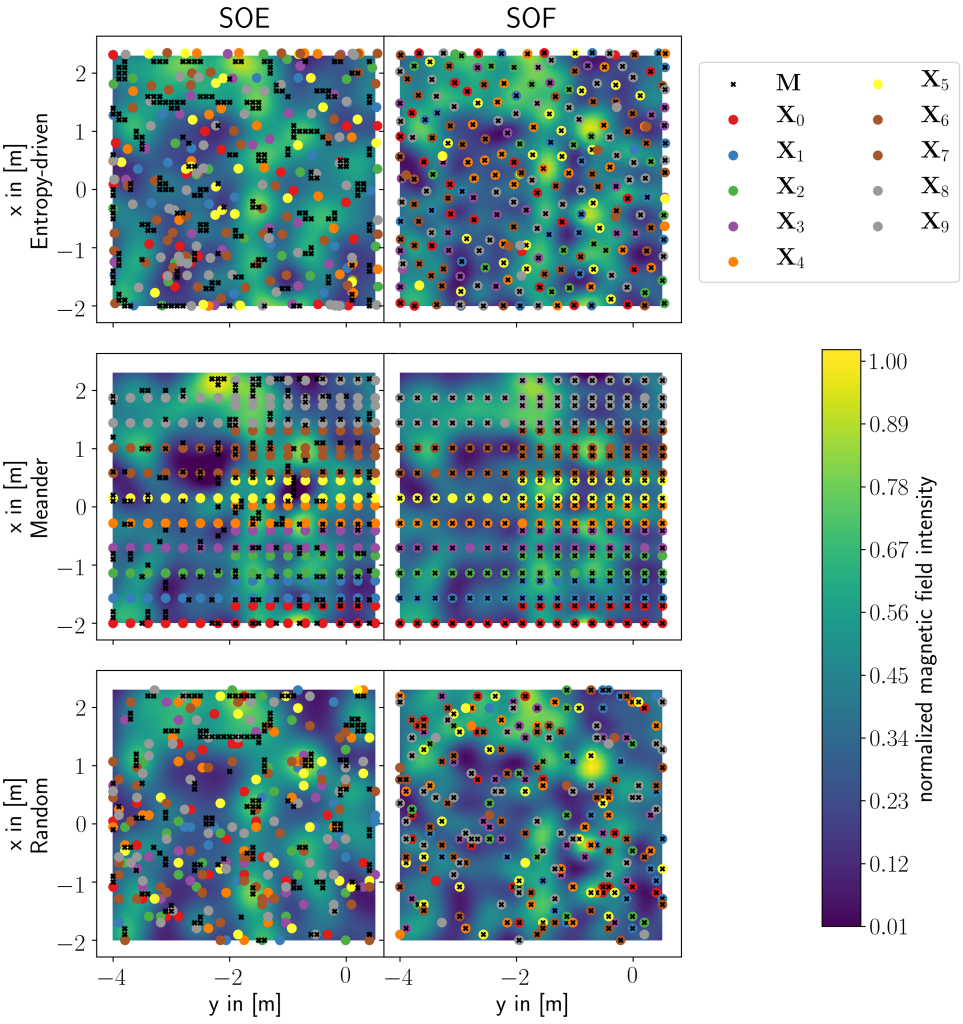


Figure 4.8: Estimates of a single run for all distribution paradigms and methods compared. All estimates are normalized to one. Colored circles represent measurement position, where each agent has a different color. The black crosses show the center positions of basis functions, which correspond to a non-zero parameter weight.

the D-optimality reduces the NRMSE faster than the benchmark methods for both distribution paradigms. Yet, the performance of the D-optimality for SOF is better compared with the performance for the D-optimality for SOE.

Because heterogeneous learning places the center of a basis function at measurement locations, the ratio of M/N is close to one, and thus the estimation might lead to smaller overfitting. The way how basis functions are introduced in SOF controls the

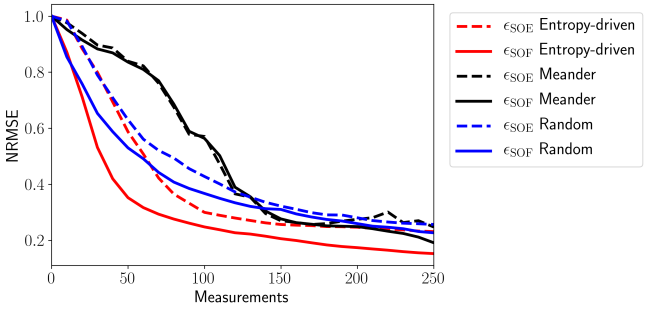
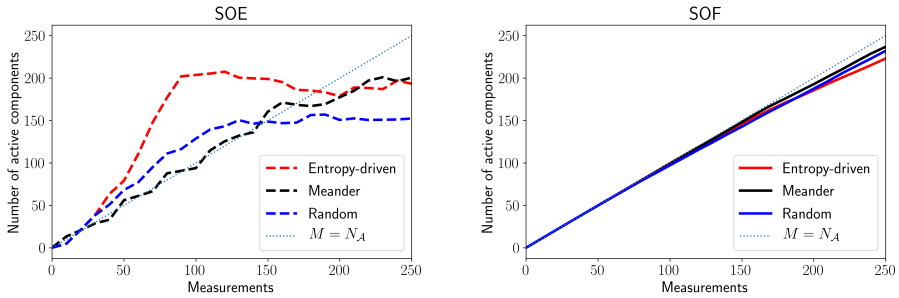


Figure 4.9: NRMSE for homogeneous learning (dashed) and heterogeneous learning (solid) for $K = 10$ agents, averaged over 10 runs.



(a) (b)

Figure 4.10: (**a**) Active components for homogeneous learning (dashed) for $K = 10$ agents averaged over 10 runs; (**b**) active components for heterogeneous learning (solid) for $K = 10$ agents averaged over 10 runs.

number of active parameter weights and they cannot be larger then the number of measurements. This can be seen in Fig. 4.10(a), which shows the number of active parameter weights and is explained later. For homogeneous learning this ratio is small because $M \leq N$ and therefore this method could tend to overfitting. This effect could be reduced by changing the penalty parameter δ which also could lead to a higher error as the result becomes sparser. Another reason might be that the criterion for convergence in SOE is not reflecting the sparsity and that the parameter estimation actually requires further iterations. This assumption could be supported by the fact that many selected center locations are close to each other. Figures 4.10(a) and 4.10(b) display the number of non-zero parameter weights, i.e. $|\mathscr{A}|$, versus the number of measurements. Indeed, for homogeneous learning, the number of active components versus the number of measurements is larger at the beginning, c.f. Fig. 4.10(a). Once a certain number of measurements is reached, the regularization becomes more aggressive because $\delta = \lambda^{-\frac{1}{2}}\sqrt{2\log(M)}$, and also the ratio M/N becomes better with increasing M. A reason for this effect could be that $\check{\boldsymbol{\mu}}$ is optimized as well by the criterion such that always relevant center positions

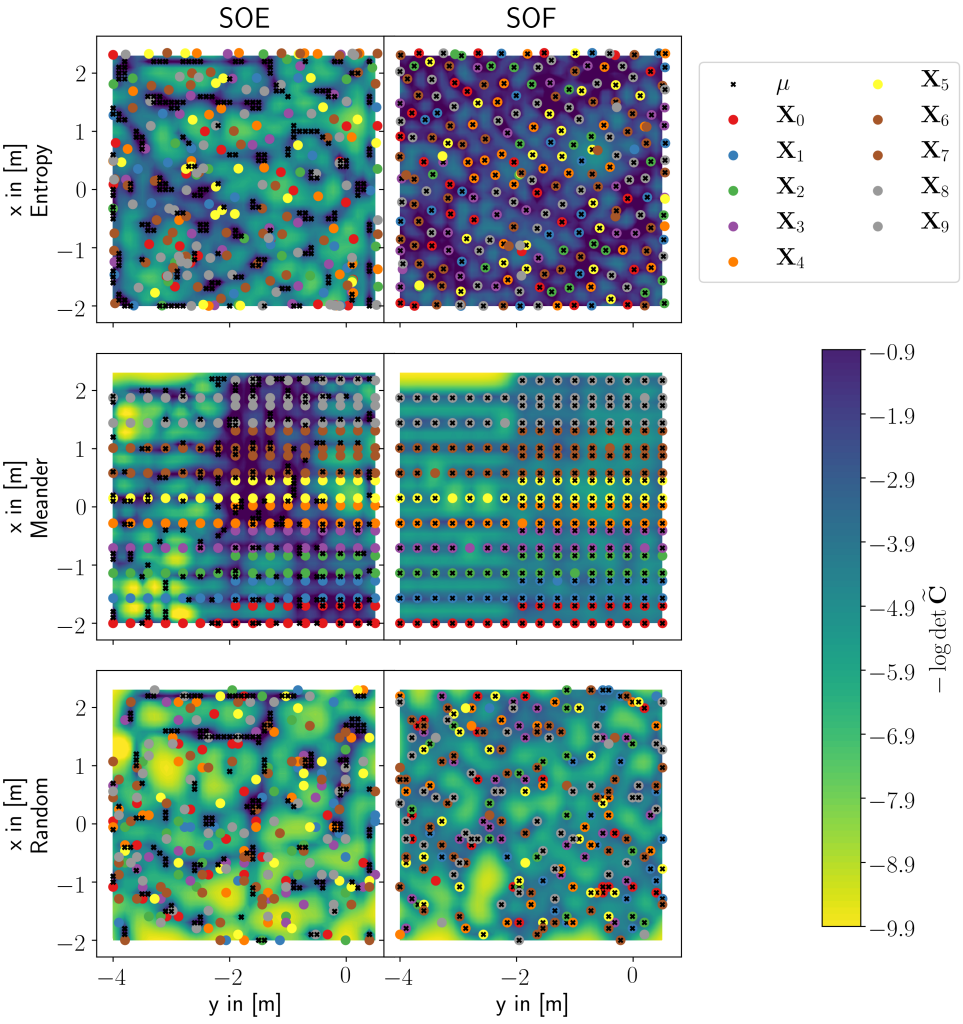
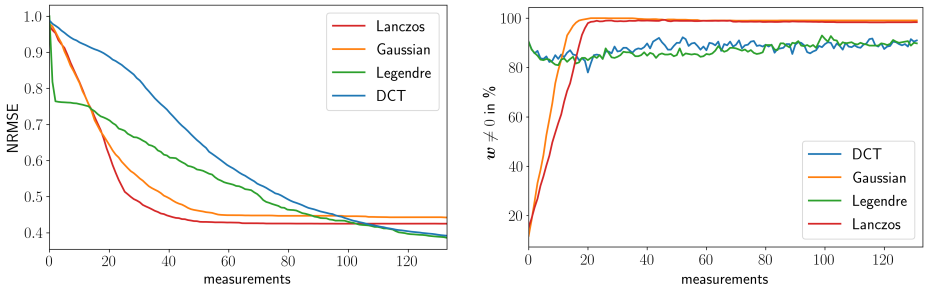


Figure 4.11: D-optimalities of a single run for SOE and SOF. The color coding of the entropies is the same for all plots. Colored circles represent measurement positions, where each agent has a different color. The black crosses show the center positions of basis functions, which correspond to a non-zero parameter weight.

are chosen. Thus, $N_{\mathscr{A}}$ is also increasing. Once the exploration area has been explored enough, the number of active basis functions – and also $N_{\mathscr{A}}$ – stabilizes as new center locations do not contribute to the criterion as much as before. For heterogeneous learning, the ratio M/N is almost constant. Only at the end, when the density of measurements is high enough, some measurements do not lead to a new basis function with non-zero parameter weight. At the end of the simulation, the D-optimality criterion is calculated for all exploration strategies and shown in Fig. 4.11 for all distribution paradigms. There,



(a) (b)

Figure 4.12: Influence of using different basis functions on the estimation performance. Figure (a) shows the NRMSE, and Fig. (b) displays the non-zero components of $\boldsymbol{w}$ in percent.

the left column refers to the SOE paradigm and the right column refers to the SOF paradigm whereas the exploration methods are depicted in the rows. The color scaling is the same for all plots. For the entropy-driven approach, the D-optimality criterion is more evenly reduced compared with the other methods. Homogeneous learning leads to a slightly worse minimization of the D-optimality criterion compared with heterogeneous learning, most likely due to because of the poor ratio M/N. For random walk and meander, the entropy yields some highly informative locations, which would yield a high information gain, but they have not been actively approached.

4.3. INFLUENCE OF BASIS FUNCTIONS ON THE D-OPTIMALITY CRITERION FOR EXPLORATION

This section analyzes the influence of different basis functions on the D-optimality. Thus, the magnetic field data, as introduced in Sec. 4.2.6, is used to test different types of basis functions, which are introduced in Sec. 2.1.3. It is again assumed that the agents can move in the environment instantly to their measurement locations. The number of agents is set to $K = 3$, and the algorithm to estimate the parameter weights is the ADMM as introduced in [18] for homogeneous learning. The penalty parameter for the ADMM is initially set to $\rho = 10$, but it is adaptive. To reduce the complexity, the data set is sampled on a regular grid with $|\mathscr{X}| = 2280$ with 30 cells in x-direction and 76 cells in y-direction.

The Gaussian basis function and the Lanczos basis function have parameters, which define the width of each basis function, see Sec. 2.1.3. These parameters are chosen as $\sigma_{\text{gauss}} = 0.2$ for the Gaussian basis function and $\sigma_{\text{lanczos}} = 0.5$ for the Lanczos basis function. Additionally, Legendre polynomials and the DCT are employed. The Legendre polynomials are two-dimensional with a degree of 25 in x- and y-direction, which leads to 626 basis functions including the bias, see Sec. 2.1.3. The DCT basis functions vary in frequency from 0 until 100 Hz in x- and y-direction in 5 Hz steps. Therefore, there are 401 basis functions, including the bias.

The simulation results are shown in Figs. 4.12(a) and 4.12(b) for the considered basis

functions with the D-optimality criterion for homogeneous learning. Considering Fig. 4.12(a), the localized basis functions — the Gaussian and Lanczos basis functions — reduce the NRMSE faster compared to the global basis functions — Legendre polynomials and the DCT. At the end, however, the global basis functions slightly outperform the localized basis functions. Fig. 4.12(a) also shows that regardless of the basis function, the D-optimality criterion still reduces the NRMSE effectively. Regarding the sparsity, presented in Fig. 4.12(b), the parameter weights of all basis functions can be considered to be not sparse in this case. Yet, the low sparsity is also influenced by the parameters δ and ρ. The first parameter determines directly the promoted sparsity. The second parameter is a regularization parameter of the ADMM algorithm and influences the convergence speed, see (2.21). Other influential parameters for the sparsity are the combination of width and the regular sampling grid for calculating the prediction for local basis functions or the relative small number of basis functions, which was small due to computational reasons, for the global basis functions.

The simulation indicates that the D-optimality is able to reduce the error faster with local basis functions compared to global basis functions. However, the global basis functions seem to represent the considered process better as fewer basis functions are used and fewer basis functions are considered in the model. Moreover, the error curves flatten out with more measurements as the evaluation depends on the discretization and the choice of the width of the basis functions. Global basis functions do not depend on a width. For local basis functions, this suggests to make either the discretization or the width adaptive but this is not considered in this thesis. However, in the next section the distributed model is exploited to reformulate the D-optimality into a more parallelizable exploration, which increases the effectiveness of the exploration.

4.4. PARALLELIZATION OF THE D-OPTIMAL DESIGN CRITERION FOR A SWARM SYSTEM

Section 4.2 introduced the D-optimal criterion as an exploration criterion. Simulations indicated that this criterion reduces the NRMSE faster compared with the benchmark exploration strategies — meander and random walk. Furthermore, it has been shown that this criterion can be computed in a distributed fashion regardless of the distribution paradigm. However, the D-optimality criterion is originally not intended for a swarm system because for an optimal computation, the criterion has to be evaluated sequentially after each measurement of each agent. This sequential evaluation leads to a decrease in time-efficiency as all agents have to wait once it is their turn. Additionally, the whole search space $\mathscr{X}$ is considered for all agents but it could also be split among agents to reduce the computation time. Therefore, this section empirically analyzes the performance of the D-optimal criterion when it is computed in parallel. Furthermore, it is looked at how to coordinate the agents to avoid inter-agent collisions. In this section, only homogeneous learning is considered as this parallelization is independent of the distribution paradigm.

4.4.1. COORDINATION STRATEGIES

For a discretized exploration space $\mathscr{X}$, the maximization of (4.25) requires in general the evaluation of all $\tilde{\boldsymbol{x}} \in \mathscr{X}$. Section 4.2 presented how to distributively solve

$$\hat{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}} \in \mathscr{X}, \tilde{\boldsymbol{x}} = \check{\boldsymbol{\mu}}}{\arg\min} \log\det \widetilde{\boldsymbol{C}}([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T). \tag{4.43}$$

Yet the agents cannot detect and approach next measurement locations at the same time. To see this, the next optimal measurement position can only be estimated after each measurement is introduced because the estimation depends on the measurements $\boldsymbol{y}$ and, furthermore, $\widetilde{\boldsymbol{C}}([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T)$ depends on the current active set $\mathscr{A}$, which is an output of the estimation. Due to this processing chain, if agent k is moving to the next measurement location, the remaining agents have to wait for agent k's new input to optimally solve (4.43). Consequently, the D-optimality cannot profit from the distributed system as everything has to be processed sequentially.

To circumvent this and facilitate a parallel optimization of (4.43), the approach here is to manipulate the set $\mathscr{X}$ such that a parallel exploration with the D-optimality is possible. In essence, the set $\mathscr{X}$ is split into subsets $\mathscr{X}_k \in \mathscr{X}, k = 1, \ldots, K$. How $\mathscr{X}_k$ is chosen and distributed in the network is referred to as *coordination method*. Depending on the coordination method the size of $\mathscr{X}_k$ also varies. The considered coordination methods are introduced in the following.

1. Coordination method *seqDOpt* as introduced in Sec. 4.2.2 and [53]. It directly uses the whole set $\mathscr{X}$ for candidate locations. As already noted before, here the agents move sequentially between the measurements. It is basically a centralized method and used as a benchmark. Furthermore, the coordination method *seqDOpt* requires no or almost no cooperation between the agents. This is because each other agent waits for the current agent to measure before a new measurement location is estimated. Thus, there is only a single next measurement location at a time.

2. In the coordination method *DOpt* proposed here, the agents sequentially select their next measurements locations according to the D-optimality criterion in a pre-defined order, where one agent is the starting agent. The first agent k chooses $\hat{\boldsymbol{x}}_k \in \mathscr{X}$ that maximizes the D-optimality criterion and communicates $\hat{\boldsymbol{x}}_k$ to the swarm. Then, to avoid collisions between agent k and the other agents, a safety area $\widehat{\mathscr{X}_k} = \{\boldsymbol{x} : \|\boldsymbol{x} - \tilde{\boldsymbol{x}}_k\| < r_s\}$ with $r_s > 0$ around $\hat{\boldsymbol{x}}_k$ is removed from $\mathscr{X}$, too. The new set for the $k+1$-th agent is then $\mathscr{X}_{k+1} = \left\{x \in \mathscr{X} \setminus \widehat{\mathscr{X}_k}\right\}$. The procedure is repeated for the other agents in the swarm, until every agent k estimates $\hat{\boldsymbol{x}}_k, \forall \mathscr{X} \setminus \widehat{\mathscr{X}_k}, \forall k = 1, \ldots, K$. Then, each agent navigates to its next measurement location at the same time. Therefore, the *DOpt* requires the communication of the estimates measurement locations during the evaluation of (4.43). This coordination method is exemplified in Fig. 4.13 on the left.

   The difference to the centralized version is that here the exploration space $\mathscr{X}$ is iteratively reduced by $\widehat{\mathscr{X}_k}$. Moreover, the movement of the agent commences after every agent has a next measurement location. Before, the current agent first

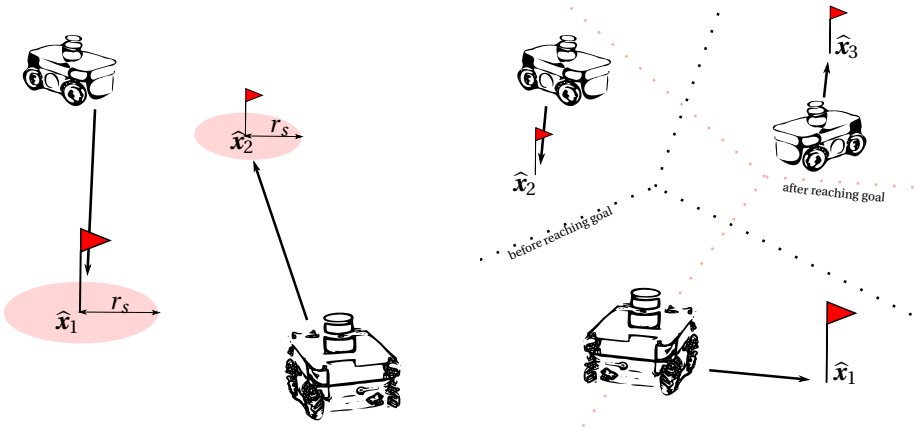


Figure 4.13: Two of the proposed coordination methods: *DOpt* on the left and *voronoiRegions* on the right. The right figure indicates that the Voronoi regions move with the agents.

moved, took a measurement, and started the estimation of the weights, and, then, the next agent estimated a new measurement location taking the recent measurement into account.

3. For the coordination method *voronoiRegions* the set $\mathcal{X}$ is partitioned into K non-overlapping Voronoi regions [145] such that $\mathcal{X} = \bigcup_{k=1}^{K} \mathcal{X}_k^{(m+1)}$ before the estimation of agent k's $m+1$-th measurement location commences. To construct the Voronoi regions $\mathcal{X}_k^{(m+1)}$, the current position of agent k – $\boldsymbol{x}_k$ from the last measurement m – acts as a Voronoi vertex, and, by design, all points in their Voronoi region have the shortest distance to their Voronoi vertex. Thus, when locally solving (4.43), each estimate will have the shortest distance to the current agent. Additionally, this method acts as a collision avoidance between other agents. Regarding the communication, this coordination method requires the communication of each agent's position before and during the evaluation of (4.43). The positions are then used to construct the Voronoi graph. This method is also shown on the right of Fig. 4.13. There, two Voronoi edges are shown: in black for before the agents started the estimation of the next measurement location and in red after the agents reached the next measurement location.

4.4.2. SIMULATION RESULTS FOR COORDINATION STRATEGIES AND D-OPTIMAL EXPERIMENT DESIGN FOR SWARM SYSTEMS

To test the coordination methods, an area without obstacles is assumed with the same size as in the simulations in Sec. 4.2.6 and the magnetic field, as introduced in Fig. 4.7(a), is explored. It is assumed that the agents can move instantly to their measurement locations, but a collision avoidance at the goal positions is included as it is used later in the experiments in Chapter 5. The number of agents in this scenario is set to $K = 3$. The algorithm to estimate the parameter weights is the ADMM introduced in [18] with $\rho = 10$ as ADMM penalty parameter. In this simulation each agent will collect $M_k = 133$

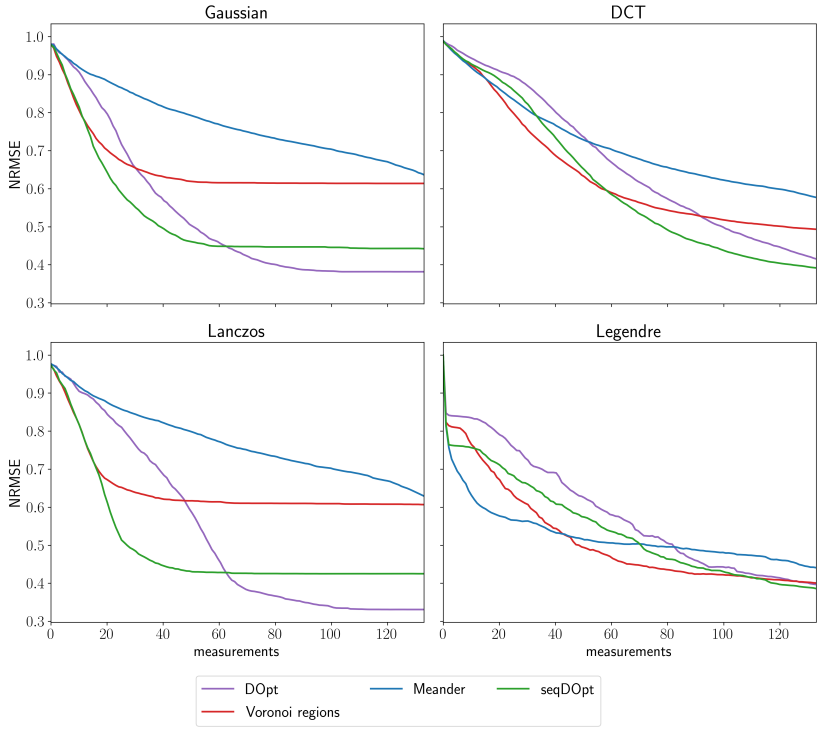


Figure 4.14: Simulation results of different coordination methods for different basis functions based on optimal experiment design (OED) with the D-optimality criterion.

measurements. Additionally, the coordination methods are evaluated for different basis functions, as introduced in Sec. 2.1.3. The parameters for the basis functions are set to $\sigma_{\text{Gauss}} = 0.2$ for the Gaussian basis functions, $\sigma_{\text{Lanczos}} = 0.55$ for the Lanczos basis functions, for the cosine basis functions the frequencies range from 0 Hz to [20 Hz] in x- and y-direction, and the Legendre polynomials have a degree of 20 in x- and y-direction.

The NRMSE of all simulations is shown in Fig. 4.14, where the two left plots show the NRMSE of the localized basis functions and the two right plots show the NRMSE of the global basis functions. For the localized basis functions, *DOpt* yields the best result after 133 measurements per each agent. In these simulations, the benchmark, *seqDOpt*, performs better at the beginning for localized basis functions, but is outperformed by the *DOpt* method at the end. One potential reason for this could be that there are regions excluded in *DOpt* for reasons of collision avoidance and this forces the other agents to select measurement locations that are suboptimal. This could be excluded in the simulation as no agents can collide, but as this is evaluated for a real scenario the collision avoidance has to be included. However, the suboptimal sampling might increase the exploration in the coordination method instead of exploitation as also less important locations are measured. The method *voronoiRegion* yields the worst performance for the localized basis functions. When using the *voronoiRegion* method it was observed that

agents might get stuck in corners as the Voronoi graph is recomputed after each agent has taken its measurement. Thus these agents can only explore in their close surrounding and it might be difficult to get out of a small exploration area.

The right plots in Fig. 4.14 present the results for the global basis functions. The Legendre polynomials identify the bias of the magnetic field, which reduces the NRMSE with the first measurements. For Legendre polynomials the *meander* strategy is able to reduce the error faster compared to the coordination strategies. Yet the coordination strategies yield a lower error at the end. Generally, the influence of the coordination methods is less prominent for Legendre polynomials and cosine functions. This could be explained by their global view; taking a measurement at some location, might influence the whole estimate. Also, due to the symmetry of the global basis function, there exist multiple measurement positions that improve the estimate equally. Therefore, it seems reasonable that coordination is less influential.

4.5. Using a Sparse Bayesian Model for Exploration with D-optimality

The D-optimality criterion can also be applied, if a Bayesian paradigm is used. The difference compared with the previous section is that for the Bayesian method, the covariance of the posterior $p(\boldsymbol{w}|\boldsymbol{y})$ is used in the D-optimality criterion. Section 3.1.1 presents how the mean $\widehat{\boldsymbol{w}}$ and the covariance $\boldsymbol{\Sigma}_w$ of $p(\boldsymbol{w}|\boldsymbol{y})$ are approximated in a Type-II SBL framework. The covariance in (3.6) is presented here again for convenience with the dependency of the center positions for localized basis functions as

$$\boldsymbol{\Sigma}_w = \left(\boldsymbol{\Phi}^T(\boldsymbol{X},\boldsymbol{M})\boldsymbol{\Lambda}\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{M}) + \widehat{\boldsymbol{\Gamma}}^{-1}\right)^{-1}. \tag{4.44}$$

There is a similarity between the covariance (4.44) of the SBL methods and the approximated Hessian (4.14). The term $\widehat{\boldsymbol{\Gamma}}^{-1}$ in (4.44) substitutes the term $\widehat{\boldsymbol{\Delta}}$ in (4.14). Yet, the difference of both is significant as indicated in the Example 4, which presents the difference in the resulting D-optimality criterion for the Bayesian approach and for the approximated approach.

Example 4. *The function $g(x)$ in Example 1 is considered here with $N = 20$ basis functions and $\boldsymbol{\phi}_n(x)$, $n = 1,\ldots,N$. The left side of Fig. 4.15 presents the estimates when using the LASSO estimator with varying δ. Also, the estimate of a Bayesian SBL algorithm is presented as a black dotted line. The right side of Fig. 4.15 shows the estimated D-optimality for the corresponding LASSO estimate and δ. The dependency of the parameter δ is clearly visible. When the covariance of estimated posterior in the Bayesian framework is used, the D-optimality is peaked at the measurement positions. For the approximated covariance a similar behavior is only observed for an appropriate δ, i.e. $\delta = 0.23334$ or $\delta = 0.46667$. If δ is set wrong, the D-optimality criterion can not be used for exploration. In this example, for $\delta = 0.7$, which might be too high, the D-optimality is minimal at $x = 1.8$ and focuses more on exploitation, or alias effects might occur as for $\delta = 1e-5$, which seems to be too low, at $x = 1.5$. The choice of δ depends on the applied basis function and the observed process.*

Because both covariances are similar the distributed computation are similar as well.

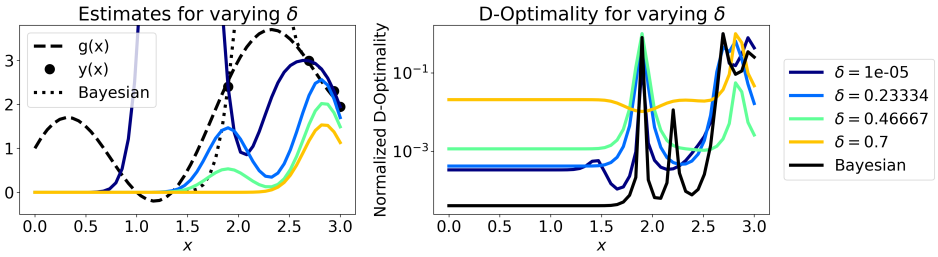


Figure 4.15: The difference between the Bayesian estimated D-optimality and the approximated D-optimality for varying δ.

If a new measurement at $\tilde{\boldsymbol{x}}$ is considered for (4.44), then a new row $\boldsymbol{\phi}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M})$ and a new column $\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}})$ are added to $\boldsymbol{\Phi}$ as

$$\widetilde{\boldsymbol{\Phi}}([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T) = \begin{bmatrix} \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{M}) & \boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ \boldsymbol{\phi}^T(\tilde{\boldsymbol{x}}, \boldsymbol{M}) & \phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) \end{bmatrix}. \tag{4.45}$$

Then, (4.45) is used to construct a new covariance

$$\widetilde{\boldsymbol{\Sigma}}_w = \left(\widetilde{\boldsymbol{\Phi}}^T([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T) \widetilde{\boldsymbol{\Lambda}} \widetilde{\boldsymbol{\Phi}}([\boldsymbol{X}^T, \tilde{\boldsymbol{x}}]^T, [\boldsymbol{M}^T, \check{\boldsymbol{\mu}}]^T) + \begin{bmatrix} \boldsymbol{\Gamma}^{-1} & 0 \\ 0 & \tilde{\gamma}^{-1} \end{bmatrix} \right)^{-1}, \tag{4.46}$$

where $\tilde{\gamma}$ is a sparsity parameter associated with a new column $[\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}})^T, \phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}})]^T$, $\widetilde{\boldsymbol{\Lambda}} = \text{diag}\{[\underbrace{\lambda, \ldots, \lambda}_{M \text{ elements}}, \tilde{\lambda}]\} \in \mathbb{R}^{M+1 \times M+1}$, and $\tilde{\lambda} \in \mathbb{R}$ is the assumed noise precision at the proposed measurement position.

In the previous section, only the columns of $\boldsymbol{\Phi}$ that belong to the active set $\mathscr{A}$ contribute to the approximation of the Hessian, but for the SBL framework *relevant* basis functions are determined by $\hat{\boldsymbol{\gamma}}$: If $\hat{\gamma}_n > 0$, the corresponding column of $\boldsymbol{\Phi}$ is used in the model; it can be interpreted as the corresponding column belonging to the active set, where for SBL the active set is defined as $\mathscr{A} = \{n \in \mathbb{N} : \hat{\gamma}_n > 0\}$. Consequently, the choice of $\tilde{\gamma}$ also influences if the basis function contributes to the model or not. The choice of $\tilde{\gamma}^{-1}$ is further elaborated in Appendix B.3.

Using the covariance (4.46), the D-optimality criterion in (4.4) is formulated as

$$\min_{\tilde{\boldsymbol{x}} \in \mathscr{X}, \check{\boldsymbol{\mu}} \in \mathscr{M} \setminus \boldsymbol{M}|_{\mathscr{A}}} \log\det\{\widetilde{\boldsymbol{\Sigma}}_w(\boldsymbol{X}, \boldsymbol{M}, \tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}})\}. \tag{4.47}$$

In the following steps, the dependency of $\tilde{\boldsymbol{x}}$ in $\widetilde{\boldsymbol{\Sigma}}_w$ is isolated to make use of the matrix determinant lemma [143]. Also, the notation is simplified for better readability. First, $\tilde{\boldsymbol{x}}$

is isolated as

$$\widetilde{\boldsymbol{\Sigma}}_w(\boldsymbol{X},\boldsymbol{M},\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}})^{-1} = \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}+\widehat{\boldsymbol{\Gamma}}^{-1} & \boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\breve{\boldsymbol{\mu}}) \\ \boldsymbol{\phi}^T(\boldsymbol{X},\breve{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\Phi} & \lambda\boldsymbol{\phi}^T(\boldsymbol{X},\breve{\boldsymbol{\mu}})\boldsymbol{\phi}(\boldsymbol{X},\breve{\boldsymbol{\mu}})+\widetilde{\gamma}^{-1} \end{bmatrix} + \widetilde{\lambda}\begin{bmatrix} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) \\ \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}\begin{bmatrix} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}. \tag{4.48}$$

Then, similarly to (4.19), the following is defined as

$$\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \triangleq \boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\breve{\boldsymbol{\mu}}), \qquad b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \triangleq \boldsymbol{\phi}^T(\boldsymbol{X},\breve{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\breve{\boldsymbol{\mu}})+\widetilde{\gamma}^{-1}, \tag{4.49}$$

where a subscript SBL indicates that these expressions are defined for an SBL method and are not mistaken for (4.19). Inserting the definitions (4.49) into (4.46) leads to

$$\widetilde{\boldsymbol{\Sigma}}_w(\boldsymbol{X},\boldsymbol{M},\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}})^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \\ \boldsymbol{c}_{\mathrm{SBL}}^T(\breve{\boldsymbol{\mu}}) & b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \end{bmatrix} + \widetilde{\lambda}\begin{bmatrix} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) \\ \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}\begin{bmatrix} \boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}. \tag{4.50}$$

As can be seen in (4.50), the second term shows the contribution of the potential measurement position to $\widetilde{\boldsymbol{\Sigma}}_w$. Then, the same derivation steps as in Sec. 4.2.2 starting from (4.21) follow, which lead to an elementwise computation of the D-optimality at location $\widetilde{\boldsymbol{x}}$. By taking the logarithm of the objective in (4.47), the inverse changes into a minus sign, and afterward the matrix determinant lemma [143] is applied as

$$\begin{aligned} \log|\widetilde{\boldsymbol{\Sigma}}_w(\boldsymbol{X},\boldsymbol{M},\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}})| = &-\log\begin{vmatrix} \boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \\ \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^T & b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \end{vmatrix} \\ &-\log\left|1+\widetilde{\lambda}\begin{bmatrix} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}\begin{bmatrix} \boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}(\breve{\boldsymbol{\mu}}) \\ \boldsymbol{c}(\breve{\boldsymbol{\mu}})^T & b(\breve{\boldsymbol{\mu}}) \end{bmatrix}^{-1}\begin{bmatrix} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) \\ \phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) \end{bmatrix}\right|. \end{aligned} \tag{4.51}$$

Using the Schur complement $q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) = b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}^T(\breve{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})$, the first term on the right hand side in (4.51) can be reformulated as

$$\log\begin{vmatrix} \boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \\ \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^T & b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \end{vmatrix} = -\log|\boldsymbol{\Sigma}_w| + \log q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}). \tag{4.52}$$

The inverted block matrix in the second term on the right hand side, can be inverted by applying inversion rules for structured matrices [116], which permit to write it as

$$\begin{bmatrix} \boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \\ \boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^T & b_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_w - \boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^{-1}\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w & -\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})/q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \\ -\boldsymbol{c}_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w/q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) & 1/q_{\mathrm{SBL}}(\breve{\boldsymbol{\mu}}) \end{bmatrix} \tag{4.53}$$

such that

$$\log\left|1+\begin{bmatrix}\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}) & \phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_w^{-1} & \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\\ \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T & b_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})\\ \phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}\right|$$
$$=\log\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})+\widetilde{\lambda}\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})-\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})\right)^2/q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\right). \quad (4.54)$$

Inserting (4.52) and (4.54) into (4.51), the D-optimality criterion (4.47) can be formulated for exploration as

$$\widehat{\boldsymbol{x}}=\underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}}}{\arg\min}\log|\widetilde{\boldsymbol{\Sigma}}_w|\equiv\underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}}}{\arg\max}\log\Big[q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M}))$$
$$+\widetilde{\lambda}\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})-\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})\right)^2\Big], \quad (4.55)$$

where $\log|\boldsymbol{\Sigma}_w|$ is dropped as it is constant in $\widetilde{\boldsymbol{x}}$ or $\check{\boldsymbol{\mu}}$.

4.5.1. D-OPTIMAL EXPERIMENT DESIGN FOR SBL AND HOMOGENEOUS LEARNING

The structure of the approximated covariance (4.14) is similar to the covariance of the posterior (4.44). The difference is that the term $\widehat{\boldsymbol{\Gamma}}^{-1}$ in (4.44) is used instead of the term $\widehat{\boldsymbol{\Delta}}$ in (4.14). Thus, the distributed computation of the D-optimality can be applied for the covariance of the posterior in a similar fashion.

For homogeneous learning, all agents locally know $\widehat{\boldsymbol{\gamma}}$ and no further distributed estimation of it is necessary. Likewise, the covariance $\boldsymbol{\Sigma}_w$ is also known to each agent due to the estimation. As shown in (4.38) and (4.39), the definitions in (4.19) can be reformulated such that they can be computed with a consensus algorithm for homogeneous learning. This is also true for the definitions made here in this section. Thus, the consensus steps can be redefined as

$$\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\triangleq\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})=\sum_{k=1}^{K}\boldsymbol{\Phi}^T(\boldsymbol{X}_k)\boldsymbol{\Lambda}_k\boldsymbol{\phi}(\boldsymbol{X}_k,\check{\boldsymbol{\mu}}), \quad (4.56)$$

$$b_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\triangleq\boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})+\widetilde{\gamma}^{-1}=\widetilde{\gamma}^{-1}+\sum_{k=1}^{K}\boldsymbol{\phi}^T(\boldsymbol{X}_k,\check{\boldsymbol{\mu}})\boldsymbol{\Lambda}_k\boldsymbol{\phi}(\boldsymbol{X}_k,\check{\boldsymbol{\mu}}), \quad (4.57)$$

where $\boldsymbol{\Lambda}_k=\lambda\boldsymbol{I}\in\mathbb{R}^{M_k\times M_k}$ is a diagonal matrix with the precision of the measurement noise on the diagonal (Please note that the size of this matrix depends on M_k). The size of this matrix is defined by the number of measurements M_k of each agent k. Subsequently, with the definitions (4.56) and (4.57), the D-optimality criterion as shown in (4.55) can be already estimated in a distributed fashion.

4.5.2. D-OPTIMAL EXPERIMENT DESIGN FOR SBL AND HETEROGENEOUS LEARNING

For heterogeneous learning, every agent has its own estimated hyper parameters $\widehat{\boldsymbol{\gamma}}_k$, i.e. $\widehat{\boldsymbol{\Gamma}}_k=\mathrm{diag}\{\widehat{\boldsymbol{\gamma}}_k\}\,\forall k=1,\ldots,K$. As a result, the definitions (4.49) cannot be used for a distributed computation of the exploration criterion (4.55). Therefore, for heterogeneous

learning define

$$\boldsymbol{H}_{\text{SBL}} \triangleq \boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T = \sum_{k=1}^{K} \boldsymbol{\Phi}_k \widehat{\boldsymbol{\Gamma}}_k \boldsymbol{\Phi}_k^T, \tag{4.58}$$

$$\boldsymbol{d}_{\text{SBL}}(\widetilde{\boldsymbol{x}}) \triangleq \boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}) = \sum_{k=1}^{K} \boldsymbol{\Phi}_k \widehat{\boldsymbol{\Gamma}}_k \boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}_k), \tag{4.59}$$

$$v_{\text{SBL}}(\widetilde{\boldsymbol{x}}) \triangleq \boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}}, \boldsymbol{M})\widehat{\boldsymbol{\Gamma}}\boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}) = \sum_{k=1}^{K} \boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}_k)^T \widehat{\boldsymbol{\Gamma}}_k \boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}_k). \tag{4.60}$$

Analog to the derivations provided in Sec. 4.2.4, the new definitions are used to express (4.55) for heterogeneous learning. The derivations for the distributed SBL follow the same steps as the derivations in Sec. 4.2.4. Starting by applying the matrix-inversion-lemma to $\boldsymbol{\Sigma}_w$ as

$$\boldsymbol{\Sigma}_w = \widehat{\boldsymbol{\Gamma}} - \widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T(\boldsymbol{\Lambda}^{-1} + \boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}} = \widehat{\boldsymbol{\Gamma}} - \widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}})^{-1}\boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}. \tag{4.61}$$

Then, (4.61) is inserted into q_{SBL}

$$\begin{aligned} q_{\text{SBL}}(\check{\boldsymbol{\mu}}) &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) - \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ &\quad - \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\Big(\boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T - \boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}})^{-1}\boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T\Big)\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}). \end{aligned} \tag{4.62}$$

Now, (4.58) is inserted into (4.62) to apply the Woodbury identity as

$$\begin{aligned} q_{\text{SBL}}(\check{\boldsymbol{\mu}}) &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ &\quad - \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\Big(\boldsymbol{H}_{\text{SBL}} - \boldsymbol{H}_{\text{SBL}}(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}})^{-1}\boldsymbol{H}_{\text{SBL}}\Big)\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) - \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\Big(\boldsymbol{\Lambda} + \boldsymbol{H}_{\text{SBL}}^{-1}\Big)^{-1}\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}). \end{aligned} \tag{4.63}$$

Afterward, $\boldsymbol{\Lambda}$ can be excluded such that the Woodbury identity is applied a second time on (4.63) such that

$$\begin{aligned} q_{\text{SBL}}(\check{\boldsymbol{\mu}}) &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\Big(\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\Big(\boldsymbol{\Lambda} + \boldsymbol{H}_{\text{SBL}}^{-1}\Big)^{-1}\boldsymbol{\Lambda}\Big)\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \\ &= \widetilde{\gamma}^{-1} + \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}})\left(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}}\right)^{-1}\boldsymbol{\phi}(\boldsymbol{X}, \check{\boldsymbol{\mu}}). \end{aligned} \tag{4.64}$$

The expression (4.55) has to be reformulated into a distributed form. This is done by looking at each summand in (4.55) separately. The first summand $q_{\text{SBL}}(\check{\boldsymbol{\mu}})$ is already derived in (4.64). Next, the aforementioned definitions (4.58), (4.59), and (4.60) are used on the second summand of (4.55) such that

$$\begin{aligned} \widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}}, \boldsymbol{M})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}) &= \widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}}, \boldsymbol{M})\left(\widehat{\boldsymbol{\Gamma}} - \widehat{\boldsymbol{\Gamma}}\boldsymbol{\Phi}^T(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}})^{-1}\boldsymbol{\Phi}\widehat{\boldsymbol{\Gamma}}\right)\boldsymbol{\phi}(\widetilde{\boldsymbol{x}}, \boldsymbol{M}) \\ &= \widetilde{\lambda}v_{\text{SBL}}(\widetilde{\boldsymbol{x}}) - \widetilde{\lambda}\boldsymbol{d}_{\text{SBL}}(\widetilde{\boldsymbol{x}})^T(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\text{SBL}})^{-1}\boldsymbol{d}_{\text{SBL}}(\widetilde{\boldsymbol{x}}). \end{aligned} \tag{4.65}$$

Finally, the last term is reformulated with (4.61), (4.58), and (4.59) which yields

$$\begin{aligned}
\boldsymbol{c}_{\mathrm{SBL}}(\boldsymbol{x})^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) &= \boldsymbol{c}_{\mathrm{SBL}}(\boldsymbol{x})^T \left(\hat{\boldsymbol{\Gamma}} - \hat{\boldsymbol{\Gamma}} \boldsymbol{\Phi}^T (\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}})^{-1} \boldsymbol{\Phi} \hat{\boldsymbol{\Gamma}}\right) \boldsymbol{\phi}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) \\
&= \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \boldsymbol{\Lambda} \boldsymbol{\Phi} \left(\hat{\boldsymbol{\Gamma}} - \hat{\boldsymbol{\Gamma}} \boldsymbol{\Phi}^T (\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}})^{-1} \boldsymbol{\Phi} \hat{\boldsymbol{\Gamma}}\right) \boldsymbol{\phi}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) \\
&= \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \boldsymbol{\Lambda} \left(\boldsymbol{\Phi} \hat{\boldsymbol{\Gamma}} \boldsymbol{\phi}(\tilde{\boldsymbol{x}}, \boldsymbol{M}) - \boldsymbol{\Phi} \hat{\boldsymbol{\Gamma}} \boldsymbol{\Phi}^T (\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}})^{-1} \boldsymbol{\Phi} \hat{\boldsymbol{\Gamma}} \boldsymbol{\phi}(\tilde{\boldsymbol{x}}, \boldsymbol{M})\right) \\
&= \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \boldsymbol{\Lambda} \left(\boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}}) - \boldsymbol{H}_{\mathrm{SBL}} (\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}})^{-1} \boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}})\right) \\
&= \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \boldsymbol{\Lambda} \left(\boldsymbol{I} - \boldsymbol{H}_{\mathrm{SBL}} \left(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}}\right)^{-1}\right) \boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}}). \qquad (4.66)
\end{aligned}$$

As a last step, the resulting summands $q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})$, (4.65), and (4.66) are inserted into (4.55) to yield the distributed criterion for the SBL method and for SOF:

$$\begin{aligned}
\hat{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}} \in \mathcal{X}, \check{\boldsymbol{\mu}} = \tilde{\boldsymbol{x}}}{\arg\max} \log\Big(& q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) \left(1 + \tilde{\lambda} \nu_{\mathrm{SBL}}(\tilde{\boldsymbol{x}}) - \tilde{\lambda} \boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}})^T (\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}})^{-1} \boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}})\right) \\
& + \tilde{\lambda} \left(\phi(\tilde{\boldsymbol{x}}, \check{\boldsymbol{\mu}}) - \boldsymbol{\phi}^T(\boldsymbol{X}, \check{\boldsymbol{\mu}}) \boldsymbol{\Lambda} \left(\boldsymbol{I} - \boldsymbol{H}_{\mathrm{SBL}} \left(\boldsymbol{\Lambda}^{-1} + \boldsymbol{H}_{\mathrm{SBL}}\right)^{-1}\right) \boldsymbol{d}_{\mathrm{SBL}}(\tilde{\boldsymbol{x}})\right)^2\Big), \qquad (4.67)
\end{aligned}$$

where for a distributed calculation $q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})$ as in (4.64) has to be applied for SOF. The performance evaluation of the exploration based on SBL is conducted through experiments in the next chapter.

4.6. SUMMARY AND DISCUSSION

The current chapter applied the D-optimality criterion on two distribution paradigms: heterogeneous learning (SOF) and homogeneous learning (SOE). Looking at the model (2.9), heterogeneous learning distributes the columns of $\boldsymbol{\Phi}$ and the parameter weights $\boldsymbol{w}$ in the network. In contrast, homogeneous learning distributes the rows of $\boldsymbol{\Phi}$ and the measurements. This chapter presented how to accumulate the distributed data to evaluate the D-optimality criterion. The data accumulation is realized by simple averaged consensus algorithms, which are well known in the literature [19, 20]. For SOE, on the one side, the amount of communicated data depends on N — the number of used basis functions — and the number of proposed center positions. Here, an agent doesn't have to transmit the proposed next measurement locations $\mathcal{X}$ because the agents do not need to know them for computing the consensus terms. For SOF, on the other side, the transmitted values are dominated by M — the number of measurements — and the proposed next measurement locations $\tilde{\boldsymbol{x}} \in \mathcal{X}$. In contrast to SOE, for SOF, the proposed center positions are not required for computing the consensus terms. However, the proposed next measurement locations $\tilde{\boldsymbol{x}} \in \mathcal{X}$ of each agent have to be known to the other agents, before the estimation of the next measurement location commences.

Furthermore, this chapter discusses how the data acquisition can also be used for different frameworks, i.e. the Bayesian framework and the frequentist framework. Both frameworks and distribution paradigms are evaluated in simulations. By the virtue of the Bayesian framework, the covariance matrix of the posterior can be used in the D-optimality criterion. For the LASSO-type estimator, the covariance of the parameter weights has to be approximated with an active set approach. This approximation depends on the penalty parameter δ of the LASSO cost function, which has to be chosen

before the estimation and exploration. A large δ makes the parameter estimate sparser, which is intended here. However, if δ is too large, the D-optimality puts too much emphasis on already measured locations, see Sec. 4.2.3. This leads to less exploration and in extreme cases the robot would repeatedly measure at the same position. Therefore, the estimation and the exploration have to use a compromise value for δ when using the approximated covariance. For the Bayesian framework, all parameters and hyper parameters are computed from the data. Also, the computed covariance of the parameter weights is used and now approximation is required. The only design choice is $\widetilde{\gamma}$, which is discussed in App. B.3.

This chapter also looked at the parallelization of the evaluation of the D-optimality, exemplified for the SOE paradigm. The division of the area based on a Voronoi tessellation before each exploration step leads to a lower performance. The best performance is yielded, if the joint D-optimality criterion is evaluated in parallel on each agent, and if the next measurement locations and a safety area around it are excluded from preceding agents. This leads to a parallelized exploration and to a more explorative behavior in the swarm. Thus, the performance increases after an initialization phase. The parallelization will then be used for the experiments, which are presented in the following chapter.

This chapter showed that the D-optimality can be evaluated for a distributed swarm and for problems with sparsity constraints. The SOF paradigm has more flexibility in deleting unused basis functions, such that the model complexity is kept low. In addition, SOF reached a lower NRMSE faster compared with SOE. On the other hand, SOE requires less data to be transmitted, which might be more practical for a real system and especially larger networks.

5

Experimental Evaluations

This chapter evaluates the distributed algorithms for parameter estimations presented in Sec. 2.4 and Chap. 3 and the exploration criteria from Chap. 4 in experiments. The goal is to show that the whole system can be implemented in real-time and with real sensors. To this end, this chapter demonstrates two experiments: The first experiment is a hardware-in-the-loop experiment, which demonstrates the real-time applicability and the effectiveness of the coordination methods presented in Sec. 4.4. The second experiment adds real-sensors and obstacles to the experiment and compares the Bayesian algorithms, see Chap. 3, with the distributed ADMM in Sec. 2.4 for both distribution paradigms. Additionally, with regard to more complex environments and real agents, both experiments require an inter-agent collision avoidance. Moreover, the experiment with obstacles, i.e. map constraints, requires a path-planner to safely reaching a target position. The following section presents the hardware-in-the-loop experiment, which also evaluates the coordination methods (see Sec. 4.4.1) in an experiment.

5.1. Different Coordination Methods for Entropy Driven Exploration

In Sec. 4.4.1 different coordination strategies to parallelize the exploration were discussed. The current section presents experimental validations of the last chapter's results in a hardware-in-the-loop experiment. Hardware-in-the-loop means in this case that the sensor readings, which are the input to the estimation, are simulated. Therefore, the experiment will evaluate,

- if the overall setting is applicable in real-time,
- if there is influence of noisy positions on the overall setting, and
- if the coordination of the agents behaves as in the simulations.

This section starts with the detailed presentation of the experimental setup. Afterward,

Figure 5.1: The Holodeck at the DLR premises. The circles highlight the VICON camera system, which is used for indoor positioning and motion tracking.

the experimental validations are shown and compared with the simulation results of Sec. 4.4.1.

5.1.1. EXPERIMENT SETUP

In this experiment, the magnetic field on the ground of the Holodeck is explored. The Holodeck is displayed in Fig. 5.1. For positioning of the agents, the laboratory is equipped with a motion capture system manufactured by VICON[1]. In particular, 16 cameras of the Bonita version are employed. The 16 cameras are marked by circles in Fig. 5.1. The VICON system is able to position and track objects in the laboratory by observing reflecting markers that are placed on the objects from different angles. At least four markers are required to position or track an object, e.g. a robotic platform, such that the position and orientation of the object can be measured. If there are multiple objects to be observed, the constellations of the reflecting markers on each object have to be unique in each experiment.

The setup comprises three holonomic ground robots, which are in the following referred to as *slider* and are displayed in Fig. 5.2. The current experiment takes place in an obstacle free map such that the experimental area comprises only these three sliders. Each slider is manufactured by Commonplace Robotics[2]. The holonomic wheels enable the slider to move in any direction without the need for turning. This simplifies the controller of the slider, as each path becomes a straight line; the controller does not need to account for dynamic constraints.

The positioning system is much more accurate than the actuators of the slider, which makes it difficult to reach an exact position. As the controller tries to correct this, it

[1] `https://www.vicon.com/`, accessed in January 2024
[2] `https://cpr-robots.com`, accessed in January 2024

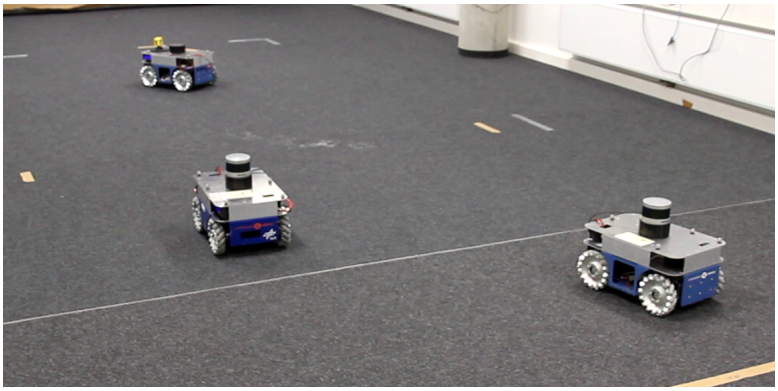


Figure 5.2: The three slider on the ground of the Holodeck.

can lead to an oscillating behavior of the slider. To avoid this oscillating behavior, the goal position is modeled as a circle with a 10 cm radius centered at the desired point. Although this introduces noise on the optimal measurement position, it increases the reliability of the whole system. For computation and control, each slider is equipped with an Intel NUC[3], which is connected to the slider wheel actuators.

The software setup for this experiment is presented in Fig. 5.3. The experiment itself is designed similar to the simulations in Sec. 4.4.1 for easier comparison. After the initialization of the system, each slider executes a sensor software, which simulates a magnetic field sensor to generate measurements. The sensor simulation gives more control about the experienced noise. Therefore, the simulated sensor reads the measurement from the ground truth, here the magnetic field, and adds i.i.d. noise to the measurement. Once the measurements are introduced into the model, the distributed regression, see Sec. 2.3.1, commences and eventually yields a parameter estimate. The algorithm used for the estimation is the DLASSO, with an augmented Lagrangian parameter $\rho = 10$ and with a sparsity parameter $\delta = 0.5$. For reasons of better interpretability, the experiment only considers Gaussian basis functions, with a width of $\sigma = 0.3$.

The parameter estimate together with the set of active features is then forwarded to the distributed exploration to evaluate the D-optimality. The D-optimality is then used in the coordination to select the measurement locations for all sliders. The coordination strategies, which are utilized in this experiment, are *seqDOpt*, *DOpt*, and *voronoiRegions*. As a benchmark the meander strategy is used. For procedural reasons the distributed regression of the magnetic field and the exploration criterion are executed on the central computer but in separated threads such that the distributed processing is simulated. In addition, this centralized computation facilitates the communication protocol of the estimation and exploration because everything is synchronized.

Once, every agent has its next measurement location, it is passed as a goal position to the controller and the collision avoidance. The collision avoidance software and the slider controller are executed on the Intel NUC. The collision avoidance for this experiment is introduced in [146] and denoted as field-of-view avoidance (FOVA). It is orig-

[3] `https://www.intel.de/`, accessed in January 2024

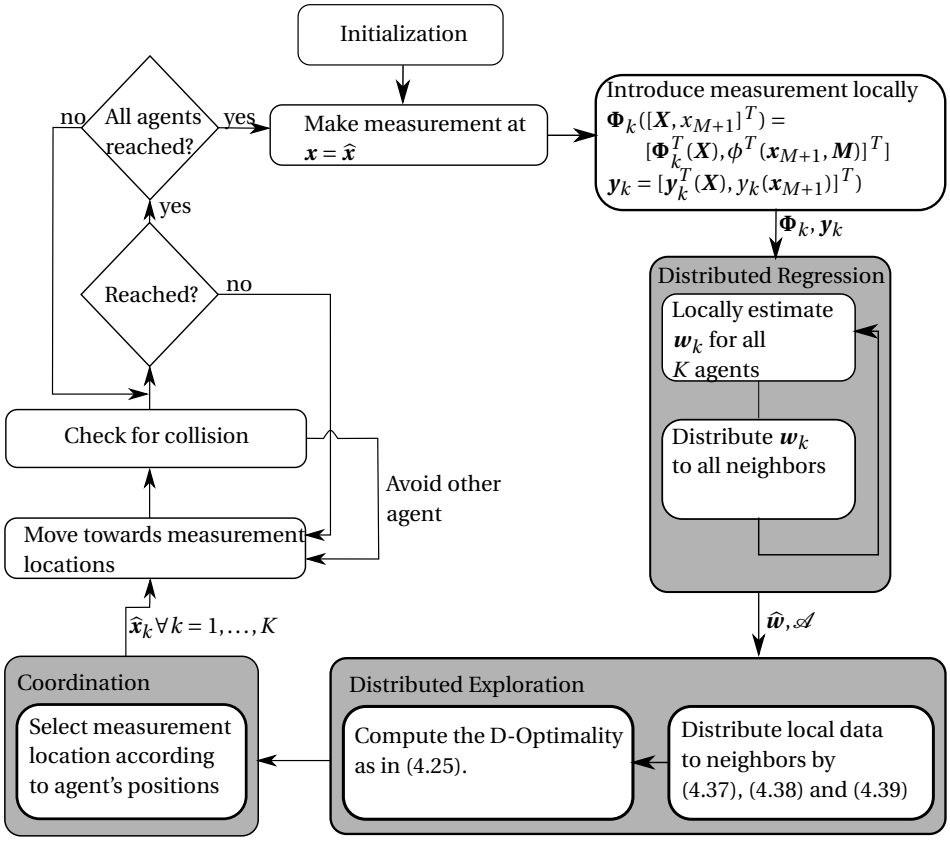


Figure 5.3: Software setup of the experiment to test coordination strategies. Each gray box represents interactions between other agents. This software setup is representative for the SOE distribution paradigm.

inally intended as an UAV collision avoidance for multi-rotor UAVs. Moreover, it is a reactive collision avoidance and, therefore, avoids collisions with approaching agents by circling around them, without changing the goal. If the turn direction is fixed for all agents this algorithm is guaranteed to avoid collisions. It can be applied in this experiment due to the holonomic wheels of the slider, which have similar dynamics in the x-y-plane as that of UAVs. Yet, in this work, the FOVA is altered and specifically parameterized for the laboratory constraints and the inertia of the sliders. When testing the collision avoidance, the sliders sometimes overshoot the goal position, if they have to avoid another approaching slider. To avoid this overshooting, the agent's velocity is reduced within the close surrounding of the goal position. Another amendment to the FOVA is a monitoring if the avoidance would make the slider exit the experimental area that is tracked by the VICON system. Otherwise, the slider would not get new position updates and would continue to move towards the walls of the laboratory. As a solution, the collision avoidance stops the slider instead and waits until the approaching agent

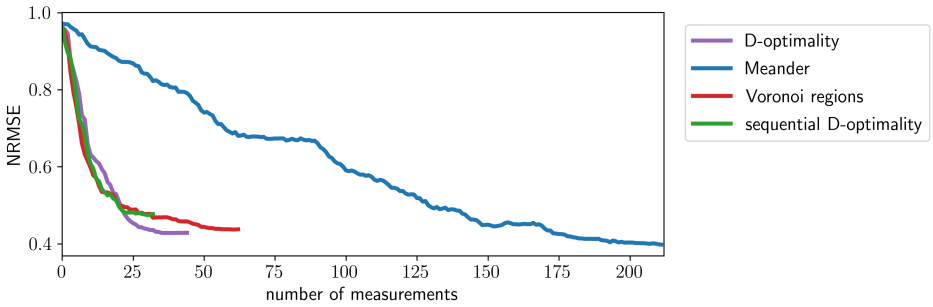


Figure 5.4: The performance of the experiments with Gaussian basis functions for different coordination methods. Each experiment was conducted for the same duration.

has passed. The specific parameters of the FOVA are presented in Appendix D.1 together with more details on the algorithm. Because this experiment considers a map without obstacles, no additional path planner is required as the best path is a straight line. If every agent has reached its measurement location, they conduct a measurement. Each run of one experiment takes approximately 30 min, and the NRMSE is used as an evaluation metric, c.f. Sec. 4.2.6.

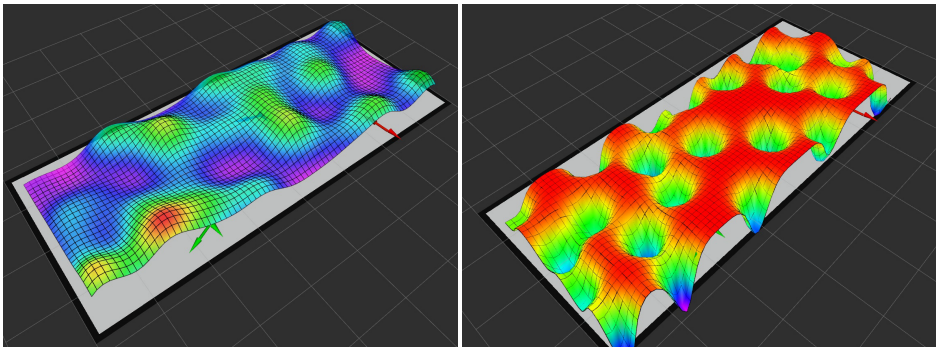
The whole experiment is controlled by a central computer because of the choice of using the robot operating system (ROS)[4]. ROS is a framework for writing robotic software, and it provides tools, libraries, and conventions to program robotic software. Especially, this experiment makes extensive use of the code conventions to distribute data between the applications, i.e. the controller, the positioning system, the FOVA, and the exploration.

5.1.2. EXPERIMENTAL RESULTS

The experiment has been conducted for all coordination methods, and the results are shown in the following. Figure 5.4 displays the NRMSE curves for a single experiment and for different coordination strategies. Figure 5.4 shows almost identical behavior compared with the simulation results in Sec. 4.4.1. The *voronoiRegions* method performs better compared with the simulation results in Sec. 4.4.1 in Fig. 4.14. Compared with the other coordination methods, the parallelized D-optimality method yields the lowest error, likewise to the simulations in Sec. 4.4.1. With regard to the number of obtained measurements, the D-optimality method performs most efficiently. As expected, the *meander* reaches the lowest error, but has to take about four times as many measurements. Thus, the *meander* is less measurement efficient.

As the time for each esperimental run is approximately 30 min, the three proposed methods are not able to make measurements as many measurements as the *meander* method for two reasons. First, each of the proposed exploration methods conducts fewer measurements compared with the benchmark because the evaluation of the covariance is computationally expensive. This is especially visible for the D-optimality method be-

[4] `http://www.ros.org`, accessed in January 2024



(a) (b)

Figure 5.5: (a) The online estimate of the magnetic field during an experiment. (b) The corresponding D-optimality criterion. In both pictures, the arrows represent the location of the slider.

cause there the exploration criterion is evaluated for the whole environment. *VoronoiRegions*, which computes the exploration criterion for each agent on a smaller and individual region, acquires more measurements compared with the D-optimality method. Second, the proposed exploration methods might yield positions that have a longer travel distance such that each slider requires more time to reach its position. The *meander* method, on the other side, chooses next measurement locations that are close to the current position such that the next measurement location can be reached in less time.

Because the experiments perform all computations in real-time the estimates and entropies can be computed at run time as well. Figure 5.5(a) shows the online estimate of an experiment, and Fig. 5.5(b) the corresponding D-optimality criterion. The criterion clearly shows the measurement locations, which yield a dip in the evaluated exploration criterion — indicated in blue color. With these few measurements, the estimate looks qualitatively similar to the ground truth.

To summarize this experiment, the meander method evaluates no criterion at all, and is thus able to conduct more measurements than the other methods. Yet, the meander method is less measurement efficient as it requires almost three times as many measurements to reach the same error compared with the D-optimality. In this experiment, this method achieves four times more measurements and almost the same NRMSE as *D-optimality*. The caveat of this hardware-in-the-loop experiment is that a measurement is not time consuming. However, if a sensor requires more time to measure, e.g. a gas-sensor, or the sensor can only be used a limited amount, e.g. the gas chromatograph on the Mars rover Curiosity [147, Sec. 6], then the exploration driven by the D-optimality criterion would lead to a better and more efficient coverage of the area. Additionally, a more effective implementation of the D-optimality would directly benefit the performance of all algorithms. It would directly lead to more measurements and hence a better performance; the performance of the meander method with regard to measurements per time cannot be improved.

Regarding the coordination methods, the experiment demonstrates also that the *D-*

optimality method indeed leads to a better performance in terms of measurements per NRMSE than the *sequential D-optimality* at the end. However, at the beginning of the exploration, the performance is marginally diminished compared with the *sequential D-optimality*.

5.2. EXPLORATION WITH MULTIPLE GROUND ROBOTS UNDER MAP CONSTRAINTS

The previous section showed an experiment, where three robots explored the magnetic field intensity in the Holodeck. The purpose of this experiment was to demonstrate the coordination methods in real-time with a hardware-in-the-loop experiment. It furthermore showed the effectiveness of the coordination methods together with the exploration. The current section extends the experiment of the previous section by using real sensors, and considers obstacles within the map. This additionally demands of a path-planning algorithm. Another drawback of the previous experiment was the slow computation of the exploration criterion. The following experiment uses an improved implementation, which is described in B.2, to accelerate the evaluation of the exploration criterion.

5.2.1. PREPARATIONS FOR THE EXPERIMENT

Before the experiment can commence, the map has to be constructed and the sensors have to be calibrated together with the collection of a ground truth. This is presented in the following.

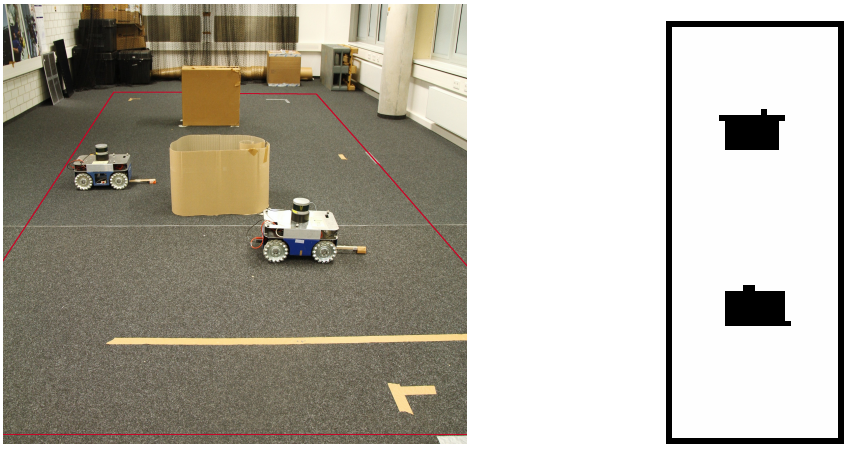
MAP CONSTRUCTION

For the experiment itself, it is assumed that the map is a-priori known to the rover. Yet, the map has to be recorded before the experiment. For this, a single rover is equipped with a light detection and ranging (LIDAR) unit manufactured by Velodyne[5], in particular a *VLP-16*, to scan the environment. The LIDAR emits a laser beam, by which it measures the distance to any surrounding object. This distance together with the position of the slider, estimated by the Vicon system, is then transformed into a map. Because the Vicon position of the slider is used for this map construction, it is simpler compared to SLAM algorithms [148, 149]. The map is then constructed as the rover traverses the environment. For the map construction and the use of the LIDAR, ROS packages are used: the steering of the slider is done with the help of ROS' *navigation stack* [150] together with the *Teb Local Planner* [151]. The map was constructed by using *Octomap* [152]. In Fig. 5.6(a) a picture of the experimental setting is displayed. A white line in this figure represents the borders of the experimental area and the placed cardboard boxes represent the obstacles. Figure 5.6(b) shows the resulting constructed map, which is subsequently used in the experiment.

SENSOR CALIBRATION

In the current experiment, each robot is equipped with a real sensor — a magnetic field sensor by XSens — which is displayed in Fig. 5.7. The sensor is an XSens MTw inertial

[5] `https://velodynelidar.com/`



(a) (b)

Figure 5.6: (a) The experimental setting with obstacles. The white line indicates the experimental area, where the slider can navigate. (b) The constructed map.

measurement unit (IMU), which is shown in Fig. 5.7. This sensor has a three-axis magneto-resistive magnetometer, and is a commercially available sensor package. It additionally provides accelerometers, gyroscopes, and a barometer, which are not used in this experiment.

Although both sensors are from the same product line of the same manufacturer, their absolute measurements differ. This requires the sensors to be calibrated relatively to each other to perceive the environment equally. The approach used in this thesis is described in [153] and is described in more detail in App. E.

Figure 5.7: Magnetic field sensor of XSens — the Xsens MTx.

The authors in [153] assume that the sensor readings of one sensor can be expressed as another sensor's reading by a linear transformation and rotation. This calibration is only useful if the orientation of both sensors is constant during the experiment, which applies to the experiment considered here. If the orientation of the sensors might change due to the navigation of the agents or collision avoidance, the sensors have to be intrinsically calibrated first before the aforementioned sensor alignment can be used. For further information on intrinsic calibration of inertial and magnetic sensors, the reader is referred to [154]. In this experiment, it is ensured that the orientation of the sensors remains the same for measuring such that no further calibration is required.

Also, the sensors react to the metal in the wheels of the robots and how the wheels

are turned. Therefore, the sensors are attached to a wooden rod. The length of the rod was chosen such that the influence of the wheels is minimal and that the stick is as short as possible.

COLLECTING A NEW GROUND TRUTH

For performance evaluations, new ground truth data has to be collected using the calibrated sensors. For collecting the ground truth data, one slider measures the area of the Holodeck on a grid with high resolution, where the distance between each measurement was set to be 5 cm. On each measurement position multiple sensor readings are taken and averaged. The resulting ground truth is displayed in Fig. 5.8

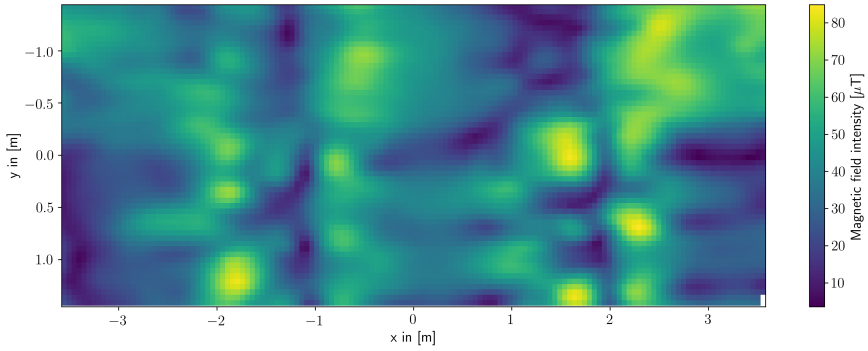


Figure 5.8: Magnetic field intensity of the Holodeck collected for the experiment with real sensors. The measurements were made in 5 cm steps.

5.2.2. EXPERIMENTAL SETTING

In this experiment, the exploration task is the same as in the previous experiment, yet for the current experiment the setting changed substantially. Due to the obstacles, the space in the laboratory becomes more limited such that only two robotic platforms are used in this experiment. Moreover, the same robotic platforms are utilized, but they include now a magnetic field sensor as shown in Fig. 5.9.

The whole system design for this experiment is shown in Fig. 5.10. In this figure the previous system is extended by the map information, the A* path-planner, and the collision avoidance based on the A*-algorithm. Here the popular A*-algorithm [155, 156] is used as a path-planner, see also Appendix D.2. The previously discussed FOVA cannot be used because it does not consider the map constraints. Thus, in this experiment a state machine, which makes use of the A*-algorithm, is created to account for inter agent collision avoidance. The state machine is designed to have a local and a global frame, which are shown in Fig. 5.11. If there is a goal position, a path is planned on a global frame, i.e. the whole map, and, if there is no other robotic system in its path, the goal is reached eventually. If another slider enters the local frame, while the robot is on its way towards the goal, the robot stops and the path within the local frame is re-planned to avoid collisions. If the planner is not able to find a solution on the local frame, the global path is

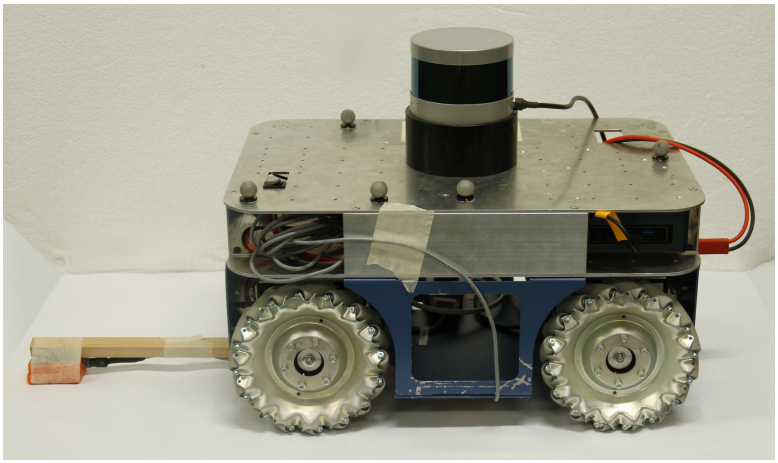


Figure 5.9: The slider, which is one of three robotic platforms in this experiment. It is equipped with an Intel NUC PC and a sensor, which is located on a wooden stick (left of the picture). On top is a lidar of the company Velodyne, but the lidar is not used in this experiment.

re-planned taking the current slider as an obstacle into account. The path planner is represented by the yellow box in Fig. 5.10. On each slider — on the software side — runs the motor controller, which translates the measurement locations into velocity commands for each wheel, the path-planner, and the sensor software. Whereas the exploration and regression is running again on a single computer but in a distributed fashion.

REGRESSION ALGORITHMS

The experiment presented in the previous Sec. 5.1 focused on different coordination strategies, which parallelized the evaluation of the exploration criterion. In this experiment two frameworks for the regression are compared with each other and their influence on the exploration. The two frameworks are the frequentist framework, where the ADMM algorithm belongs to, and the Bayesian framework, where the D-R-ARD algorithm belongs to. Both algorithms can be implemented for the homogeneous learning paradigm and the heterogeneous learning paradigm. Thus, there are four algorithms to compare — ADMM for SOE, ADMM for SOF, D-R-ARD for SOE, and D-R-ARD for SOF. Table 5.1 lists where the algorithms are discussed and where the corresponding exploration criterion is derived in this thesis.

5.2.3. EXPERIMENTAL RESULTS

The NRMSE of all conducted experiments with respect to time and to the number of measurements is shown in Fig. 5.12. The performance is plotted as a scatter plot since each experimental run has a different duration and the time that a measurement takes is not equal due to path lengths and collision avoidance. Also, the number of measurements in each experiment varies. The time steps are asynchronous as the ROS system uses asynchronous interprocess communication, and the number of measurements varies because the computation time for each measurement could be different. All of this

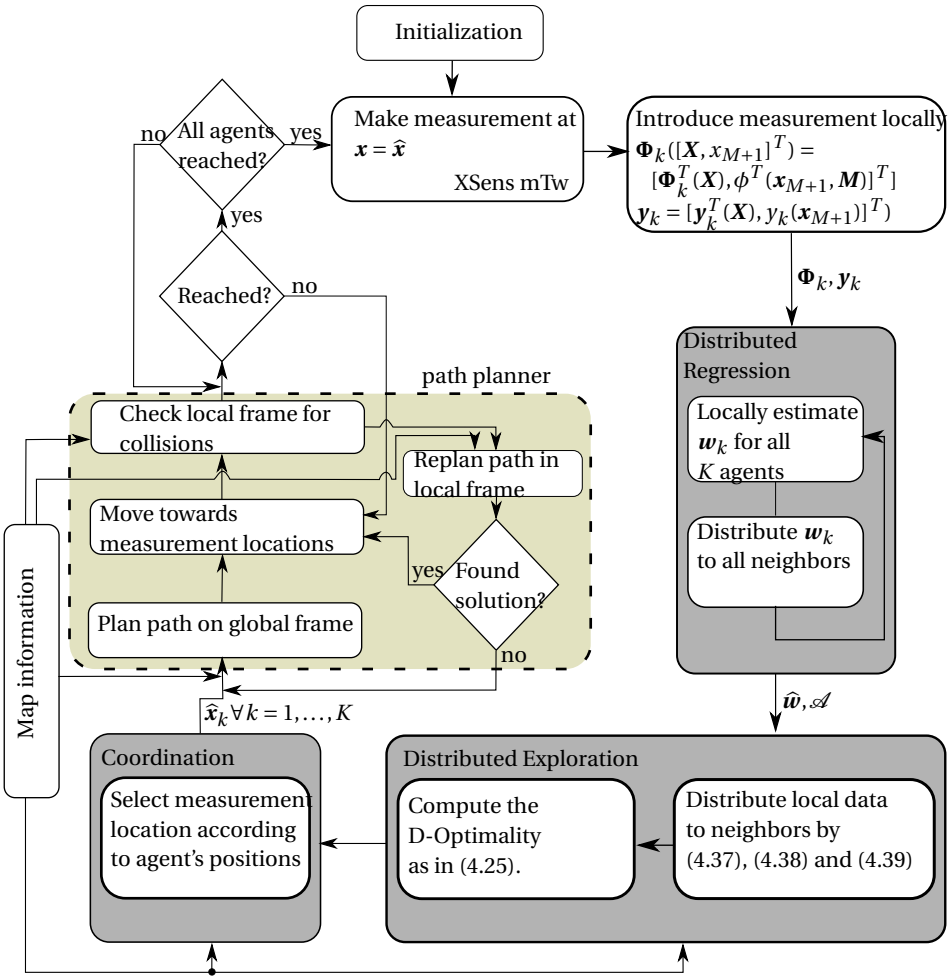


Figure 5.10: System design with additional path planner and map constraints. Each gray box represents interactions between agents. This software setup is representative for the SOE distribution paradigm.

complicates the averaging over multiple algorithm runs such that a scatter plot is more appropriate.

As can be observed in this figure, the variation of the NRMSE for each experiment is low, except for the D-R-ARD for SOF. For both ADMM algorithms four experiments are displayed, whereas for each D-R-ARD algorithm two experiments are shown.

Considering the NRMSE with respect to time (the top part of Fig. 5.12), the D-R-ARD for SOE achieves the best performance. Regarding the ADMM algorithms, the SOE paradigm performs initially better until 1200 s until the SOF paradigm outperforms the SOE paradigm. The weakest performance has the D-R-ARD for SOF. This follows from the distributed structure of the algorithm, which requires computing a matrix inversion in

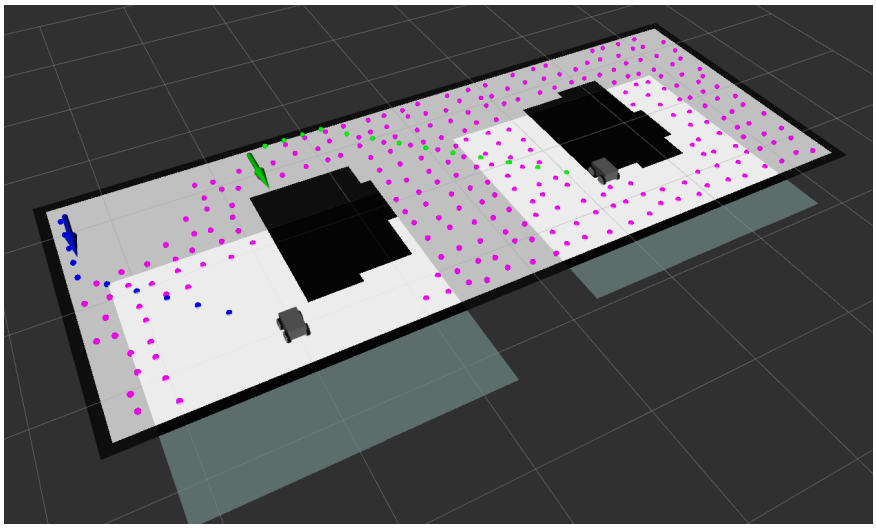


Figure 5.11: This figure exemplifies the local and global frame for the path planner. The global frame is basically the whole map. The local frame is a small highlighted area around each robot. The green and blue dots represent the calculated paths of each robot, and the arrow the target orientation. The pink dots represent the considered measurement locations $\mathcal{X}_k$ of the robot k. Here it is the robot on the right that corresponds to the green path.

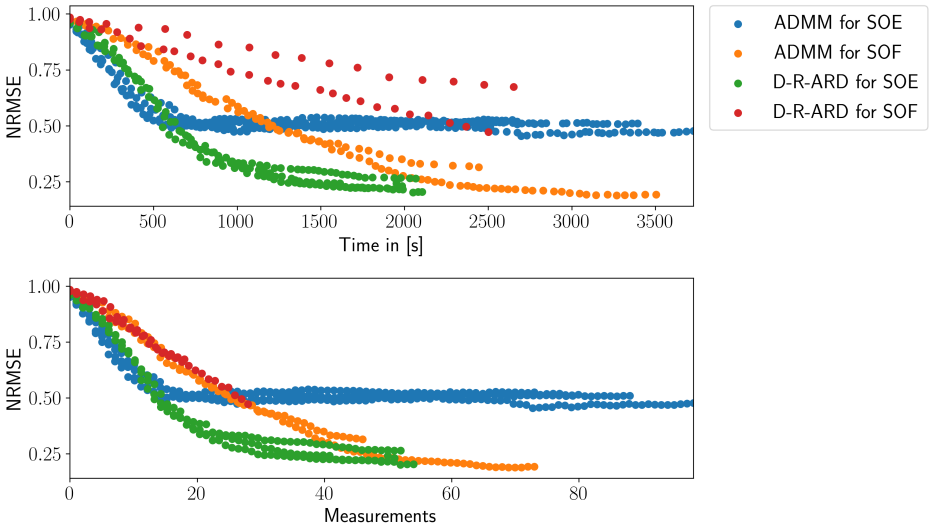


Figure 5.12: The NRMSE of the conducted experiments with respect to time and with respect to the number of measurements.

	Regression introduced in	Exploration introduced in
ADMM for SOE	Sec. 2.4.2	Sec. 4.2.5
ADMM for SOF	Sec. 2.4.3	Sec. 4.2.4
D-R-ARD for SOE	Sec. 3.2.2	Sec. 4.5.1
D-R-ARD for SOF	Sec. 3.3.1	Sec. 4.5.2

Table 5.1: The algorithms that are used in this experiment and the section where they are introduced.

	ADMM for SOE	ADMM for SOF	D-R-ARD for SOE	D-R-ARD for SOF
$\frac{\text{meas}}{\text{min}}$	1.8	1.2	1.8	0.6

Table 5.2: The number of measurements per minute for each algorithm.

each of its iterations. Whereas the corresponding algorithm with the SOE distribution paradigm requires to invert the matrix only once, which drastically increases the performance. To emphasize the aspect of computational efficiency, Table 5.2 shows the acquired measurements per minute. There, the algorithms that are distributed by the SOF paradigm require fewer measurements because these algorithms cannot cache the matrix inversion.

The D-R-ARD algorithms have generally a higher computational complexity compared with the ADMM algorithms. This is because that the D-R-ARD algorithms utilize the gradient and its curvature, which are also referred to as first order and second order derivative, respectively. Using the second order derivative involves the inversion of the Hessian, which can be computationally complex. However, the D-R-ARD algorithms require therefore fewer iterations to converge than the ADMM algorithms, which utilize only the gradient.

Looking at the NRMSE with respect to the number of measurements (the bottom part of Fig. 5.12), the performance of the D-R-ARD for SOF is almost equal to the ADMM for SOF, while the ADMM for SOF is able to achieve more measurements, see also Tab. 5.2. This is explained by the computational complexity of the Bayesian framework and the SOF paradigm, where matrix caching is not possible. Therefore, regarding the SOF distribution paradigm ADMM for SOF is the winner.

Whereas, for the SOE distribution paradigm, the Bayesian framework performs better. In the experiments presented here, the D-R-ARD for SOE achieved a lower NRMSE in total, and it is more measurement efficient compared with the ADMM for SOE algorithm. An explanation for this could be that D-R-ARD for SOE computes the covariance of the parameter weights and does not approximate it. The computed covariance seems then to be better for the D-optimality criterion than the approximated version for the ADMM for SOE shown in Sec. 4.2.5. Also, the ADMM for SOE flattens out at the end, which could result from the gridding in the model. Moreover, the sparsity regularization also intro-

duces some error, as small weights are set to zero. These weights can then not contribute to the prediction. Either the Bayesian methods appear to be much more robust for that, or the ADMM methods require more time to converge if such a discretization and basis functions are used.

To support the claim that the Bayesian framework estimates a better covariance of the parameter weights when the SOE paradigm is applied, Figs. 5.13(a) and 5.13(b) display the estimated magnetic fields and the calculated D-optimalities, which result from the covariance, at different time steps during the experiment. In both figures, the left most plots display the beginning of the experiment and the most right plots show the end result of the experiment. At the beginning of the experiments, both algorithms — ADMM and D-R-ARD for SOE — estimate a sparse covariance with not much difference. As the number of measurements increases the approximated covariance in the frequentist framework becomes smooth whereas the covariance estimated in the Bayesian framework stays sparse with respect to the used basis functions. Looking back at Fig. 4.4 in Sec. 4.2.3 this smoothing effect seems reasonable, as δ is chosen as a compromise between sparsity and a reasonably well approximated covariance.

As a second conclusion, the ADMM for SOE involves a thresholding operator, which is controlled by the same parameter δ. This operator sets all not used basis functions to zero such that in a second step these basis functions cannot be considered for the exploration. The D-R-ARD for SOE, on the other side, estimates a hyper-parameter for each basis function. This way, each basis function and consequently each parameter weight have an individual sparsity parameter, which leads to a covariance that is better conditioned.

5.3. SUMMARY AND CONCLUSION

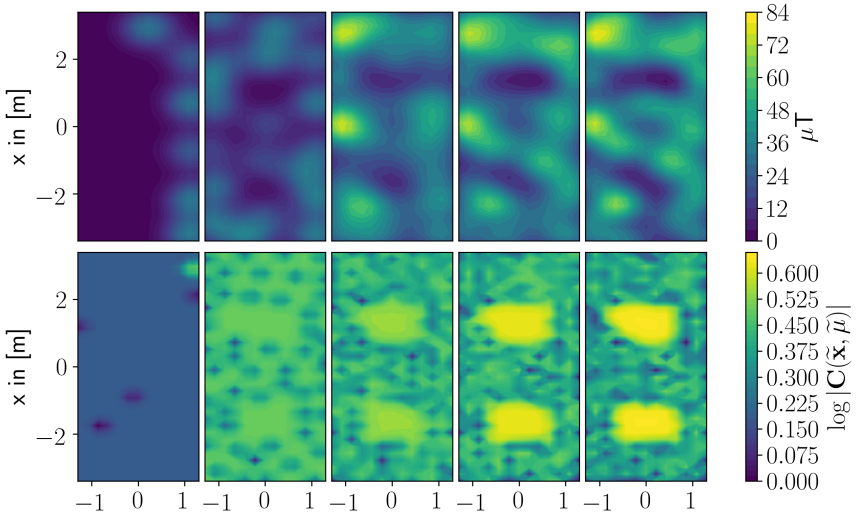
Chapter 5 verified the theoretical results from the Chapter 3 and Chapter 4 in two experiments. The first experiment considered the parallelization of the D-optimality criterion in a hardware-in-the-loop experiment in an obstacle-free environment. Because the experiment involved real robotic platforms, a reactive collision avoidance, FOVA, was utilized. This experiment demonstrated that the *D-optimality* can be evaluated in parallel by using the coordination methods introduced in Sec. 4.4.1. With respect to the number of measurements, a simple parallel sampling of the D-optimality leads to a more effective exploration.

The second experiment evaluated the algorithmic frameworks — the Bayesian and the frequentist framework — in an environment with obstacles and with real sensors. As the environment was not obstacle-free, a path planner had to be used and a collision avoidance that avoided the other agents and the obstacles. For both tasks an A* path planner was used, which featured a global and a local map. Also, the sensors had to be calibrated and a new ground-truth had to be collected.

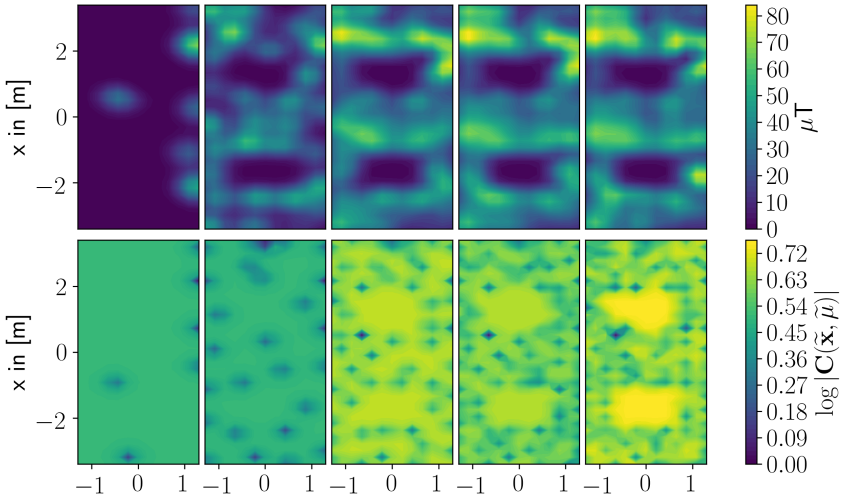
This experiment was conducted multiple times for each of the four proposed algorithms, which are defined in either the Bayesian or frequentist framework and utilize one of the two distribution paradigms. For the homogeneous learning (SOE) the algorithm in the Bayesian framework performed more efficient than the algorithm in the frequentist framework. Although the algorithm in the Bayesian framework is computationally more complex, the estimates and the resulting exploration led to a lower error although fewer

measurements have been collected. This indicates that the measurements are more informative. It might be explained by that the use of an approximated covariance seems to be inferior to the estimated covariance within the Bayesian framework. This seems to be particularly true for the SOE distribution paradigm. For the heterogeneous learning (SOF) the Bayesian method was computationally too demanding and therefore did not collect as many measurements as the corresponding algorithm in the frequentist framework. When comparing the first 30 measurements, the performance of both algorithms coincided, yet this cannot be generalized as the experiment was stopped after approx. 30 min because the NRMSE was not substantially decreasing or the time between the next measurements became too long.

From both experiments it can be concluded that a parallel evaluation of the D-optimality criterion is definitely useful. The experiments show that the whole system with multiple robotic platforms, a distributed inference, and a distributed exploration criterion is real-time applicable.



(a)



(b)

Figure 5.13: (a) SOE with a frequentist framework. (b) SOE with a Bayesian framework. In both figures, the upper row displays the estimates at different time steps and the lower row shows the entropy at the same time steps.

6

Conclusion and Outlook

In this chapter, the main contribution of the thesis is summarized, followed by a reflection on the research questions that have been derived in the first chapter of this thesis. Afterward, the limitations of this thesis are discussed, followed by an outlook and future work.

6.1. Summary of the Main Contribution

This thesis considered the problem of using a spatially distributed robotic system or swarm for exploratory tasks. Although the exploratory task is generic, this thesis is motivated by space exploration. Because of maintenance and control reasons, the swarm had to autonomously decide for its measurement locations and the coordination of each individual robotic system within the swarm. This thesis thus investigated an information driven approach for exploration. Here, a generalized linear model, which uses basis functions, is utilized to estimate parameter weights, from which a resulting covariance is used for the exploration based on the theory of optimal experiment design. Additionally, this thesis assumes that the parameter weights of the generalized linear model are sparse, which means that there are more zero parameter weights than nonzero parameter weights, due to the following reasons: First, most processes found in nature could be represented as a linear combination of some overcomplete basis vectors, where the coefficients of this basis become sparse. This makes it possible to present the model in a compact manner. Second, at the beginning of the exploration the number of measurements is much smaller than the number of parameter weights, which makes the model not uniquely identifiable without a regularization – here a sparse regularization was applied.

To learn these sparse parameter weights from the measurements in a distributed fashion, the ADMM algorithm was selected. This algorithm can be used for multiple different distribution paradigms, and this thesis chose the heterogeneous learning (SOF) and the homogeneous learning (SOE) paradigm. Here, the ADMM algorithm is applied to a sparsity constrained problem, which makes the regularized cost function not twice

differentiable. Hence, the algorithm cannot provide a covariance of the parameter weights. Therefore, this thesis also investigated into Bayesian methods for solving sparse generalized linear models and particularly SBL methods. Due to the nature of Bayesian methods, the variance of the posterior PDF of the parameter weights can be used. SBL methods of the so called type-II were identified to be promising for the considered problem. So in this thesis the development of distributed type-II SBL algorithms is shown based on consensus techniques to solve for the parameter weights. In particular, this thesis presents the derivation of distributed versions of two SBL algorithms, the FMLM and the R-ARD, for the distribution paradigm SOE (DFMLM and D-R-ARD respectively). It turned out that for a small number of measurements the DFMLM is not as robust as the D-R-ARD and required more iterations to converge to a solution. For a larger number of measurements, the DFMLM is superior to the D-R-ARD. Simulations with different swarm topologies showed that the DFMLM has a reduced communication load compared with the D-R-ARD. Nonetheless, the robustness especially for a small number of measurements was considered to be of higher importance for the exploration, especially at the beginning, and therefore the D-R-ARD was used; the DFMLM had problems to converge with only very few measurements.

Afterward the thesis presented the derivation of a D-R-ARD algorithm for the SOF distribution paradigm. It happened that the distributed version of D-R-ARD required an approximation of an intermediate variable during the estimation as it could not be estimated in a distributed manner otherwise. This thesis shows an empirical investigation of the convergence properties of the new algorithm with the approximation for the considered problems. The result was that the D-R-ARD for SOF converged to the same solution. Also, multiple simulations showed that the difference between the exact and approximated solution is negligible during the estimation. As a side effect the distributed algorithm leads to sparser results due to the used approximation at the expense of an increase NRMSE.

The exploration utilized the D-optimality criterion, which originates from the theory of optimal experiment design. Due to the distribution of the data — either the distribution of the measurements for SOE or the distribution of the models for SOF — this thesis presented for both distribution paradigms the derivation of a distributed version of the D-optimality criterion. For the cost-function that is sparsity regularized the Hessian, which can be used as a covariance on the parameter weights, cannot be calculated. It turns out that the sparsity induced cost-function can be altered into a Ridge-regression based on the sparse estimates. This way, it is possible to get an approximation of the Hessian matrix of the sparsity induced cost-function for the nonzero parameter estimates, which can then be used in the D-optimality exploration criterion. The analysis shows however that the penalty parameter for sparsity of the sparse cost-function could influence the approximation negatively if it is chosen to be too large. For the Bayesian methods, the covariance of the weight posterior PDF is used for the D-optimality criterion. In the considered setting, it does not require an approximation and can be computed exactly.

As the system is distributed, the exploration has to be formulated in a distributed fashion as well. In particular, the problem was that the data is distributed within the swarm. Therefore, this thesis showed how to estimate the D-optimality criterion for a

potential measurement location for each distribution paradigm by means of a consensus algorithm. Unfortunately, the optimization of the D-optimality criterion is NP-hard such that this criterion has to be evaluated for each potential measurement location. This thesis shows in simulations that the distributed D-optimality criterion leads to a better performance than randomly selected measurement locations or a systematic traversing of an area.

Furthermore, this thesis also investigated into the influence of different basis functions on the D-optimality criterion and how to coordinate the robotic systems. One result was that the parallel sampling of the D-optimality had a better performance than the sequential sampling. For the sequential sampling, the agents moved in turns to the location with the lowest D-optimality, whereas in the parallel sampling the first agent moved to the location with the lowest D-optimality, the second agent to the location with the second lowest D-optimality, and so forth. It turned out that the parallel sampling was more robust against the discretization effects of the environment's gridding, which led to aliases in the D-optimality criterion.

After the analysis of the building blocks of this work — the distributed parameter weight estimation and the distributed exploration — experimental validations demonstrated how the proposed system might work together. First, this thesis shows a hardware-in-the-loop experiment, where the controller of the robotic platforms and the positioning system introduces position noise into the system. This experiment validated the results from the coordination strategies without a real sensor, as the sensor might introduce additional measurement noise into the system. Thus, in a second experiment, a sensor device was added to each robotic platform and, additionally, the environment was populated with obstacles. Also in this experiment, the different algorithms, i.e. the D-R-ARD and the ADMM, and their resulting D-optimality criteria were compared. The result of this experiment showed that the estimated covariance of the Bayesian methods led to a better performance for the SOE distribution paradigm although the Bayesian methods are computationally more complex. For the SOF distribution paradigm, the performance with respect to the number of measurements was not worse but the D-R-ARD for SOF is computationally complex such that the computation is very time demanding. Hence, the D-R-ARD for SOF was not able to acquire as many measurements during the time of the experiment as the corresponding ADMM algorithm.

This summary ends by reflecting the guiding research questions from the introduction by providing comprehensive and concise answers.

Q 1:Can model based exploration applied to a swarm with a distributed structure in real-time when using the entropy of a model's parameters? Although the calculation of the D-optimality and the estimation of the parameter weights require more time than "just measuring", the experiments indicate that it is possible to use such a system for exploration in real-time.

Q 1.1: If optimal design strategies are used for entropy driven exploration, what is the effect of the regularization on the exploration? This work considers a sparse regularization for the parameter weight estimation such that the resulting covariance also becomes sparse. Nevertheless, such regularizations are often controlled via a penalty parameter in the estimation. For the parameter estimation with ADMM, as shown in Sec. 4.2.3, the value of this penalty term can negatively influence the entropy driven ex-

ploration. If the value of δ is too high, the D-optimality results in an exploiting behavior, i.e. the agent visits already measured locations, while if the value of δ is small the D-optimality results in an explorative behavior. However, this penalty parameter is chosen for the *estimation* and not for the *exploration*. Consequently, this penalty parameter has to be chosen adequately with respect to the data and the desired exploration behavior. This is different for the Bayesian methods, as they learn the hyper parameter from the data and for each parameter weight individually. As the hyper parameters influence on the covariance, the D-optimality criterion results in an exploiting behavior, if there is less information about a parameter weight. Likewise, if there is enough information about the parameter weights, the D-optimality results in an explorative behavior.

Q 1.2: What are the benefits of a sparse regularization for the exploration with a swarm? As zero parameter weights do not contribute to the covariance matrix, the size of the covariance matrix becomes reduced. For the Bayesian methods, the sparse regularization lead to a more distinct exploration behavior, where the algorithm decides between exploration and exploitation. Generally, zero parameter weights lead to a lower computational complexity. For a distributed case, it could lead also to a smaller load on the communication, as the matrix and vector sizes become reduced, but this has not been investigated in this thesis.

Q 2: How should a model be distributed to be most suitable for model based exploration employed for a swarm, and how well does the distributed model approximate the measurements? The heterogeneous learning often leads to a smaller error metric and thus leads to a better approximation, but for the estimation it requires to distribute the measurements and intermediate steps of the estimation in the swarm. Whereas the homogeneous learning only communicates intermediate parameter weight estimates, and it can be implemented more efficiently, due to caching of matrix inversions. Therefore, the exploration is equally possible with both distribution paradigms, but depending on the application the communication load can vary. The communication load is, however, not considered in this thesis. Regarding the estimation, the heterogeneous distribution paradigm yielded the lowest error with an equal number of measurements.

Q 2.1: How do Bayesian methods perform compared with non-Bayesian methods when applied to distributed spatial regression for swarm exploration? Bayesian methods do not approximate the covariance matrix, and the covariance matrix is not dependent on a predefined penalty parameter. Hence, the exploration leads to better measurement locations and consequently to a better performance. That being said, Bayesian methods are computationally more complex but require fewer measurements compared with classical methods such as ADMM. Thus Bayesian methods are more effective and could be used in applications where the measurement duration or cost is high, and ADMM methods are more effective, if the measurement duration and cost is small.

Q 2.2: When using a distributed model, what data is required to communicate for exploration and distributed spatial regression? This question cannot shortly be answered, as it not only depends on the distribution paradigm but also on the used algorithm. Generally speaking, for homogeneous learning the communication for estimation and exploration is dominated by the number of used parameter weights and the sparsity, whereas the estimation and exploration for heterogeneous learning is dominated by the number of measurements. Tables 2.1, 3.1, and 4.1 additionally provide an

answer to this question for the estimation with classical methods, for the estimation with Bayesian methods, and for exploration, respectively.

6.2. DISCUSSION AND LIMITATIONS

The following chapter puts the thesis in a broader context and discusses limitations and simplifications that have been made in this work. In particular, this thesis' introduction started with the extraterrestrial exploration of the planet Mars. Nowadays, the fifth rover — Perseverance [157] — and the first helicopter — ingenuity [158] — of the NASA are roving/flying on Mars with the objective to find previous life on Mars. Furthermore, another nation, China, landed its first Rover on Mars as well [159]. Regarding Perseverance, for the search of previous life on Mars this rover has cameras [160, 161], gas chromatographs, mass spectrometers, and further sensors [162] on board, whereas this thesis only considered the magnetic field only. However, the methods provided in this thesis could be considered as sensor agnostic; the magnetic field was only considered here for the simplicity in measuring and understanding of the methods investigated in this thesis. Other static processes such as temperature distributions, a map of a distribution of certain elements, and the spatial distribution of radiation could be measured more time effectively with a higher grade of autonomy using the proposed methods. The assumptions for these processes are that they are static or slowly changing and smooth.

The swarm aspect considered in this thesis is also becoming more and more a realistic. With Perseverance and Ingenuity [157, 158], two robotic platforms for exploration have been sent to Mars as the same payload. It is therefore only reasonable, also regarding to colonization plans for Mars [163, 164], that multiple robotic platforms, i.e. swarms, will be sent to Mars. For more robotic platforms on Mars a holistic and autonomous control of the robotic platforms is crucial; and this thesis provided there some new insights to the research field. Yet without loss of generality, this thesis considered swarms up to a size of ten swarm elements in simulations and up to three swarm elements in experiments. Regarding the experiments, the reasons for this number were mainly the space in the laboratory and availability of robotic platforms. For the simulations the reasons were computational complexity as the whole system was simulated on a single computer. More efficient implementations could however reduce the computational load, as partially shown in Appendix B.2.

The problem of the scalability of the swarm, e.g. increasing the number of swarm elements or increasing the model size, has not been addressed in this thesis. For example, if considering that each sensor has only a limited spatial view, the measurements and the estimations could be distributed more efficiently. With regard to the scalability of the parameter estimation, it is probably not necessary to distribute all parameters for homogeneous learning or all measurements for heterogeneous learning. This way larger models and swarms could be realized as the computational load is not equally increasing. A realization of this might result in mixing the distribution paradigms discussed in this thesis. The effect of this mixing on the exploration is not discussed in this thesis, but it can be assumed that both properties of both distribution paradigms are inherited to a mixed version. Hence, the distribution of parameter and measurements could become a complex protocol, which should maintain where to send which measurements and parameter values for the estimation. For dynamic robotic platforms, such a protocol could

become even more complex as the robotic platforms change their location after each measurement. It is therefore also difficult to argue if one distribution paradigm is better compared with the other, as for larger swarms probably mixtures of both paradigms are employed.

As the topic presented in this thesis is highly interdisciplinary other aspects have been simplified. The aspects of a delay due to communication and data loads for specific communication channels have been more or less neglected. Also, the thesis almost always assumed a perfect localization of the swarm. Just recently, the introduction of a communication delay, a load of the communication channel, and the introduction of a localization are investigated in [14]. For the last experiment, this thesis assumed that the map of all obstacles is known beforehand. A robotic system on Mars would require that such a map is estimated accurately.

6.3. OUTLOOK

The following section denotes future research topics that either emerge from the results of this thesis or that could extend the presented system. One aspect that addresses both of the aforementioned topics would be the investigation of the scalability of a swarm with respect to the number of swarm elements and the number of parameters. Thus, scalability means on the one hand how the robotic systems in the swarm scale up such that they can still explore effectively. This would involve research on mixtures of the distribution paradigm [91] and how the estimation results can be distributed for exploration. On the other hand, scalability could also be invested regarding the size of the parameter space and the discretization of the exploration area. Both influence the computational complexity of the exploration method in this thesis.

For the frequentist approach in this thesis, the hyper-parameter δ could be further investigated. This hyper-parameter was introduced to primarily control the degree of sparsity, but it also controls the exploration versus exploitation ratio. The idea would be to actively control this parameter to put more emphasis on exploration or exploitation. This thesis assumed that δ is constant and that it is adjusted once for the estimation, but it is unclear what would happen if this parameter is tuned for the evaluation of the exploration criterion.

It would also be useful to investigate a more effective evaluation of the exploration criterion. Currently, the exploration criterion is evaluated for each potential measurement location. A gradient based evaluation of this criterion would require to convexify the non-convex criterion or put constraints on its evaluation. Furthermore, if multiple processes should be explored at the same time, would a superposition of exploration criteria still be effective? In particular, the exploration of a map together with the exploration of unknown processes, would be a realistic scenario because in a real scenario the map might be unknown. In a follow-up step, the exploration could be combined with the estimation algorithm; i.e. during the estimation algorithm, the exploration criterion is evaluated as well. The movement of the swarm depends then of the current intermediate parameter estimate. Then, it might not make sense to estimate next measurement locations, but to estimate directions, which provide — based on the current iteration — the most information. A hypothesis could be that the robots in the swarm would traverse a path in the environment that follows partly the gradient of the estimation and partly

the gradient of the D-optimality.

Likewise, for a more effective evaluation of the exploration criterion, the distribution of the agents could be exploited more. If the next measurement location should be close to a particular agent, the whole D-optimality has not to be evaluated. Consequently some gathered information can be left out for exploration, which reduces the computational complexity of the exploration. This way the coordination of the agents could be integrated into this evaluation.

As another approach to the exploration, the path of the robot could be included. When a path-planner for the navigation in a map is used, the intermediate steps in the resulting path could also be influenced by the exploration. Then, the robotic platforms in the swarm could decide for longer paths as they maximize the information, but at the same time have to decide for paths such that they reach their destination. First thoughts on single agent systems with an informative path planner have been looked at in [140, 165]. Another question in this context would be, when to change the original destination when information has been gathered along the trajectory; may be the originally estimated measurement location is not useful anymore.

This work could furthermore be extended by learning the basis functions that are used for the regression. The problem then — especially at the beginning of the exploration — would be the low number of measurements because learning the basis functions introduces even more unknowns to the system than measurements. Potentially, the exploration might be influenced by learning the basis functions and vice-versa.

A

Crossvalidation for the Magnetic Field in the Holodeck

This section briefly presents the reconstruction of a part of the magnetic field in the Holodeck when using the ADMM algorithm. Two important parameter for the estimation are δ and σ. The first parameter δ controls the sparsity, i.e. the number parameter weights that are non-zero. Thus, a large value for δ would result in few non-zero parameter weights. The second parameter σ controls the width of the Gaussian basis functions. To find good values for these parameters, a cross validation is used.

Figure A.1 presents the reconstruction of the magnetic field in the Holodeck when using the ADMM algorithm depending on the two parameters σ and δ. For generating Fig. A.1 an SNR of 30 dB is assumed and a measurement to parameter ratio of $M/N = 40/400 = 0.1$. With the aforementioned argumentation and with respect to to Fig. A.1 reasonable values for σ and δ are $\sigma = [0.25 \ldots 0.4]$ and $\delta = [0.08 \ldots 0.25]$. Here σ is large enough that the basis functions spatially spans over a relatively large area, and δ is large enough that the resulting estimation is sparse.

The NRMSE for each of the cross validations in A.1 is shown in Fig. A.2. There, it can be observed that a narrow Gaussian basis function (small σ) leads to a lower error, but then the basis function can only represent a smaller area. However, if σ is chosen to be larger, less sparsity (small δ) is required to yield a lower NRMSE. Thus, there is no optimal choice for either variables and a trade-off has to be made regarding the coverage of the basis functions (σ) and the sparsity regularization (δ). When δ is chosen according to [74, Section 5.9], i.e. $\delta = \lambda^{-\frac{1}{2}}\sqrt{2\log(M)}$, then $\delta = 0.015$. Figure A.3 evaluates the number of measurements vs δ for this scenario for different SNR values. If the SNR is high, δ is low, and if the SNR is low, δ is large. This expresses that a low SNR requires more regularization as the measurements are noisy. On the other side a high SNR means that the measurements posses only little noise and therefore require less regularization.

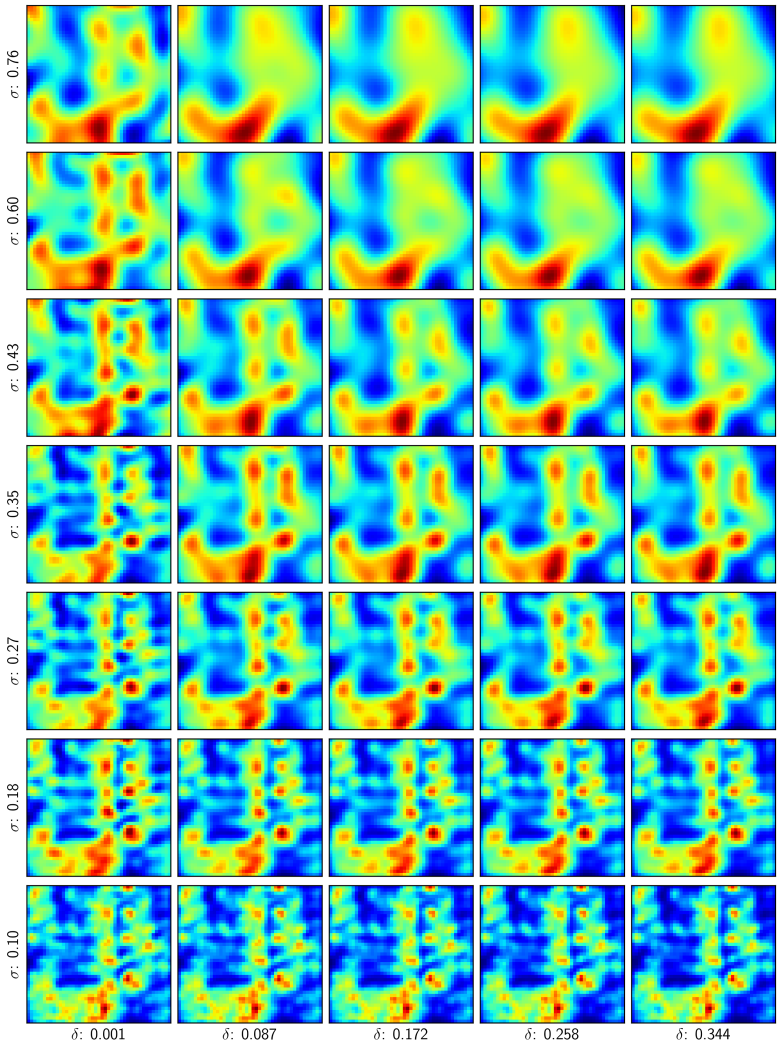


Figure A.1: The reconstruction of the magnetic field in the Holodeck with the LASSO solver for varying lasso parameter δ and varying kernel widths σ for Gaussian basis functions.

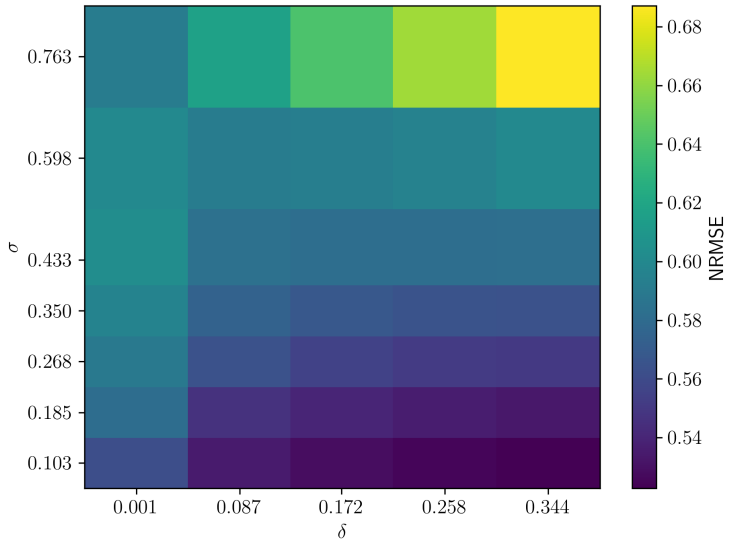


Figure A.2: The NRMSE for the cross validation considered in Fig. A.1.

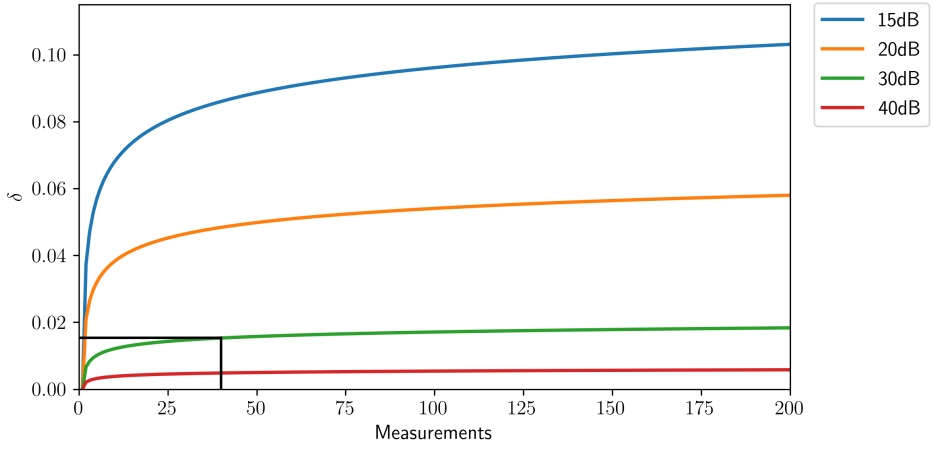


Figure A.3: The choice of δ according to [74, Section 5.9] for varying SNR and number of measurements. The black lines indicate the choice of δ for the cross validation considered in Fig. A.1.

A small remark: the parameter σ is also important for the exploration because small values of σ reduce the spatial size of the Gaussian basis functions; they become narrower. As the next measurement locations are chosen according to the model's ability to predict values at unseen locations, the model can only predict at very close locations. Another problem can occur it the Gaussian basis function is significantly smaller than the discretization. Then, the number of basis functions is not increased, and the parameter weights can become very large to account for measurements between the basis functions. This leads to instabilities in the model and high variances at these locations. Therefore, the exploration is going to consider those measurement locations more often, although it might not be necessary.

B

D-Optimality for Exploration

B.1. Adding of Basis Functions to the D-optimality Criterion

The D-optimality is used to estimate new measurement locations $\hat{\boldsymbol{x}}$ by solving

$$\hat{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}} \in \mathcal{X}}{\arg\min} \det \left\{ \left(\widetilde{\boldsymbol{\Phi}}^T(\tilde{\boldsymbol{x}}) \widetilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{x}}) + \alpha \boldsymbol{I} \right)^{-1} \right\}, \tag{B.1}$$

where $\widetilde{\boldsymbol{\Phi}} \in \mathbb{R}^{M \times N}$ and $\alpha > 0$ ensures through diagonal loading that the inverse always exists. For the problems considered in this thesis, the diagonal loading yields from the sparsity regularization of the regression problems. Due to this sparsity regularization and due to localized basis functions, it is necessary to add also an additional column to $\widetilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{x}})$ as shown in the following. Consider therefore a scenario with one measurement, $M = 1$, and one potential new measurement, but no column is added to $\widetilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{x}})$, i.e.

$$\widetilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \boldsymbol{u}^T(\boldsymbol{x}) \\ \boldsymbol{v}^T(\tilde{\boldsymbol{x}}) \end{bmatrix}, \tag{B.2}$$

where $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^N$ and $\boldsymbol{v}(\tilde{\boldsymbol{x}}) \in \mathbb{R}^N$ are two measurement rows in $\widetilde{\boldsymbol{\Phi}} \in \mathbb{R}^{2 \times N}$, and $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are a measured position and a potential measurement location, respectively. Please note that $\boldsymbol{u}$ and $\boldsymbol{v}$ are constructed by N local, yet only relevant basis functions due to the sparsity constraints; irrelevant basis functions have been removed through the sparsity regularization. In the following analysis the dependency on $\boldsymbol{x}$ is dropped for better readability.

Then, consider

$$\begin{aligned}\log\det\{(\widetilde{\boldsymbol{\Phi}}^T\widetilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I})^{-1}\} &= -\log\det\{\widetilde{\boldsymbol{\Phi}}^T\widetilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I}\}\\ &= -\log\det\{\boldsymbol{u}\boldsymbol{u}^T+\boldsymbol{v}(\widetilde{\boldsymbol{x}})\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})+\alpha\boldsymbol{I}\}\\ &= -\log\{\det\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\}\det\{1+\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\widetilde{\boldsymbol{x}})\}\}\\ &= -\log\det\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\}-\log\det\{1+\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\widetilde{\boldsymbol{x}})\}.\end{aligned} \tag{B.3}$$

The first term in (B.3) is independent of $\widetilde{\boldsymbol{x}}$ and therefore constant if the D-optimality is optimized over $\widetilde{\boldsymbol{x}}$. The second term can be reformulated with the matrix inversion lemma as

$$\begin{aligned}&\log\det\{1+\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\widetilde{\boldsymbol{x}})\}\\ &\quad= \log\det\{1+\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\left(\alpha^{-1}\boldsymbol{I}-\alpha^{-1}\boldsymbol{u}(1+\boldsymbol{u}^T\alpha^{-1}\boldsymbol{u})^{-1}\boldsymbol{u}^T\alpha^{-1}\right)\boldsymbol{v}(\widetilde{\boldsymbol{x}})\}\\ &\quad= \log\det\left\{1+\alpha^{-1}\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2-\frac{\alpha^{-2}(\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\boldsymbol{u})^2}{1+\alpha^{-1}\|\boldsymbol{u}\|^2}\right\}\\ &\quad= \log\det\left\{1+\frac{\alpha^{-1}\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2+\alpha^{-2}\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\|\boldsymbol{u}\|^2-\alpha^{-2}\left(\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\boldsymbol{u}\right)^2}{1+\alpha^{-1}\|\boldsymbol{u}\|^2}\right\}\\ &\quad= \log\det\left\{1+\frac{\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}+\frac{\alpha^{-1}\left(\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\|\boldsymbol{u}\|^2-(\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\boldsymbol{u})^2\right)}{\alpha+\|\boldsymbol{u}\|^2}\right\}.\end{aligned} \tag{B.4}$$

Now, the last term in the determinant of (B.4) is basically a Cauchy-Schwarz inequality:

$$\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\|\boldsymbol{u}\|^2 \geq (\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\boldsymbol{u})^2. \tag{B.5}$$

Note that from the Cauchy-Schwarz inequality, the term under the logarithm in (B.4) is positive. As such, there are now two cases to analyze: first, either the new measurement location $\widetilde{\boldsymbol{x}}$ is far away from the other measurement location, or, second, $\widetilde{\boldsymbol{x}}$ is close to the other measurement location $\boldsymbol{x}$.

1. If $\widetilde{\boldsymbol{x}}$ is "far away" from $\boldsymbol{x}$, i.e. $\boldsymbol{u}\perp\boldsymbol{v}(\widetilde{\boldsymbol{x}})$, and $\boldsymbol{v}^T(\widetilde{\boldsymbol{x}})\boldsymbol{u}=0$. This can be considered in (B.4) such that

$$\begin{aligned}&\log\det\left\{1+\frac{\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}+\frac{\alpha^{-1}\left(\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\|\boldsymbol{u}\|^2-0\right)}{\alpha+\|\boldsymbol{u}\|^2}\right\}\\ &\quad= \log\det\left\{1+\frac{\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2+\alpha^{-1}\left(\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\|\boldsymbol{u}\|^2\right)}{\alpha+\|\boldsymbol{u}\|^2}\right\}\\ &\quad= \log\det\left\{1+\frac{\alpha^{-1}\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\left(\alpha+\|\boldsymbol{u}\|^2\right)}{\alpha+\|\boldsymbol{u}\|^2}\right\}=\log\det\{1+\alpha^{-1}\|\boldsymbol{v}(\widetilde{\boldsymbol{x}})\|^2\}.\end{aligned} \tag{B.6}$$

   Now, if $\widetilde{\boldsymbol{x}}$ is far away from $\boldsymbol{x}$, the N *local* basis functions in the model are not defined for these far location. Subsequently, under these conditions, $\boldsymbol{v}(\widetilde{\boldsymbol{x}})$ becomes zero and therefore the D-optimality yields

$$\log\det\{(\widetilde{\boldsymbol{\Phi}}^T\widetilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I})^{-1}\}=-\log\det\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\}. \tag{B.7}$$

2. If $\tilde{\boldsymbol{x}}$ is close to $\boldsymbol{x}$ such that $\boldsymbol{v}^T(\tilde{\boldsymbol{x}})\boldsymbol{u} \rightarrow \|\boldsymbol{v}(\tilde{\boldsymbol{x}})\|\|\boldsymbol{u}\|$, i.e. $\boldsymbol{u}$ and $\boldsymbol{v}(\tilde{\boldsymbol{x}})$ are similar (for localized basis functions, this is the case if the measurement locations are close). Then, the last term in the determinant of (B.4) becomes zero such that

$$\log\det\left\{1+\frac{\|\boldsymbol{v}(\tilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}+\frac{0}{\alpha+\|\boldsymbol{u}\|^2}\right\}=\log\det\left\{1+\frac{\|\boldsymbol{v}(\tilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}\right\}. \tag{B.8}$$

Thus, under these conditions the D-optimality yields

$$\log\det\left\{(\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I})^{-1}\right\}=-\log\det\left\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\right\}-\log\det\left\{1+\frac{\|\boldsymbol{v}(\tilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}\right\}. \tag{B.9}$$

As the D-optimality criterion, as shown in (B.1), gets minimized, it would always measure at already measured measurement locations, as the results in (B.7) and (B.9) indicate

$$-\log\det\left\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\right\}-\log\det\left\{1+\frac{\|\boldsymbol{v}(\tilde{\boldsymbol{x}})\|^2}{\alpha+\|\boldsymbol{u}\|^2}\right\}<-\log\det\left\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\right\}. \tag{B.10}$$

Therefore, locations close to already measured locations are preferred by this criterion if no column is added. If, however, a column at the hypothetical measurement location is added, it is ensured that the model is defined for each considered location $\tilde{\boldsymbol{x}}$ as shown in the following. When adding a column to the design matrix, it becomes

$$\tilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})=\begin{bmatrix}\boldsymbol{u}^T & \boldsymbol{v}(\boldsymbol{x},\check{\boldsymbol{\mu}})\\ \boldsymbol{v}^T(\tilde{\boldsymbol{x}}) & v(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})\end{bmatrix}, \tag{B.11}$$

where $\check{\boldsymbol{\mu}}$ is the center position of a newly added local basis function. As shown in Sec. 4.2.2, the D-optimality becomes

$$\begin{aligned}\log\det\left\{(\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I})^{-1}\right\}&=-\log\det\left\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\right\}-\log q(\check{\boldsymbol{\mu}})\\ -\log\det\Bigg\{&\underbrace{1+\boldsymbol{v}^T(\tilde{\boldsymbol{x}})(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\tilde{\boldsymbol{x}})}_{\text{(B.4)}}+\frac{\left(v(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})-\boldsymbol{v}^T(\boldsymbol{x},\check{\boldsymbol{\mu}})\boldsymbol{u}^T(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\tilde{\boldsymbol{x}})\right)^2}{q(\check{\boldsymbol{\mu}})}\Bigg\},\end{aligned} \tag{B.12}$$

with $q(\check{\boldsymbol{\mu}})=\|v(\boldsymbol{x},\check{\boldsymbol{\mu}})\|^2-\boldsymbol{v}(\boldsymbol{x},\check{\boldsymbol{\mu}})\boldsymbol{u}^T(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{u}\boldsymbol{v}^T(\boldsymbol{x},\check{\boldsymbol{\mu}})$. Adding a column, as shown in (B.12), adds an additional term, which considers that a new basis function can be activated. Considering the cases from above

1. If $\tilde{\boldsymbol{x}}$ is far away from $\boldsymbol{x}$, then the D-optimality yields

$$\begin{aligned}\log\det\left\{(\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}}+\alpha\boldsymbol{I})^{-1}\right\}&=-\log\det\left\{\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I}\right\}-\log q(\check{\boldsymbol{\mu}})\\ -\log\det\Bigg\{&1+\frac{\left(v(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})-\boldsymbol{v}^T(\boldsymbol{x},\check{\boldsymbol{\mu}})\boldsymbol{u}^T(\boldsymbol{u}\boldsymbol{u}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{v}(\tilde{\boldsymbol{x}})\right)^2}{q(\check{\boldsymbol{\mu}})}\Bigg\}.\end{aligned} \tag{B.13}$$

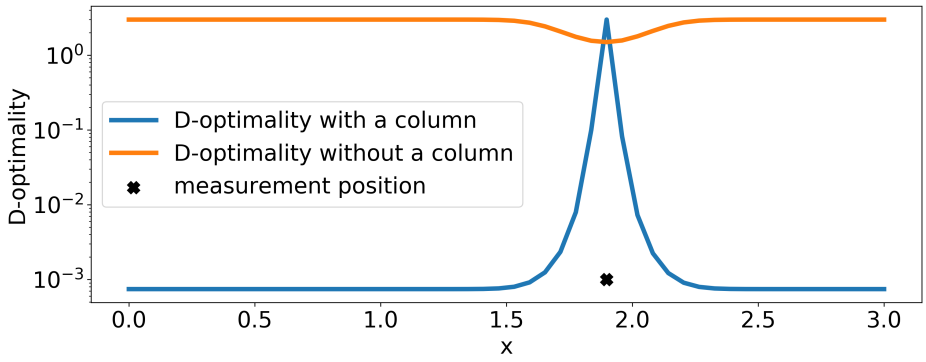


Figure B.1: The resulting D-optimality for a one-dimensional environment for adding a column to $\tilde{\boldsymbol{\Phi}}$ and for not adding a column to $\tilde{\boldsymbol{\Phi}}$. In each cases, the exploration algorithm would choose the minimum of either the orange curve or the blue curve.

2. If $\tilde{\boldsymbol{x}} \approx \boldsymbol{x}$, then the additional term that considers the newly added basis function becomes zero because $\boldsymbol{u}$ and $\boldsymbol{v}$ are evaluated at the same positions. Thus the D-optimality yields

$$\log\det\{(\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}} + \alpha\boldsymbol{I})^{-1}\} = -\log\det\{\boldsymbol{u}\boldsymbol{u}^T + \alpha\boldsymbol{I}\}. \tag{B.14}$$

The additional term results in a more explorative behavior. Figure B.1 shows this also graphically for one measurement position and the D-optimality evaluated along $\boldsymbol{x}$.

B.2. FAST EVALUATION OF THE D-OPTIMALITY

The computations of the D-optimality criterion can be cumbersome if applied for each measurement location sequentially. A vectorized computation of the criterion however could exploit the benefits of faster computations. Considering (4.25), which is put here again for convenience

$$\begin{aligned}\log\det\tilde{\boldsymbol{C}} = \log|\boldsymbol{A}| + \log\Big(&q(\check{\boldsymbol{\mu}}) + q(\check{\boldsymbol{\mu}})\boldsymbol{\phi}_{\mathscr{A}}^T(\tilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M}) \\ &+ \left(\phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}(\check{\boldsymbol{\mu}})^T\boldsymbol{A}^{-1}\boldsymbol{\phi}_{\mathscr{A}}(\tilde{\boldsymbol{x}},\boldsymbol{M})\right)^2\Big),\end{aligned} \tag{B.15}$$

where the first log-term can be neglected as it is independent of the potential measurement location $\tilde{\boldsymbol{x}}$ and $q(\check{\boldsymbol{\mu}}) = b(\check{\boldsymbol{\mu}}) - \boldsymbol{c}^T(\check{\boldsymbol{\mu}})\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{\mu}})$. Please note that according to (4.19) $\boldsymbol{A}^{-1}$ always exists. For the following discussion the term $\phi(\tilde{\boldsymbol{x}},\check{\boldsymbol{\mu}})$, which represents a basis function centered at $\check{\boldsymbol{\mu}}$ and evaluated at a potential measurement location $\tilde{\boldsymbol{x}}$, is important. Assuming the two-dimensional case and that there are in total $T \in \mathbb{N}$ potential measurement locations and $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1^T,\ldots,\tilde{\boldsymbol{x}}_T^T]^T \in \mathbb{R}^{T\times 2}$ are all potential measurement locations in matrix notation, then it is possible to construct the vector $\boldsymbol{\phi}_t(\tilde{\boldsymbol{X}}) = [\phi(\tilde{\boldsymbol{x}}_1,\tilde{\boldsymbol{x}}_t),\ldots,\phi(\tilde{\boldsymbol{x}}_T,\tilde{\boldsymbol{x}}_t)]^T \in \mathbb{R}^T$. The vector $\boldsymbol{\phi}_t(\tilde{\boldsymbol{X}})$ evaluates a basis function at all potential measurement locations, if the basis function is centered at the t-th potential

measurement location $\widetilde{\boldsymbol{x}}_t$. Then the matrix, $\widetilde{\boldsymbol{\Phi}}(\widetilde{\boldsymbol{X}}) \triangleq [\boldsymbol{\phi}_1(\widetilde{\boldsymbol{X}}),\dots,\boldsymbol{\phi}_T(\widetilde{\boldsymbol{X}})] \in \mathbb{R}^{T\times T}$ can be constructed, which comprises all basis functions, evaluated and centered at all potential measurement locations. Likewise, the matrix $\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X}) \triangleq [\boldsymbol{\phi}_1(\boldsymbol{X}),\dots,\boldsymbol{\phi}_T(\boldsymbol{X})] \in \mathbb{R}^{M\times T}$ consists of all basis functions, evaluated at all existing measured positions $\boldsymbol{X}$ but centered at all potential measurement locations, and the matrix $\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}}) \triangleq [\boldsymbol{\phi}_1(\widetilde{\boldsymbol{X}}),\dots,\boldsymbol{\phi}_N(\widetilde{\boldsymbol{X}})] \in \mathbb{R}^{T\times N}$ consists of all basis functions, evaluated at all potential measurement positions and centered at all center locations $\boldsymbol{M}$. In the next two paragraphs, these definitions will be used to define faster matrix calculations for multiple potential measurement locations $\widetilde{\boldsymbol{X}}$ for both distribution paradigms.

B.2.1. FAST EVALUATION OF THE D-OPTIMALITY FOR SOE

Now, (B.15) will be reformulated for a faster computation for the SOE paradigm. Therefore the following definitions, which can also be calculated with a consensus algorithm, are introduced:

$$\widetilde{\boldsymbol{D}} \triangleq \widetilde{\boldsymbol{\Phi}}^T(\boldsymbol{X})\,\boldsymbol{\Phi}_{\mathscr{A}} = \sum_{k=1}^{K} \boldsymbol{\Phi}^T(\boldsymbol{X}_k,\widetilde{\boldsymbol{X}})\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X}_k,\boldsymbol{X}), \tag{B.16}$$

$$\widetilde{\boldsymbol{b}} \triangleq \operatorname{diag}\{\widetilde{\boldsymbol{\Phi}}^T(\boldsymbol{X})\,\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X})\} = \sum_{k=1}^{K} \operatorname{diag}\{\boldsymbol{\Phi}^T(\boldsymbol{X}_k,\widetilde{\boldsymbol{X}})\,\boldsymbol{\Phi}(\boldsymbol{X}_k,\widetilde{\boldsymbol{X}})\}, \tag{B.17}$$

where $\boldsymbol{\Phi}_{\mathscr{A}} \in \mathbb{R}^{M\times|\mathscr{A}|}$ as introduced in Sec. 4.2.2 and $\boldsymbol{\Phi}(\boldsymbol{X}_k,\widetilde{\boldsymbol{X}}) = \left[\boldsymbol{\phi}_1^T(\widetilde{\boldsymbol{X}}),\dots,\boldsymbol{\phi}_{M_k}^T(\widetilde{\boldsymbol{X}})\right]^T \in \mathbb{R}^{M_k\times T}$ is the part of the k-th agent from $\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X})$, i.e.

$$\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{\Phi}(\boldsymbol{X}_{M_1},\widetilde{\boldsymbol{X}}) \\ \vdots \\ \boldsymbol{\Phi}(\boldsymbol{X}_{M_K},\widetilde{\boldsymbol{X}}) \end{bmatrix}. \tag{B.18}$$

With the term $\boldsymbol{A}$, defined in (4.19), these defintions can then be used such that

$$\boldsymbol{q} = \widetilde{\boldsymbol{b}} - \operatorname{diag}\{\widetilde{\boldsymbol{D}}^T\boldsymbol{A}^{-1}\widetilde{\boldsymbol{D}}\}. \tag{B.19}$$

Then (B.15) can be formulated into a matrix vector expression (neglecting the first logarithmic term) as

$$\begin{aligned}\log\det\widetilde{\boldsymbol{C}} = \log(&\boldsymbol{q} + \boldsymbol{q}\circ\operatorname{diag}\{\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}})\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T(\widetilde{\boldsymbol{X}})\} \\ &+ (\operatorname{diag}\{\widetilde{\boldsymbol{\Phi}}(\widetilde{\boldsymbol{X}})\} - \operatorname{Sq}\{\operatorname{diag}\{\widetilde{\boldsymbol{D}}^T\boldsymbol{A}^{-1}\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}})\}\})),\end{aligned} \tag{B.20}$$

where $\operatorname{Sq}\{\cdot\}$ is an element-wise square operation on a vector and $\circ$ is the Hadamard product [166].

The computation of (B.20) involves the $\operatorname{diag}\{\cdot\}$ function, which takes the diagonal of a matrix. Consequently, the off-diagonal entries of this matrix are not used and the computation can be accelerated further by exploiting Einstein-summation [167] in the implementation. When using Einstein-summations for calculating the diagonal of a matrix product, only the corresponding columns and rows are multiplied. Thus, the computational complexity is reduced from quadratic to linear.

B.2.2. FAST EVALUATION OF THE D-OPTIMALITY FOR SOF

If the SOF distribution paradigm is applied, the fast computation changes accordingly. For convenience, the exploration criterion from Chap. 4 for the SOF distributiion paradigm is shown here again

$$\begin{aligned}\log\det\widetilde{\boldsymbol{C}} = \log|\boldsymbol{A}| + \log\Big(&q \\ &+ q(v(\widetilde{\boldsymbol{x}}) - \boldsymbol{d}(\widetilde{\boldsymbol{x}})^T(\boldsymbol{I}+\boldsymbol{H})^{-1}\boldsymbol{d}(\widetilde{\boldsymbol{x}})) \\ &+ \left(\phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) - \left(\boldsymbol{I}-\boldsymbol{H}(\boldsymbol{I}+\boldsymbol{H})^{-1}\right)\boldsymbol{d}(\widetilde{\boldsymbol{x}})\right)^2\Big),\end{aligned} \tag{B.21}$$

where $\boldsymbol{H}$ is defined in (4.27), $\boldsymbol{d}(\widetilde{\boldsymbol{x}})$ in (4.28), and $v(\widetilde{\boldsymbol{x}})$ in (4.29). First, the term $\boldsymbol{\Phi}\left(\widetilde{\boldsymbol{X}},\boldsymbol{M}_k\right) \triangleq \left[\boldsymbol{\phi}_1(\widetilde{\boldsymbol{X}}),\ldots,\boldsymbol{\phi}_{N_k}(\widetilde{\boldsymbol{X}})\right] \in \mathbb{R}^{T\times N_k}$ is defined, which represents the contribution of agent k to $\boldsymbol{\Phi}\left(\widetilde{\boldsymbol{X}}\right)$. Next, following an equal approach as before, the definitions, which can also be calculated by a consensus, are introduced as

$$\widetilde{\boldsymbol{E}} \triangleq \boldsymbol{\Phi}_{\mathscr{A}}\widehat{\boldsymbol{\Delta}}^{-1}\boldsymbol{\Phi}^T(\widetilde{\boldsymbol{X}}) = \sum_{k=1}^{K}\boldsymbol{\Phi}_{\mathscr{A}}(\boldsymbol{X},\boldsymbol{M}_k)\,\widehat{\boldsymbol{\Delta}}_k^{-1}\boldsymbol{\Phi}^T\left(\widetilde{\boldsymbol{X}},\boldsymbol{M}_k\right), \tag{B.22}$$

$$\widetilde{\boldsymbol{g}} \triangleq \operatorname{diag}\{\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}})\widehat{\boldsymbol{\Delta}}^{-1}\boldsymbol{\Phi}^T(\widetilde{\boldsymbol{X}})\} = \sum_{k=1}^{K}\boldsymbol{\Phi}\left(\widetilde{\boldsymbol{X}},\boldsymbol{M}_k\right)\widehat{\boldsymbol{\Delta}}_k^{-1}\boldsymbol{\Phi}^T\left(\widetilde{\boldsymbol{X}},\boldsymbol{M}_k\right). \tag{B.23}$$

First, by using the defintion of $\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X})$, the Schur-Complement from before, can be defined as

$$\boldsymbol{q} = \operatorname{diag}\{\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X})(\boldsymbol{I}-\boldsymbol{H})^{-1}\widetilde{\boldsymbol{\Phi}}^T(\boldsymbol{X})\}, \tag{B.24}$$

where $\widetilde{\boldsymbol{\Phi}}(\boldsymbol{X})$ is known to all agents by virtue of the SOF distribution paradigm. Next, the last two terms in (B.21) are reformulated for a faster computation such that

$$q(v(\widetilde{\boldsymbol{x}}) - \boldsymbol{d}(\widetilde{\boldsymbol{x}})^T(\boldsymbol{I}+\boldsymbol{H})^{-1}\boldsymbol{d}(\widetilde{\boldsymbol{x}}))\big|_{\text{for all } T} = \boldsymbol{q}\circ\left(\widetilde{\boldsymbol{g}} - \operatorname{diag}\{\widetilde{\boldsymbol{E}}^T(\boldsymbol{I}-\boldsymbol{H})^{-1}\widetilde{\boldsymbol{E}}\}\right), \tag{B.25}$$

and

$$\left(\phi(\widetilde{\boldsymbol{x}},\breve{\boldsymbol{\mu}}) - \left(\boldsymbol{I}-\boldsymbol{H}(\boldsymbol{I}+\boldsymbol{H})^{-1}\right)\boldsymbol{d}(\widetilde{\boldsymbol{x}})\right)^2\Big|_{\text{for all } T} = \operatorname{Sq}\left\{\operatorname{diag}\left\{\widetilde{\boldsymbol{\Phi}}(\widetilde{\boldsymbol{X}})\right\} - \operatorname{diag}\left\{\left(\boldsymbol{I}-\boldsymbol{H}(\boldsymbol{I}+\boldsymbol{H})^{-1}\right)\widetilde{\boldsymbol{E}}\right\}\right\}. \tag{B.26}$$

If the criterion (B.21) is used in a vectorized formulation, the exploration when using the SOF paradigm is then

$$\begin{aligned}\widehat{\boldsymbol{x}} = \arg\min_{\widetilde{\boldsymbol{x}}}\ \log\Big(&\boldsymbol{q} + \boldsymbol{q}\circ\left(\widetilde{\boldsymbol{g}} - \operatorname{diag}\{\widetilde{\boldsymbol{E}}^T(\boldsymbol{I}-\boldsymbol{H})^{-1}\widetilde{\boldsymbol{E}}\}\right) \\ &+ \operatorname{Sq}\left\{\operatorname{diag}\left\{\widetilde{\boldsymbol{\Phi}}(\widetilde{\boldsymbol{X}})\right\} - \operatorname{diag}\left\{\left(\boldsymbol{I}-\boldsymbol{H}(\boldsymbol{I}+\boldsymbol{H})^{-1}\right)\widetilde{\boldsymbol{E}}\right\}\right\}\Big).\end{aligned} \tag{B.27}$$

B.3. THE CHOICE OF $\widetilde{\gamma}$ FOR EXPLORATION

In Sec. 4.5, it is mentioned that for the exploration the choice of $\widetilde{\gamma}^{-1}$ for newly added columns influences the exploration. In the following, (4.55) is analyzed regarding the

value of $\widetilde{\gamma}^{-1}$. For convenience (4.55) is shown here again as

$$\widehat{\boldsymbol{x}} = \underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}}}{\arg\min} \log|\widetilde{\boldsymbol{\Sigma}}_w| \equiv \underset{\substack{\widetilde{\boldsymbol{x}}\in\mathscr{X},\\ \check{\boldsymbol{\mu}}\in\mathscr{M}\setminus\boldsymbol{M}}}{\arg\max} \log\Big[q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{M})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})) + \widetilde{\lambda}\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{M})\right)^2\Big]. \tag{B.28}$$

The parameter $\widetilde{\gamma} > 0$ only appears in $b(\check{\boldsymbol{\mu}})$, which is only used in $q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})$. As the logarithm is not defined for negative values, the following will evaluate $\widetilde{\gamma}$, if it can be chosen such that the argument in the logarithm becomes zero. For having a positive term inside of the logarithm, it therefore needed that

$$q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})) + \widetilde{\lambda}\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2 \overset{!}{\geq} 0. \tag{B.29}$$

As $\widetilde{\gamma}^{-1}$ is the only term that could be chosen, it is expressed in the following. For this, the definition $q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) = b_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}^T(\check{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})$ and $b_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) \triangleq \boldsymbol{\phi}^T(\boldsymbol{X},\check{\boldsymbol{\mu}})\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}) + \widetilde{\gamma}^{-1}$ are used such that

$$\begin{aligned}
q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})\underbrace{\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)}_{>0} &\overset{!}{\geq} -\widetilde{\lambda}\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2 \\
b(\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}^T(\check{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) = q_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) &\overset{!}{\geq} -\widetilde{\lambda}\frac{\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2}{\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)} \\
b(\check{\boldsymbol{\mu}}) &\overset{!}{\geq} -\widetilde{\lambda}\frac{\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2}{\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)} + \boldsymbol{c}_{\mathrm{SBL}}^T(\check{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) \\
\underbrace{\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})}_{>0} + \widetilde{\gamma}^{-1} &\overset{!}{\geq} -\widetilde{\lambda}\frac{\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2}{\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)} + \boldsymbol{c}_{\mathrm{SBL}}^T(\check{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) \\
\widetilde{\gamma}^{-1} &\overset{!}{\geq} \boldsymbol{c}_{\mathrm{SBL}}^T(\check{\boldsymbol{\mu}})\boldsymbol{\Sigma}_w\boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}}) - \widetilde{\lambda}\frac{\left(\phi(\widetilde{\boldsymbol{x}},\check{\boldsymbol{\mu}}) - \boldsymbol{c}_{\mathrm{SBL}}(\check{\boldsymbol{\mu}})^T\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)^2}{\left(1+\widetilde{\lambda}\boldsymbol{\phi}^T(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\widetilde{\boldsymbol{x}},\boldsymbol{X_N})\right)} - \boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}})^T\boldsymbol{\Lambda}\boldsymbol{\phi}(\boldsymbol{X},\check{\boldsymbol{\mu}}).
\end{aligned} \tag{B.30}$$

Thus $\widetilde{\gamma}^{-1}$ has to be larger than the right-hand side of (B.30) to ensure a positive term within the logarithm of the exploration criterion. Also, by definition, all hyper-parameter and, therefore also $\widetilde{\gamma}^{-1}$ have to be larger than zero.

Here another remark on $\widetilde{\gamma}$: This parameter can also be chosen such that the agents do more exploration or exploitation. The agents would explore more, if $\widetilde{\gamma}$ becomes as small as possible. On the other side, if $\widetilde{\gamma}$ becomes very large, the agents would exploit more already measured locations; the algorithm aims then at maximizing the confidence of the already estimated values. This scenario is, however, not investigated in this thesis.

C

CONSENSUS ALGORITHMS

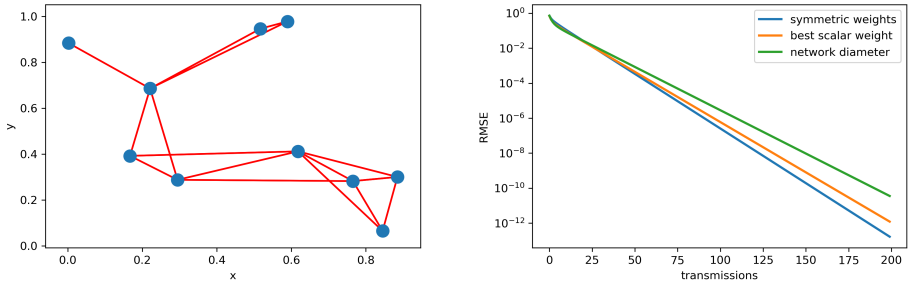
Consensus, which is also known as the distributed averaging problem, arises in the context of coordination of networks of autonomous agents but also in development of distributed processing algorithms, as presented in this thesis. There exist multiple realizations of the consensus algorithm:

1. Distributed averaging in a synchronous [168] and an asynchronous fashion [169].
2. Gossip algorithms [170] and in particular randomized gossiping [171] and geographic gossiping [172], where a peer to peer connection between two nodes is used.
3. Gossiping with eavesdropping [173], where intermediate nodes listen to the peer to peer communication.
4. Broadcasting realizations, e.g. the broadcast weighted gossip [174].

This thesis uses the synchronous distributed averaging. Distributed averaging requires weighting parameters for the network. The weighting parameters can be derived from [168]

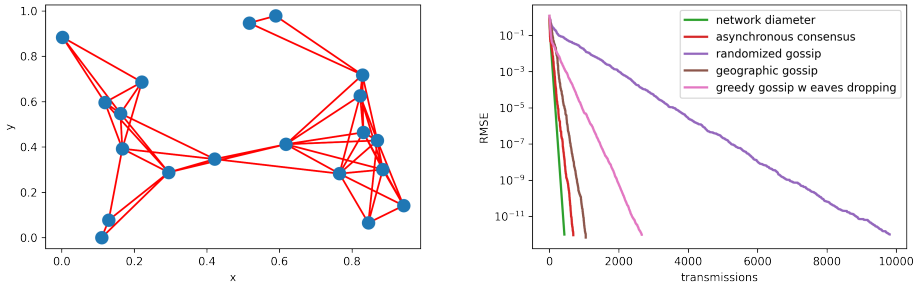
- the minimization of the spectral norm of the network, which results in a weighting matrix with different weighting factors for each node,
- the eigenvalues of the network's Laplacian matrix,
- or the network diameter, which is the longest path between two nodes within the graph.

As an example, Fig. C.1(a) presents a network with the diameter of 4. The convergence of the three aforementioned methods for the example network is presented in Fig. C.1(b). Although this is only an example, the result coincides with the results presented in [168]. Therefore, all these three algorithms converge with a geometric rate and, with an increasing number of iterations, the symmetric weights method performs best. Yet the



(a) (b)

Figure C.1: (a) An example network with diameter 4. (b) Error versus the number of transmissions of three synchronous distributed averaging methods for the example network in (a).



(a) (b)

Figure C.2: (a) An example network (b) The performance of asynchronous distributed averaging methods for the network presented in (a).

symmetric weights method involves an estimation of the weighting parameter for each network.

For comparison reasons, the plot in Fig. C.2(b) displays the performance of the asynchronous methods for solving the distributed averaging problem [169, 171–173] against the network diameter method for the example network in Fig. C.2(a). This example shows that asynchronous communication comes with a loss of performance, i.e. more transmissions are required to achieve an equal error compared with synchronous methods.

D

Collision Avoidance and Path Planning

D.1. Collision Avoidance

The collision avoidance was realized using a FOVA algorithm [146]. Originally, this algorithm was tailored for UAVs, which are holonomic and have no obstacles except for other UAVs. The benefit of this algorithm is that it does not necessarily require any communication between the UAVs, yet the UAVs have to see each other for distance measuring. The FOVA can then be explained as follows.

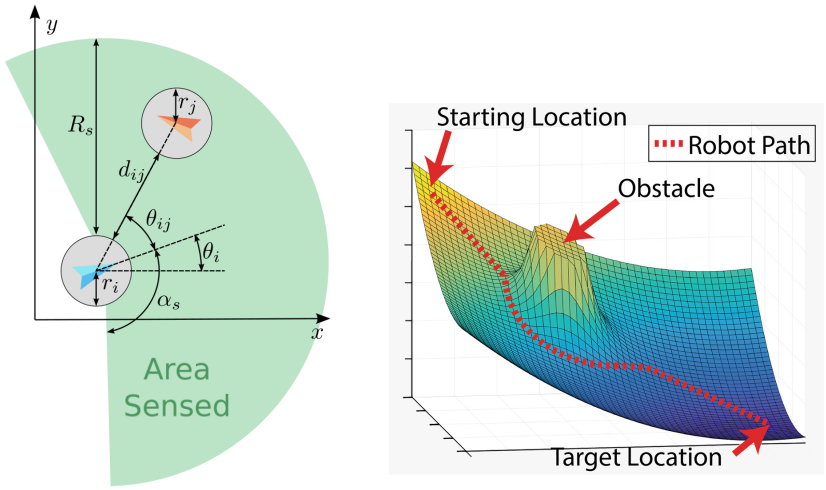
1. Once an UAV j has been detected within the field-of-view of UAV i, the algorithm models the UAV j by a potential field, where the center of j has the highest potential. Figure D.1(a) graphically explains the modeling of UAV i's field of view.

   It is determined by the range $R_s \in \mathbb{R}$ and an angle $0 \leq \alpha_s \leq 180$. Once another UAV j enters the field of i, the distance $d_{ij} \in \mathbb{R}$ and the relative angle θ_{ij} are estimated. As a safety measure each UAV has an area, which models the physical dimensions of each UAV, denoted as r_i or r_j.

2. The sensing UAV i also has a potential field and both fields are combined over a system of equations to create a spring-like mechanism, which results in a force pushing away from UAV j. The following presents the equations, but for a detailed description and derivation the reader is referred to [146]. The first equation controls the velocity $V_{\min} < v_i < V_{\max} \in \mathbb{R}^+$ as

   $$v_i = V_{\max} - \Delta V \max_j(\beta_{a_j}, \beta_{d_j}), \tag{D.1}$$

   where $\Delta V = V_{\max} - V_{\min}$ and $0 \leq \beta_{a_j} \leq 1$ and $0 \leq \beta_{d_j} \leq 1$ are so-called barrier functions with respect to UAV j that depend on the relative angle θ_{ij} and the distance



(a) (b)

Figure D.1: (a) This figure is taken from [146] and it graphically introduces the field-of-view and the distance variables of the FOVA algorithm. (b) This figure exemplifies a resulting potential field and a path of a UAV when confronted with an obstacle.

d_{ij}, respectively. The second equation controls the turning rate u_i

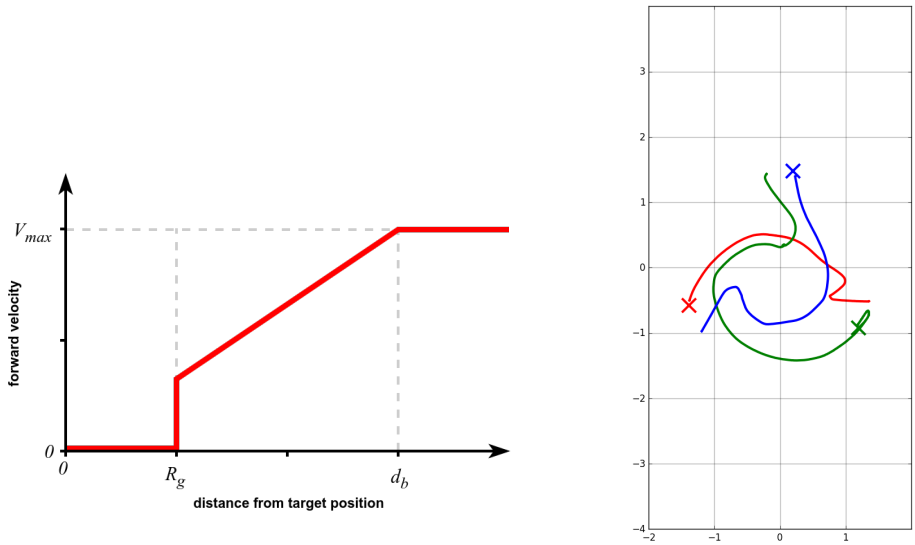
$$u_i = K(1 - \max_j(\beta_{a_j}, \beta_{d_j}))(\theta_t - \theta_i) + \sum_j \beta_{a_j}\beta_{d_j}\frac{-\pi V_{\max}}{d_{ij}}, \tag{D.2}$$

with some constant $K > 0$ and the angle to the target θ_t.

3. The velocity v_i and u_i are then sent to the motion controller of UAV i to evade UAV j while still going towards the target location. Figure D.1(b) exemplifies the resulting potential field and the UAV path denoted as "Robot Path".

The original algorithm was slightly modified to account for holonomic ground vehicles and for the laboraty conditions in the Holodeck. Additionally, the ground vehicles did not have an active sensing system for distance measuring, e.g. a Lidar or stereoscopic camera. Thus, the limited view of the robots was simulated with positioning data obtained from the VICON[1] camera system in the laboratory; the VICON system also provided the location of the agents. To account for the walls in the laboratory, the walls were added as static "agents" to the algorithm. When using the orignal algorithm, it was observed that the ground robots tend to osczillate at their target location. This is due to the built-in controller, which cannot reach the goal perfectly. Thus, the velocity of the ground vehicle was modified according to the distance to the goal as presented in Fig. D.2(a). There, at a distance $d_b \in \mathbb{R}$ the velocity was reduced until the distance $R_g \in \mathbb{R}$ where the velocity was set to zero. This approach removed the osczillating behaviour for the ground vehicles.

[1] https://www.vicon.com/



(a) (b)

Figure D.2: (a) Velocity reduction, when reaching the target location, to avoid osczillations at the destination. (b) Recorded test of the modified FOVA for three ground vehicles. Each vehicle drove towards to cross while avoiding the other ground vehicles.

Figure D.2(b) presents a recorded test with three ground vehicles, which are differentiated by color. Each ground vehicle moved towards the target location denoted by the cross. The line represents the trajectory of each ground vehicle. Apparently the ground vehicles did not collide and avoided each other by circling around each other.

D.2. A* - PATH PLANNER

The A* algorithm is according to [10] the standard search algorithm for the shortest path problem in a graph [175]. Essentially, this algorithm minimizes the fitness function

$$f(n) = g(n) + h(n), \tag{D.3}$$

which is comprised of the terms $g(n)$ – the spent cost of moving from start to position n – and $h(n)$ – a heuristic estimation of the cost of moving from n to the goal position. The A* algorithm minimizes $f(n)$ by managing two lists of open positions and closed positions. The open positions are considered for the minimization and the closed positions contain at the end of the algorithm the path to the goal. The whole algorithm is summarized in Alg. 10. An example scenario is presented in Fig. D.3, where the A* algorithm finds a path in a grid-environment.

Algorithm 10 A* algorithm

```
open_list ← Start position
closed_list ← empty list
while destination not reached do
    n̂ ← arg min_n f(n) ∀ n ∈ open_list
    if n̂ == destination then
        append closed_list by n̂ return closed_list
    else
        Remove n̂ from open_list
        Append closed_list by n̂
        for neighboring positions m of n̂ do
            if not m ∈ open_list then
                Append open_list by m
```

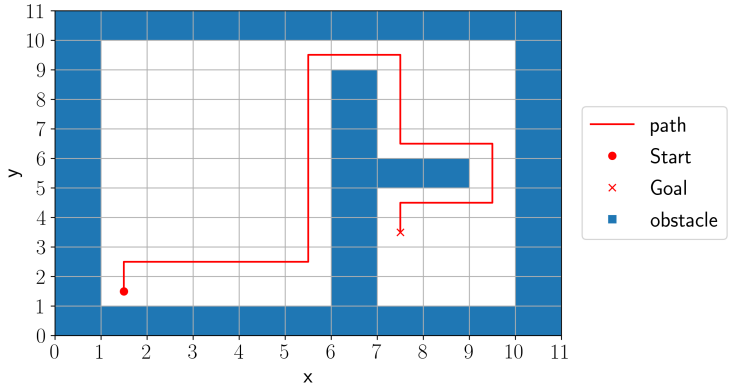



Figure D.3: Example scenario, where a solution of the A* algorithm is presented.

E

SENSOR CALIBRATION

The authors in [153] assume that the sensor readings of one sensor can be expressed as another sensor's reading by an affine transformation and rotation. A single measurement taken at a position $\boldsymbol{x} \in \mathbb{R}^3$ has three components; one component of the magnetic field per Euclidean axis. The measurements of the sensors $\{1,2\}$ are arranged in the vectors $\boldsymbol{z}_1 \in \mathbb{R}^3$ and $\boldsymbol{z}_2 \in \mathbb{R}^3$, respectively. Then a translation $\boldsymbol{b} \in \mathbb{R}^3$ and rotation $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is computed to align $\boldsymbol{z}_2$ to $\boldsymbol{z}_1$ such that

$$\boldsymbol{z}_1(\boldsymbol{x}_m) = \boldsymbol{R}\boldsymbol{z}_2(\boldsymbol{x}_m) + \boldsymbol{b}, \tag{E.1}$$

where $\boldsymbol{x}_m$ means the m-th measurement location. The estimation of $\boldsymbol{R}$ and $\boldsymbol{b}$ can be formulated as an optimization problem. Therefore, the rotation matrix $R = [\boldsymbol{r}_1^T, \ldots, \boldsymbol{r}_3^T]^T$, with the rows $\boldsymbol{r}_i^T \in \mathbb{R}^3,\ i = 1,\ldots,3$, is used to formulate an optimization parameter $\boldsymbol{\theta} = [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_3, \boldsymbol{b}^T]^T \in \mathbb{R}^{12}$ such that

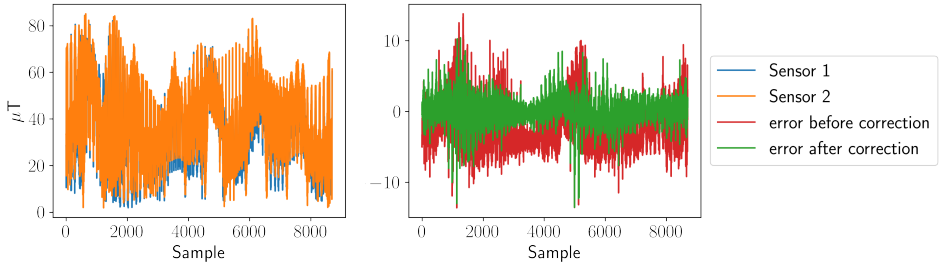
$$\boldsymbol{z}_1(\boldsymbol{x}_m) = \boldsymbol{H}'(\boldsymbol{x}_m)\boldsymbol{\theta}, \tag{E.2}$$

with

$$\boldsymbol{H}'(\boldsymbol{x}_m) = \begin{bmatrix} \boldsymbol{z}_2^T(\boldsymbol{x}_m) & & \mathbf{0} & 1 & 0 & 0 \\ & \boldsymbol{z}_2^T(\boldsymbol{x}_m) & & 0 & 1 & 0 \\ \mathbf{0} & & \boldsymbol{z}_2^T(\boldsymbol{x}_m) & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times12}. \tag{E.3}$$

To achieve a better calibration multiple measurement locations have to be considered. Thus, if $D \in \mathbb{N}$ measurement locations defined as $\boldsymbol{X}_D = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_D]^T \in \mathbb{R}^{D\times3}$ are considered, then $\boldsymbol{H}(\boldsymbol{X}_D) = \left[\boldsymbol{H}'^T(\boldsymbol{x}_1), \ldots, \boldsymbol{H}'^T(\boldsymbol{x}_D)\right]^T \in \mathbb{R}^{3D\times12}$, and $\boldsymbol{Z}_1(\boldsymbol{X}_D) = \left[\boldsymbol{z}_1^T(\boldsymbol{x}_1), \ldots, \boldsymbol{z}_1^T(\boldsymbol{x}_D)\right]^T \in \mathbb{R}^{3D}$. For multiple measurements (E.2) can then be formulated as a maximum likelihood problem as

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{H}(\boldsymbol{X}_D)\boldsymbol{\theta} - \boldsymbol{Z}_1(\boldsymbol{X}_D)\|_2^2, \tag{E.4}$$



(a) (b)

Figure E.1: (a) absolute values of the magnetic field samples of two sensors. Each sensor measured on the same locations. (b) absolute values of the magnetic field samples after the corrections. The sensors are now aligned.

which is then solved by least squares method, see Sec. 2.1.1.

The sensor readings before the calibration are presented in Fig. E.1(a). Figure E.1(b) displays the differences of both sensor readings before and after the calibration. It can be seen that the used method reduces the bias and it also reduces the variance. However, the error does not go down to zero. This is because each sensor involves non-linearities, which cannot be considered by the affine transform of this calibration. Also electrical components in the robots might influence the sensor readings, but this has not been looked at.

REFERENCES

[1] M. P. Golombek, R. C. Anderson, J. R. Barnes, J. F. Bell, N. T. Bridges, D. T. Britt, J. Brückner, R. A. Cook, D. Crisp, J. A. Crisp, T. Economou, W. M. Folkner, R. Greeley, R. M. Haberle, R. B. Hargraves, J. A. Harris, A. F. C. Haldemann, K. E. Herkenhoff, S. F. Hviid, R. Jaumann, J. R. Johnson, P. H. Kallemeyn, H. U. Keller, R. L. Kirk, J. M. Knudsen, S. Larsen, M. T. Lemmon, M. B. Madsen, J. A. Magalhães, J. N. Maki, M. C. Malin, R. M. Manning, J. Matijevic, H. Y. McSween, H. J. Moore, S. L. Murchie, J. R. Murphy, T. J. Parker, R. Rieder, T. P. Rivellini, J. T. Schofield, A. Seiff, R. B. Singer, P. H. Smith, L. A. Soderblom, D. A. Spencer, C. R. Stoker, R. Sullivan, N. Thomas, S. W. Thurman, M. G. Tomasko, R. M. Vaughan, H. Wänke, A. W. Ward, and G. R. Wilson, *Overview of the Mars Pathfinder Mission: Launch through landing, surface operations, data sets, and science results,* Journal of Geophysical Research: Planets **104**, 8523 (1999).

[2] R. E. Arvidson, S. W. Ruff, R. V. Morris, D. W. Ming, L. S. Crumpler, A. S. Yen, S. W. Squyres, R. J. Sullivan, J. F. Bell, N. A. Cabrol, B. C. Clark, W. H. Farrand, R. Gellert, R. Greenberger, J. A. Grant, E. A. Guinness, K. E. Herkenhoff, J. A. Hurowitz, J. R. Johnson, G. Klingelhöfer, K. W. Lewis, R. Li, T. J. McCoy, J. Moersch, H. Y. McSween, S. L. Murchie, M. Schmidt, C. Schröder, A. Wang, S. Wiseman, M. B. Madsen, W. Goetz, and S. M. McLennan, *Spirit Mars Rover Mission to the Columbia Hills, Gusev Crater: Mission overview and selected results from the Cumberland Ridge to Home Plate,* Journal of Geophysical Research: Planets **113** (2008), 10.1029/2008JE003183.

[3] S. W. Squyres, R. E. Arvidson, D. Bollen, J. F. Bell, J. Brückner, N. A. Cabrol, W. M. Calvin, M. H. Carr, P. R. Christensen, B. C. Clark, L. Crumpler, D. J. D. Marais, C. d'Uston, T. Economou, J. Farmer, W. H. Farrand, W. Folkner, R. Gellert, T. D. Glotch, M. Golombek, S. Gorevan, J. A. Grant, R. Greeley, J. Grotzinger, K. E. Herkenhoff, S. Hviid, J. R. Johnson, G. Klingelhöfer, A. H. Knoll, G. Landis, M. Lemmon, R. Li, M. B. Madsen, M. C. Malin, S. M. McLennan, H. Y. McSween, D. W. Ming, J. Moersch, R. V. Morris, T. Parker, J. W. Rice, L. Richter, R. Rieder, C. Schröder, M. Sims, M. Smith, P. Smith, L. A. Soderblom, R. Sullivan, N. J. Tosca, H. Wänke, T. Wdowiak, M. Wolff, and A. Yen, *Overview of the Opportunity Mars Exploration Rover Mission to Meridiani Planum: Eagle Crater to Purgatory Ripple,* Journal of Geophysical Research: Planets **111** (2006), 10.1029/2006JE002771.

[4] A. R. Vasavada, J. P. Grotzinger, R. E. Arvidson, F. J. Calef, J. A. Crisp, S. Gupta, J. Hurowitz, N. Mangold, S. Maurice, M. E. Schmidt, R. C. Wiens, R. M. E. Williams,

and R. A. Yingst, *Overview of the Mars Science Laboratory mission: Bradbury Landing to Yellowknife Bay and beyond,* Journal of Geophysical Research: Planets **119**, 1134 (2014).

[5] L. Crane, *China launches mission to Mars,* New Scientist **247**, 15 (2020).

[6] J. L. Vago, F. Westall, L. S. S. W. G. Pasteur Instrument Teams, and Other Contributors, A. J. Coates, R. Jaumann, O. Korablev, V. Ciarletti, I. Mitrofanov, J.-L. Josset, M. C. De Sanctis, J.-P. Bibring, F. Rull, F. Goesmann, H. Steininger, W. Goetz, W. Brinckerhoff, C. Szopa, F. Raulin, F. Westall, H. G. M. Edwards, L. G. Whyte, A. G. Fairén, J.-P. Bibring, J. Bridges, E. Hauber, G. G. Ori, S. Werner, D. Loizeau, R. O. Kuzmin, R. M. E. Williams, J. Flahaut, F. Forget, J. L. Vago, D. Rodionov, O. Korablev, H. Svedhem, E. Sefton-Nash, G. Kminek, L. Lorenzoni, L. Joudrier, V. Mikhailov, A. Zashchirinskiy, S. Alexashkin, F. Calantropio, A. Merlo, P. Poulakis, O. Witasse, O. Bayle, S. Bayón, U. Meierhenrich, J. Carter, J. M. García-Ruiz, P. Baglioni, A. Haldemann, A. J. Ball, A. Debus, R. Lindner, F. Haessig, D. Monteiro, R. Trautner, C. Voland, P. Rebeyre, D. Goulty, F. Didot, S. Durrant, E. Zekri, D. Koschny, A. Toni, G. Visentin, M. Zwick, M. van Winnendael, M. Azkarate, C. Carreau, and the ExoMars Project Team, *Habitability on Early Mars and the Search for Biosignatures with the ExoMars Rover,* Astrobiology **17**, 471 (2017).

[7] S. Squyres, *Roving Mars: Spirit, Opportunity, and the Exploration of the Red Planet,* first edition ed. (Hyperion, New York, 2015).

[8] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, *NASA's swarm missions: The challenge of building autonomous software,* IT Professional **6**, 47 (2004).

[9] C. Goerzen, Z. Kong, and B. Mettler, *A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance,* Journal of Intelligent and Robotic Systems **57**, 65 (2010).

[10] N. Sariff and N. Buniyamin, *An Overview of Autonomous Mobile Robot Path Planning Algorithms,* in *2006 4th Student Conference on Research and Development* (IEEE, 2006) pp. 183–188.

[11] M. Sutoh, M. Otsuki, S. Wakabayashi, T. Hoshino, and T. Hashimoto, *The Right Path: Comprehensive Path Planning for Lunar Exploration Rovers,* IEEE Robotics & Automation Magazine **22**, 22 (2015).

[12] Y. Dimopoulos and P. Moraitis, *Multi-Agent Coordination and Cooperation through Classical Planning,* in *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (2006) pp. 398–402.

[13] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, *Artificial Intelligence for Long-Term Robot Autonomy: A Survey,* IEEE Robotics and Automation Letters **3**, 4023 (2018).

[14] E. Staudinger, S. Zhang, R. Pöhlmann, and A. Dammann, *The Role of Time in a Robotic Swarm: A Joint View on Communications, Localization, and Sensing,* IEEE Communications Magazine (2021).

[15] A. Sayed, *Adaptation, Learning, and Optimization over Networks,* Foundations and Trends® in Machine Learning **7**, 311 (2014).

[16] J. Montiel, A. Bifet, V. Losing, J. Read, and T. Abdessalem, *Learning Fast and Slow: A Unified Batch/Stream Framework,* in *2018 IEEE International Conference on Big Data (Big Data)* (2018) pp. 1065–1072.

[17] M. Rabbat and R. Nowak, *Distributed optimization in sensor networks,* in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks* (ACM Press, 2004) p. 20.

[18] S. Boyd, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,* Foundations and Trends® in Machine Learning **3**, 1 (2010).

[19] A. Nedić, A. Olshevsky, and M. G. Rabbat, *Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization,* arXiv:1709.08765 [cs, math] (2017), arxiv:1709.08765 [cs, math] .

[20] R. Olfati-Saber, J. A. Fax, and R. M. Murray, *Consensus and Cooperation in Networked Multi-Agent Systems,* Proceedings of the IEEE **95**, 215 (2007).

[21] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Mathematics and Its Applications (Springer Netherlands, 2000).

[22] A. E. Hoerl and R. W. Kennard, *Ridge Regression: Biased Estimation for Nonorthogonal Problems,* Technometrics **12**, 55 (1970), 1267351 .

[23] R. Tibshirani, *Regression Shrinkage and Selection via the Lasso,* Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996), 2346178 .

[24] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net,* Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**, 301 (2005).

[25] R. Giri and B. Rao, *Type I and Type II Bayesian Methods for Sparse Signal Recovery Using Scale Mixtures,* IEEE Transactions on Signal Processing **64**, 3418 (2016).

[26] T. Buchgraber and D. Shutin, *Distributed variational sparse Bayesian learning for sensor networks,* in *2012 IEEE International Workshop on Machine Learning for Signal Processing* (2012) pp. 1–6.

[27] S. Khanna and C. R. Murthy, *Decentralized Joint-Sparse Signal Recovery: A Sparse Bayesian Learning Approach,* IEEE Transactions on Signal and Information Processing over Networks **3**, 29 (2017).

[28] P. Whaite and F. P. Ferrie, *Autonomous Exploration: Driven by Uncertainty,* IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **19**, 13 (1997).

[29] A. Krause, A. Singh, and C. Guestrin, *Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies,* Journal of Machine Learning Research **9**, 235 (2008).

[30] S. Thrun and K. Möller, *Active Exploration in Dynamic Environments,* , 8 (1992).

[31] F. Pukelsheim, *Optimal Design of Experiments* (SIAM, 1993).

[32] A. Viseras, T. Wiedemann, C. Manss, V. Karolj, D. Shutin, and J. Marchal, *Beehive-Inspired Information Gathering with a Swarm of Autonomous Drones,* Sensors **19**, 4349 (2019).

[33] T. Wiedemann, C. Manss, and D. Shutin, *Multi-agent exploration of spatial dynamical processes under sparsity constraints,* Autonomous Agents and Multi-Agent Systems **32**, 134 (2018).

[34] X. Zhou, W. Wang, T. Wang, M. Li, and F. Zhong, *Online Planning for Multiagent Situational Information Gathering in the Markov Environment,* IEEE Systems Journal , 1 (2019).

[35] D. Carmel and S. Markovitch, *Exploration Strategies for Model-based Learning in Multi-agent Systems: Exploration Strategies,* Autonomous Agents and Multi-Agent Systems **2**, 141 (1999).

[36] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, *Coordinated multi-robot exploration,* IEEE Transactions on Robotics **21**, 376 (2005).

[37] O. Esrafilian and D. Gesbert, *3D City Map Reconstruction from UAV-Based Radio Measurements,* in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference* (IEEE, 2017) pp. 1–6.

[38] P. Lanillos, S. K. Gan, E. Besada-Portas, G. Pajares, and S. Sukkarieh, *Multi-UAV target search using decentralized gradient-based negotiation with expected observation,* Information Sciences **282**, 92 (2014).

[39] A. A. Redwan Newaz, S. Jeong, H. Lee, H. Ryu, and N. Y. Chong, *UAV-based multiple source localization and contour mapping of radiation fields,* Robotics and Autonomous Systems **85**, 12 (2016).

[40] M. Andries and F. Charpillet, *Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendezvous using ant algorithms,* (IEEE, 2013) pp. 5571–5578.

[41] E. Galceran and M. Carreras, *A survey on coverage path planning for robotics,* Robotics and Autonomous Systems **61**, 1258 (2013).

[42] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, *Efficient Boustrophedon Multi-Robot Coverage: An algorithmic approach,* Annals of Mathematics and Artificial Intelligence **52**, 109 (2008).

[43] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos, *Multi-robot three-dimensional coverage of unknown areas,* The International Journal of Robotics Research **31**, 738 (2012).

[44] S. Dhillon and K. Chakrabarty, *Sensor placement for effective coverage and surveillance in distributed sensor networks,* in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, Vol. 3 (2003) pp. 1609–1614 vol.3.

[45] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, *Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks,* Proceedings of the IEEE **9**, 1541 (2011).

[46] Y. Mostofi, *Compressive Cooperative Sensing and Mapping in Mobile Networks,* IEEE Transactions on Mobile Computing **10**, 1769 (2011).

[47] X. Huang, F. Arvin, C. West, S. Watson, and B. Lennox, *Exploration in Extreme Environments with Swarm Robotic System,* in *2019 IEEE International Conference on Mechatronics (ICM)* (IEEE, Ilmenau, Germany, 2019) pp. 193–198.

[48] D. Albani, T. Manoni, A. Arik, D. Nardi, and V. Trianni, *Field Coverage for Weed Mapping: Toward Experiments with a UAV Swarm,* in *Bio-Inspired Information and Communication Technologies*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, edited by A. Compagnoni, W. Casey, Y. Cai, and B. Mishra (Springer International Publishing, Cham, 2019) pp. 132–146.

[49] C. Di Franco and G. Buttazzo, *Coverage Path Planning for UAVs Photogrammetry with Energy and Resolution Constraints,* Journal of Intelligent & Robotic Systems **83**, 445 (2016).

[50] C. Manss, D. Shutin, A. V. Ruiz, T. Wiedemann, and J. Mueller, *Exploration under sparsity constraints,* in *2015 European Conference on Mobile Robots (ECMR)* (2015) pp. 1–6.

[51] C. Manss, D. Shutin, T. Wiedemann, A. Viseras, and J. Mueller, *Decentralized multi-agent entropy-driven exploration under sparsity constraints,* in *2016 4th International Workshop on Compressed Sensing Theory and Its Applications to Radar, Sonar and Remote Sensing (CoSeRa)* (2016) pp. 143–147.

[52] C. Manss, T. Wiedemann, and D. Shutin, *Entropy Driven Height Profile Estimation with Multiple UAVs under Sparsity Constraints,* in *2017 IEEE Globecom Workshops (GC Wkshps)* (IEEE, Singapore, Singapore, 2017) pp. 1–6.

[53] C. Manss and D. Shutin, *Global-Entropy Driven Exploration with Distributed Models under Sparsity Constraints,* Applied Sciences **8**, 21 (2018).

[54] Manss, D. Shutin, and G. Leus, *Distributed Splitting-Over-Features Sparse Bayesian Learning with Alternating Direction Method of Multipliers,* in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018-04-15/2018-04-20) pp. 3654–3658.

[55] C. Manss, D. Shutin, and G. Leus, *Coordination methods for entropy-based multi-agent exploration under sparsity constraints,* in *CAMSAP 2019* (Le Gosier, 2019).

[56] J. Müller, A. V. Ruiz, C. Manss, and T. Wiedemann, *C-ABT: A continuous control layer for inter-agent collision avoidance based on asynchronous backtracking,* in *2015 IEEE Conference on Control Applications (CCA)* (2015) pp. 539–544.

[57] E. Staudinger, D. Shutin, C. Manss, A. Viseras, and S. Zhang, *Swarm Technologies For Future Space Exploration Missions,* in *ISAIRAS '18: FOURTEENTH INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, ROBOTICS AND AUTOMATION IN SPACE* (Madrid, Spanien, 2018).

[58] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, *Decentralized multi-agent exploration with online-learning of Gaussian processes,* in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016) pp. 4222–4229.

[59] T. Wiedemann, C. Manss, D. Shutin, A. J. Lilienthal, V. Karolj, and A. Viseras, *Probabilistic modeling of gas diffusion with partial differential equations for multi-robot exploration and gas source localization,* in *2017 European Conference on Mobile Robots (ECMR)* (2017) pp. 1–7.

[60] C. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics (Springer-Verlag, New York, 2006).

[61] P. Mc Cullagh and J. Nelder, *Generalized Linear Models*, Chapman & Hall/CRC Monographs on Statistics and Applied Probability (Chapman and Hall/CRC, 1989).

[62] R. Penrose, *A generalized inverse for matrices,* Mathematical Proceedings of the Cambridge Philosophical Society **51**, 406 (1955).

[63] E. Candes and M. Wakin, *An Introduction To Compressive Sampling*, IEEE Signal Processing Magazine **25**, 21 (2008).

[64] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images,* SIAM Review **51**, 34 (2009).

[65] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications* (Society for Industrial and Applied Mathematics, 2001).

[66] R. G. Baraniuk, *Compressive Sensing [Lecture Notes],* IEEE Signal Processing Magazine **24**, 118 (2007).

[67] E. J. Candès, M. B. Wakin, and S. P. Boyd, *Enhancing Sparsity by Reweighted l1 Minimization,* Journal of Fourier Analysis and Applications **14**, 877 (2008).

[68] S. Chen, D. Donoho, and M. Saunders, *Atomic Decomposition by Basis Pursuit,* SIAM Review **43**, 129 (2001).

[69] M. Zibulevsky and M. Elad, *L1-L2 Optimization in Signal and Image Processing,* IEEE Signal Processing Magazine **27**, 76 (2010).

[70] S. Boyd and L. Vandenberghe, *Convex Optimization* (Cambridge University Press, 2004).

[71] Y. Sharon, J. Wright, and Y. Ma, *Computation and Relaxation of Conditions for Equivalence between l1 and l0 Minimization,* Coordinated Science Laboratory Report no. UILU-ENG-07-2208, DC-231 (2007).

[72] N. Huri and M. Feder, *Selecting the LASSO regularization parameter via Bayesian principles,* in *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)* (2016) pp. 1–5.

[73] N. Ternès, F. Rotolo, and S. Michiels, *Empirical extensions of the lasso penalty to reduce the false discovery rate in high-dimensional Cox regression models,* Statistics in Medicine **35**, 2561 (2016).

[74] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, 2nd ed., Springer Series in Statistics (Springer-Verlag, New York, 2009).

[75] M. Fornasier and H. Rauhut, *Iterative thresholding algorithms,* Applied and Computational Harmonic Analysis **25**, 187 (2008).

[76] N. Parikh and S. Boyd, *Proximal Algorithms,* Foundations and Trends® in Optimization **1**, 127 (2014).

[77] S. Tao, D. Boley, and S. Zhang, *Local Linear Convergence of ISTA and FISTA on the LASSO Problem,* SIAM Journal on Optimization **26**, 313 (2016).

[78] A. Beck and M. Teboulle, *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems,* SIAM Journal on Imaging Sciences **2**, 183 (2009).

[79] Yu. Nesterov, *Gradient methods for minimizing composite functions,* Mathematical Programming **140**, 125 (2013).

[80] X. Chen, Z. Nashed, and L. Qi, *Smoothing Methods and Semismooth Methods for Nondifferentiable Operator Equations,* SIAM Journal on Numerical Analysis **38**, 1200 (2000).

[81] M. Hintermuller, *Semismooth Newton Methods and Applications,* (2010).

[82] N. Aronszajn, *Theory of reproducing kernels,* Transactions of the American Mathematical Society **68**, 337 (1950).

[83] T. Hofmann, B. Schölkopf, and A. J. Smola, *Kernel methods in machine learning,* The Annals of Statistics **36**, 1171 (2008), arxiv:math/0701907 .

[84] S. Schölkopf, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, Adaptive Computation and Machine Learning (MIT Press, Cambridge, MA, USA, 2002).

[85] C. E. Rasmussen, *Gaussian Processes in Machine Learning*, in *Advanced Lectures on Machine Learning*, Vol. 3176, edited by O. Bousquet, U. von Luxburg, and G. Rätsch (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004) pp. 63–71.

[86] W. Burger and M. Burge, *Principles of Digital Image Processing: Core Algorithms*, Undergraduate Topics in Computer Science (Springer, London, 2009).

[87] S. Chapman and J. Bartels, *Geomagnetism, Vol. II: Analysis of the Data, and Physical Theories*, Geomagnetism, Vol. II: Analysis of the Data, and Physical Theories, by S. Chapman and J. Bartels. London: Oxford Univ. Press, 1940 (1940).

[88] N. M. Temme, *Special Functions: An Introduction to the Classical Functions of Mathematical Physics* (Wiley, New York, 1996).

[89] M. Rabbani, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Journal of Electronic Imaging **11**, 286 (2002).

[90] B. Ying and A. H. Sayed, *Information Exchange and Learning Dynamics Over Weakly Connected Adaptive Networks*, IEEE Transactions on Information Theory **62**, 1396 (2016).

[91] T. van Waterschoot and G. Leus, *Distributed estimation of static fields in wireless sensor networks using the finite element method*, in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2012) pp. 2853–2856.

[92] H. Zheng, S. R. Kulkarni, and H. V. Poor, *Attribute-Distributed Learning: Models, Limits, and Algorithms*, IEEE Transactions on Signal Processing **59**, 386 (2011).

[93] H. Zheng, S. R. Kulkarni, and H. V. Poor, *Dimensionally distributed learning models and algorithm*, in *2008 11th International Conference on Information Fusion* (2008) pp. 1–8.

[94] Y. Chen, J. Lu, X. Yu, and D. J. Hill, *Multi-Agent Systems with Dynamical Topologies: Consensus and Applications*, IEEE Circuits and Systems Magazine **13**, 21 (thirdquarter 2013).

[95] I. Jawhar, N. Mohamed, J. Wu, and J. Al-Jaroodi, *Networking of Multi-Robot Systems: Architectures and Requirements*, Journal of Sensor and Actuator Networks **7**, 52 (2018).

[96] F. D. Côté, I. N. Psaromiligkos, and W. J. Gross, *In-network linear regression with arbitrarily split data matrices*, in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2016) pp. 580–584.

[97] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, *A Hybrid Dictionary Approach for Distributed Kernel Adaptive Filtering in Diffusion Networks,* 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) , 5 (2018).

[98] G. Mateos, J. A. Bazerque, and G. B. Giannakis, *Distributed Sparse Linear Regression,* IEEE Transactions on Signal Processing **58**, 5262 (2010).

[99] A. Amini, A. Asif, and A. Mohammadi, *An Event-Triggered Average Consensus Algorithm with Performance Guarantees for Distributed Sensor Networks,* in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018-04-15/2018-04-20) pp. 3409–3413.

[100] E. Wei and A. Ozdaglar, *On the O(1=k) convergence of asynchronous distributed alternating Direction Method of Multipliers,* in *2013 IEEE Global Conference on Signal and Information Processing* (IEEE, 2013) pp. 551–554.

[101] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, *Robust Decentralized Learning Using ADMM with Unreliable Agents,* arXiv:1710.05241 [cs, stat] (2017), arxiv:1710.05241 [cs, stat] .

[102] J. Chen, Z. J. Towfic, and A. H. Sayed, *Dictionary Learning Over Distributed Models,* IEEE Transactions on Signal Processing **63**, 1001 (2015).

[103] Z. Xu, M. A. T. Figueiredo, and T. Goldstein, *Adaptive ADMM with Spectral Penalty Parameter Selection,* arXiv:1605.07246 [cs] (2016), arxiv:1605.07246 [cs] .

[104] B. S. He, H. Yang, and S. L. Wang, *Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities,* Journal of Optimization Theory and Applications **106**, 337 (2000).

[105] R. Griesse and D. A. Lorenz, *A semismooth Newton method for Tikhonov functionals with sparsity constraints,* Inverse Problems **24**, 035007 (2008).

[106] M. E. Tipping, *Sparse Bayesian Learning and the Relevance Vector Machine,* Journal of Machine Learning Research **1**, 211 (2001).

[107] M. E. Tipping and A. C. Faul, *Fast Marginal Likelihood Maximisation for Sparse Bayesian Models,* Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics , 14 (2003).

[108] D. P. Wipf and S. S. Nagarajan, *A New View of Automatic Relevance Determination,* in *Advances in Neural Information Processing Systems 20*, edited by J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Curran Associates, Inc., 2008) pp. 1625–1632.

[109] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis, Third Edition*, 3rd ed. (CRC Press, Boca Raton, 2013).

[110] N. L. Pedersen, C. Navarro Manchón, M.-A. Badiu, D. Shutin, and B. H. Fleury, *Sparse estimation using Bayesian hierarchical prior modeling for real and complex linear models,* Signal Processing **115**, 94 (2015).

[111] J. M. Mooij and H. J. Kappen, *Sufficient Conditions for Convergence of the Sum–Product Algorithm,* IEEE Transactions on Information Theory **53**, 4422 (2007).

[112] S. Tatikonda and M. I. Jordan, *Loopy Belief Propogation and Gibbs Measures,* arXiv:1301.0605 [cs] (2012), arxiv:1301.0605 [cs] .

[113] S. Khanna and C. R. Murthy, *Communication-Efficient Decentralized Sparse Bayesian Learning of Joint Sparse Signals,* IEEE Transactions on Signal and Information Processing over Networks **3**, 617 (2017).

[114] J. Hua and C. Li, *Distributed Jointly Sparse Bayesian Learning With Quantized Communication,* IEEE Transactions on Signal and Information Processing over Networks **4**, 769 (2018).

[115] G. H. Golub and Charles F. Van Loan, *Matrix Computations, Fourth Edition* (Johns Hopkins University Press Books, 2013).

[116] D. Tylavsky and G. Sohie, *Generalization of the matrix inversion lemma,* Proceedings of the IEEE **74**, 1050 (1986).

[117] T. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*, 621.39: 51 MON (Prentice Hall, Upper Saddle River, 2000).

[118] C. Manss, D. Shutin, and G. Leus, *Consensus Based Distributed Sparse Bayesian Learning By Fast Marginal Likelihood Maximization,* IEEE Signal Processing Letters , 1 (2020).

[119] A. Nedic, A. Ozdaglar, and P. Parrilo, *Constrained Consensus and Optimization in Multi-Agent Networks,* IEEE Transactions on Automatic Control **55**, 922 (2010).

[120] D. Shutin, S. R. Kulkarni, and H. V. Poor, *Incremental Reformulated Automatic Relevance Determination,* IEEE Transactions on Signal Processing **60**, 4977 (2012).

[121] B. Wohlberg, *ADMM Penalty Parameter Selection by Residual Balancing,* arXiv e-prints **1704**, arXiv:1704.06209 (2017).

[122] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, *Optimal Parameter Selection for the Alternating Direction Method of Multipliers (ADMM): Quadratic Problems,* IEEE Transactions on Automatic Control **60**, 644 (2015).

[123] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, *A General Analysis of the Convergence of ADMM,* arXiv:1502.02009 [cs, math] (2015), arxiv:1502.02009 [cs, math] .

[124] M. Park, S. Lee, and S. Lee, *Dynamic Topology Reconstruction Protocol for UAV Swarm Networking,* Symmetry **12**, 1111 (2020).

[125] Y. Xu, M. Liu, Q. Lin, and T. Yang, *ADMM without a Fixed Penalty Parameter: Faster Convergence with New Adaptive Penalization,* in *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017).

[126] S. M. Kay, *Fundamentals of statistical signal processing: Estimation theory,* (1993).

[127] N. N. Chan, *A-optimality for regression designs,* Journal of Mathematical Analysis and Applications **87**, 45 (1982).

[128] F. Chen, J. Wang, T. Shan, and B. Englot, *Autonomous Exploration Under Uncertainty via Graph Convolutional Networks,* **Proceedings of the International Symposium on Robotics Research**, 16 (2019).

[129] J. Wang and B. Englot, *Autonomous Exploration with Expectation-Maximization,* in *Robotics Research*, Springer Proceedings in Advanced Robotics, edited by N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti (Springer International Publishing, Cham, 2020) pp. 759–774.

[130] Ch. Hajiyev, *Determination of optimum measurement points via A-optimality criterion for the calibration of measurement apparatus,* Measurement **43**, 563 (2010).

[131] R. Sim and N. Roy, *Global A-Optimal Robot Exploration in SLAM,* in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (2005) pp. 661–666.

[132] H. Wynn, *Maximum Entropy Sampling and General Equivalence Theory,* in *mODa 7 — Advances in Model-Oriented Design and Analysis*, Contributions to Statistics, edited by A. Di Bucchianico, H. Läuter, and H. P. Wynn (Physica-Verlag HD, Heidelberg, 2004) pp. 211–218.

[133] S. W. Wanyonyi, A. A. Okango, and J. K. Koech, *Exploration of D-, A-, I- and G-Optimality Criteria in Mixture Modeling,* Asian Journal of Probability and Statistics , 15 (2021).

[134] M. J. Box and N. R. Draper, *Factorial Designs, the |X′X| Criterion, and Some Related Matters,* Technometrics **13**, 731 (1971).

[135] M. Choueiki and C. Mount-Campbell, *Training data development with the D-optimality criterion,* IEEE Transactions on Neural Networks **10**, 56 (1999).

[136] D. C. Montgomery, E. N. Loredo, D. Jearkpaporn, and M. C. Testik, *Experimental Designs for Constrained Regions,* Quality Engineering **14**, 587 (2002).

[137] H. Carrillo, I. Reid, and J. A. Castellanos, *On the comparison of uncertainty criteria for active SLAM,* in *2012 IEEE International Conference on Robotics and Automation* (2012) pp. 2080–2087.

[138] V. V. Fedorov, *Theory of optimal experiments,* https://cds.cern.ch/record/228979 (1972).

[139] T. Patten and R. Fitch, *Large-Scale Near-Optimal Decentralised Information Gathering with Multiple Mobile Robots,* Proceedings of Australasian Conference on Robotics and Automation , 9 (2013).

[140] D.-H. Cho, J.-S. Ha, S. Lee, S. Moon, and H.-L. Choi, *Informative Path Planning and Mapping with Multiple UAVs in Wind Fields,* arXiv:1610.01303 [cs] (2016), arxiv:1610.01303 [cs] .

[141] H. L. Van Trees and K. L. Bell, *Detection Estimation and Modulation Theory, Detection, Estimation, and Filtering Theory* (2013).

[142] E. Candes, *Mathematics of sparsity (and a few other things),* in *Proceedings of the International Congress of Mathematicians* (Citeseer, Seoul, Sout Korea, 2014) p. 27.

[143] J. Ding and A. Zhou, *Eigenvalues of rank-one updated matrices with some applications,* Applied Mathematics Letters **20**, 1223 (2007).

[144] C. Brezinski, *Schur Complements and Applications in Numerical Analysis,* in *The Schur Complement and Its Applications*, Numerical Methods and Algorithms, edited by F. Zhang (Springer US, Boston, MA, 2005) pp. 227–258.

[145] D. F. Watson, *Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes,* The Computer Journal **24**, 167 (1981).

[146] S. Roelofsen, A. Martinoli, and D. Gillet, *Distributed deconfliction algorithm for Unmanned Aerial Vehicles with limited range and field of view sensors,* in *2015 American Control Conference (ACC)* (IEEE, Chicago, IL, USA, 2015) pp. 4356–4361.

[147] P. R. Mahaffy, C. R. Webster, M. Cabane, P. G. Conrad, P. Coll, S. K. Atreya, R. Arvey, M. Barciniak, M. Benna, L. Bleacher, W. B. Brinckerhoff, J. L. Eigenbrode, D. Carignan, M. Cascia, R. A. Chalmers, J. P. Dworkin, T. Errigo, P. Everson, H. Franz, R. Farley, S. Feng, G. Frazier, C. Freissinet, D. P. Glavin, D. N. Harpold, D. Hawk, V. Holmes, C. S. Johnson, A. Jones, P. Jordan, J. Kellogg, J. Lewis, E. Lyness, C. A. Malespin, D. K. Martin, J. Maurer, A. C. McAdam, D. McLennan, T. J. Nolan, M. Noriega, A. A. Pavlov, B. Prats, E. Raaen, O. Sheinman, D. Sheppard, J. Smith, J. C. Stern, F. Tan, M. Trainer, D. W. Ming, R. V. Morris, J. Jones, C. Gundersen, A. Steele, J. Wray, O. Botta, L. A. Leshin, T. Owen, S. Battel, B. M. Jakosky, H. Manning, S. Squyres, R. Navarro-González, C. P. McKay, F. Raulin, R. Sternberg, A. Buch, P. Sorensen, R. Kline-Schoder, D. Coscia, C. Szopa, S. Teinturier, C. Baffes, J. Feldman, G. Flesch, S. Forouhar, R. Garcia, D. Keymeulen, S. Woodward, B. P. Block, K. Arnett, R. Miller, C. Edmonson, S. Gorevan, and E. Mumm, *The Sample Analysis at Mars Investigation and Instrument Suite,* Space Science Reviews **170**, 401 (2012).

[148] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, *An online multi-robot SLAM system for 3D LiDARs,* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017) pp. 1004–1011.

[149] J. Vallvé and J. Andrade-Cetto, *Mobile robot exploration with potential information fields,* in *2013 European Conference on Mobile Robots* (2013) pp. 222–227.

[150] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, *The Office Marathon: Robust navigation in an indoor office environment,* in *2010 IEEE International Conference on Robotics and Automation* (2010) pp. 300–307.

[151] C. Rösmann, F. Hoffmann, and T. Bertram, *Planning of multiple robot trajectories in distinctive topologies,* in *2015 European Conference on Mobile Robots (ECMR)* (2015) pp. 1–6.

[152] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, *OctoMap: An efficient probabilistic 3D mapping framework based on octrees,* Autonomous Robots **34**, 189 (2013).

[153] B. Siebler, S. Sand, and U. D. Hanebeck, *Localization with Magnetic Field Distortions and Simultaneous Magnetometer Calibration,* IEEE Sensors Journal , 1 (2020).

[154] S. Bonnet, C. Bassompierre, C. Godin, S. Lesecq, and A. Barraud, *Calibration methods for inertial and magnetic sensors,* Sensors and Actuators A: Physical **156**, 302 (2009).

[155] W. Zeng and R. L. Church, *Finding shortest paths on real road networks: The case for A*,* (2009), 10.1080/13658810801949850.

[156] P. Renton, M. Greenspan, H. A. ElMaraghy, and H. Zghal, *Plan-N-Scan: A Robotic System for Collision-Free Autonomous Exploration and Workspace Mapping,* Journal of Intelligent and Robotic Systems **24**, 207 (1999).

[157] P. Voosen, *Perseverance will explore history of ancient lake,* Science **371**, 870 (2021).

[158] N. Potter, *A Mars helicopter preps for launch: The first drone to fly on another planet will hitch a ride on NASA's Perseverance rover - [News],* IEEE Spectrum **57**, 06 (2020).

[159] P. Ye, Z. Sun, W. Rao, and L. Meng, *Mission overview and key technologies of the first Mars probe of China,* Science China Technological Sciences **60**, 649 (2017).

[160] J. F. Bell, J. N. Maki, G. L. Mehall, M. A. Ravine, M. A. Caplinger, Z. J. Bailey, S. Brylow, J. A. Schaffner, K. M. Kinch, M. B. Madsen, A. Winhold, A. G. Hayes, P. Corlies, C. Tate, M. Barrington, E. Cisneros, E. Jensen, K. Paris, K. Crawford, C. Rojas, L. Mehall, J. Joseph, J. B. Proton, N. Cluff, R. G. Deen, B. Betts, E. Cloutis, A. J. Coates, A. Colaprete, K. S. Edgett, B. L. Ehlmann, S. Fagents, J. P. Grotzinger, C. Hardgrove, K. E. Herkenhoff, B. Horgan, R. Jaumann, J. R. Johnson, M. Lemmon, G. Paar, M. Caballo-Perucha, S. Gupta, C. Traxler, F. Preusker, M. S. Rice, M. S. Robinson, N. Schmitz, R. Sullivan, and M. J. Wolff, *The Mars 2020 Perseverance Rover Mast Camera Zoom (Mastcam-Z) Multispectral, Stereoscopic Imaging Investigation,* Space Science Reviews **217**, 24 (2021).

[161] J. N. Maki, D. Gruel, C. McKinney, M. A. Ravine, M. Morales, D. Lee, R. Willson, D. Copley-Woods, M. Valvo, T. Goodsall, J. McGuire, R. G. Sellar, J. A. Schaffner, M. A. Caplinger, J. M. Shamah, A. E. Johnson, H. Ansari, K. Singh, T. Litwin, R. Deen, A. Culver, N. Ruoff, D. Petrizzo, D. Kessler, C. Basset, T. Estlin, F. Alibay, A. Nelessen, and S. Algermissen, *The Mars 2020 Engineering Cameras and Microphone on the Perseverance Rover: A Next-Generation Imaging System for Mars Exploration,* Space Science Reviews **216**, 137 (2020).

[162] B. L. Teece, S. C. George, O. B. A. Agbaje, S. M. Jacquet, and G. A. Brock, *Mars Rover Techniques and Lower/Middle Cambrian Microbialites from South Australia: Construction, Biofacies, and Biogeochemistry*, Astrobiology **20**, 637 (2020).

[163] I. Levchenko, S. Xu, S. Mazouffre, M. Keidar, and K. Bazaka, *Mars Colonization: Beyond Getting There,* Global Challenges **3**, 1800062 (2019).

[164] A. Swaminathan and V. Malhotra, *A Spatial Perspective of Space Colonization on Mars,* in *2021 IEEE Aerospace Conference (50100)* (2021) pp. 1–7.

[165] A. Viseras, D. Shutin, and L. Merino, *Robotic Active Information Gathering for Spatial Field Reconstruction with Rapidly-Exploring Random Trees and Online Learning of Gaussian Processes,* Sensors **19**, 1016 (2019).

[166] R. Horn, *The hadamard product,* in *Proc. Symp. Appl. Math*, Vol. 40 (1990) pp. 87–169.

[167] K. Åhlander, *Einstein summation for multidimensional arrays,* Computers & Mathematics with Applications **44**, 1007 (2002).

[168] L. Xiao and S. Boyd, *Fast linear iterations for distributed averaging,* Systems & Control Letters **53**, 65 (2004).

[169] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, *Asynchronous Distributed Averaging on Communication Networks,* IEEE/ACM Transactions on Networking **15**, 512 (2007).

[170] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, *Gossip Algorithms for Distributed Signal Processing,* Proceedings of the IEEE **98**, 1847 (2010).

[171] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms,* IEEE Transactions on Information Theory **52**, 2508 (2006).

[172] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, *Geographic Gossip: Efficient Averaging for Sensor Networks,* IEEE Transactions on Signal Processing **56**, 1205 (2008).

[173] D. Ustebay, B. Oreshkin, M. Coates, and M. Rabbat, *Rates of convergence for greedy gossip with eavesdropping,* in *2008 46th Annual Allerton Conference on Communication, Control, and Computing* (2008) pp. 367–374.

[174] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, *Broadcast Gossip Algorithms for Consensus,* IEEE Transactions on Signal Processing **57**, 2748 (2009).

[175] A. V. Goldberg and C. Harrelson, *Computing the Shortest Path: A∗ Search Meets Graph Theory*, Tech. Rep. (2003).

ABBREVIATIONS

ADMM	alternating direction method of multipliers
ARD	automatic relevance determination
AWGN	additive white Gaussian noise
DCT	discrete cosine transformation
DE-SOF-LASSO	distributed exploration with SOF for LASSO
DE-SOE-LASSO	distributed exploration with SOE for LASSO
DFMLM	distributed FMLM
DLASSO	distributed LASSO
DLR	German Aerospace Center
D-R-ARD	distributed reformulated ARD
EM	expectation maximization
FISTA	fast ISTA
FMLM	fast marginalized likelihood maximization
FOVA	field-of-view avoidance
GP	Gaussian process
i.i.d.	identically and independent distributed
IMU	inertial measurement unit
ISTA	iterative soft-thresholding algorithm
JSM	joint sparsity model
LASSO	least absolute shrinkage and selection operator
LIDAR	light detection and ranging
MAP	maximum a-posteriori
MER-A	Mars Exploration Rover - A
NRMSE	normalized root mean square error

OED	optimal experiment design
PDF	probability density function
R-ARD	reformulated ARD
RBF	radial basis function
ROS	robot operating system
SBL	sparse Bayesian learning
SLAM	simultaneous localization and mapping
SNR	signal to noise ratio
SOF	splitting-over-features
SOE	splitting-over-examples
UAV	unmanned aerial vehicle

ACKNOWLEDGEMENTS

First I would like to express my gratitude to both of my esteemed supervisors Prof. Geert Leus at the TUDelft and Dr. Dmitriy Shutin at the DLR. Their support, patience during lengthy derivations, and guidance in paper writing have been invaluable to me. You both have played an integral role in shaping my growth and development as a researcher.

I extend my gratitude to all colleagues at the DLR and particularly the colleagues at the Department Communication Systems. Throughout my PhD journey, I've deeply appreciated the accessibility of each office for both professional insights and emotional support. I am thankful for each Weißwurstfrühstück, all measurement campaigns, and festive occasions. A heartfelt thanks goes out to the colleagues in the Swarm Exploration group: Alberto, Joachim, Juan, Thomas, and Valentina. Your kindness during challenging and demanding periods has been vital to me, and every moment spent with you has been cherished.

Furthermore, I am grateful to the DLR for facilitating my research stay at TUDelft, where I had the opportunity to collaborate with the EEMCS group. In particular, I'd like to thank Mario for his strong support and engaging philosophical discussions. I also extend my thanks to my new colleagues at the DFKI, who supported me during the writing process.

I want to express my gratitude to my friends, Benjamin, Paul, and Mohammad, with whom I spent countless hours playing board games and engaging in deep discussions about PhD challenges. Additionally, a special thanks to the friends i made when I have been an undergraduate who embarked on their own PhD journeys elsewhere. I am particularly thankful to Moritz for our extended writing sessions, both online and offline, which made the writing process not only bearable but also enjoyable.

Lastly but definitely not least, a heartfelt thanks to my mother, Regina, and my partner, Jule, who have provided emotional support and encouraged me to pursue this PhD journey.

Thank you!

CURRICULUM VITÆ

Christoph Manß was born in Eckernförde, Germany in 1987. He received the Bachelors and the Master of Science degree in Electrical Engineering and Informationtechnology from the Christian-Albrechts-Universität zu Kiel in 2012 and 2014, respectively. The topic of this master thesis was "Towards a Swarm Navigation System on Mars - Test Data Generation and Evaluation" in cooperation with the Institute of Communication and Navigation at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. He then did his research for his PhD at the same institute from 2014 until 2020 in the Swarm Exploration group together with Dr. Dmitriy Shutin. During that time, he was enrolled as a part-time PhD student at the TUDelft in the Netherlands supervised by Prof. Geert Leus. Since 2021 until now, he is working at the German Research Center for Artificial Intelligence (DFKI) at the institute of Marine Perception in Oldenburg, Germany. There, he is currently team leader of the research group Machine Learning Systems.