



Hybrid Electronic Systems
Technical University of Munich
Department of Electrical and Computer Engineering
Univ.-Prof. Dr. rer. nat. Franz Kreupl



MASTER'S THESIS

Sample-Efficient Hyperparameter Optimization of an Aim Point Controller for Solar Power Tower Plants by Bayesian Optimization

Author:	Barbara Marie Anna Lenz
Matriculation Number:	03658801
Address:	Entenbachstraße 12 81541 München
Email:	lenzbarbara@t-online.de
Supervisor:	Herr David Zanger
Begin:	01.12.2021
End:	31.07.2022

Abstract

In this work, sample-efficient algorithms for a controller hyperparameter optimization of an arbitrary aim point controller for solar power tower plants are introduced. The objective is to find controller parameters, which optimize the performance of the aim point controller, and thus increase the efficiency of the plant. This should be accomplished within a minimum number of optimization steps, which implies the need of a sample-efficient optimization strategy. The algorithms, proposed in this work, are based on the Bayesian Optimization (BO) approach [41] and enhance the algorithm's sample efficiency by leveraging simulation data as prior information. It is assumed that the utilized simulation data is possibly corrupted by mismatches to the system's real behavior and thus does not contain information about the optimal controller parameter configurations. Therefore, it is not possible to choose them directly from the simulation data, however it can still contain helpful information to accelerate the optimization. The controller parameters, selected by an optimization algorithm, have to be evaluated on the plant, after every optimization iteration. Testing the controller parameters on the real system is a time-consuming procedure, which explains the need to reduce the optimization iterations to a minimum. The algorithms, proposed for this purpose, are mostly based on the methods for leveraging prior information in BO of Antonova and Rai et al. [4], [45] and extended to the use of multiple sets of simulation data, which was not sufficiently covered in literature so far. Moreover, a novel approach for utilizing simulation data in BO is introduced in this work, named Prior-Guided Expected Improvement. The algorithms were tested on a six-dimensional test function, which imitates the performance of an aim point controller, dependent on six controller hyperparameters. Several sets of simulation data were deployed, that partly resemble the function and do not contain the function's global optimum. As a reference, the standard BO algorithm was used. Two of the proposed approaches outperformed the reference by reaching close to optimal controller hyperparameters within 33 % less optimization steps, than the standard BO. In addition, the prior-informed algorithms seemed to be less prone to get stuck in local optima, than the standard BO. Moreover, in case of high simulation to reality mismatches or unsuitable simulation data, the prior-informed algorithms still yielded results similar to the reference. In a second test case, the proposed approaches were used to optimize a simulated Vant-Hull aim point controller with two hyperparameters, where they needed 23 % less optimization iterations than the standard BO. However, to test the prior-informed aim point controller optimization on a real solar power tower plant, further development has to be done to guarantee save controller behavior during the hyperparameter optimization. Thereby, damages to the receiver, caused by overheating, can be prevented.

Contents

List of Figures	XII
List of Algorithms	XIII
List of Tables	XVI
List of Symbols	XVIII
List of Abbreviations	XIX
1 Introduction	1
1.1 Motivation	1
1.2 Objective	2
2 Fundamentals and Literature Review	1
2.1 Solar Power Tower Plants	1
2.1.1 System Components	1
2.1.2 Aim Point Control	3
2.2 Overview on Black-Box Hyperparameter Optimization	5
2.2.1 Grid Search and Random Search	5
2.2.2 Population-Based Algorithms	5
2.2.3 Model-Based Algorithms	6
2.3 Bayesian Optimization Fundamentals	8
2.3.1 Gaussian Process Regression	8
2.3.2 Handling Hyperparameters of the Gaussian Process	12
2.3.3 Acquisition Function	15
2.3.4 Bayesian Optimization Algorithm	18
2.4 Enhancing Sample Efficiency in Bayesian Optimization	20
2.4.1 Prior-Informed Mean Functions	21
2.4.2 Prior-Informed Kernel Functions	23
2.4.3 Bayesian Model Selection	26
2.4.4 Combining Models	30
3 Bayesian Optimization with Multiple Simulation Data Sets	33
3.1 Interpolated Simulation Data	33
3.2 Prior-Informed Mean Function	34
3.2.1 Handling Hyperparameters	36

3.2.2	Pitfalls of the Prior-Informed Mean Functions	36
3.3	Prior-Informed Kernel Function	40
3.3.1	Handling Hyperparameters	41
3.3.2	Mismatch Correction	41
3.4	Prior-Guided Expected Improvement	43
3.4.1	Determining the Weight Parameter	44
3.4.2	Mismatch Correction	45
3.5	Expansion to Multiple Sets of Simulation Data	46
3.5.1	Multiple Prior-Informed Models	46
3.5.2	Prior-Guided EI with Multiple Sources of Prior Information	49
3.6	Resulting Algorithms	50
3.6.1	Algorithms with Prior-Informed Kernels	50
3.6.2	Algorithm with Prior-Guided Acquisition Function	53
3.6.3	Hybrid Algorithms	55
3.6.4	Algorithm Overview	56
4	Algorithm Evaluations and Performance Analysis	57
4.1	Hartmann6 Test Function	57
4.2	Simulation Data	58
4.3	Baseline	58
4.4	Evaluation Criteria	59
4.4.1	Number of Function Evaluations	60
4.4.2	Accuracy of the Located Optimum	61
4.4.3	Overcoming Local Optima	63
4.4.4	Dealing with Unsuitable Simulation Data	65
4.5	Algorithm Evaluations	66
4.5.1	Test Case	66
4.5.2	Number of Function Evaluations	66
4.5.3	Accuracy of the Located Optimum	68
4.5.4	Overcoming Local Optima	71
4.5.5	Dealing with Unsuitable Simulation Data	76
4.5.6	Comparison of the Algorithms	87
4.5.7	Conclusion	89
5	Application: Vant-Hull-Controller	91
5.1	Vant-Hull Control Strategy with Two Controller Parameters	91
5.2	Simulation Data	92
5.3	Test Case	93
5.4	Results	94
5.5	Conclusion	98
6	Summary and Outlook	101
A	Supplementary Material	105
A.1	Neural Networks for Interpolating Simulation Data	105
A.1.1	Regressing Hartmann6 Simulation Data	105
A.1.2	Regressing Vant-Hull Simulation Data	105

A.2	Supplementary Test Results for Hartmann6	106
A.2.1	All Simulation Data Sets	106
A.2.2	All Simulation Data Sets with Fixed Initial Parameters	107
A.2.3	Simulation Data Set 1	108
A.2.4	Simulation Data Set 4	109
Bibliography		111

List of Figures

2.1	Simplified schematic structure of a solar power tower plant	3
2.2	Function regression with a Gaussian Process (GP) model	11
2.3	Impact of the length-scale parameter l on the Gaussian Process (GP) prior and posterior distribution, shown for two different values of the length-scale parameter	13
2.4	Gaussian Process (GP) posterior with Upper Confidence Bound (UCB) for different values of the parameter γ	16
2.5	Visualization of the Probability of Improvement (PI) acquisition function	17
2.6	Gaussian Process (GP) posterior with Expected Improvement (EI) acquisition function	18
2.7	One-dimensional Bayesian Optimization for three optimization steps .	20
3.1	Corrupted simulation data for a sinus function	39
3.2	Degenerate scenario of Bayesian Optimization with prior-informed mean function and vanishing posterior standard deviation for three optimization steps	39
4.1	Hartmann6 function values vs. simulation data predictions for four sets of simulation data	59
4.2	Hartmann6 test: Mean distance to the global optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 30 optimization runs	63
4.3	Hartmann6 test: mean distance to the global optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 5 optimization runs with fixed initial parameter configurations	65
4.4	Hartmann6 test: mean distance to the local optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 5 optimization runs with fixed initial parameter configurations	65
4.5	Hartmann6 tests with all simulation data sets for best performing algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	69

4.6	Hartmann6 tests with all simulation data sets for best performing algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	70
4.7	Hartmann6 tests with all simulation data sets and fixed initial parameters for best performing algorithms: mean distance to the global optimum and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, in function output space (a) and parameter space (b)	74
4.8	Hartmann6 tests with simulation data set 1 for best performing algorithms: mean distance to the global optimum and confidence interval 95 %, averaged over 30 optimization runs, in function output space (a) and parameter space (b)	77
4.9	Hartmann6 tests with simulation data set 1 for selected algorithms: mean distance to the local optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for best performing approaches (a) and worst performing approaches (b) . . .	78
4.10	Hartmann6 tests with simulation data set 4: weights of used models in the Weighted Mixture Expected Improvement, averaged over 30 optimization runs, without mismatch correction (a) and with mismatch correction (b)	81
4.11	Hartmann6 tests with simulation data set 4 for algorithms with prior-informed kernel: used model per iteration for 30 optimization runs, for algorithms with and without mismatch correction and different acquisition functions	82
4.12	Hartmann6 tests with simulation data set 4 for algorithms with Weighted Mixture Kernel: used model per iteration for 30 optimization runs, for algorithms with and without mismatch correction and different acquisition functions	84
4.13	Hartmann6 tests with simulation data set 4: weights of used sub-kernels in the Weighted Mixture Kernel, averaged over 30 optimization runs with Expected Improvement, without mismatch correction (a) and with mismatch correction (b)	84
4.14	Hartmann6 tests with simulation data set 4: weighting of the simulation data in the Prior-Guided Expected Improvement, averaged over 30 optimization runs, without mismatch correction (a) and with mismatch correction (b)	85
5.1	Heliostat field layout: northern field of the Jülich plant	92
5.2	Vant-Hull objective function values vs. simulation data predictions for two sets of simulation data	94
5.3	Vant-Hull tests on all simulation data sets: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	96

5.4	Vant-Hull tests on all simulation data sets: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	97
5.5	Mean prediction and standard error of the Vant-Hull objective function, based on objective function observations	98
A.1	Hartmann6 tests with all simulation data sets for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	106
A.2	Hartmann6 tests with all simulation data sets for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	106
A.3	Hartmann6 tests with all simulation data sets and fixed initial parameters for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	107
A.4	Hartmann6 tests with all simulation data sets and fixed initial parameters for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	107
A.5	Hartmann6 tests with simulation data set 1 for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	108
A.6	Hartmann6 tests with simulation data set 1 for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	108
A.7	Hartmann6 tests with simulation data set 4: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	109

A.8	Hartmann6 tests with simulation data set 4: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)	109
A.9	Hartmann6 tests with simulation data set 4: weights of used sub-kernels in the Weighted Mixture Kernel, averaged over 30 optimization runs, with Most Likely Expected Improvement acquisition function without mismatch correction (a) and with mismatch correction (b)	110

List of Algorithms

1	Bayesian Optimization	19
2	Bayesian Optimization with prior-informed kernels	52
3	Bayesian Optimization with Weighted Mixture Kernels	53
4	Bayesian Optimization with Weighted Mixture Expected Improvement	54
5	Bayesian Optimization with prior-guided acquisition function	54
6	Bayesian Optimization with Prior-Guided Expected Improvement and prior-informed kernels	55

List of Tables

3.1	Overview on resulting prior-informed algorithms for leveraging multiple sets of simulation data	56
4.1	Local minima of 6-dimensional Hartmann6 test function	58
4.2	Hartmann6 test: number of required function evaluations (eval.) of standard Bayesian Optimization to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum, averaged over 30 optimization runs	61
4.3	Hartmann6 test: accuracy (acc.) of standard Bayesian Optimization after 25 % (= 19 eval.), 50 % (= 38 eval.), 75 % (= 57 eval.) and 100 % of maximum function evaluations (eval.) (= 75 eval.), averaged over 30 optimization runs	63
4.4	Hartmann6 test: accuracy of standard Bayesian Optimization in parameter space after 25 % (= 19 eval.), 50 % (= 38 eval.), 75 % (= 57 eval.) and 100 % of maximum function evaluations (eval.) (= 75 eval.), averaged over 30 optimization runs	63
4.5	Hartmann6 test: accuracy (acc.) of standard Bayesian Optimization in function output space after 19, 38, 57 and 75 evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations	64
4.6	Hartmann6 test: accuracy of standard Bayesian Optimization in parameter space (acc. p.) after 19, 38, 57 and 75 evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations	64
4.7	Hartmann6 tests with all simulation data sets: number of required function evaluations (eval.) to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum in comparison to the baseline, averaged over 30 optimization runs	67
4.8	Hartmann6 tests with all simulation data sets: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs	69
4.9	Hartmann6 tests with all simulation data sets: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs . . .	71
4.10	Hartmann6 tests with all simulation data sets and fixed initial parameters: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations .	72

4.11	Hartmann6 tests with all simulation data sets and fixed initial parameters: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations	74
4.12	Hartmann6 tests with simulation data set 1: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs	75
4.13	Hartmann6 tests with simulation data set 1: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs	77
4.14	Hartmann6 tests with simulation data set 4: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations	79
4.15	Hartmann6 tests with simulation data set 4: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations	80
4.16	Comparison of the algorithms regarding the evaluation criteria 1. number of required function evaluations, 2. accuracy of the located optimum, 3. overcoming local optima and 4. dealing with unsuitable or "bad" simulation data, where "r" denotes reasonable performance, "g" denotes good performance, "vg" denotes very good performance and "x" denotes bad performance	88
5.1	Generating 14 sets of simulation data of the solar power tower plant in Jülich, by varying the following simulation hyperparameters: maximum allowable flux density [W m^{-2}], mirror error [mrad] and mirror reflectivity factor	93
5.2	Vant-Hull tests with all simulation data sets: number of required function evaluations (eval.) to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum in comparison to the baseline, averaged over 30 optimization runs	95
5.3	Vant-Hull tests on all simulation data sets: accuracy (acc.) in function output space after 25 % (= 14 eval.), 50 % (= 27 eval.) and 75 % (= 39 eval.) of maximum function evaluations (eval.) (= 52 eval.), averaged over 30 optimization runs	96
5.4	Vant-Hull tests on all simulation data sets: accuracy in parameter space (acc. p.) after 25 % (= 14 eval.), 50 % (= 27 eval.), 75 % (= 39 eval.) and 100 % of maximum function evaluations (eval.) (= 52 eval.), averaged over 30 optimization runs	97

List of Symbols

α	Control parameter for the prior-guided acquisition function
β	Control parameter of the prior-informed mean function
$\boldsymbol{\theta}$	Vector of Gaussian Process prior hyperparameters
ϵ	Observation noise
γ	Control parameter for the Upper Confidence Bound
μ_n	Posterior mean under n observations
$\phi_{\max,i}$	Maximum allowed flux density at receiver bin i
ϕ_i	Flux density at receiver bin i
Ψ	Cumulative density function
ψ	Probability density distribution
σ_k^2	Output variance of the kernel function
σ_ϵ^2	Variance of the observation noise
σ_n^2	Posterior variance under n observations
\mathbf{f}_*	Vector of unevaluated function values
\mathbf{x}^*	Vector of optimal parameters
\mathbf{x}^+	Vector of best observed parameters so far
\mathbf{x}_*	Vector of unevaluated parameters
\mathbf{x}	Vector of parameters
\mathbf{y}	Vector of observed function evaluations
\hat{a}	Marginalized acquisition function

A	Feasible set of parameter vectors
a_{EI}	Expected Improvement
a_{UCB}	Upper Confidence Bound
a_{MLEI}	Most Likely Expected Improvement
a_{PI}	Probability of Improvement
a_{UMEI}	Utility Mean Expected Improvement
a_{WMEI}	Weighted Mixture Expected Improvement
c	Prior mean constant
CV_{MC}	Monte Carlo Cross-Validation score
f	Objective function to be optimized
$f_{\text{NN},i}$	Function interpolating simulation data set i
I	Identity matrix
K	Covariance matrix
k	Covariance function
$k_{\text{NN},\text{mis}}$	Prior-informed kernel with neural network and mismatch correction
k_{NN}	Prior-informed kernel with neural network
k_{SE}	Squared exponential kernel
l	Kernel length-scale parameter
m	Prior mean
N	Budget of objective function evaluations
n_{s}	Number of available simulation data sets
p_{ML}	Marginal likelihood of a Gaussian Process model
p_{pp}	Predictive posterior probability of a Gaussian Process model
X	Set of evaluated parameter vectors
X_*	Set of unevaluated parameter vectors

List of Abbreviations

BBK	behavior-based kernels
BO	Bayesian Optimization
CDF	cumulative distribution function
CI	confidence interval
CV	Cross-Validation
DNI	direct normal irradiation
EI	Expected Improvement
GAs	Genetic Algorithms
GP	Gaussian Process
MLEI	Most Likely Expected Improvement
MPC	Model Predictive Control
NN	Neural Network
PDF	probability density distribution
PGEI	Prior-Guided Expected Improvement
PI	Probability of Improvement
PID	Proportional Integral Derivative
PIK	prior-informed kernel
PSO	Particle Swarm Optimization
RL	Reinforcement Learning
SE	squared exponential
SEK	squared exponential kernel
TPE	Tree-Structured Parzen Estimator
UCB	Upper Confidence Bound
WMEI	Weighted Mixture Expected Improvement
WMK	Weighted Mixture Kernel

Chapter 1

Introduction

1.1 Motivation

To mitigate climate change, the climate target plan of the European Union aims to cut greenhouse gas emissions of the base year 1990 by at least 55% by 2030 and become climate neutral by 2050 [1]. To achieve these urgent objectives, the transition to renewable energy supplies has become an important topic in politics, as well as research. Additionally, the current global energy crisis, caused by Russia's war against Ukraine, has added new urgency to accelerate renewable energy transitions and highlighted its key role. Especially, wind and solar energy have the potential to reduce the dependence of the European Union's power sector on natural gas from the Russian Federation [3].

The field of solar energy offers a variety of different technologies to produce climate friendly energy, such as photovoltaic modules, linear Fresnel systems, parabolic through systems or solar power tower plants. Among those technologies, the main advantage of solar power tower plants is their ability of storing energy. This is of great importance, since there are possible temporal variations and regional differences in seasonal, monthly and daily power demand. By storing energy, these load fluctuations and mismatches between the plant's power generation and the power demand can be compensated. Furthermore, they are regarded as the concentrated solar power technology with the highest cost reduction potential and can be upscaled easily and cost-effectively [5].

A solar power tower plant possesses a multitude of mirrors, called heliostats, which focus the sunlight onto a receiver mounted on top of a tower. The concentrated radiation is absorbed by a heat transfer medium (e.g. molten salt) inside the receiver. Consequently, the thermal energy of the medium is fed into a connected energy conversion process (e.g. steam power process) for electricity production. Due to high achievable concentration ratios, solar power tower plants yield a high theoretical efficiency, compared to other concentrated solar power technologies.

Worldwide, currently around 25 solar power tower plants are operational, such as the Ashalim Power Station in Israel with a maximum capacity of 121 MW or the recently commissioned Atacama-1 in Chile with a maximum capacity of 110 MW [18]. In China, another solar power tower plant, called Shouhang Yumen, with a maximum capacity of 100 MW, is under construction [19].

Increasing the efficiency of solar power tower plants is of great importance, since it considerably raises their economical value. One possibility to increase the efficiency of solar power tower plants, is to use an aim point control strategy, which intends to maximize the power on the receiver, while simultaneously not exceeding an allowable maximum irradiation. This prevents damages to the receiver, caused by overheating. An optimal aim point control algorithm is still a research question. However, most of the potential aim point control strategies possess tunable hyperparameters with complex dependencies to the control behavior. Here, the choice of the hyperparameters often considerably influences the controller's performance and thus the efficiency of the plant. This leads to a non-trivial hyperparameter optimization problem. For simplicity, the hyperparameters of a control strategy are often referred to as controller parameters.

Usually, simulations can be used to find optimal controller parameters. Unfortunately, deviations from expected performance occur when there is a mismatch between simulation and reality due to modeling errors. Thus, to ensure optimal control behavior, hyperparameters have to be readjusted on the real plant. This involves testing various parameter configurations on the real system and observing the resulting performance of the aim point controller, which requires expert knowledge and is a time-consuming procedure without guarantee of finding close to optimal parameter configurations. A more sophisticated approach would be the use of a sample-efficient optimization algorithm and further enhance sample efficiency by leveraging available data from simulations. In this context, sample or data efficiency means the amount of information the optimization algorithm is able to use from observed controller parameter evaluations. Ideally, an increase in sample efficiency decreases the number of required optimization iterations to find nearly optimal controller parameters and thus makes the procedure less time-consuming.

In literature, there exist approaches to sample-efficient hyperparameter optimization. The most commonly used algorithm is called Bayesian Optimization (BO). In fact, the BO framework allows to incorporate simulation data as prior information to increase the sample efficiency of the algorithm. Former research in this area mainly stems from the field of model-based Reinforcement Learning (RL). However, they do not sufficiently engage with the question of how to deal with multiple sets of simulation data, which are likely to have high mismatches to the real system. This aspect is necessary for the application of solar power tower plants, since some simulation variables of the plant, like mirror errors, rely on possibly inaccurate estimations. Therefore, it would be beneficial to generate multiple simulation data sets for varying values of the simulation variable in question and jointly use them to enhance the sample efficiency of the optimization algorithm.

1.2 Objective

The objective of this thesis is to develop hyperparameter optimization approaches, which increase sample efficiency in BO, by incorporating multiple sets of simulation data, for the application of finding nearly optimal controller parameters of an arbitrary aim point controller for solar power tower plants. An increase in sample efficiency ideally leads to a decrease in optimization iterations, which are required

to find a close to optimal controller parameter configuration. Consequently, this results in lowering the expenditure of time, required for the optimization procedure. Finally, a suitable parameter configuration will improve the performance of the aim point control strategy of a solar power tower plant and thus increase the efficiency of the plant by maximizing the power on the receiver.

The approaches should be able to decide which sets of simulation data to use within the optimization process and which simulation data sets to discard, based on observation data, gathered from previous controller parameter evaluations. Furthermore, it is desirable to identify mismatches between simulation and reality and correct them in order to take advantage of partly unsuitable simulation data or simulation data with offsets.

To develop algorithms, which achieve these objectives, the fundamentals of the BO algorithm and its individual components, as well as its advantages compared to other hyperparameter optimization algorithms, will be elaborated and presented. Subsequently, existing approaches from literature, which aim to increase sample efficiency in BO by leveraging simulation data, will be reviewed and evaluated regarding the possibility of transferring it to the application of aim point controllers for solar power tower plants. Promising methods will be adapted to the application and extended to fulfill all relevant requirements, mentioned above. Finally, the developed approaches will be tested and analyzed to prove a possible enhancement in sample efficiency and number of required optimization steps for reaching a close optimal parameter configuration, in comparison to a standard BO algorithm without simulation data.

Chapter 2

Fundamentals and Literature Review

2.1 Solar Power Tower Plants

A solar power tower plant converts solar radiation to electricity via an energy conversion process or uses it to support thermochemical processes. The plant consists of several individual components, namely a heliostat field, a receiver, an energy conversion system and a storage module. The heliostats, which are a multitude of mirrors, focus the solar radiation onto the receiver which is mounted on top of the tower. The exact points on the receiver surface, where the heliostats aim to, are called aim points. These are determined by an aim point controller. The resulting concentrated radiation is absorbed by a heat transfer medium inside the receiver. The medium transfers the absorbed energy to an attached energy conversion process. The following sections will give a deeper insight into the system fundamentals of solar power tower plants and its individual components, as well as the aim point control strategy.

2.1.1 System Components

The system structure of a solar power tower plant and its individual components is depicted in fig. 2.1, which shows the heliostat field and the receiver, as well as the energy conversion process and the storage module.

Heliostat Field

A heliostat is an optical aperture, which is used to always direct the solar radiation to the same fixed point on the receiver, also called aim point, regardless of the change of the sun's position [9]. It consists of a reflecting surface, a supporting structure and a tracking mechanism, which guarantees that the heliostat always targets the same aim point. A multitude of usually identical heliostats are aggregated in concentric circles or in rows around the solar power tower plant and create a heliostat field. The heliostats as well as their arrangement are suspect to physical limitations and deficiencies, called optical losses, which divert the radiation reflected by them and thus lower the energy absorbed by the receiver. Examples for optical losses are

mirror errors and tracking errors. Mirror errors denote a form deviation of a mirror which can lead to a wider reflection image compared to an ideal mirror [9]. Tracking errors cause a deviation in the tracking of a heliostat which results in an offset of the reflected image onto the receiver [47]. Optical losses like tracking errors can vary over time, caused by dirt and abrasion of the heliostats. This is of great relevance for this work because it can considerably influence the tracking behavior and therefore result in the need of re-tuning the hyperparameters of the aim point controller, which directs the movement of the heliostats.

Receiver

The receiver, mounted on top of the tower, absorbs the reflected radiation by the heliostats and conducts the stored energy within the receiver material to a heat transfer medium via convection. The medium can be air, molten salt, water, liquid metals or particles [36]. Depending on the transfer medium, the receiver surface consists of different materials, such as metallic tubes for molten salt or porous ceramics for air [8]. Certain receiver characteristics, such as the corrosion limit as well as the maximum allowed thermal stresses of the receiver must not be exceeded, to prevent damages to the receiver material [49]. Those physical limitations impose restrictions on the intensity of the radiation on the receiver and thus on the allowed flux density. Here, the flux density ϕ is defined as the radiant flux Φ per area dA , which can be expressed as

$$\begin{aligned} \phi &= \frac{d\Phi}{dA}, \\ \text{with } \Phi &= \frac{dQ}{dt}, \end{aligned} \tag{2.1}$$

with the radiant flux Φ , as the radiant energy Q per time unit. The maximum limit of the allowed flux density for a certain receiver aperture can be calculated based on the receiver characteristics [9]. The resulting power P , that is fed into the energy conversion process by the heat transfer medium, can be calculated from integrating the flux density ϕ , measured at every point on the receiver aperture, over the receiver surface A as

$$P = \int_A \phi \, dA. \tag{2.2}$$

Energy Conversion Process

The heat transfer medium is fed into an energy conversion process. In thermal receivers, the energy conversion process may represent a conventional power plant process (e.g. a Rankine cycle to produce electricity), while in thermochemical receivers, the heat will be used for a chemical reaction (e.g. hydrogen production). Instead of being transferred to the energy conversion process, the medium can also be directed to an energy storage module for later use.

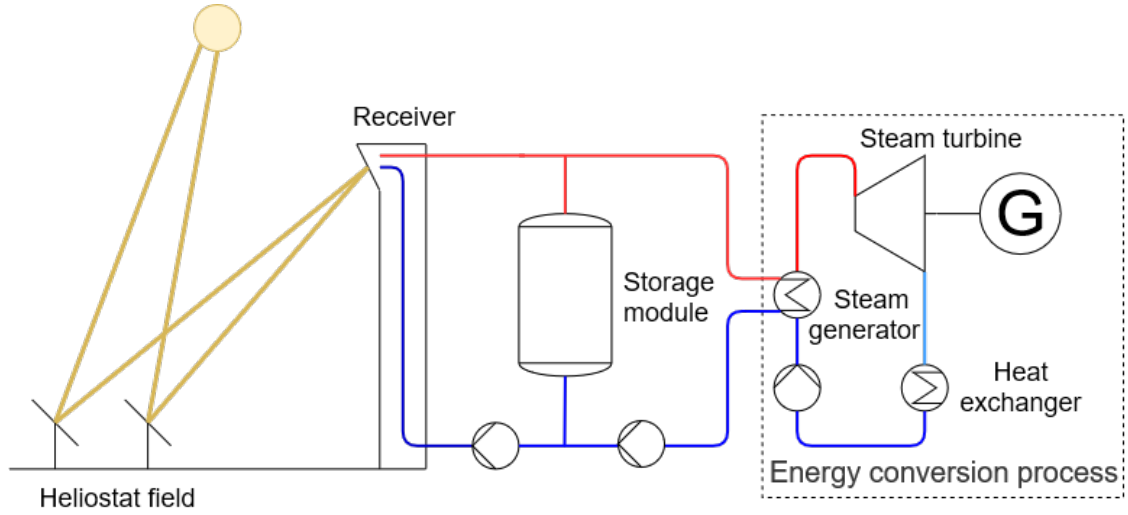


Figure 2.1: Simplified schematic structure of a solar power tower plant

2.1.2 Aim Point Control

The aim point controller optimizes the distribution of aim points on the receiver surface, which determines how much and where the radiation is concentrated. An optimal distribution of the aim points maximizes the power on the receiver. This increases the plant's efficiency, and thus makes the plant more economically attractive. However, at the same time the maximum limit of allowed flux density shall not be exceeded, to prevent damage to the receiver.

Control Objective

The objective of the controller is to maximize the power P_{receiver} on the receiver, while not violating the allowed flux density and being robust to disturbances, such as clouds. Therefore, the receiver surface is divided into sub-areas, called bins. Optimal heliostat aiming is then achieved when the flux density at each bin on the receiver is equal to the allowed flux density at the respective bin. However, since one heliostat correlates with many bins on the receiver, this is not a trivial task. It may even be impossible to find a solution where each bin value equals the allowed flux density, because the flux density on one bin can not be adjusted without changing it on other bins.

Consequently, the optimization problem of the control objective for a rectangular receiver can be formulated as

$$\begin{aligned}
 &\text{maximize} && P_{\text{receiver}} = \sum_i \phi_i \cdot dA \\
 &\text{subject to} && \phi_i \leq \phi_{\max,i}, \forall i = 1, \dots, n_{\text{bins}} \\
 &&& (x_{a,h}, y_{a,h}) \in \mathbb{R}^2, \forall h = 1, \dots, n_{\text{heliostats}}
 \end{aligned} \tag{2.3}$$

where ϕ_i is the flux density at receiver bin i , $\phi_{\max,i}$ the respective maximum allowed flux density and $(x_{a,h}, y_{a,h})$ the aim point coordinates for heliostat h . As stated

in eq. (2.3), the intercepted power on the receiver P_{receiver} can be calculated by multiplying the flux density with the bin area and sum them up for each bin.

Existing Aim Point Control Strategies

In literature, most developed aim point controllers are open-loop controllers, which face the requirements of minimizing the spillage and not violating the flux constraints. Here, the constrained optimization problem in eq. (2.3) can be solved with an optimization algorithm. An example would be a population-based optimization approach used by Belhomme et al. [10] or a modified population-based approach used by Maldonado et al. [35], constrained to a local neighborhood region of aim points. However, there also exists research on closed-loop aim point control. For instance, a controller introduced by Vant-Hull [57], is a simple method to prevent exceeding the allowed flux conditions. When violation is measured at one bin, the heliostat producing the greatest flux on the respective bin is identified and removed from tracking. Recent research also focuses on using popular control strategies like Proportional Integral Derivative (PID) [26] or Model Predictive Control (MPC) [27] for aim point optimization, which also covers the rejection of disturbances. However, a PID controller is usually used to control a single-input single-output system. For a solar power tower plant, the aiming control strategy of a field of heliostats, which is considered as a multiple-input multiple-output system, would have to be decoupled into several single-input single-output systems. This approach is only reasonable if the coupling between the inputs and outputs of the system is rather loose. Since a heliostat can influence several output variables, an approach with a decoupled system may not handle the interactions between inputs and outputs appropriately [27].

Controller Hyperparameters

Most control strategies have tunable hyperparameters, which have to be chosen beforehand and have an appreciable impact on the control behavior. For classic control approaches, like a PID controller, there exist heuristic hyperparameter tuning methods, like the popular Ziegler-Nichols method [61]. For other control strategies, choosing controller hyperparameters is a non-trivial task and results in a hyperparameter optimization problem, if there are complex dependencies between the controller's tunable parameters and its performance. Here, the performance function, also called objective function, can be defined as any function indicating the performance of an aim point controller. This could be a weighted sum of separate performance metrics, like power and violation of allowable flux conditions. Since the analytical expression of the objective function is usually unknown, the term black-box optimization problem is suitable. Therefore, obtaining objective function observations is only possible by observing the resulting performance of the aim point controller for various controller parameter configurations. This is a costly procedure, since it involves running the plant. Alternatively, simulations can be used to find an optimal parameter configuration for the aim point controller. In particular, the simulation, used in this work, leverages a ray tracing model to simulate optical effects like reflection and dispersion. However, deviations from expected performance

occur when there is a mismatch between simulation and reality due to modeling errors. To ensure optimal control behavior, the controller parameters have to be readjusted on the real plant. Here, manual controller parameter tuning by trial and error is a possible strategy, but often requires expert knowledge and is a time-consuming and therefore costly task. Moreover, there is no guarantee to actually find optimal parameter configurations when selecting them by hand. Therefore, the following section 2.2 gives an overview on most important black-box hyperparameter optimization algorithms to avoid parameter tuning by hand.

2.2 Overview on Black-Box Hyperparameter Optimization

Optimization algorithms automate the search for optimal parameters and are usually able to deliver better results in a less time-consuming way than manual hyperparameter tuning [58]. However, when having a black-box objective function, whose analytical expression is not known, traditional optimization algorithms, which rely on gradients or relaxations, are not suitable [55]. Therefore, this section focuses on black-box hyperparameter optimization and gives an overview on most important state-of-the-art approaches.

2.2.1 Grid Search and Random Search

Grid Search and Random Search are two of the most basic hyperparameter optimization approaches and belong to the category of gradient-free and model-free exhaustive search algorithms.

In Grid Search, the search space of each hyperparameter is discretized. Then, every possible combination of discretized parameters is evaluated. Theoretically, optimal parameters can always be found as long as sufficient computational resources are given.

In Random Search, a randomized search over hyperparameters is performed, by randomly selecting parameters to evaluate from an assumed probability distribution. The procedure continues until a previously specified time budget is exhausted, a desired accuracy or a performance value is reached. Compared to Grid Search, Random Search performs better if hyperparameters are not uniformly distributed within its search space and an assumption about the underlying distribution exists. Both algorithms implicate performing many function evaluations to yield a good parameter configuration, which makes them only suitable for functions which are computationally cheap and not time-consuming to evaluate. Information gathered from previous observations is not used in the optimization process, thus these approaches are not considered as sample efficient. [59]

2.2.2 Population-Based Algorithms

Population-based algorithms are adaptive meta-heuristic approaches, which include Evolutionary Algorithms and Swarm Intelligence Algorithms. These nature-inspired

algorithms intelligently exploit random search by using previously observed data to direct the search into promising regions of the search space. Because they are independent of gradients, they can tackle problems like objective functions, whose analytical expression is unknown, as well as non-continuous or non-differentiable functions. Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) are the two most important representatives of Evolutionary Algorithms and Swarm Intelligence Algorithms, respectively. Both approaches were successfully used in many applications and research areas. [15]

GAs are used to generate high-quality solutions for optimization problems [38]. They simulate the iterative process of natural selection. This means that within a population, only successful individuals will survive and contribute to the creation of the next generation. GAs consist of three main steps. Firstly, within an initialization step, a population, consisting of sample points, is randomly distributed over the search space. Then the optimization loop is executed. Here, the population is evaluated with a performance metric, that selects individuals with the highest scores. Subsequently, a reproduction step is performed in order to create the next generation of the population. The next generation results from best performing individuals and a random mutation in order to explore unknown areas of the search space. [59] The second popular approach within the class of population-based algorithms is PSO [15], an iterative algorithm based on the concept of social interaction and the intelligence and movement of swarms. Individual agents, called particles, constitute a swarm, which moves around the search space, looking for the best solution. Each particle tracks its positional coordinates, associated with the best solution so far found by the respective particle, as well as the coordinates associated with the globally best observed value so far, obtained by any other particle in the neighborhood of that particle. In every iteration, an updating rule trades off leading the swarm to the direction of the best observed value so far and further exploring of the search space. [29]

PSO is considered as more computational efficient in terms of speed and memory requirements compared to GAs [33]. Nevertheless, both GAs and PSO rely on a high number of function evaluations, and thus are not suitable for objective functions which are computationally costly or time-consuming to evaluate. Both approaches are not considered as sample efficient since they discard objective function observations, obtained in previous iterations [55].

2.2.3 Model-Based Algorithms

In contrast to previously outlined model-free methods, model-based search algorithms deploy a surrogate model which is fitted on observation data, gathered from former objective function evaluations. Based on the surrogate model, predictions for the optimum can be made and used within the optimization process. An advantage of model-based optimization is that prior beliefs about the objective function can be easily incorporated into the surrogate model. The most important representatives within the class of model-based hyperparameter optimization for black-box functions, are the BO algorithm and approaches using a Tree-Structured Parzen Estimator (TPE). In addition, there are several variants of the BO algorithm, com-

binning it with components from other optimization approaches to enable for example parallelization of the optimization process. [59]

BO is an iterative method that aims to find the global optimum of an objective function, whose analytical expression is usually not known, within a minimum number of trials [59]. It is considered as best-suited for optimization problems with less than 20 dimensions [25]. In BO, a surrogate model of the objective function is learned by fitting a probabilistic regression model, which is fast and computationally cheap to evaluate and robust to noisy observations [55]. For this purpose, usually a Gaussian Process (GP) regression model is used, which can be understood as a probability distribution over functions, conditioned on former objective function evaluations. Based on the model, a previously defined function, called acquisition function, decides which sample to evaluate next. Acquisition functions usually achieve a trade-off between exploitation of areas that look promising from previous function evaluations and exploration of unknown areas of the search space. In an iterative procedure, the acquisition function picks a new promising point from the search space, where the objective function will be evaluated next. Then the probabilistic model is refined with the newly gathered data, before the acquisition function again chooses the next point to evaluate. In this manner, BO takes advantage of information gathered throughout the whole optimization process. Therefore, it is considered as a sample efficient approach, which enables the algorithm to find nearly optimal parameter configurations in only a few iteration steps. This makes the algorithm particularly useful for optimization problems with objective functions that are costly to evaluate. [51]

Another popular model-based optimization approach is the TPE, which is a favored choice for hyperparameter optimization in machine learning [30]. The basic idea is similar to BO, but instead of building a probabilistic surrogate model, models with a graph structure are used. This technique is especially useful for problems with a graph structured configuration space, meaning that parameters are conditionally dependent on each other. An example would be a neural network architecture, where the number of neurons in one layer can only be set if this particular layer exists. Because TPE uses the history of previously evaluated hyperparameter configurations to sample the following ones, it is considered as sample efficient [11].

The tree structure can enable TPE to even outperform BO in dealing with conditional variables [56]. However, for applications which does not have the attribute of a tree-structured configuration space, the performance is questionable. Additionally, unlike Gaussian Processes, TPE is not able to model interactions between parameters [12].

Hence, BO outperforms other commonly used black-box hyperparameter optimization algorithms in many relevant criteria such as sample efficiency, having an objective function, which is computationally costly or time-consuming to evaluate, the assumption of obtaining possibly noisy observations, modeling interactions between hyperparameters and most importantly, the possibility of leveraging prior assumptions about the objective function. An explanation of the BO algorithm and its components in detail can be found in the following section.

2.3 Bayesian Optimization Fundamentals

The BO algorithm was first proposed by Jonas Mockus in a series of publications on global optimization in the 1970s and 1980s [42], [40], [41].

In BO, the goal is to find a vector of parameters \mathbf{x}^* , in the following also denoted as the optimal parameter configuration, which yields the optimum of an objective function $f(\mathbf{x})$. However, the objective function's explicit analytical expression, depending on the parameter vector \mathbf{x} , is usually not known. The general optimization problem can be stated as

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in A} f(\mathbf{x}), \quad (2.4)$$

where A denotes the set of feasible parameter vectors. Additionally, the function f is required to be continuous to make building a probabilistic model of the function applicable. Of course, it can also be desired to find an optimal parameter vector, yielding the minimum of an objective function. For that case, the maximization problem in eq. (2.4) holds true for simply maximizing the negative objective function.

In general, the BO approach consists of two main components. Firstly, fitting a statistical surrogate model, most commonly a GP model, and secondly, an acquisition function which decides where to sample next. In an iterative procedure, the acquisition function chooses a new promising vector of parameters, where the objective function will be evaluated next. The probabilistic model is then refined with the newly obtained objective function observation, before the acquisition function decides for the next parameter configuration to evaluate. Usually, it is assumed that an objective function value $f(\mathbf{x})$ differs from the observed function value y by additive Gaussian noise ϵ , which is independent and identically distributed with zero mean and a noise variance σ_ϵ^2 [46]

$$y = f(\mathbf{x}) + \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2). \quad (2.5)$$

2.3.1 Gaussian Process Regression

A GP is a multivariate normal distribution over functions, specified by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. Thus, the function values themselves are random variables. A GP is defined as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad (2.6)$$

with mean function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad (2.7)$$

and covariance function

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \quad (2.8)$$

The resulting distribution over functions is also called the GP prior distribution. In order to use a GP as a regression model, the predictive posterior distribution, conditioned on previously gathered objective function evaluations, is derived.

Deriving the Predictive Posterior Distribution

Supposing that there exists a vector of objective function evaluations \mathbf{y} for a set of n parameter vectors $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$, one can derive the predictive posterior distribution of the GP, conditioned on the vector of function observations. Calculating the predictive posterior leads to a normal distribution that can be completely described by a mean and covariance. This is only trackable, because the prior distribution was chosen to be Gaussian. With the posterior, it is possible to make predictions for any unevaluated parameter configuration \mathbf{x}_* in a set of unevaluated parameter vectors X_* . The predictive posterior can be written as

$$f(\mathbf{x}_*) | \mathbf{y}, X \sim \mathcal{N}(\mu_n(\mathbf{x}_*), \sigma_n^2(\mathbf{x}_*)), \quad (2.9)$$

with posterior mean μ_n and posterior variance σ_n^2 . To calculate mean and variance of the predictive posterior, it is necessary to set up the joint distribution of the vector of observations \mathbf{y} and the vector of objective function values \mathbf{f}_* at unevaluated test locations under the prior. [46]

The general definition of a joint Gaussian distribution of two random variables A and B is

$$\begin{bmatrix} A \\ B \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_A \\ m_B \end{bmatrix}, \begin{bmatrix} \text{cov}(A, A) & \text{cov}(A, B) \\ \text{cov}(B, A) & \text{cov}(B, B) \end{bmatrix}\right), \quad (2.10)$$

where m_A and m_B are the means and $\text{cov}(\cdot)$ denotes the covariance, for respective random variables. [53]

Transferred to the vector of objective function observations \mathbf{y} and function values \mathbf{f}_* at unevaluated test locations, the observation noise from eq. (2.5) has to be considered. From the independence assumption of the observation noise, it follows that

$$\text{cov}(\mathbf{y}, \mathbf{y}) = K(X, X) + \sigma_\epsilon^2 I, \quad (2.11)$$

with the covariance matrix K , generated by evaluating the covariance function $k(\cdot, \cdot)$ elementwise for all evaluated parameter vectors in the set X [46].

Thus, the joint distribution results in

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma_\epsilon^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \quad (2.12)$$

where I is the identity matrix. K denotes the covariance matrix, generated by evaluating the covariance function $k(\cdot, \cdot)$ elementwise for all evaluated parameter vectors in the set X and all unevaluated parameter vectors in the set X_* , respectively. Equivalently, \mathbf{m} and \mathbf{m}_* denote the vectors generated by evaluating the mean function $m(\cdot)$ for all parameter configurations in the sets X and X_* , respectively. [46]

By applying Bayesian statistics, the joint Gaussian prior in eq. (2.12) is conditioned on the observations, which yields the mean μ_n and covariance σ_n of the predictive posterior distribution $f(\mathbf{x}_*) | \mathbf{y}, X$ as

$$\mu_n(\mathbf{x}_*) = m(\mathbf{x}_*) + \mathbf{k}_*^T [K(X, X) + \sigma_\epsilon^2 I]^{-1} (\mathbf{y} - \mathbf{m}) \quad (2.13)$$

$$\sigma_n^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T [K(X, X) + \sigma_\epsilon^2 I]^{-1} \mathbf{k}_*, \quad (2.14)$$

with $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_1), \dots, k(\mathbf{x}_*, \mathbf{x}_n)]^T$. [46]

Thus, with eq. (2.13) and eq. (2.14), objective function predictions for an arbitrary unevaluated parameter configuration \mathbf{x}_* can be calculated. The predictive posterior mean from eq. (2.13) can be explained graphically. At evaluated parameter configurations, the predictive posterior mean will take values of the objective function observations, while at points far away from those observations, the predictive posterior mean will transition into the prior mean function.

The concept of GP regression is visualized in fig. 2.2. In fig. 2.2a, random function samples were drawn from a GP prior distribution with zero mean function and a squared exponential covariance function. The GP prior mean and covariance function will be further explained in the following two sections, respectively. In fig. 2.2b, random function samples were drawn from a GP posterior, conditioned on function evaluations from a sinus function. It was assumed that there is no observation noise.

Mean Function of the GP Prior Distribution

The prior mean function can incorporate any prior belief about the function to be modeled. For the case, that no prior assumption is available, most commonly a zero mean is chosen for the GP prior distribution [46]. If observation data from previous objective function evaluations exists, a simple and in most cases effective technique to customize the mean function, is to define a constant mean

$$m(\mathbf{x}) = c, \quad (2.15)$$

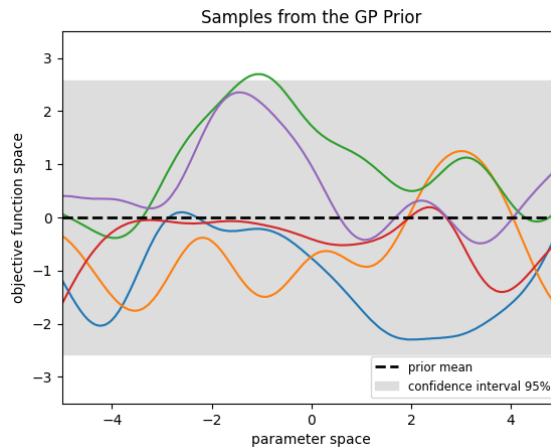
where c is chosen based on previous objective function observations [6].

Covariance Function of the GP Prior Distribution

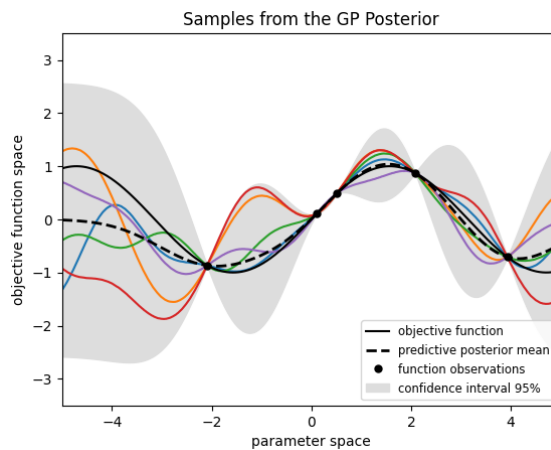
The covariance function $k(\mathbf{x}, \mathbf{x}')$ of the GP, also called kernel function, returns a modelled covariance between parameters \mathbf{x} and \mathbf{x}' [46]. Therefore, a distance metric is included in every kernel function, which usually determines the distance between two points in parameter space. In most kernel functions, the distance metric depends on an adjustable parameter, which controls the smoothness of the function. Kernels are typically defined, so that points close together in parameter space are considered as strongly correlated, whereas points farther away in parameter space have little correlation [25]. By choosing a specific kernel, it is possible to set prior assumptions about the objective function to model. Thus, the choice of the kernel function can significantly influence the GP regression. In general, there is a multitude of different kernel classes, which can be divided into two subcategories. Namely, there are stationary and non-stationary kernels, referring to the invariance to translations in the input space [51]. Stationary kernel functions depend only on the radial distance between points in some user-defined metric and are therefore shift invariant, as

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} + \mathbf{c}, \mathbf{x}' + \mathbf{c}), \quad (2.16)$$

with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{c} \in \mathbb{R}^n$.



(a) Random samples drawn from a GP's prior distribution with zero mean and squared exponential covariance function



(b) Random samples drawn from the GP's posterior distribution, conditioned on sinus function evaluations

Figure 2.2: Function regression with a Gaussian Process (GP) model

On the contrary, non-stationary kernels depend on the values of the input coordinates themselves, which implies more prior knowledge about the objective function. Thus, eq. (2.16) does not hold true for non-stationary kernel functions.

A widely used kernel function is the squared exponential (SE) kernel, which is a stationary kernel and thus shift invariant.

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right), \quad (2.17)$$

where σ_k^2 denotes the output variance, which can be chosen as the average distance of the observed function values from the prior mean. l is the length-scale parameter which determines the smoothness of the function. In other words, the length-scale determines how far two data points can lie away from each other to be still considered as close. In general, it is not possible to extrapolate more than l units away

from a data point [22]. However, for objective functions, where strong smoothness assumptions are unrealistic, it was argued that using a kernel function from other kernel classes, like the Matérn kernel, can be beneficial [54].

In fig. 2.3, the impact of the length-scale parameter choice on the regression model is visualized for a GP with zero prior mean function and SE kernel. Random samples from a GP with length-scale $l = 0.1$ were drawn and plotted in fig. 2.3a. In fig. 2.3b, the length-scale was changed to $l = 2$. In fig. 2.3c and fig. 2.3d function samples drawn from the GP posterior distribution, conditioned on function evaluations from a sinus function, are plotted. Again, it was assumed that there is no observation noise. For the case of a SE kernel, a large length-scale results in smooth functions. Especially from fig. 2.3c and fig. 2.3d it becomes apparent that the choice of the length-scale parameter can significantly influence the quality of the regression model, depending on the underlying objective function. The length-scale parameter $l = 2$ accounts for a model, which provides a better fit for the underlying sinus objective function than the length-scale parameter $l = 0.1$. Consequently, the smoothness parameter of a kernel function can be seen as a tunable hyperparameter of the GP. The following section will deal with the question of how to handle the GP's hyperparameters.

2.3.2 Handling Hyperparameters of the Gaussian Process

In previous sections, a GP regression model was derived. However, a question not answered yet, is how to determine the hyperparameters of a GP. This could be for instance a vector $\boldsymbol{\theta} = [c, l, \sigma_k^2, \sigma_\epsilon^2]$ with mean function constant c , kernel length-scale l , kernel output variance σ_k^2 and observation noise variance σ_ϵ^2 . The most commonly advocated approach to prior hyperparameter estimation is to fit the parameters to observed data [52]. This can be done by optimizing either the marginal likelihood or the unnormalized posterior. In both cases, the results are fixed point estimates for the hyperparameters in question and can be computed analytically for GP regression models [51]. It should be explicitly noted, that within this section, the term "hyperparameters" always refers to the GP hyperparameters $\boldsymbol{\theta}$.

Optimizing the Marginal Likelihood

Optimizing the marginal likelihood $p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta})$ with respect to the GP's hyperparameters $\boldsymbol{\theta}$, chooses values for $\boldsymbol{\theta}$, that make the GP prior distribution most likely to have generated a set of function observations \mathbf{y} for a set of evaluated parameter configurations $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$. Due to its practical success, this approach is a widely-used method [34].

The general optimization problem can be stated as

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} \log p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta}), \quad (2.18)$$

where the marginal likelihood is usually used in logarithmic scale for reasons of algebraic simplification. The term "marginal" refers to the fact that the unknown latent function f is marginalized out [51].

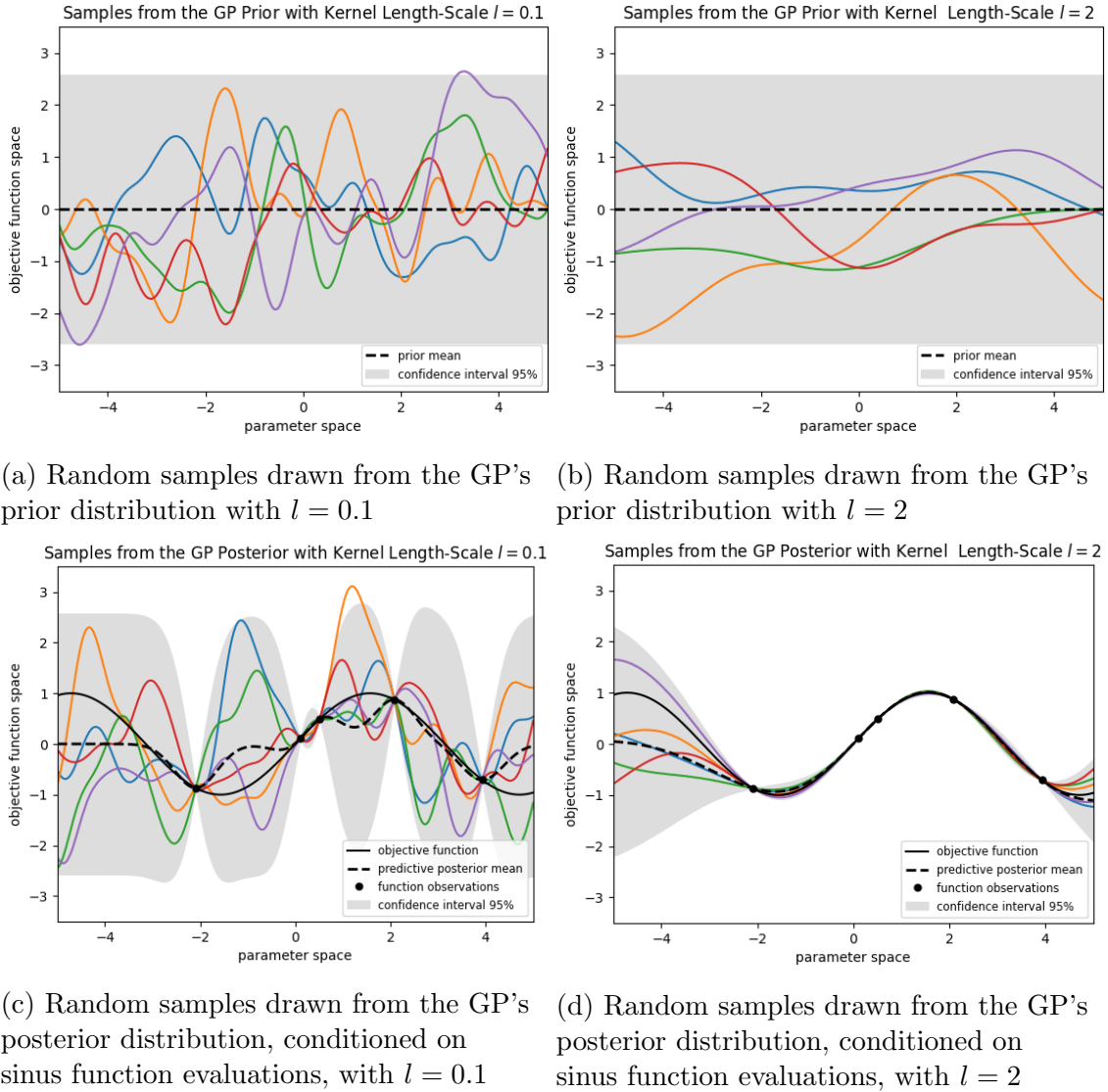


Figure 2.3: Impact of the length-scale parameter l on the Gaussian Process (GP) prior and posterior distribution, shown for two different values of the length-scale parameter

The likelihood function of a GP itself is Gaussian and therefore can be analytically expressed by

$$p_{\text{ML}}(\mathbf{y} | X, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^n |K_{\boldsymbol{\theta}} + \sigma_{\epsilon}^2 I|}} \exp \left(-\frac{1}{2} (\mathbf{y} - \mathbf{m}_{\boldsymbol{\theta}})^T (K_{\boldsymbol{\theta}} + \sigma_{\epsilon}^2 I)^{-1} (\mathbf{y} - \mathbf{m}_{\boldsymbol{\theta}}) \right), \quad (2.19)$$

where $\mathbf{m}_{\boldsymbol{\theta}}$ is the vector of prior mean function evaluations and $K_{\boldsymbol{\theta}}$ the covariance matrix for a set of evaluated parameter configurations X , given a certain vector of GP hyperparameters $\boldsymbol{\theta}$. n denotes the number of observations contained in X . Taking the logarithm of eq. (2.19) and performing some algebraic simplifications

yields the marginal log-likelihood of a GP as

$$\begin{aligned} \log p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta}) &= -\frac{1}{2}(\mathbf{y} - \mathbf{m}_{\boldsymbol{\theta}})^T (K_{\boldsymbol{\theta}} + \sigma_{\epsilon}^2 I)^{-1} (\mathbf{y} - \mathbf{m}_{\boldsymbol{\theta}}) \\ &\quad - \frac{1}{2} \log |K_{\boldsymbol{\theta}} + \sigma_{\epsilon}^2 I| - \frac{n}{2} \log(2\pi). \end{aligned} \quad (2.20)$$

[51] The parameters in question can now be determined by gradient-based optimization, like gradient descent.

Optimizing the Unnormalized Posterior

Optimizing the unnormalized prior with respect to $\boldsymbol{\theta}$, also known as the *a posteriori* approach, is particularly useful if there exists a prior belief $p(\boldsymbol{\theta})$ about the distribution of the hyperparameters. With $p(\boldsymbol{\theta})$ and the marginal likelihood $p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta})$, the posterior belief over $\boldsymbol{\theta}$, given observations X can be decomposed using Bayes' Rule as

$$p(\boldsymbol{\theta}|X) = \frac{p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(X)}. \quad (2.21)$$

The distribution $p(X)$ is usually not trackable but can be neglected since it is independent of $\boldsymbol{\theta}$ [25]. Discarding $p(X)$ yields the unnormalized posterior. Thus, the optimization problem results in

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} p_{\text{ML}}(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta}). \quad (2.22)$$

Fully Bayesian Approach

The point estimates, which are the result of the previously described approaches, cannot capture uncertainty, which plays a key role in guiding the exploration in BO. Therefore, another approach is frequently proposed in literature, namely the Fully Bayesian Approach or Fully Bayesian Gaussian Process Regression [32]. The main idea is to compute an integrated acquisition function \hat{a} , which is marginalized over all possible values of hyperparameters

$$\begin{aligned} \hat{a}(\mathbf{x}) &= \mathbb{E}_{\boldsymbol{\theta}|X}[a(\mathbf{x}, \boldsymbol{\theta})] \\ &= \int a(\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|X) d\boldsymbol{\theta}, \end{aligned} \quad (2.23)$$

where $a(\mathbf{x}, \boldsymbol{\theta})$ is the acquisition function, depending on the parameter vector \mathbf{x} and the vector of GP hyperparameters $\boldsymbol{\theta}$. More details about the acquisition function a and possible acquisition function choices will follow in the next section.

The resulting marginal acquisition function from eq. (2.23) now incorporates the uncertainty in $\boldsymbol{\theta}$ directly into the process of choosing the next parameter configuration per optimization iteration. This integral is typically intractable, but it is possible to approximate it as

$$\mathbb{E}_{\boldsymbol{\theta}|X}[a(\mathbf{x}, \boldsymbol{\theta})] \approx \sum_{i=1}^M \frac{1}{M} a(\mathbf{x}, \boldsymbol{\theta}_i). \quad (2.24)$$

To solve eq. (2.24), M samples, drawn from the posterior over GP hyperparameters $p(\boldsymbol{\theta}|X)$, are required. In practice, it is not possible to directly sample from this posterior. As stated before, $p(X)$ is usually not known, which makes $p(\boldsymbol{\theta}|X)$ (eq. (2.21)) not fully trackable. However, there are techniques like sequential Monte Carlo or Monte Carlo Markov Chains algorithms, which are able to produce a sequence of samples, that are approximately distributed according to $p(\boldsymbol{\theta}|X)$. Once the M samples are obtained, eq. (2.24) can be computed, by evaluating the acquisition function and averaging over all samples. [51]

2.3.3 Acquisition Function

This section explains the technique to select the next point in parameter space to evaluate. As already mentioned, in BO, this is done by deploying an acquisition function. This function uses the GP predictive posterior distribution to determine which point in parameter space looks most promising for optimizing the objective function. One of the main objectives of an acquisition function is to trade off the exploration of the search space and the exploitation of areas which were already discovered as promising. There is rich literature on selection strategies that use the posterior model to guide the iterative search. Two traditional approaches, the optimistic acquisition functions and the improvement-based acquisition functions, will be explained in detail. For simplicity, observation noise will be neglected.

Optimistic Acquisition Function

The guiding principle behind this class of acquisition functions is to be optimistic in the face of uncertainty. This means, that using an optimistic acquisition function for every new point \mathbf{x}_{n+1} to evaluate, corresponds to effectively using a fixed probability best case scenario according to the model. [51]

The Upper Confidence Bound (UCB) acquisition function is such a strategy, which is particularly popular for negotiating exploration and exploitation. For the aim of maximizing an objective function $f(\mathbf{x})$, it is defined as

$$a_{\text{UCB,max}}(\mathbf{x}) = \mu_n(\mathbf{x}) + \gamma\sigma_n(\mathbf{x}). \quad (2.25)$$

Whereas for the aim of minimizing an objective function $f(\mathbf{x})$, it changes to

$$a_{\text{UCB,min}}(\mathbf{x}) = -\mu_n(\mathbf{x}) + \gamma\sigma_n(\mathbf{x}). \quad (2.26)$$

Consequently, to select the next promising point \mathbf{x}_{n+1} in parameter space from the feasible set of parameter vectors A , the UCB acquisition function (for the aim of maximizing and minimizing an objective function) is maximized

$$\mathbf{x}_{n+1} = \arg \max_{\mathbf{x} \in A} a_{\text{UCB}}(\mathbf{x}), \quad (2.27)$$

where $a_{\text{UCB}}(\mathbf{x})$ is calculated based on n previous function evaluations. [51]

The first part of the sum in eq. (2.25) is the predictive posterior mean of the GP regression model. The second part of the sum is the standard deviation of the predictive posterior, weighted by a parameter γ . This parameter tunes the trade-off

between exploration and exploitation and has to be chosen manually. As an example, the following plot depicts a GP predictive posterior and three UCB acquisition functions for three values of γ , respectively. A small value for γ will result in exploitation of regions, proposed by the predictive posterior mean, whereas a high value for γ will account for more exploratory behavior. It can be seen that the UCB for $\gamma = 4$ has a different maximal value than the UCB for $\gamma = 1.2$. Respectively, different points would be chosen to be evaluated next.

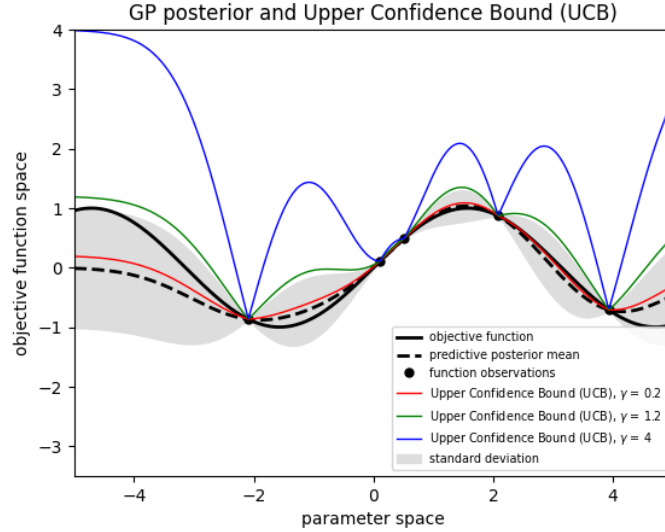


Figure 2.4: Gaussian Process (GP) posterior with Upper Confidence Bound (UCB) for different values of the parameter γ

Improvement-Based Acquisition Function

Improvement-based acquisition functions favor points that are likely to bring improvement, compared to the best observed value so far. Probability of Improvement (PI) and Expected Improvement (EI) are the most popular representatives of improvement-based acquisition functions.

PI chooses the next query point as the one, which has the highest probability of improvement. More precisely, it computes the probability that a point \mathbf{x} will lead to an improvement over the current best observed objective function value $f(\mathbf{x}^+)$ at the sample point \mathbf{x}^+ . Since the posterior distribution of the target values is Gaussian, the PI is trackable. For the aim of maximizing an objective function $f(\mathbf{x})$, the PI acquisition function is defined as

$$a_{\text{PI,max}}(\mathbf{x}) = P(f(\mathbf{x}) > f(\mathbf{x}^+)) = \Psi\left(\frac{\mu_n(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_n(\mathbf{x})}\right). \quad (2.28)$$

Ψ denotes the standard normal cumulative distribution function (CDF).

Analogously, for the aim of minimizing an objective function $f(\mathbf{x})$ the PI is defined as

$$a_{\text{PI,min}}(\mathbf{x}) = P(f(\mathbf{x}) < f(\mathbf{x}^+)) = \Psi\left(-\frac{\mu_n(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_n(\mathbf{x})}\right). \quad (2.29)$$

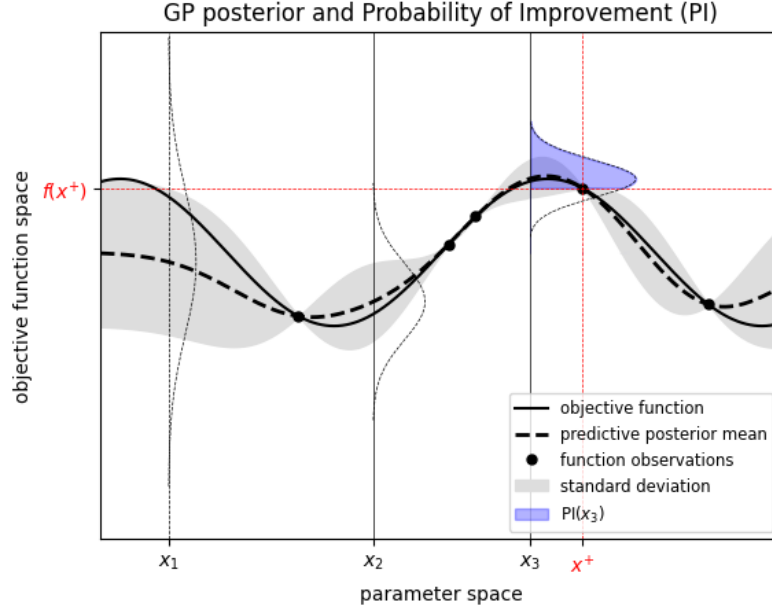


Figure 2.5: Visualization of the Probability of Improvement (PI) acquisition function

To select the next point \mathbf{x}_{n+1} in parameter space, the PI acquisition function (for the aim of maximizing and minimizing an objective function) is maximized

$$\mathbf{x}_{n+1} = \arg \max_{\mathbf{x} \in A} a_{\text{PI}}(\mathbf{x}), \quad (2.30)$$

where $a_{\text{PI}}(\mathbf{x})$ is calculated based on n previous function evaluations. [31]

In fig. 2.5 the main idea of the PI is visualized for three discrete example points in parameter space. At each point a Gaussian curve is placed with a mean, that equals the predictive posterior mean and a variance, that equals the predictive posterior variance of the GP at respective points. The blue shaded area under the curve gives the PI $f(x_3) > f(x_+)$ at x_3 . The PI at x_1 and x_2 is negligible.

The EI acquisition function can be seen as an enhancement of the PI. Other than only measuring the probability of improvement, the expected magnitude of improvement is estimated. For the aim of maximizing the objective function $f(\mathbf{x})$, EI is defined as

$$a_{\text{EI,max}}(\mathbf{x}) = \mathbb{E}[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)]. \quad (2.31)$$

The EI can be evaluated analytically under the GP model as

$$a_{\text{EI,max}}(\mathbf{x}) = \begin{cases} (\mu_n(\mathbf{x}) - f(\mathbf{x}^+))\Psi(Z) + \sigma_n(\mathbf{x})\psi(Z), & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}, \quad (2.32)$$

where Ψ and ψ are the CDF and probability density distribution (PDF), respectively. Z is defined as

$$Z = \begin{cases} \frac{\mu_n(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_n(\mathbf{x})}, & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}. \quad (2.33)$$

Analogously, the EI acquisition function is defined for the aim of minimizing an objective function $f(\mathbf{x})$ as

$$a_{\text{EI},\min}(\mathbf{x}) = \begin{cases} -(\mu_n(\mathbf{x}) - f(\mathbf{x}^+))\Psi(Z) + \sigma_n(\mathbf{x})\psi(Z), & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}, \quad (2.34)$$

$$\text{with } Z = \begin{cases} -\frac{\mu_n(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_n(\mathbf{x})}, & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}.$$

To select the next promising point \mathbf{x}_{n+1} in parameter space, the EI acquisition function (for the aim of maximizing and minimizing an objective function) is maximized

$$\mathbf{x}_{n+1} = \arg \max_{\mathbf{x} \in A} a_{\text{EI}}(\mathbf{x}), \quad (2.35)$$

where $a_{\text{EI}}(\mathbf{x})$ is calculated based on n previous function evaluations. [28]

From eq. (2.32), it is easy to see that EI will yield high probability scores when either the expected value of $\mu_n(\mathbf{x}) - f(\mathbf{x}^+)$, or the value $\sigma_n(\mathbf{x})$ is high. Here $\mu_n(\mathbf{x}) - f(\mathbf{x}^+)$ denotes the distance between the predicted posterior mean at the point \mathbf{x} in parameter space and the best observed objective function value so far. $\sigma_n(\mathbf{x})$ can be interpreted as the uncertainty around the point \mathbf{x} . Thus, the first term in eq. (2.32) can be considered as the exploitation term, the second addend controls the exploration.

In fig. 2.6, an example for a GP posterior with respective EI acquisition function is plotted, where the maximum value of EI determines the next point to evaluate.

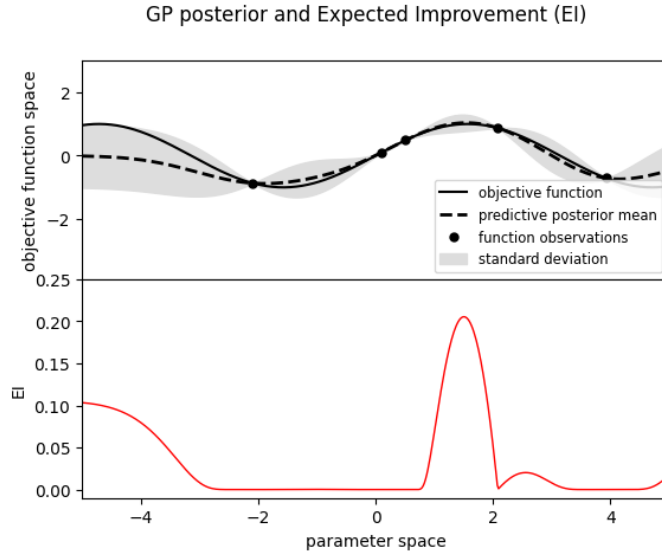


Figure 2.6: Gaussian Process (GP) posterior with Expected Improvement (EI) acquisition function

2.3.4 Bayesian Optimization Algorithm

Based on the theory of GP regression, acquisition functions and choosing hyperparameters for the GP prior distribution, the complete Bayesian Optimization algorithm can be set up. A basic pseudocode can be found in alg. 1. In the following, it

is assumed that the hyperparameters θ of the GP regression model are determined by fitting them to observed data, as it was explained in section 2.3.2.

The algorithm starts with the initialization of the GP prior, which involves choosing a mean and a covariance function. As an initialization step of the optimization, i objective function evaluations are performed. Usually, this is done by randomly sampling i points in parameter space. From the evaluated parameters and the obtained objective function observations, the set of evaluated parameter vectors X and the observed function values \mathbf{y} are initialized. Then a loop over a fixed number of iteration steps is performed, until a budget N of maximum objective function evaluations is exhausted.

Firstly, the GP is fitted on all available data by determining its parameters θ . Then the predictive posterior distribution is (re)calculated on all available data X , \mathbf{y} . Next, the posterior distribution is used to set up and optimize the acquisition function. The parameter configuration \mathbf{x}_{n+1} , that optimizes the acquisition function is selected and evaluated, which yields a new observation y_{n+1} . After incrementing the budget counter, the next iteration starts.

When budget N is exhausted, the parameter configuration \mathbf{x}^* is returned, which yields the best observed value \mathbf{y}^* . [25]

Three optimization iterations of the BO algorithm are visualized in fig. 2.7 for the objective of maximizing a sinus objective function with one-dimensional parameter space. The predictive posterior mean and the standard deviation, obtained from the predictive posterior variance, are depicted for the GP regression models. The EI acquisition function was used, where the highest acquisition function score indicates which point in parameter space will be evaluated next.

Algorithm 1: Bayesian Optimization

```

Initialize a GP with prior mean  $m(\mathbf{x})$  and kernel  $k(\mathbf{x}, \mathbf{x}')$ .
Perform  $i$  objective function evaluations. Set counter  $n = i$ .
Initialize  $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$  and  $\mathbf{y} = [y_0, \dots, y_i]$ 
while  $n \leq N$  do
    Fit the GP's hyperparameters  $\theta$  on  $X$ ,  $\mathbf{y}$ .
    Update the predictive posterior distribution, using  $X$ ,  $\mathbf{y}$ .
    Set up the acquisition function, using the current posterior distribution.
    Select the next point  $\mathbf{x}_{n+1}$  by optimizing the acquisition function.
    Add  $\mathbf{x}_{n+1}$  to  $X$ .
    Evaluate the objective function at  $\mathbf{x}_{n+1}$ , observe  $y_{n+1}$ 
    Add  $y_{n+1}$  to  $\mathbf{y}$ .
    Increment  $n$ .
end
Return parameter vector  $\mathbf{x}^*$  that yields best observed value  $\mathbf{y}^*$ .

```

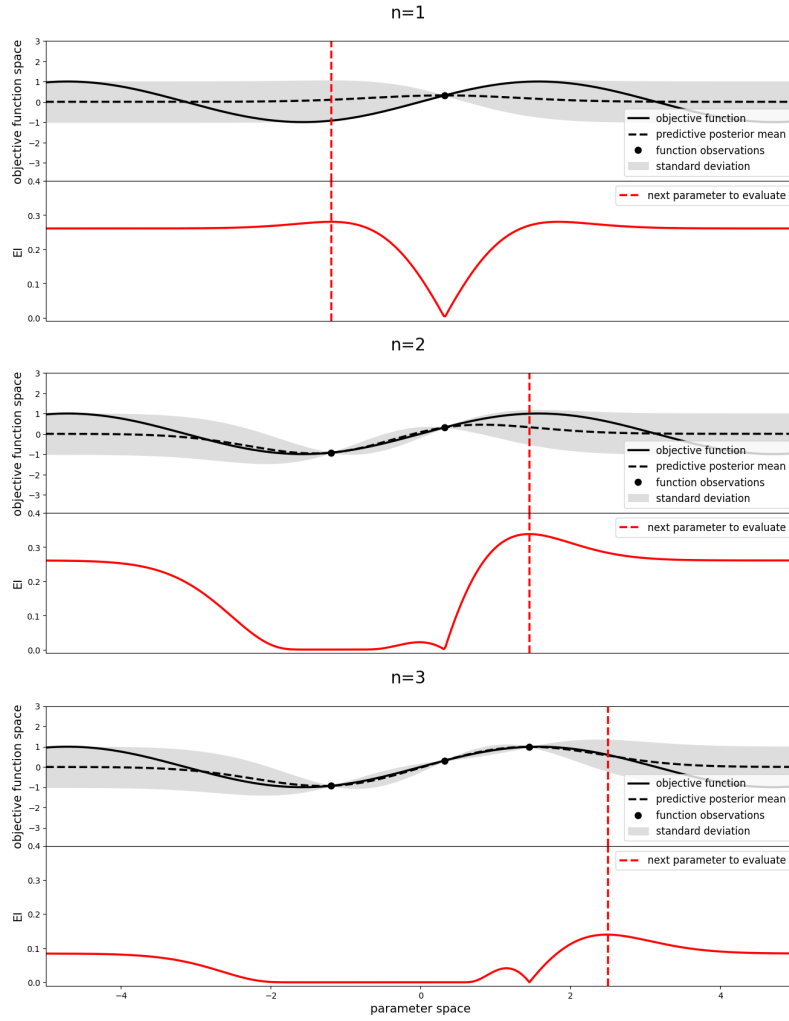


Figure 2.7: One-dimensional Bayesian Optimization for three optimization steps

2.4 Enhancing Sample Efficiency in Bayesian Optimization

In some applications, there is prior information available about the process to be optimized, for instance in the form of simulation data. Then the sample efficiency of the BO algorithm can be increased by incorporating the information as a prior belief into the GP model. In this context, sample or data efficiency means the amount of information the BO algorithm is able to use from observed samples. Increasing sample efficiency ideally results in decreasing the number of required optimization steps to reach optimal parameter configurations. As already mentioned in section 2.3, prior information can be utilized in BO to formulate a suitable mean function, as well as an appropriate kernel function for the GP prior distribution. Former research on this topic mainly stems from the field of model-based RL in robotics, where it is of crucial importance that a robotic agent profits from all available information in order to minimize number of trials and time to learn a certain behavior. The following sections provide an insight into existing approaches on incorporating sim-

ulation data into a Bayesian surrogate model. The approaches can be divided into creating either a prior-informed mean function or a prior-informed kernel function of the GP model. In case of dealing with possibly unsuitable or multiple sets of simulation data, it is necessary to choose or discard one or multiple models which contain prior information. Therefore, section 2.4.3 elaborates different model selection criteria from literature. Additionally, it is possible to use more than one source of prior information during one optimization step. This can be achieved by composing multiple models, which will be explained in section 2.4.4.

2.4.1 Prior-Informed Mean Functions

Defining a suitable mean function for the GP’s prior distribution is a possible strategy for incorporating prior knowledge into the surrogate model. The following three approaches from literature, use simulation data to create a so called prior-informed mean function.

Preselecting Promising Points in Simulation

Cully et al. [20] propose an approach that enables robots to adapt to damages, such as broken or missing legs. By tuning the hyperparameters of a parametrized controller, behaviors, that compensate those damages, are learned efficiently. Therefore, simulations are utilized to collect best performing parameters in simulation. Those are used to define a metric for the behavior of the robot. Controller, that result in desired behaviors, lie close together in behavior space, while a controller, that results in undesired behavior, lies further away. This behavior metric is incorporated into the GP model as the prior mean function and guides BO to quickly find controllers on hardware which can compensate the undesired behavior resulting from damages on the robot. However, it has to be considered that the search on hardware is then strongly limited to preselected successful points from simulation. This helps to make the search faster and safer on hardware. But if an optimal point was not preselected, BO can not sample it during optimization.

Tuning the Influence of the Simulation Data

In contrast to the previously described approach, Wilson et al. [2] consider the setting where the simulator is not an accurate model of the true domain and may not be trusted blindly. The main objective of the work proposed in [2], is to increase sample efficiency in model-based RL by efficiently retraining models and choosing a robot’s next actions to take, based on those models with BO. Whenever an action results in desired behavior, a function which evaluates the behavior gives back a high value, called reward. Accordingly, actions which result in undesired behaviors are poorly rewarded. Thus, a sequence of actions, called policy, should maximize the expected reward. To choose optimal policies, the system has to be modeled. Therefore, several actions taken by the robot, which are denoted as trajectories, are simulated. The simulated trajectory data D is then used to learn domain models of the robot. In this context, domain models are several RL specific functions, such as the reward function and the initial state distribution of the robot. By utilizing

domain models, the reward of an arbitrary policy $m(p, D)$ can be predicted by computing Monte-Carlo estimates of generated trajectory roll-outs. Here, the term trajectory roll-out means that an initial state is sampled from the learned initial state distribution and policy p is executed from there until termination. Since domain models are difficult to specify and learn, it is desired to efficiently refine them on new data. To find the next policy to evaluate, a GP is used to model the expected reward, which is used as the objective function to be optimized. Then, a BO algorithm finds the next optimal point in policy space, which is evaluated to consequently retrain the domain models efficiently. Within the GP, the estimated policy reward $m(p, D)$ is used as the prior mean function of the GP. The predictive distribution of the GP can be calculated following the procedure in section 2.3.1. By using $m(p, D)$ as the prior mean function, the predictive posterior mean, given a set D of n data points, results in

$$\mu_n(p_*) = m(p_*, D) + \mathbf{k}_*^T [K(X, X) + \sigma_\epsilon^2 I]^{-1} (\mathbf{y} - \mathbf{m}(p, D)), \quad (2.36)$$

where p_* is an unevaluated policy and $\mathbf{m}(p, D)$ the vector of Monte-Carlo estimates with an element for each policy in D . Wilson et al. assert, that the first part of this sum is the Monte-Carlo approximation of the expected reward, whereas the second part of the sum is the GP's prediction of the residual to the true underlying reward function, to be optimized.

For the case, that the domain models cannot be effectively approximated, the model-based estimates of the expected return may have a bad influence on the optimization procedure. An example would be the case where $m(p, D)$ underestimates the true mean function, resulting in zero expected improvement in areas of high deviations. Therefore, a new model of the expected reward is introduced, regulated by a parameter β , which controls the influence of the Monte-Carlo estimates of the expected return

$$f(p) = (1 - \beta)f_1(p) + \beta f_2(p). \quad (2.37)$$

Here the function f_1 is modeled by a GP as

$$f_1(p) \sim \mathcal{GP}(0, k(p, p')), \quad (2.38)$$

with zero mean. The second function f_2 is modeled by an additional GP as

$$f_2(p) \sim \mathcal{GP}(m(p, D), k(p, p')), \quad (2.39)$$

with the model-based mean $m(p, D)$, introduced above, and the same kernel as f_1 . The resulting distribution for $f(p)$ can be written as

$$f(p) \sim \mathcal{GP}(\beta m(p, D), d(\beta)k(p, p')), \quad (2.40)$$

with prior mean $\beta m(p, D)$ and a covariance computed by the kernel function $k(p, p')$ and a factor $d(\beta)$, which incorporates variance changes. The predictive posterior mean from eq. (2.36) consequently changes to

$$\mu_{n,\beta}(p) = \beta m(p, D) + \mathbf{k}_*^T [K(X, X) + \sigma_\epsilon^2 I]^{-1} (\mathbf{y} - \beta \mathbf{m}(p, D)). \quad (2.41)$$

In [2] the parameter β is determined by maximizing the marginal likelihood on gathered trajectory data D , as explained in section 2.3.2. The gradient of eq. (2.20) in section 2.3.2 can be taken with respect to β and simply solved for β , which yields

$$\beta^* = \frac{\mathbf{y}^T K^{-1} \mathbf{m}(p, D)}{\mathbf{m}(p, D)^T K^{-1} \mathbf{m}(p, D)}. \quad (2.42)$$

Optimizing the parameter β in this manner allows the model to control the impact of the prior-informed mean.

Parameterized Models

Another algorithm, called Black-DROPS, introduced by Chatzilygeroudis et al. [14], leverages prior information within the mean function of a GP model. The goal is to speed up model-based RL in robotics by being able to tune the parameters of a parameterized prior model, as well as select the most suitable model from multiple prior model. It can be seen as an enhancement to related approaches, proposed in [21] by Culter et al. and [50] by Saveriano et al.. It should be noted that Black-DROPS and related approaches do not use classic BO in their methods. In fact, they employ popular techniques for policy search in model-based RL but build their models with GP regression, which explains the relevance for this work. In Black-DROPS, the dynamics of a robotic system is modeled with a GP surrogate. It is stated that analytic equations of a robot's dynamics alone are usually not able to fully capture the system, for example in case of complex friction effects. Furthermore, analytic equations depend on robot specific parameters, which have to be determined in a model identification process beforehand. Therefore, a GP is used to model the analytically known part of the dynamics, depending on robot specific parameters and an additional part of the dynamics, which is analytically unknown. Here, the analytic equations for the dynamics are used as the prior mean function of the GP, depending on tunable robot specific parameters. These parameters can then be determined by maximizing the likelihood function of the GP, as described in section 2.3.2, which can be seen as the model identification procedure. In [14], it is stated that the marginal likelihood can also be used to choose between multiple parameterized models. More precisely, the model which yields the highest marginal likelihood on previously evaluated data is selected. Finally, the resulting GP model of the system is used in combination with popular techniques for model-based RL to find optimal control policies.

2.4.2 Prior-Informed Kernel Functions

Defining a customized kernel function is another possible way of incorporating prior knowledge into the GP model. The following approaches from literature define prior-informed kernels to use information from simulation data.

Balancing between Effort and Accuracy

Marco et al. [37] tunes the parameters of a control policy for a cart-pole system by taking a simulation as a complementary source of noisy data. The algorithm decides

whether to gather accurate real world observations, which are costly to obtain, or to use a simulator where evaluations are cheap to obtain but noisy. This trade-off is achieved by deploying an additive kernel structure.

Therefore, the vector of parameters \mathbf{x} to be optimized is extended by an additional binary variable δ , where $\delta = 0$ if the objective function is evaluated in simulation or $\delta = 1$ if it is evaluated on the physical system. With the extended vector of parameters $\mathbf{a} = [\mathbf{x}, \delta]$ an additive kernel is defined as

$$k(\mathbf{a}, \mathbf{a}') = k_{\text{sim}}(\mathbf{x}, \mathbf{x}') + k_{\delta}(\delta, \delta')k_{\text{err}}(\mathbf{x}, \mathbf{x}') \quad (2.43)$$

where the kernels $k_{\text{sim}}(\cdot, \cdot)$ and $k_{\text{err}}(\cdot, \cdot)$ model the objective function on the simulator and its difference to the objective function on the physical system, respectively. The kernel k_{δ} is defined as

$$k_{\delta}(\delta, \delta') = \delta\delta'. \quad (2.44)$$

Thus, $k_{\delta} = 1$ if two parameter configurations x and x' are both evaluated on the physical system, and $k_{\delta} = 0$ otherwise. Consequently, if either δ or δ' is zero, the covariance between controller parameters is only captured by k_{sim} . Effectively, the error covariance is not used in case of performing evaluations in simulation. On the contrary, in case of performing a parameter evaluation on the physical system, the kernel k_{error} is used and corrects the simulator assumptions with information from real observations. To enable BO to properly trade off between evaluating the simulator and the physical system, a so-called effort measure is introduced and included into the acquisition function. This effort measure is an estimator of the expected effort, such as the amount of time taken by a simulation relative to a physical experiment.

Behavior-Based Kernels

Another contribution from Wilson et al. [2] are behavior-based kernels (BBK) for RL in robotics. The objective is to choose an optimal policy for a robotic application by BO. However, Policies are often non-parametric. In the context of BO, this is problematic because standard kernels usually use a metric to determine the distance between two points in parameter space. Therefore, within a BBK a new distance metric is defined to determine the similarity of two policies, independent of any parameterization. The metric compares two policies regarding their resulting behavior on the robot. More precisely, the metric computes an estimate of a symmetric variant of the Kullback-Leibler divergence between trajectories, induced by two controllers. However, computing $k(p, p')$ for arbitrary policies would require an evaluation of every policy p and p' . Since an evaluation of every policy is considered as impractical, BBKs are suggested to be combined with previously described model-based Monte-Carlo policy roll-outs by Wilson et al. [2].

Employing Neural Networks

Antonova and Rai et al. [4], [45] optimize parameters of locomotion controllers for robotic applications with BO. They aim to increase sample efficiency in BO by

incorporating a Neural Network (NN) into their surrogate models, which was previously trained on simulation data. To collect simulation data, short simulator runs are performed. The resulting trajectories are summarized by a vector \mathbf{t}_x of several generic aspects of locomotion, like walking time or position of the robot's torso. With these trajectory summaries \mathbf{t}_x , collected for a range of controller parameters \mathbf{x} , a NN is trained, so that

$$f_{\text{NN}}(\mathbf{x}) = \hat{\mathbf{t}}_x, \quad (2.45)$$

where \mathbf{x} denotes the controller parameter configuration and $\hat{\mathbf{t}}_x$ is a so-called reconstructed trajectory summary. To train a NN on the trajectory summaries, can be interpreted as a feature transform. Subsequently, the reconstructed trajectory summaries can be incorporated into the kernel of a GP. They are utilized as the distance metric within the kernel, which considers the available prior information. In [4] a SE kernel is modified to include the reconstructed trajectory summaries as

$$\begin{aligned} k_{\text{NN}}(\mathbf{x}, \mathbf{x}') &= \\ &= \sigma_k^2 \exp \left(-\frac{1}{2} [f_{\text{NN}}(\mathbf{x}) - f_{\text{NN}}(\mathbf{x}')]^T \text{diag}(\mathbf{l})^{-2} [f_{\text{NN}}(\mathbf{x}) - f_{\text{NN}}(\mathbf{x}')] \right). \end{aligned} \quad (2.46)$$

By incorporating the reconstructed trajectory summaries, the stationary SE kernel is turned into a prior-informed and non-stationary kernel function, which is no longer invariant to shifts in parameter space. With the prior-informed kernel $k_{\text{NN}}(\mathbf{x}, \mathbf{x}')$, BO is biased towards promising regions in the space of the recovered trajectory summaries.

Correcting Mismatches

Antonova and Rai et al. [4] presented a prior-informed kernel which successfully increased sample efficiency in BO. However, mismatches between simulation and hardware can result in a considerable performance drop and can even prevent BO from finding optimal controller parameters. Therefore, Antonova and Rai et al. [45] propose an extended approach to prior-informed kernels to deal with mismatches between simulation and hardware. For the sake of simplicity, it is for now assumed that a trajectory summary t_{x_i} is one-dimensional.

Firstly, an additional GP will be used to model the deviations d_1, \dots, d_n between trajectory summaries t_{x_1}, \dots, t_{x_n} , which were previously evaluated on hardware, and the reconstructed trajectory summaries $f_{\text{NN}}(\mathbf{x}_1), \dots, f_{\text{NN}}(\mathbf{x}_n)$, obtained from simulation, for a set of controller parameters $\mathbf{x}_1, \dots, \mathbf{x}_n$. The deviations are calculated as

$$d_i = f_{\text{NN}}(\mathbf{x}_i) - t_{x_i} \text{ for } i = 1, \dots, n. \quad (2.47)$$

The GP, which models the deviations, is then defined as

$$d(\mathbf{x}) \sim \mathcal{GP}(0, k_{\text{SE}}(\mathbf{x}, \mathbf{x}')), \quad (2.48)$$

with zero mean function and SE kernel. Conditioning the GP prior distribution on previously observed deviations d_1, \dots, d_n will yield the GP predictive posterior

distribution. The predictive posterior mean $\mu_{n,\text{mis}}(\mathbf{x})$ from eq. (2.13) can now be used to predict mismatches between recovered trajectory summaries and hardware for arbitrary unevaluated parameter configurations. To extend this approach to multidimensional trajectory summaries $\mathbf{t}_{\mathbf{x}_1}, \dots, \mathbf{t}_{\mathbf{x}_n}$, the resulting multidimensional deviations $\mathbf{d}_1, \dots, \mathbf{d}_n$ are modeled with a multi-output GP. However, since this is not of importance for this work, further details on multi-output GPs will be omitted. The predicted mismatch $\mu_{n,\text{mis}}(\mathbf{x})$ is subsequently included in the prior-informed kernel function. This is done by extending the kernel with an additional dimension, as

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} f_{\text{NN}}(\mathbf{x}) \\ \mu_{n,\text{mis}}(\mathbf{x}) \end{bmatrix}, \quad (2.49)$$

$$k_{\text{NN},\text{mis}}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{1}{2}[\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}')]^T \text{diag}\left(\begin{bmatrix} l_1 \\ l_2 \end{bmatrix}\right)^{-2} [\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}')]\right), \quad (2.50)$$

with a diagonal matrix constructed from two independent length-scale parameters l_1 and l_2 . Introducing independent length-scale parameters for the simulated objective function values and the predicted deviations is necessary, because the scales can differ considerably. In case of multidimensional trajectory summaries, the length-scale parameters l_1 and l_2 turn into length-scale vectors \mathbf{l}_1 and \mathbf{l}_2 .

By using the $k_{\text{NN},\text{mis}}$ kernel function, the correlation between parameter configurations \mathbf{x} and \mathbf{x}' is determined by two components: the similarity in interpolated simulation data space and the predicted mismatch. Intuitively speaking, parameter configurations will only be considered as strongly correlated if they have similar simulated objective function values, as well as similar predicted mismatches. The benefit becomes clear from a short example. A controller may look promising in simulation, but result in undesired behavior, when tested on hardware. Then, all controllers, which are considered as "close" by the prior-informed kernel function without mismatch correction, will not be evaluated, even if they lie in different areas of the parameter space. However, when using the mismatch correction, the predicted mismatch for controllers in different areas of the parameter space will most likely not be equal. Thus, BO can still sample points that look promising in simulation, and are in a different region of the controller parameter space.

2.4.3 Bayesian Model Selection

Yet another question, which is not covered in most approaches, is how to deal with multiple sets of prior information. In this context, it is assumed that one set of simulation data will be incorporated into one GP model, respectively. A model \mathcal{M} will from now on refer to a GP prior, which contains information from one simulation data set. A model $\mathcal{M}|\mathbf{y}$ will refer to the respective GP posterior, conditioned on observations \mathbf{y} . Hence, the arising question is how to evaluate these models and select the one, which represents the underlying objective function best. In general, criteria to evaluate the fit of a probabilistic surrogate are called probabilistic model selection criteria. Since a GP is a Bayesian surrogate, the term Bayesian model selection is used. Criteria for Bayesian model selection were already proposed several decades ago. A traditionally used criterion is the marginal likelihood, for instance utilized

within the previously described BLACK-DROPS approach by Chatzilygeroudis et al. [14]. However, there is a strong polarization in literature regarding the marginal likelihood, motivating current research to reconsider the topic of Bayesian model evaluation and selection [34]. Therefore, this section outlines traditional model selection criteria, as well as a recently suggested approach to the Bayesian model selection problem.

Marginal Likelihood

The marginal likelihood p_{ML} was already introduced in section 2.3.2 in the context of choosing hyperparameters of the GP prior distribution, where it is used as a popular approach with great success. In addition, the marginal likelihood is also widely applied to model selection problems, where from multiple fitted models one particular model should be chosen, which is most likely to have generated a set of observed data points. This model will yield the highest marginal likelihood score, which is usually stated in logarithmic scale to simplify its analytic expression. The optimization problem results in

$$\mathcal{M}^* = \arg \max_{\mathcal{M} \in \mathcal{M}_1, \dots, \mathcal{M}_k} \log p_{\text{ML}}(\mathbf{y}|X, \mathcal{M}), \quad (2.51)$$

where the models $\mathcal{M}_1, \dots, \mathcal{M}_k$ are k GP priors. X is a set of n evaluated parameter vectors and \mathbf{y} the respective objective function observations. [46]

As already introduced in eq. (2.20), the marginal log-likelihood can be analytically expressed as

$$\begin{aligned} \log p_{\text{ML}}(\mathbf{y}|X, \mathcal{M}) = & -\frac{1}{2}(\mathbf{y} - \mathbf{m}_{\mathcal{M}})^T (K_{\mathcal{M}} + \sigma_{\epsilon}^2 I) (\mathbf{y} - \mathbf{m}_{\mathcal{M}}) \\ & -\frac{1}{2} \log |K_{\mathcal{M}} + \sigma_{\epsilon}^2 I| - \frac{n}{2} \log(2\pi), \end{aligned} \quad (2.52)$$

where $\mathbf{m}_{\mathcal{M}}$ and $K_{\mathcal{M}}$ refer to the mean function and the covariance matrix of a prior model \mathcal{M} , respectively. Since \mathcal{M} is a GP prior distribution, the marginal likelihood gives information about the likelihood of a prior model to have generated a set of data. However, usually it is desired to select a model that provides good predictions on unseen data. Lofti et al. [34] state, that a meaningful model selection criterion should therefore give information about the likelihood of the model's posterior $\mathcal{M}|\mathbf{y}_{\text{train}}$, conditioned on the set of training data, to have generated a set of withheld test data points. This would yield information about a model's predictive performance on unseen data. Hence, the marginal likelihood still provides a meaningful criterion for evaluating priors, but is generally not well-aligned with model selection.

Cross-Validation

Cross-Validation (CV) is one of the most commonly used methods in machine learning for evaluating the predictive performance of a model M . It is not a specific method for Bayesian model selection, however it can be transferred to Bayesian models. The method is based on splitting an available set of observation data

$Z = (X, \mathbf{y})$, consisting of n data samples, into disjoint parts. One part of the data, called training data $Z_{\text{train}} = (X_{\text{train}}, \mathbf{y}_{\text{train}})$, contains m data samples and is used for fitting each available model. The other part of the data, denoted as hold-out data $Z_{\text{hold-out}} = (X_{\text{hold-out}}, \mathbf{y}_{\text{hold-out}})$, contains $p = n - m$ data samples and is used to estimate the predictive performances of the models by the validation errors. The splitting ratio is optional and can be chosen by the user. However, a typical choice is for example to take 80 % of the available data for training and to hold out 20 % for testing.

Subsequently, the models in question are fitted on the training data set and the CV is computed on the withheld data. Finally, the model with the best overall performance is selected. [60]

The CV performance of a model M is evaluated by

$$CV_M(Z_{\text{hold-out}}) = \frac{1}{p} \sum_{i=1}^p s_M(Z_{\text{hold-out},i}), \quad (2.53)$$

where s_M is a scoring function for the model M . For the general evaluation of machine learning models, usually the squared distance between a withheld observation $y_{\text{hold-out},i}$ and the model's prediction $\hat{y}_i = M(\mathbf{x}_{\text{hold-out},i})$ is used as the scoring function. Logically, a small CV score denotes good performance of the model.

Delete-p Cross Validation To obtain a more stable assessment of a model's predictive performance, usually the average of multiple versions of data splittings, which is called the delete-p CV, is considered as

$$CV_{\text{delete-p}}(Z, p) = \frac{1}{|\mathcal{R}(p)|} \sum_{r \in \mathcal{R}(p)} CV_M(Z_{\text{hold-out}}(r)), \quad (2.54)$$

where $\mathcal{R}(p)$ is a collection of data splittings at the same splitting ratio with p withheld data samples and $r \in \mathcal{R}(p)$ denotes a specific splitting, producing the data sets $Z_{\text{train}}(r)$ and $Z_{\text{hold-out}}(r)$.

There are different types of the delete-p CV, regarding the choice of the data splittings \mathcal{R} .

Leave-p-Out Cross Validation One opportunity is to average over all possible data orders $|\mathcal{R}| = \binom{n}{p}$ for a constant splitting ratio, which is called the leave-p-out CV and is usually computationally expensive to calculate.

Alternatively, the delete-p CV can be calculated by choosing the splittings \mathcal{R} , so that $1 \leq |\mathcal{R}| \leq \binom{n}{p}$. This can be done by randomly choosing the $|\mathcal{R}|$ data orders, either with replacement or without replacement.

Monte Carlo Cross Validation In this context, with replacement means, that a withheld point can occur within the set of withheld data points for multiple data splittings. Choosing the data orders randomly, with replacement, yields the so-called Monte Carlo CV.

Repeated Learning-Testing and k-Fold Cross-Validation On the contrary, without replacement means, that a withheld data point can not be part of the test data set for more than one data splittings. Choosing the data orders randomly, without replacement, results in the so-called repeated learning-testing. The simplest version of repeated learning-testing is the k-fold CV, where the data is randomly partitioned into k equal-sized and mutually exclusive subsets. Which CV technique to use depends on the application.

In general, the Monte Carlo CV yields results with higher confidence and is more repeatable, compared to k-fold, since its variance is low. However, the Monte Carlo CV will have a higher bias than the k-fold CV, since some data points will be represented in more data splittings than others. [24]

Bayesian Cross-Validation The CV framework can be transferred to a Bayesian model selection problem by choosing a Bayesian scoring function. Since it is desired to evaluate the predictive performance of a GP posterior $\mathcal{M}|\mathbf{y}_{\text{train}}$, the predictive posterior probability p_{pp} will be used as a scoring function [24]. Logically, a high probability value and thus a large CV score will now denote good performance of the model. The predictive posterior probability on a set of withheld data for a GP is Gaussian, and can thus be derived analogously to the marginal likelihood in eq. (2.19). Instead of the prior mean and the kernel function, the posterior mean and posterior variances are used. Consequently, the predictive posterior probability of a Gaussian Process model \mathcal{GP} can be expressed as

$$\begin{aligned} \log p_{\text{pp}}(\mathbf{y}_{\text{hold-out}}|X_{\text{hold-out}}, Z_{\text{train}}, \mathcal{GP}) = & -\frac{1}{2}(\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_m)^T K^{-1}(\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_m) \\ & -\frac{1}{2} \log |K| - \frac{p}{2} \log(2\pi), \end{aligned} \quad (2.55)$$

where μ_m is the posterior mean (eq. (2.13)), conditioned on m training samples, and K is the matrix of posterior variances, generated by evaluating the predictive posterior variance $\sigma_m^2(\mathbf{x})$ (eq. (2.14)) for all withheld parameter configurations $X_{\text{hold-out}}$.

Cumulative Cross-Validation

Fong et al. [24] derive a novel Bayesian model selection criterion, based on a cumulative CV score. In [24], they first prove an equivalence of the marginal likelihood $p_{\text{ML}}(\mathbf{y}|X, \mathcal{M})$ of a Bayesian model with prior \mathcal{M} , and the cumulative leave-p-out CV, when using the predictive posterior probability from eq. (2.55) as a scoring function. This equality can be expressed as

$$\begin{aligned} \log p_{\text{ML}}(\mathbf{y}|X, \mathcal{M}) &= \sum_{p=1}^n CV_{\text{leave-p-out}}(Z, p) \\ &= \sum_{p=1}^n \frac{1}{\binom{n}{p}} \sum_{r=1}^{\binom{n}{p}} CV_M(Z_{\text{hold-out}}(r)), \end{aligned} \quad (2.56)$$

where Z again consists of the set X with evaluated parameter vectors and respective objective function observations \mathbf{y} . For simplicity, M denotes the GP predictive posterior model, conditioned on the respective training data splitting. Fong et al. state, that the representation of the marginal likelihood as a cumulative cross-validation score provides insight into the sensitivity to the prior \mathcal{M} . For instance, the last term of the summation in eq. (2.56) denotes the case $p = n$, where no training data is involved, and evaluates the model entirely on how well the prior \mathcal{M} is specified. In fact, 10% of terms contributing to the marginal likelihood come from predictions, using on average less than 5% of the available training data [24]. A criterion, which compares models with a score that includes contributions from predictions, made by using only a handful of training points, may not be suitable for model selection. Therefore, Fong et al. propose to begin evaluating the model performance after a preparatory phase, where for example 10% of the data is used for preparatory training, before testing. This leads to a preparatory cross validation score

$$PCV_{\text{leave-p-out}}(Z, P) = \sum_{p=P}^n CV_{\text{leave-p-out}}(Z, p), \quad (2.57)$$

which is a sum of terms containing none and few samples of training data and thus should at most be used for evaluating the prior \mathcal{M} .

The respective Bayesian cumulative leave-p-out CV score, which is recommended to be used for Bayesian model selection, the results in

$$CCV_{\text{leave-p-out}}(Z, P) = \sum_{p=1}^P CV_{\text{leave-p-out}}(Z, p), \quad (2.58)$$

where it is recommended to set P between $0.9n$ and $0.5n$.

Fong et al. provide test results, where both CV scores, namely the cumulative leave-p-out CV from eq. (2.58) and standard leave-p-out CV from eq. (2.54), averaged over all possible data orders, are able to determine the true model. However, the cumulative leave-p-out CV provides more distinct scores, while being computationally more expensive to calculate, than the standard CV.

2.4.4 Combining Models

Previously, possible techniques were introduced for choosing a model, that fits the underlying objective function best, based on observed data. However, in many applications, it is not guaranteed that the true model is included in a set of available models. In fact, in many cases the model, resembling the underlying function, lies somewhere in between available models. Therefore, it would be practical to choose k best performing models, based on their model selection scores, and use a composition of their information for selecting the next promising point in parameter space. This can be achieved by two methods, namely, by either creating a composed acquisition function, or a composed kernel function, which are presented in the following.

Composed Acquisition Function

Most Likely Expected Improvement Pautrat et al. [43] in general advocate the marginal likelihood as a suitable model selection criterion. However, they propose an approach where the marginal likelihood is combined with expected improvement scores within the acquisition function. In [43], Pautrat et al. generally aim to achieve a data-efficient policy search in RL for robotics by automatically selecting a suitable prior mean function for a GP. Thereby, it is assumed to have multiple priors to select from. To handle the available prior mean functions, they suggest combining the step of model selection and the optimization step of choosing the next promising point in parameter space. Therefore, they introduce a new acquisition function, based on the EI acquisition function. In contrast to the EI, this new acquisition function, called Most Likely Expected Improvement (MLEI), is able to choose a promising point in parameter space from all available GP models simultaneously. This is done by determining the EI scores of all available GP models, with respective prior means, and weighting them by their marginal likelihood scores. By optimizing the MLEI, the most promising point from all models is chosen. The MLEI acquisition function can be stated as

$$\begin{aligned} a_{\text{MLEI}}(\mathbf{x}, \mathcal{M}) &= \log p_{\text{ML}}(\mathbf{y}|X, \mathcal{M}) a_{\text{EI}} \\ &= \log p_{\text{ML}}(\mathbf{y}|X, \mathcal{M}) \mathbb{E}[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)] \end{aligned} \quad (2.59)$$

where a_{EI} denotes the EI acquisition function and \mathcal{M} a prior model.

Thus, the next point in parameter space is chosen by maximizing the product of marginal likelihood and EI scores, with respect to the parameter vector \mathbf{x} from the set of feasible parameter vectors A and the model \mathcal{M} from k prior models $\mathcal{M}_1, \dots, \mathcal{M}_k$. The resulting optimization problem of the MLEI acquisition function can be stated as

$$\mathbf{x}_{n+1} = \arg \max_{\mathbf{x} \in A, \mathcal{M} \in \mathcal{M}_1, \dots, \mathcal{M}_k} a_{\text{MLEI}}(\mathbf{x}, \mathcal{M}). \quad (2.60)$$

Utility Mean Roman et al. [48] performed an experimental study on adaptive kernel selection for BO, where they propose different approaches for choosing and utilizing GP models with varying kernels in BO. Their suggestions for choosing a model correspond to already introduced methods like the marginal likelihood or taking a MLEI acquisition function. However, they contribute new approaches on weighting multiple models within the acquisition function in each optimization iteration. Firstly, they introduce the so-called Utility Mean. In this method, a combination of multiple EI acquisition functions is used to select a new point in parameter space to evaluate. The next point will be chosen by optimizing not one single acquisition function, but the mean of EIs of all available models as

$$\begin{aligned} \mathbf{x}_{n+1} &= \arg \max_{\mathbf{x} \in A} a_{\text{UMEI}}(\mathbf{x} | \mathcal{GP}_{1, \dots, k}) \\ &= \arg \max_{\mathbf{x} \in A} \left(\frac{1}{k} \sum_{i=1}^k a_{\text{EI}}(\mathbf{x} | \mathcal{GP}_i) \right), \end{aligned} \quad (2.61)$$

where a_{EI} is the EI acquisition function, k is the number of GP models $\mathcal{GP}_1, \dots, \mathcal{GP}_k$ to be used in the Utility Mean a_{UMEI} .

Weighted Mixture Expected Improvement Another method, proposed by Roman et al. [48], is called Weighted Mixture a_{WMEI} and is based on the Utility Mean acquisition function. Other than only taking the average of acquisition functions from k GP models, they are additionally weighted by their marginal likelihood ratios as

$$\begin{aligned} \mathbf{x}_{n+1} &= \arg \max_{\mathbf{x} \in A} a_{\text{WMEI}}(\mathbf{x} | \mathcal{GP}_{1,\dots,k}) \\ &= \arg \max_{\mathbf{x} \in A} \left(\frac{1}{k} \sum_{i=1}^k w_i a_{\text{EI}}(\mathbf{x} | \mathcal{GP}_i) \right), \\ \text{with } w_i &= \frac{\log p_{\text{ML}}(\mathbf{y} | X, \mathcal{GP}_i)}{\sum_{j=1}^k \log p_{\text{ML}}(\mathbf{y} | X, \mathcal{GP}_j)}. \end{aligned} \quad (2.62)$$

The composed acquisition functions from eq. (2.61) and eq. (2.62) are used in [48] to weight information from multiple GP models with varying kernels. However, commonly used acquisition functions also incorporate information from the prior mean. Therefore, the Utility Mean and the Weighted Mixture could also be used to utilize multiple models with varying prior mean functions.

Composed Kernel Function

Duvenaud et al. [23] investigate different kernel structures, such as additive or multiplicative kernels, and propose using combined kernels in BO. They state, that kernels, composed by adding k sub-kernels, like

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + \dots + k_k(\mathbf{x}, \mathbf{x}'), \quad (2.63)$$

can be understood as an OR-operation between respective sub-kernels. Consequently, two points yield a high covariance value if at least one sub-kernel yields a high value. Analogously, kernels, composed by multiplying k sub-kernels, like

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot \dots \cdot k_k(\mathbf{x}, \mathbf{x}'), \quad (2.64)$$

can be understood as an AND-operation between respective sub-kernels. Thus, two points have a high covariance value only if all sub-kernels yield high covariance. Using either eq. (2.63) or eq. (2.64) as the kernel function of a GP, yields a GP model with combined prior information from both kernels. This model can then be further used by the acquisition function to select the next parameter configuration to evaluate.

Chapter 3

Bayesian Optimization with Multiple Simulation Data Sets

In this chapter, different algorithms are developed for leveraging simulation data within the BO framework and thus increasing sample efficiency in BO. Compared to previous research, the developed algorithms focus more on the problem of having multiple sets of simulation data, which are assumed as likely to have mismatches to the real objective function to be optimized.

The first section of this chapter will give more details on available simulation data and its properties. Then, three possible approaches of incorporating simulation data into the BO framework are selected and presented for the simple case of having a single set of simulation data. The approaches are, namely, the prior-informed mean function, the prior-informed kernel function and the Prior-Guided Expected Improvement (PGEI) acquisition function, which is a novel approach, introduced in this work. To expand the methods to the deployment of multiple simulation data sets, possible model selection criteria, as well as strategies to combine models, are assessed and adjusted regarding the requirements of the application covered in this work. Lastly, the resulting algorithms are summarized and complemented with pseudocode.

3.1 Interpolated Simulation Data

In the context of this work, simulation data refers to sets of discrete data points, previously obtained from a simulation of the aim point controller and its interaction with the solar power tower plant. One set of simulation data contains various controller parameter configurations and their corresponding objective function value, indicating the performance of the aim point controller. Since some simulation variables of solar tower plants, like mirror errors, rely on possibly inaccurate estimations, multiple simulation data sets for different values of the simulation variable in question are generated. To obtain objective function values for a continuous range of parameter configurations, every set of simulation data is interpolated by a regression NN, trained on the respective simulation data set. Certainly, other interpolation methods could be used as an alternative to training a NN. However, NNs are capable of representing even complicated relations in several dimensions

and thus appear most suitable for this application. From now on, $f_{\text{NN},i}(\mathbf{x})$ denotes the output of a NN for a controller parameter configuration \mathbf{x} , which interpolates the simulation data set i , where $i = 1, \dots, n_s$, for n_s available sets of simulation data.

3.2 Prior-Informed Mean Function

Customizing the prior mean function of a GP surrogate model is one possibility for the aim of incorporating prior information in BO. In section 2.4.1, approaches from literature for this purpose were introduced. One of the methods, proposed by Cully et al. [20], selects promising points in simulation. However, this approach seems not suitable for the application covered in this work, because it constrains the optimization to preselected points from simulation. In case of considerable mismatches between the true objective function and the simulation, which are likely to happen for the application, covered in this work, BO could be prevented from finding optimal controller parameters. Likewise, the Black-DROPS approach, proposed by Chatzilygeroudis et al. [14] is not convincing when transferred to the problem of controller parameter optimization for solar power tower plants. Although Black-DROPS generated promising results for a robot, transferred to other applications it may be unclear how to find a suitable parametric model, describing the system with analytic equations. For the case of having a ray tracing model, as assumed for the solar power tower plant, a closed analytic expression for the model does not exist. Finally, a prior mean function, weighted by a parameter, which controls the influence of simulation data on the model, as used by Wilson et al. [2], seems to be a promising approach. It considers the case of having simulation data, that possibly does not resemble the true objective function and can be transferred easily to applications different from the original robotics use case. For now, it is assumed that only one set of interpolated simulation data is available, which will be denoted as f_{NN} .

Analogous to [2], the objective function $f(\mathbf{x})$ will be described by a convex combination of two GP function surrogates, regulated by a parameter β

$$f(\mathbf{x}) = (1 - \beta)f_1(\mathbf{x}) + \beta f_2(\mathbf{x}), \quad (3.1)$$

where the function f_1 is modeled by a GP as

$$f_1(\mathbf{x}) \sim \mathcal{GP}(0, k_{\text{SE}}(\mathbf{x}, \mathbf{x}')). \quad (3.2)$$

As proposed in [2], a zero mean function is used for the first GP. As a covariance function, the SE kernel was selected, since it is most commonly used in literature and advocated as a suitable choice for most objective functions.

The second function f_2 is modeled by an additional GP as

$$f_2(\mathbf{x}) \sim \mathcal{GP}(f_{\text{NN}}(\mathbf{x}), k_{\text{SE}}(\mathbf{x}, \mathbf{x}')). \quad (3.3)$$

Here the prior mean function equals the values of the interpolated simulation data f_{NN} . According to [2], the same covariance function should be used for both GPs.

The linear combination of f_1 and f_2 in eq. (3.1) results in new GP, whose prior distribution is again defined by a mean function and a covariance function. The latter constitutes the covariance matrix of the prior distribution.

In general, the linear combination of statistically independent multivariate Gaussian random variables can be determined analytically. With two statistically independent multivariate random variables X and Y

$$\begin{aligned} X &\sim \mathcal{N}(\mathbf{m}_X, \Sigma_X) \\ Y &\sim \mathcal{N}(\mathbf{m}_Y, \Sigma_Y) \end{aligned} \quad (3.4)$$

the linear combination $Z = (1 - \beta)X + \beta Y$ can be stated as

$$\begin{aligned} Z &\sim \mathcal{N}(\mathbf{m}_Z, \Sigma_Z), \\ \text{with } \mathbf{m}_Z &= (1 - \beta)\mathbf{m}_X + \beta\mathbf{m}_Y \\ \text{and } \Sigma_Z &= (1 - \beta)^2\Sigma_X + \beta^2\Sigma_Y. \end{aligned} \quad (3.5)$$

Furthermore, an analytical expression for the covariance Σ_Z can also be found for the case, where the random variables are statistically dependent and jointly normally distributed, but then the correlations between X and Y have to be known. In case of statistical dependence, the mean \mathbf{m}_Z would stay the same as in eq. (3.5). [53]

However, instead of determining the resulting covariance function of the new GP analytically, Wilson et al. [2] again define the covariance by a SE kernel function. Moreover, they add an unknown multiplicative factor $d(\beta)$ to the kernel function, which indicates the variance changes, resulting from the use of the prior-informed mean function, dependent on β .

With the equations for the mean from eq. (3.5) and the weighted kernel function, the complete distribution for $f(\mathbf{x})$ assembles to

$$f(\mathbf{x}) \sim \mathcal{GP}(\beta f_{\text{NN}}(\mathbf{x}), d(\beta)k_{\text{SE}}(\mathbf{x}, \mathbf{x}')), \quad (3.6)$$

which is a GP with prior mean function

$$m(\mathbf{x}) = \beta f_{\text{NN}}(\mathbf{x}), \quad (3.7)$$

consisting of the β -weighted simulation data.

For the case of simulation data, resembling the objective function, β should take a value close to one. Consequently, the prior mean function will take the values of the interpolated simulation data f_{NN} . On the contrary, in case of having unsuitable simulation data, β should take a value close to zero to suppress its influence.

To obtain a GP regression model, the predictive posterior mean is calculated, according to section 2.3.1, by conditioning the GP on observed data. When using a prior mean function different from zero, the predictive posterior mean consequently changes to

$$\mu_{n,\beta}(\mathbf{x}_*) = (\beta f_{\text{NN}}(\mathbf{x}_*)) + \mathbf{k}_*^T [K(X, X) + \sigma_\epsilon^2 I]^{-1} (\mathbf{y} - \beta \mathbf{f}_{\text{NN}}), \quad (3.8)$$

where \mathbf{x}_* denotes an unevaluated parameter configuration, \mathbf{y} is a vector of objective function observations and \mathbf{f}_{NN} a vector of interpolated simulation data values with

one element for each parameter configuration \mathbf{x} in the set of evaluated parameter configurations X .

The equation for the predictive posterior variance remains the same as in eq. (2.14). However, the factor $d(\beta)$ will have an influence on the kernel function itself and therefore also on the predictive posterior variance. This topic will be further elaborated after handling the GP's hyperparameters.

3.2.1 Handling Hyperparameters

The resulting hyperparameters of the GP can be summarized in a vector $\boldsymbol{\theta}$, which includes the kernel parameters σ_k^2 and l (in case of using the SE kernel), as well as the parameter β and the noise variance σ_ϵ^2 . Wilson et al. suggest tuning the parameter β by maximizing the marginal likelihood of the GP on observed data. In that way, β is able to control the influence of the simulation data, based on gathered information about the true objective function. This corresponds to the most commonly used method of handling the hyperparameters $\boldsymbol{\theta}$ by optimizing the marginal likelihood on observed function evaluations, as previously explained in section 2.3.2. This approach is especially practical, since β can be optimized jointly with the other hyperparameters of the GP.

3.2.2 Pitfalls of the Prior-Informed Mean Functions

Wilson et al. [2] successfully assert improvements in sample efficiency and performance deriving from a β -weighted prior-informed mean function. However, when having a closer look at the approach, transferred to the optimization problem examined in this work, an issue will become apparent. In [2], it is assumed that there is a deviation between the output of a simulator and the objective function, modeled by the GP. However, in case of well suited interpolated simulation data, it is possible that the prior mean function resembles the observed objective function values well for several evaluations. The problem arising from this circumstance will be explained with the help of an example, where for simplicity no observation noise is assumed.

In the following, the SE kernel function is used

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right). \quad (3.9)$$

The GP's hyperparameters $\boldsymbol{\theta} = [l, \sigma_k^2]$ are fitted on objective function observations in every BO iteration, yielding the maximum marginal likelihood on observed data. It is assumed, that the interpolated simulation data f_{NN}^* is deployed, which resembles the true objective function f at n previously gathered objective function observations

$$f_{\text{NN}}^*(\mathbf{x}_i) \approx f(\mathbf{x}_i) \quad \forall i = 1, \dots, n. \quad (3.10)$$

Thus, it can be assumed that β will take a value close to one, which allows the prior mean function to approximately take the values of the interpolated simulation data

$$\begin{aligned} \text{with } m(\mathbf{x}) &= \beta f_{\text{NN}}^*(\mathbf{x}) \text{ and } \beta \approx 1 \\ \implies m(\mathbf{x}) &\approx f_{\text{NN}}^*(\mathbf{x}). \end{aligned} \quad (3.11)$$

In [2], it is stated, that the covariance function of the linear combination of GPs changes by a multiplicative factor $d(\beta)$, dependent on β . Therefore, the SE kernel function changes to

$$k_{\text{SE},\beta}(\mathbf{x}, \mathbf{x}') = d(\beta)k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = d(\beta)\sigma_k^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right), \quad (3.12)$$

with the kernel parameters σ_k^2 and l , which are contained in the vector of GP hyperparameters $\boldsymbol{\theta}$.

The factor $d(\beta)$ indicates, that using the prior-informed mean function, dependent on β , has an influence on the covariance of the GP prior distribution. This influence becomes visible for the kernel parameters, which incorporate the changes, after fitting them on observed data. For the case, that the prior mean function resembles the objective function observations, it was observed, that the kernel variance σ_k^2 takes values close to zero or zero, while the length-scale parameter l increases by a multiple of the search space. Both changes have the effect, that the kernel function takes values close to zero. **The vanishing covariance function is problematic in BO, because it also lets the predictive posterior variance from eq. (3.12) approach zero.**

A neglectable small predictive posterior variance will prevent explorative behavior in BO and will result in exhaustive sampling in areas, which are considered as optimal by the posterior mean. Furthermore, a predictive posterior variance equal to zero for all points of the search space, will even result in an EI of zero for respective points, which can be easily verified from the definition of the EI in eq. (2.32). This would prevent the acquisition function from choosing a new point to evaluate. In this case, the next parameter configuration would have to be chosen randomly to continue the optimization.

Visual Example: Degenerate Case of the Prior-Informed Mean Function

A visual example of the problem, stated above, is given in fig. 3.1 and fig. 3.2. In fig. 3.1, a NN was trained on random sinus function samples, to imitate interpolated simulation data. However, the simulation data is corrupted, since the true maxima of the function are not well represented.

Using the corrupted simulation data in the prior-informed mean function, yields the results, shown in fig. 3.2. Three BO steps are depicted, where a case without observation noise is considered. For every iteration, the number of function evaluations n , as well as the values for β and the kernel parameters σ_k^2 and l are stated.

In the first optimization step, the sinus objective function was already evaluated $n = 2$ times at randomly chosen points during the BO initialization step. The observations yield a good fit for the simulation data, as well as the true objective function, which leads to a β -value of almost one from maximizing the marginal likelihood. Consequently, the prior mean function approximately takes the values of the corrupted simulation data. In this case, where the objective function observations resemble the prior mean function, it can be observed, that the kernel variance σ_k^2 approaches zero, while the length-scale l takes a value, which is a multiple of the search space. Both of these kernel parameters lead to a vanishing covariance function, which again leads to a neglectable small posterior variance and accordingly small standard deviation. Therefore, the acquisition function solely relies on

the predictive posterior mean. As already mentioned in section 2.3.1, the predictive posterior mean takes values of the objective function observations at evaluated points and transitions into the prior mean function at locations far away from the evaluated points. Consequently, with a β -value close to one, the predictive posterior mean approximately resembles the corrupted simulation data, which does not contain the true function's maxima. Therefore, the acquisition function will select the maximum of the simulation data, which underestimates the true optimum of the objective function in the stated example.

In the next iteration $n = 3$, β is one, which leads to a prior mean equal to the simulation data. Again, the kernel parameters cause vanishing posterior variances. From the plot it can be seen, that in this iteration all EI acquisition function values become zero and the acquisition function is not able to select a new promising point. This is an indicator, that the posterior variance equals exactly zero for all points within the search space. Thus, a random point is evaluated, which is plotted in the optimization step $n = 4$.

At this new point, the simulation data is similar to the objective function observation, but has a small offset. Here, β takes the value 1.04 and the length-scale takes a reasonable value of $l = 0.408$. However, σ_k^2 is still quite small, which leads to small posterior variances greater than zero. The acquisition function also takes values greater than zero and selects a new point, but since the variance is still suppressed by σ_k^2 , any form of exploration is prevented. Therefore, a point in the immediate vicinity of the maximum within the simulation data is selected to be evaluated.

In further optimization steps, the observed behavior would be repeated until a point gets evaluated, which has a considerable offset to the prior mean. Then, the variance would increase, allowing further exploration and eventually finding the true optimum of the objective function. For the stated example in two dimensions, this could be achieved in a reasonable number of function evaluations. However, for objective functions with higher dimensionality, it possibly takes numerous function evaluations. Since the aim of this work is to increase the sample efficiency and thus reduce the number of function evaluations, the prior-informed mean could lead to counterproductive behavior.

This problem may be fixed by approaches like setting the kernel parameters to pre-defined values, instead of fitting them to observed data. However, this procedure requires prior domain knowledge and decreases the flexibility of the surrogate model. Therefore, the prior-informed mean function was not further investigated. Instead, two alternative methods of leveraging simulation data in BO will be presented in following sections.

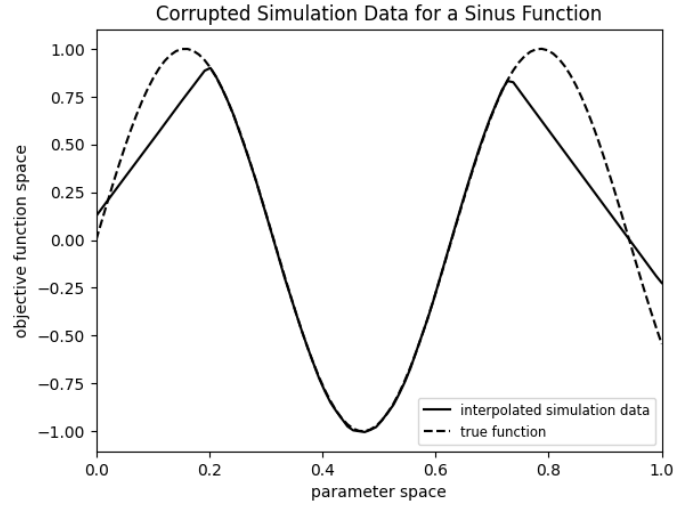


Figure 3.1: Corrupted simulation data for a sinus function

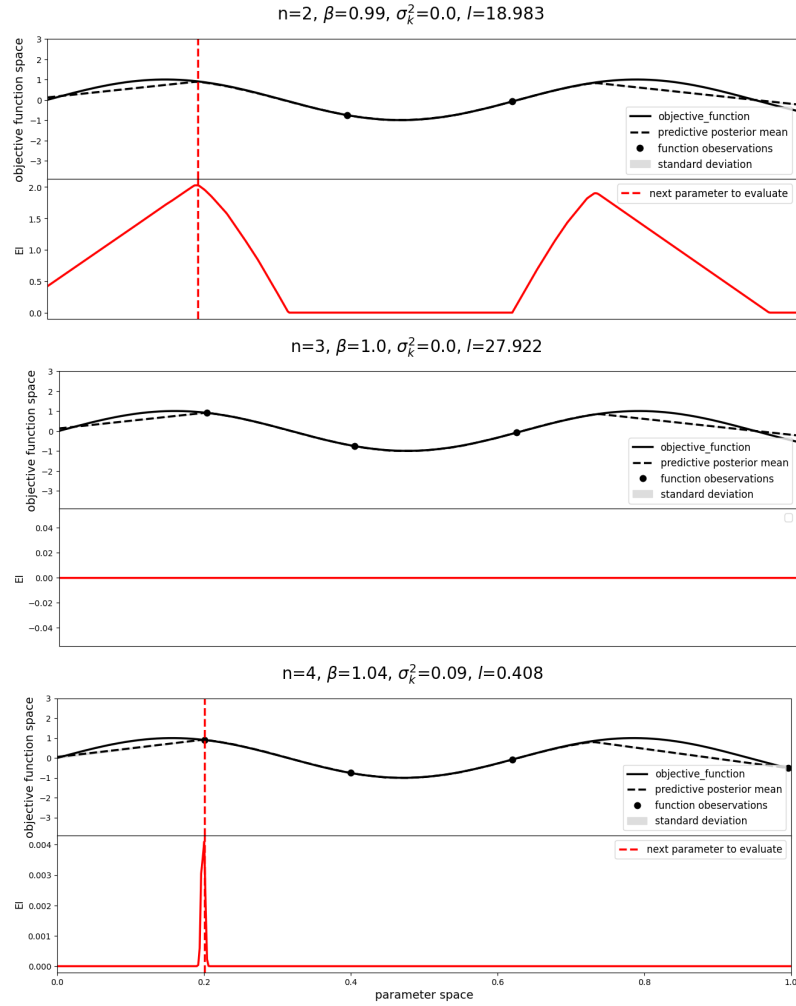


Figure 3.2: Degenerate scenario of Bayesian Optimization with prior-informed mean function and vanishing posterior standard deviation for three optimization steps

3.3 Prior-Informed Kernel Function

An alternative approach for incorporating prior information into a GP model, is to use a suitable prior-informed kernel function. In section 2.4.2, methods from literature were introduced, that use prior knowledge from simulations in their kernel functions. One of these algorithms, proposed by Marco et al. [37], trades off cost and accuracy by switching between a simulator and the real system, to obtain objective function observations. This method can be successful when simulations are not significantly different from the true system behavior. Otherwise, one needs to carefully tune the influence of simulation data over real world observations. Therefore, this approach seems not suitable for the application dealt with in this work, whose simulation data has possibly high deviations from the real objective function. In another contribution from Wilson et al. [2], a BBK kernel function is introduced, which aims to compute a distance between actions, taken by a robot, not in any parameter space, but regarding their resulting behavior on a robot. This method is very specific for robotic applications and rather hard to transfer to other optimization problems. The approach, suggested by Antonova and Rai et al. [4], also calculates the kernel's distance metric no longer in parameter space, but in simulated and interpolated objective function space. To reconstruct simulation data at unevaluated data points, they use NNs. Furthermore, they suggest an extension to their prior-informed kernel function, which corrects mismatches between simulation and reality, based on previous objective function observations. This method, which was already explained in section 2.4.2, will now be transferred to the problem of optimizing an objective function with respect to a vector of parameters \mathbf{x} , while having a possibly unsuitable set of simulation data as prior information.

Based on [4], a standard SE kernel function is turned into a prior-informed kernel by incorporating predictions of a NN, previously trained on available prior information. Evaluating the NN at any arbitrary parameter vector \mathbf{x} yields a predicted objective function value \hat{y} as

$$\hat{y}(\mathbf{x}) = f_{\text{NN}}(\mathbf{x}). \quad (3.13)$$

Consequently, the NN predictions are used as prior information within the modified kernel function and thus directly provide information about the correlation of parameter vectors in objective function space

$$k_{\text{NN}}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{(f_{\text{NN}}(\mathbf{x}) - f_{\text{NN}}(\mathbf{x}'))^2}{2l^2}\right). \quad (3.14)$$

Compared to a standard SE kernel, the prior-informed kernel function is no longer stationary. Assuming two pairs of parameter vectors $(\mathbf{x}_1, \mathbf{x}_1')$ and $(\mathbf{x}_2, \mathbf{x}_2')$, that have the same distance $d(\mathbf{x}_1, \mathbf{x}_1') = d(\mathbf{x}_2, \mathbf{x}_2')$ in parameter space, may have different distances $d(f_{\text{NN}}(\mathbf{x}_1), f_{\text{NN}}(\mathbf{x}_1')) \neq d(f_{\text{NN}}(\mathbf{x}_2), f_{\text{NN}}(\mathbf{x}_2'))$ in interpolated simulation data space. Thus, the kernel is no longer shift-invariant in parameter space, but uses specific information about the objective function from simulation data.

In [4], a zero mean function is proposed as the GP's prior mean. However, as

already explained in section 3.2, a constant mean function $m(\mathbf{x}) = c$, with constant c fitted on objective function observations is beneficial and will be used instead of the zero mean. The resulting GP surrogate model of the objective function with the prior-informed kernel can be written as

$$f(\mathbf{x}) \sim \mathcal{GP}(c, k_{\text{NN}}(\mathbf{x}, \mathbf{x}')). \quad (3.15)$$

Finally, the mean and variance of the GP predictive posterior distribution can be calculated without any modifications by eq. (2.13) and eq. (2.14), respectively.

3.3.1 Handling Hyperparameters

Fitting the hyperparameters θ of a GP to objective function observations by maximizing the marginal likelihood is a convenient technique, which yields in general suitable results and will therefore also be used to determine the hyperparameters of a GP with prior-informed kernel function.

However, when using the prior-informed kernel, it should be considered to introduce a fixed upper bound for the lengths-scale l , before fitting the parameter to observed data. This is particularly useful when having simulation data, that does possibly not or only partly resemble the objective function observations. For this case, objective function observations will be gathered, which are unlikely to be drawn from the GP prior distribution with the prior-informed kernel function. Consequently, the marginal likelihood will decrease. However, sometimes a comparatively high marginal likelihood can still be achieved by smoothing the GP model with a disproportional large length-scale and thus suppressing the influence of the simulation data within the kernel. This is problematic, since all unevaluated points are considered as close to the evaluated ones, which again yields a vanishing posterior variance and prevents further exploration of the search space. Moreover, it also corrupts the model at unevaluated points in parameter space, where the simulation data information could possibly still be useful. An upper length-scale bound can prevent this undesired behavior. When there is no further reliable prior knowledge about the true objective function available, the maximum distance within the domain, which is used in the kernel's distance metric, can be chosen as the upper bound for l . For the prior-informed kernel function, the length-scale bound would be chosen as the maximum distance of objective function values within the simulation data space. This is a suitable choice, since it will not be necessary to extrapolate more than this maximum distance away from any data point.

3.3.2 Mismatch Correction

Compared to a standard kernel function, the prior-informed kernel can yield a more accurate GP model of the underlying objective function by accessing prior knowledge, and thus accelerate the BO algorithm in finding the optimum. However, this is only the case if the set of interpolated simulation data f_{NN} , used in the prior-informed kernel, resembles the objective function well. For the case of using unsuitable simulation data within the prior-informed kernel function, the resulting

GP model will possibly not yield a good fit for the objective function and thus decrease the performance of BO. Therefore, the previously proposed prior-informed kernel function eq. (3.14) will be extended by a mismatch correction term, based on the approach of Antonova and Rai et al. [45].

For this purpose, an additional GP will be used to model the deviations between previously collected objective function observations $\mathbf{y} = [y_1, \dots, y_n]$ for a set of evaluated parameter configurations $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ and the interpolated simulation data values $\mathbf{f}_{\text{NN}} = [f_{\text{NN}}(\mathbf{x}_1), \dots, f_{\text{NN}}(\mathbf{x}_n)]$. The vector of observed deviations can be written as

$$\mathbf{d} = [d_1, \dots, d_n], \quad (3.16)$$

with $d_i = f_{\text{NN}}(\mathbf{x}_i) - y_i$ for $i = 1, \dots, n$.

The GP, modeling the deviations is then defined as

$$d(\mathbf{x}) \sim \mathcal{GP}(c, k_{\text{SE}}(\mathbf{x}, \mathbf{x}')), \quad (3.17)$$

with standard SE kernel and constant prior mean function.

Conditioning the mismatch GP prior distribution on previously observed deviations \mathbf{d} will yield the GP posterior distribution with predictive posterior mean $\mu_{n,\text{mis}}(\mathbf{x})$ according to eq. (2.13). The predictive posterior mean can then be used to predict mismatches between interpolated simulation data and arbitrary unevaluated parameter configurations.

The predicted mismatch can be directly included in the prior-informed kernel function. This is done by extending the kernel k_{NN} from eq. (3.14) with an additional dimension to

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} f_{\text{NN}}(\mathbf{x}) \\ \mu_{n,\text{mis}}(\mathbf{x}) \end{bmatrix}, \quad (3.18)$$

$$k_{\text{NN},\text{mis}}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{1}{2}[\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}')]^T \text{diag}\left(\begin{bmatrix} l_1 \\ l_2 \end{bmatrix}\right)^{-2} [\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}')]\right), \quad (3.19)$$

with a diagonal matrix constructed from two independent length-scale parameters l_1 and l_2 . Introducing independent length-scale parameters for the simulated objective function values and the predicted deviations is beneficial because its scales can differ.

By using the prior-informed kernel function $k_{\text{NN},\text{mis}}$ with mismatch correction, the correlation between parameter configurations \mathbf{x} and \mathbf{x}' is determined by two components: the similarity in interpolated simulation data space and the predicted mismatch. Intuitively speaking, parameter configurations will only be considered as strongly correlated if they have similar simulated objective function values, as well as similar predicted mismatches.

This can be shown mathematically by reshaping eq. (3.19) to

$$\begin{aligned} k_{\text{NN},\text{mis}}(\mathbf{x}, \mathbf{x}') &= \\ &= \sigma_k^2 \exp\left(-\frac{(f_{\text{NN}}(\mathbf{x}) - f_{\text{NN}}(\mathbf{x}'))^2}{2l_1}\right) \cdot \sigma_k^2 \exp\left(-\frac{(\mu_{n,\text{mis}}(\mathbf{x}) - \mu_{n,\text{mis}}(\mathbf{x}'))^2}{2l_2}\right). \end{aligned} \quad (3.20)$$

The result is a multiplicative kernel structure, where the first part of the multiplication is a sub-kernel, which determines the covariance by computing a distance in

parameter space, while the second part of the multiplication is a sub-kernel, which calculates the covariance by computing a distance in predicted mismatch space. As previously stated in section 2.4.4, a multiplicative kernel structure equals an AND-operation between sub-kernels, which confirms the statement, that parameter configurations will only be considered as strongly correlated if they have similar simulated objective function values, as well as similar predicted mismatches.

The resulting benefit can be seen in the following example. It is assumed, that parameter configuration \mathbf{x}_1 was already evaluated and yielded bad performance. A second parameter configuration \mathbf{x}_2 lies in another region of the parameter space, but has a similar predicted objective function value in simulation data space. Without using a mismatch correction, \mathbf{x}_2 will not be selected and evaluated, even if the point could still be promising. However, with mismatch correction, \mathbf{x}_2 can still be selected, if the predicted mismatch for \mathbf{x}_2 yields another value than the predicted mismatch for \mathbf{x}_1 , which is likely for points in different areas of the parameter space.

3.4 Prior-Guided Expected Improvement

Within this work, a new method for leveraging prior information from simulation data in BO is introduced. This method is fundamentally different from previously described approaches. Instead of incorporating a set of interpolated simulation data into the GP model, it is used to directly guide the EI acquisition function towards promising areas in simulated objective function space. In the broadest sense, this approach is inspired by the prior-informed mean function by Wilson et al. [2], while overcoming the limitation of a vanishing posterior variance. Instead of incorporating the simulation data into the prior mean function of the GP model, the idea is to add them to the posterior mean function $\mu_n(\mathbf{x})$ (eq. (2.13)) which is then used within the acquisition function.

Firstly, the posterior mean is calculated from the GP model and adjusted to

$$\mu_{n,\alpha}(\mathbf{x}) = \mu_n(\mathbf{x}) + \alpha f_{\text{NN}}(\mathbf{x}), \quad (3.21)$$

with the parameter α , controlling the influence of the interpolated simulation data $f_{\text{NN}}(\mathbf{x})$.

Then, within the acquisition function, the original posterior mean $\mu_n(\mathbf{x})$ is substituted with the adjusted posterior mean $\mu_{n,\alpha}(\mathbf{x})$, which results in guiding the search towards promising areas of the interpolated simulation data space. Other than the prior-informed mean in section 3.2, there is no risk of a vanishing covariance if the adjusted posterior means equals or resembles observed objective function evaluations. The EI acquisition function with modified predictive posterior mean (for the aim of maximizing an objective function) yields

$$a_{\text{PGEI}}(\mathbf{x}) = \begin{cases} (\mu_{n,\alpha}(\mathbf{x}) - f(\mathbf{x}^+))\Psi(Z) + \sigma_n(\mathbf{x})\psi(Z), & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}, \quad (3.22)$$

$$\text{with } Z = \begin{cases} \frac{\mu_{n,\alpha}(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_n(\mathbf{x})}, & \text{if } \sigma_n(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma_n(\mathbf{x}) = 0 \end{cases}.$$

The PGEI will be likely to favor points, which are promising in simulated objective function space. For the case, that the interpolated simulation data and the original predicted posterior mean μ_n are contradictory, the PGEI will yield a medium score, allowing to first evaluate promising points, where the original model and the interpolated simulation data agree. Moreover, it has to be considered, that a modification of the posterior mean also affects points of the regression model, where objective function observations already exist. For other acquisition functions, like UCB, this can lead to the undesired behavior, that the modified posterior mean of a point, which was already evaluated, again results in a higher acquisition function score than unevaluated points and therefore is again selected to be evaluated. However, for the EI acquisition function this is unlikely to happen, since the posterior variance at those points vanishes, which yields vanishing EI scores, as it can easily be seen from eq. (3.22).

3.4.1 Determining the Weight Parameter

Wilson et al. [2] suggest determining the weight parameter, which controls the influence of the simulation data, by maximizing the marginal likelihood on observation data. By doing so, the probability that the observation data was drawn from the GP's prior distribution is maximized. This seems reasonable, since they modify the prior mean function of the GP. However, for adjusting the acquisition function, it is desired to modify the posterior mean of the GP. Thus, maximizing the marginal likelihood is not a suitable approach. In fact, the weight parameter α should rather maximize the probability that some withheld observation data samples are drawn from the posterior distribution of the GP. This can be achieved by using the predictive posterior probability p_{pp} , which was introduced in section 2.4.3, eq. (2.55). Therefore, the set $Z = (X, \mathbf{y})$ of evaluated parameter configurations and respective objective function observations is separated into disjoint data sets. $Z_{\text{train}} = (X_{\text{train}}, \mathbf{y}_{\text{train}})$ will denote m data points on which the GP's prior distribution will be conditioned on. The data split ration can be chosen freely. However, a suitable approach is to take 80 % of the available data for training and to hold out the remaining 20 % for testing.

Subsequently, the predictive posterior mean $\mu_m(\mathbf{x})$ of the GP and the predictive posterior variance $\sigma_m^2(\mathbf{x})$ can be computed, based on the m training points. $Z_{\text{hold-out}} = (X_{\text{hold-out}}, \mathbf{y}_{\text{hold-out}})$ will be the set of p withheld data points.

Maximizing the predictive posterior probability, to obtain the parameter α^* yields an optimization problem which, can be stated as

$$\alpha^* = \arg \max_{\alpha} \log p_{pp}(\mathbf{y}_{\text{hold-out}} | X_{\text{hold-out}}, Z_{\text{train}}, \mathcal{GP}, \alpha). \quad (3.23)$$

The predictive posterior probability, depending on parameter α , can then be expressed (analogously to eq. (2.55)) by

$$\begin{aligned} \log p_{pp}(\mathbf{y}_{\text{hold-out}} | X_{\text{hold-out}}, Z_{\text{train}}, \mathcal{GP}, \alpha) &= \\ &= -\frac{1}{2}(\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_{m,\alpha})^T K^{-1}(\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_{m,\alpha}) \\ &\quad - \frac{1}{2} \log |K| - \frac{n_{\text{hold-out}}}{2} \log(2\pi), \end{aligned} \quad (3.24)$$

where K is the matrix of posterior variances, generated by evaluating the predictive posterior variance $\sigma_m^2(\mathbf{x})$ for all withheld parameter configurations $X_{\text{hold-out}}$. Analogously, $\boldsymbol{\mu}_{m,\alpha}$ is the vector of modified predictive posterior means, evaluated at withheld parameter configurations.

After plugging in the modified posterior mean from eq. (3.27) into the predictive posterior probability in eq. (3.24), the gradient of eq. (3.24) can be taken with respect to α . Consequently, solving the equation for α will yield the parameter which maximizes the predictive posterior probability on withheld data points

$$\alpha^* = \frac{(\mathbf{y} - \boldsymbol{\mu}_m)^T K^{-1} \mathbf{f}_{\text{NN}}}{\mathbf{f}_{\text{NN}}^T K^{-1} \mathbf{f}_{\text{NN}}}, \quad (3.25)$$

where $\boldsymbol{\mu}_m$ and \mathbf{f}_{NN} are the vectors of posterior means and interpolated simulation data, evaluated at parameter configurations $X_{\text{hold-out}}$, respectively.

However, the parameter α^* now depends on the choice of withheld data points. To make the estimation of α^* more robust, the procedure is repeated for r random splits of withheld data with a constant splitting ratio. This results in r values for the parameter α , which will be averaged to

$$\bar{\alpha}^* = \frac{1}{r} \sum_{i=1}^r \alpha_i^*. \quad (3.26)$$

The predictive posterior mean with optimal control parameter yields

$$\mu_{n,\alpha}^*(\mathbf{x}) = \mu_n(\mathbf{x}) + \bar{\alpha}^* f_{\text{NN}}(\mathbf{x}), \quad (3.27)$$

which can then be used in within the acquisition function. In case of suitable simulation data, the parameter $\bar{\alpha}^*$ will take values unequal to zero, guiding the acquisition function to promising areas in simulation data space. In case of untrustworthy simulation data, the parameter will take the value zero or close to zero, which yields the unmodified acquisition function without influence of simulation data.

3.4.2 Mismatch Correction

So far, the described method of a prior-guided acquisition function just discards simulation data if it appears unsuitable, based on objective function observations. Alternatively, a mismatch correction step can be executed before calculating an optimal averaged $\bar{\alpha}^*$. Therefore, an additional GP will model the observed mismatch between n already gathered objective function evaluations and interpolated simulation data values, as previously explained in section 3.3.2. The posterior mean $\mu_{n,\text{mis}}(\mathbf{x})$ of the mismatch GP can again be calculated, which predicts the deviation between simulation data and reality for arbitrary parameter configurations \mathbf{x} .

Consequently, the corrected simulation data can be written as

$$\hat{f}_{\text{NN}}(\mathbf{x}) = f_{\text{NN}}(\mathbf{x}) - \mu_{n,\text{mis}}(\mathbf{x}). \quad (3.28)$$

Therefore, the adjusted posterior mean from eq. (3.21) is further modified to

$$\hat{\mu}_{n,\alpha}(\mathbf{x}) = \mu_n(\mathbf{x}) + \alpha \hat{f}_{\text{NN}}(\mathbf{x}). \quad (3.29)$$

The optimal parameter $\overline{\alpha}^*$ can then be calculated following eqs. (3.24) to (3.26). Finally, the predictive posterior mean with mismatch correction and optimal control parameter yields

$$\hat{\mu}_{n,\alpha}^*(\mathbf{x}) = \mu_n(\mathbf{x}) + \overline{\alpha}^* \hat{f}_{NN}(\mathbf{x}), \quad (3.30)$$

which is used to guide the acquisition function in eq. (3.22).

3.5 Expansion to Multiple Sets of Simulation Data

So far, approaches for incorporating a single set of simulation data were explained in detail. However, for the application covered in this work, it is essential to leverage multiple sets of simulation data. The previously introduced approaches for incorporating prior information into a GP model will be extended to multiple sets of prior information in section 3.5.1. Therefore, multiple prior-informed GP models are generated, where one model contains one set of simulation data, respectively. Thus, the first question to answer is how to decide which models should be used during the optimization process and which should be discarded. Therefore, a suitable model selection criterion has to be chosen. Because it can not be assumed that there is one model, perfectly representing the true objective function, it may be desired to use more than one source of prior-information during one optimization iteration. Thus, the second question is how to combine multiple models. Similar to the approaches that involve a prior-informed model, the concept of having a prior-guided acquisition function will be extended to multiple sets of prior information in section 3.5.2, by deciding which set of simulation data to trust in guiding the acquisition function to promising regions in simulation data space.

3.5.1 Multiple Prior-Informed Models

For n_s sets of simulation data, n_s GP models are generated, where each model incorporates prior information from one set of simulation data, respectively. The GPs can be generated by using either a prior informed mean function, as explained in section 3.2, or a prior-informed kernel function, as described in section 3.3. However, since it is not ensured that any set of simulation data resembles the true objective function, one additional GP model without prior information will be generated. This additional GP has a constant prior mean function and the commonly used SE kernel, which is suitable for most objective functions. Consequently, it solely relies on previous function observations and thus will be called the standard GP. In every BO iteration, the resulting $n_s + 1$ GP models are updated on gathered objective function observations. Since it is assumed that controller parameters are time-consuming to evaluate on the real system, it would be impractical to let an acquisition function choose a promising point from all $n_s + 1$ GP models and thus evaluate $n_s + 1$ parameters per iteration. Therefore, in each BO iteration, it has to be decided which model to use for choosing the next point in parameter space to evaluate.

Choosing a Model Selection Criterion

In section 2.4.3 four possible Bayesian model selection criteria were introduced, which are either commonly used or previously proposed in literature.

The most-widely used Bayesian model selection criterion is the marginal likelihood. Based on observation data, the marginal likelihood would be able to choose the GP prior $\mathcal{M}^* \in \{\mathcal{M}_1, \dots, \mathcal{M}_{n_s+1}\}$, which is most likely to have generated the observation data. In general, this does not guarantee that the GP posterior performs well when predicting on unseen data. However, this is the crucial factor in model selection for BO, since the acquisition function uses the GP posterior to predict the performance of unseen points in parameter space. Therefore, the marginal likelihood and related criteria seem not suitable for the problem of Bayesian model selection in this work. A meaningful model selection criterion should give information about the likelihood of a posterior $\mathcal{M}|\mathbf{y}_{\text{train}}$, conditioned on a set of training data, to have generated a set of withheld test data.

To fulfill this requirement, in section 2.4.3, eq. (2.55), the predictive posterior probability was introduced. However, the predictive posterior probability alone, evaluated on only one split of training data and withheld data, would also yield no robust criterion for choosing a model. Thus, promising methods are the presented CV related techniques with the predictive posterior probability as a scoring function. Here, the cumulative leave-p-out CV outperforms classic CV in terms of more distinct scores when applied to multiple models. This seems especially useful under the assumption that there is one model, which represents the true objective function well. However, for the case of having multiple models that possibly mismatch the true function or only partly resemble it, it is unclear if using a cumulative CV score is worth the additional computational effort.

Therefore, it was decided to use a classic CV criterion for model selection. The leave-p-out CV, where the average over all possible orders of data is calculated, is in general costly to compute. Usually, it is reasonable to average over a smaller number of data splits by choosing random data orders for a constant split ratio. These data splits can be generally chosen with or without replacement, where without replacement means, that a withheld data point can not be part of the test data set for another data split. On the contrary, with replacement means, that a withheld point can occur within the set of withheld data points for multiple data splits. Generally, it is rather hard to tell if methods, that choose data splits with replacement are more or less beneficial, compared to methods, choosing the data splits without replacement. It was decided to take a method with replacement, namely the Monte Carlo CV, which in general has a low variance and a high bias. However, the k-fold CV, which has a higher variance and lower bias, would have been a suitable choice as well. The formula for the Monte Carlo CV can be found in eq. (2.54), where the ratio for the split in training data and withheld data set was chosen to 80 % for training and 20 % for testing. This split is used, whenever the Monte Carlo CV is computed in this work. The number of data splits $|\mathcal{R}|$, on which the CV scores are determined and averaged, was chosen to 10. This rather low number for $|\mathcal{R}|$ is reasonable, since the data set of gathered objective function observations contains only a couple of data points. Therefore, choosing a higher number of data splits $|\mathcal{R}|$ would be computationally more costly, while hardly yielding any benefits.

Combining Models

As explained in section 2.4.4, there are methods to combine multiple models, either within the acquisition function, or by directly using a composed kernel. Consequently, k models can be combined and used in BO to select the next promising point in parameter space.

Composed Acquisition Function The first option is to combine predictions from multiple GP posteriors within a composed acquisition function. The MLEI acquisition function, from eq. (2.59), chooses the next parameter configuration to evaluate from k GP models simultaneously. However, it incorporates the marginal likelihood, which was assessed as rather unsuitable for determining a model's predictive performance. Nevertheless, the MLEI could still be a promising approach for considering multiple GP models, when the marginal likelihood gets substituted with another criterion for the model's predictive performance. Here, the Monte Carlo CV of the considered GP model can be simply used to replace the marginal likelihood. For simplicity, the Monte Carlo CV of a GP model, computed with a share of 20 % withheld data points from available observation data, will be denoted as $CV_{MC}(\mathcal{GP})$. For the MLEI acquisition function with Monte Carlo CV scores, this results in

$$\begin{aligned} a_{MLEI}(\mathbf{x}, \mathcal{GP}) &= CV_{MC}(\mathcal{GP}) a_{EI} \\ &= CV_{MC}(\mathcal{GP}) \mathbb{E}[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)], \end{aligned} \quad (3.31)$$

which can be optimized with respect to the parameter vector and the GP model, as stated in eq. (2.60).

Another composed acquisition function, the Utility Mean from eq. (2.61) yields the mean of the EI of multiple models. Therefore, it possibly tends to favor points, suggested by optimistic models. A more suitable approach is the Weighted Mixture acquisition function from eq. (2.62), where the EI of the models is weighted by their ratios of model selection scores. Originally, the marginal likelihood is utilized to compute the weights. However, after assessing that the marginal likelihood is an unsuitable criterion, it is replaced by the Monte Carlo CV. The resulting Weighted Mixture Expected Improvement (WMEI) with Monte Carlo CV can be stated as

$$\begin{aligned} \mathbf{x}_{n+1} &= \arg \max_{\mathbf{x} \in A} a_{WMEI}(\mathbf{x} | \mathcal{GP}_{1,\dots,k}) \\ &= \arg \max_{\mathbf{x} \in A} \left(\frac{1}{k} \sum_{i=1}^k w_i a_{EI}(\mathbf{x} | \mathcal{GP}_i) \right), \\ \text{with } w_i &= \frac{CV_{MC}(\mathcal{GP}_i)}{\sum_{j=1}^k CV_{MC}(\mathcal{GP}_j)}. \end{aligned} \quad (3.32)$$

Composed Kernel Function The second possibility of combining k models, is to create a composed kernel function from the models' kernels by using them as k sub-kernels for a multiplicative or additive kernel structure. Using a multiplicative kernel structure would limit BO to points where all k sub-kernels agree, whereas an additive kernel structure is able to consider the information of all sub-kernel

functions. Therefore, the method of adding kernels seems more reasonable than multiplying kernels. Since it may not be desired to weigh the information from each sub-kernel equally, the k sub-kernels will be weighted by their respective Monte Carlo CV ratios as

$$k_{\text{WM}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^k w_i k_i(\mathbf{x}, \mathbf{x}'), \quad (3.33)$$

$$\text{with } w_i = \frac{CV_{\text{MC}}(\mathcal{GP}_i)}{\sum_{j=1}^k CV_{\text{MC}}(\mathcal{GP}_j)}.$$

The resulting kernel function in eq. (3.33) will be referred to as Weighted Mixture Kernel (WMK), which yields a composed GP regression model. The composed model can then be further used by the acquisition function to select the next parameter configuration to evaluate.

Unfortunately, a downside of combining too many models is, that it can easily lead to untraceable results. This means, the more models are combined, the harder it gets to reconstruct how much information is used from which model throughout the optimization. Therefore, it was decided to follow a specific approach, where at most $k = 2$ models are combined, concerning the composed acquisition function, as well as the composed kernel. The first model will always be the standard GP model with the GP's hyperparameters $\boldsymbol{\theta}$ fitted on observation data, since it can be assumed that it becomes a good representative of the true objective function, when enough objective function observations are gathered. The second model will be one prior-informed model, yielding the highest model selection score.

3.5.2 Prior-Guided EI with Multiple Sources of Prior Information

Lastly, the approach of the PGEI acquisition function has to be extended to the use of multiple sets of prior information. The difference to the approaches in section 3.5.1 is, that here only one GP model is deployed, namely a standard GP, without prior information from simulation data. After setting up the modified prior-guided posterior mean from section 3.4 eq. (3.21), depending on parameter α , the optimal weighting parameter $\bar{\alpha}_i^*$ is computed for every $i = 1, \dots, n_s$ set of available interpolated simulation data $f_{\text{NN},1}, \dots, f_{\text{NN},n_s}$. Therefore, the steps of maximizing the predictive posterior probability, described in section 3.4.1, can be followed. A mismatch correction step can be performed, analogously to the explanation in section 3.4.2. As a result, the respective modified posterior mean for simulation data set i can be written as

$$\mu_{n,\alpha_i}^*(\mathbf{x}) = \mu_n(\mathbf{x}) + \bar{\alpha}_i^* f_{\text{NN},i}(\mathbf{x}). \quad (3.34)$$

To evaluate the modified predictive posterior means for every set of simulation data, the Monte Carlo CV is used with the predictive posterior probability p_{pp} from eq. (3.24) as a scoring function. The Monte Carlo CV of the GP model \mathcal{GP} , with a posterior mean, which is modified by the simulation data set i , is then determined

by

$$\begin{aligned}
CV_{MC}^i(Z, p, \mathcal{GP}) &= \frac{1}{|\mathcal{R}(p)|} \sum_{r \in \mathcal{R}(p)} CV_M(Z_{\text{hold-out}}(r)) \\
&= \frac{1}{|\mathcal{R}(p)|} \sum_{r \in \mathcal{R}(p)} -\frac{1}{2} (\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_{m, \alpha_i}^*)^T K^{-1} (\mathbf{y}_{\text{hold-out}} - \boldsymbol{\mu}_{m, \alpha_i}^*) \\
&\quad - \frac{1}{2} \log |K| - \frac{p}{2} \log(2\pi),
\end{aligned} \tag{3.35}$$

where M for simplicity denotes the posterior model, conditioned on respective training data splitting. Again, 20 % of the data were withheld for testing. $\mathcal{R}(p)$ is the set of random data splittings for a constant number of p withheld data samples and $\boldsymbol{\mu}_{m, \alpha_i}^*$ is the vector of modified predictive posterior means, based on m training data samples and evaluated at p withheld data points. Finally, the particular set of simulation data, which yields the highest predictive posterior probability, indicated by the Monte Carlo CV, will be selected to guide the EI acquisition function.

3.6 Resulting Algorithms

This section summarizes all promising approaches for incorporating simulation data sets into the BO framework, by proposing concrete algorithms. Since the theory of all the algorithms' components was explained in detail in previous sections, the focus lies on putting together the individual components. For more detailed explanation on individual components, it will be referred to respective previous sections. Because of its downsides, the approach of utilizing a prior-informed mean function will be omitted. Thus, three classes of algorithms are elaborated. Firstly, algorithms that use prior-informed kernels for incorporating simulation data. Secondly, algorithms that deploy a prior-guided acquisition function. And lastly, hybrid algorithms, which deploy prior-informed kernels, as well as a prior-guided acquisition function.

Whenever a standard acquisition function is used within one of the algorithms, it was decided to utilize the EI acquisition function, because of its popularity and well performance, stated in literature.

3.6.1 Algorithms with Prior-Informed Kernels

For the sake of clarity, the algorithms with prior-informed kernels had to be divided into three sub-categories. Firstly, a basic algorithm with prior-informed kernel and either standard EI acquisition function or MLEI acquisition function will be presented and complemented with pseudocode in alg. 2. Secondly, an algorithm with a Weighted Mixture Kernel (WMK) structure will be summarized in alg. 3. The last algorithm, given by alg. 4, deploys the Weighted Mixture EI acquisition function.

Basic Algorithm with Prior-Informed Kernels

A basic algorithm, stated in alg. 2, which uses prior-informed kernels, starts with choosing whether the standard EI or the MLEI acquisition function should be used

for the optimization process. Subsequently, a standard GP, denoted as \mathcal{GP}_s , and $1, \dots, n_s$ GP models with prior-informed kernel function for n_s sets of simulation data, respectively, are initialized. To initialize the sets of gathered objective function data X and \mathbf{y} , i objective function evaluations are performed.

Then the optimization loop is performed, until the pre-defined budget of function evaluations N is exhausted. Firstly, the hyperparameters $\boldsymbol{\theta}$ of all $n_s + 1$ GPs are fitted on all available data, by maximizing the marginal likelihood. Then the Monte Carlo CV is determined, as explained in section 2.4.3, for all GPs, which provides information about the models' predictive performance.

For the case of setting up the EI acquisition function, firstly the best scoring GP model \mathcal{GP}^* is selected from all GPs. Then its predictive posterior distribution is updated on all available objective function observations. Based on the predictive posterior mean and variance, EI selects the next promising point in parameter space. On the contrary, in case of using the MLEI acquisition function instead of EI, the Monte Carlo CV scores are computed as well, but directly used to weigh the predictions of all models within the acquisition function. Therefore, the predictive posterior distributions of all GP models are updated on available objective function observations. Then, in contrast to the previous approach, the MLEI acquisition function selects the next promising point from all GP posteriors.

Evaluating the new point in parameter space yields a new observation y_{n+1} . Consequently, the budget counter is incremented and the next iteration starts. After budget N being exhausted, the parameter configuration \mathbf{x}^* is returned, yielding the best observed value \mathbf{y}^* .

Additionally, it has to be noted, that the option of performing a mismatch correction within the prior-informed kernel is not explicitly stated in the pseudocode in alg. 2, in order to keep the structure of the algorithm simple. For correcting the mismatches between objective function observations and every set of interpolated simulation data, an additional GP has to be fitted for each set of simulation data $i = 1, \dots, n_s$, respectively, and further used as explained in section 3.3.2. In every BO iteration, the mismatch GP predictive posterior distributions are updated on available observation data.

Algorithm with Weighted Mixture Kernels

When using a WMK, as presented in section 3.5.1 in eq. (3.33), some steps are added to the basic algorithm with prior-informed kernels. This results in a new algorithm, stated in alg. 3. After the initialization steps, the optimization loop starts analogously to the basic algorithm with fitting the GP hyperparameters by maximizing the marginal likelihood on observation data and computing the Monte Carlo CV scores. Then the best performing model \mathcal{GP}^* , among the prior-informed models, is selected for building the WMK function of an additional GP model, denoted as \mathcal{GP}_c . The composed kernel is constructed, following eq. (3.33), by combining the standard GP's kernel function and the kernel of the best performing prior-informed model. The kernel parameters of the WMK can be adopted from the standard kernel and the best performing prior-informed kernel, respectively. Consequently, the Monte Carlo CV of the additional composed GP is computed as previously done for the other models. Finally, the overall best model is chosen, which yields the highest

model selection score (denoted as \mathcal{GP}^{**}) and the algorithm proceeds like alg. 2. Again, the mismatch correction, which can be performed when using a prior-informed kernel, was not explicitly stated in the pseudocode of alg. 3. For correcting the mismatches between objective function observations and every set of interpolated simulation data, again an additional GP has to be fitted for each set of simulation data $i = 1, \dots, n_s$, following section 3.3.2, and updated on available observation data in each BO iteration.

Algorithm 2: Bayesian Optimization with prior-informed kernels

```

Choose an acquisition function  $a$ , with  $a \in \{a_{\text{EI}}, a_{\text{MLEI}}\}$ .
Initialize  $\mathcal{GP}_s$  with mean  $m(\mathbf{x}) = c$  and kernel  $k_{\text{SE}}(\mathbf{x}, \mathbf{x}')$ .
Initialize  $\mathcal{GP}_j$  with mean  $m(\mathbf{x}) = c_j$  and prior-informed kernel  $k_{\text{NN}_j}(\mathbf{x}, \mathbf{x}')$ 
  for  $j = 1, \dots, n_s$ .
Perform  $i$  objective function evaluations. Set counter  $n = i$ .
Initialize  $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$  and  $\mathbf{y} = [y_0, \dots, y_i]$ 
while  $n \leq N$  do
  Fit the hyperparameters  $\boldsymbol{\theta}_j$  of  $\mathcal{GP}_j$  on  $X, \mathbf{y}$  for all  $j = 1, \dots, n_s + 1$ .
  Compute  $CV_{\text{CV}}(\mathcal{GP}_j)$  for all  $j = 1, \dots, n_s + 1$ .
  if  $a = a_{\text{EI}}$  then
    Choose  $\mathcal{GP}^*$ , that yields the highest  $CV_{\text{CV}}$  score.
    Update the predictive posterior of  $\mathcal{GP}^*$ , using  $X, \mathbf{y}$ .
    Set up  $a_{\text{EI}}$  with current predictive posterior of  $\mathcal{GP}^*$ .
  else
    Update the predictive posterior of  $\mathcal{GP}_j$  for all  $j = 1, \dots, n_s + 1$ , using
       $X, \mathbf{y}$ .
    Set up  $a_{\text{MLEI}}$  with current predictive posteriors of all GPs.
  end
  Select the next point  $\mathbf{x}_{n+1}$  by optimizing the acquisition function.
  Add  $\mathbf{x}_{n+1}$  to  $X$ .
  Evaluate the objective function at  $\mathbf{x}_{n+1}$ , observe  $y_{n+1}$ 
  Add  $y_{n+1}$  to  $\mathbf{y}$ .
  Increment  $n$ .
end
Return parameter vector  $\mathbf{x}^*$  that yields best observed value  $\mathbf{y}^*$ .

```

Algorithm with Weighted Mixture Acquisition Function

An algorithm, which deploys the WMEI acquisition function, is stated in alg. 4 and proceeds similar to alg. 3. Instead of creating a composed kernel structure, from the standard GP and the overall best performing GP, the WMEI function, which was introduced in section 3.5.1 in eq. (3.32), is set up, using the Monte Carlo CV for computing the weights. Unlike the WMK function, the WMEI is only set up if there is a prior-informed GP model which performs better than the standard GP. If the standard GP outperforms all other available models, it can be assumed that no set of simulation data resembles the true objective function. To use an additional prior-

informed model within the acquisition function could then possibly worsen the choice of the next parameter vector to evaluate. Therefore, the standard EI acquisition function is used in this case, which selects the next parameter configuration from the standard GP. As usual, the optimization loop is repeated until termination.

Algorithm 3: Bayesian Optimization with Weighted Mixture Kernels

```

Choose an acquisition function  $a$ , with  $a \in \{a_{\text{EI}}, a_{\text{MLEI}}\}$ .
Initialize  $\mathcal{GP}_s$  with mean  $m(\mathbf{x}) = c$  and kernel  $k_{\text{SE}}(\mathbf{x}, \mathbf{x}')$ .
Initialize  $\mathcal{GP}_j$  with mean  $m(\mathbf{x}) = c_j$  and prior-informed kernel  $k_{\text{NN}_j}(\mathbf{x}, \mathbf{x}')$ 
  for  $j = 1, \dots, n_s$ .
Perform  $i$  objective function evaluations. Set counter  $n = i$ .
Initialize  $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$  and  $\mathbf{y} = [y_0, \dots, y_i]$ 
while  $n \leq N$  do
  Fit the hyperparameters  $\theta_j$  of  $\mathcal{GP}_j$  on  $X, \mathbf{y}$  for all  $j = 1, \dots, n_s + 1$ .
  Compute  $CV_{\text{CV}}(\mathcal{GP}_j)$  for all  $j = 1, \dots, n_s + 1$ .
  Choose prior-informed  $\mathcal{GP}^*$ , that yields the highest  $CV_{\text{CV}}$  score.
  Initialize  $\mathcal{GP}_c$  with composed kernel function from  $\mathcal{GP}^*$  and  $\mathcal{GP}_s$ .
  Compute  $CV_{\text{CV}}(\mathcal{GP}_c)$ .
  if  $a = a_{\text{EI}}$  then
    Choose  $\mathcal{GP}^{**}$ , that yields the highest  $CV_{\text{CV}}$  score.
    Update the predictive posterior of  $\mathcal{GP}^{**}$ , using  $X, \mathbf{y}$ .
    Set up  $a_{\text{EI}}$  with current predictive posterior of  $\mathcal{GP}^{**}$ .
  else
    Update the predictive posterior of  $\mathcal{GP}_j$  for all  $j = 1, \dots, n_s + 2$ , using
       $X, \mathbf{y}$ .
    Set up  $a_{\text{MLEI}}$  with current predictive posteriors of all GPs.
  end
  Select the next point  $\mathbf{x}_{n+1}$  by optimizing the acquisition function.
  Add  $\mathbf{x}_{n+1}$  to  $X$ .
  Evaluate the objective function at  $\mathbf{x}_{n+1}$ , observe  $y_{n+1}$ 
  Add  $y_{n+1}$  to  $\mathbf{y}$ .
  Increment  $n$ .
end
Return parameter vector  $\mathbf{x}^*$  that yields best observed value  $\mathbf{y}^*$ .

```

3.6.2 Algorithm with Prior-Guided Acquisition Function

An algorithm with prior-informed acquisition function can be found in alg. 5 has the same structure as the classic BO algorithm from section 2.3.4. However, instead of utilizing a commonly used acquisition like EI or UCB, the PGEI acquisition function is set up, as previously explained in section 3.4 for one set of simulation data, and extended for multiple sets of simulation data in section 3.5.2.

Algorithm 4: Bayesian Optimization with Weighted Mixture Expected Improvement

Initialize \mathcal{GP}_s with mean $m(\mathbf{x}) = c$ and kernel $k_{SE}(\mathbf{x}, \mathbf{x}')$.
Initialize \mathcal{GP}_j with mean $m(\mathbf{x}) = c_j$ and prior-informed kernel $k_{NN_j}(\mathbf{x}, \mathbf{x}')$
for $j = 1, \dots, n_s$.
Perform i objective function evaluations. Set counter $n = i$.
Initialize $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$ and $\mathbf{y} = [y_0, \dots, y_i]$
while $n \leq N$ **do**
 Fit the hyperparameters $\boldsymbol{\theta}_j$ of \mathcal{GP}_j on X, \mathbf{y} for all $j = 1, \dots, n_s + 1$.
 Compute $CV_{CV}(\mathcal{GP}_j)$ for all $j = 1, \dots, n_s + 1$.
 Choose \mathcal{GP}^* , that yields the highest CV_{CV} score.
 if $\mathcal{GP}^* \neq \mathcal{GP}_s$ **then**
 Update the predictive posteriors of \mathcal{GP}^* and \mathcal{GP}_s , using X, \mathbf{y} .
 Set up a_{WMEI} with current predictive posteriors from \mathcal{GP}^* and \mathcal{GP}_s .
 else
 Update the predictive posterior of \mathcal{GP}_s , using X, \mathbf{y} .
 Set up a_{EI} with current predictive posterior from \mathcal{GP}_s .
 end
 Select the next point \mathbf{x}_{n+1} by optimizing the acquisition function.
 Add \mathbf{x}_{n+1} to X .
 Evaluate the objective function at \mathbf{x}_{n+1} , observe y_{n+1}
 Add y_{n+1} to \mathbf{y} .
 Increment n .
end
Return parameter vector \mathbf{x}^* that yields best observed value \mathbf{y}^* .

Algorithm 5: Bayesian Optimization with prior-guided acquisition function

Initialize \mathcal{GP}_s with mean $m(\mathbf{x}) = c$ and kernel $k_{SE}(\mathbf{x}, \mathbf{x}')$.
Perform i objective function evaluations. Set counter $n = i$.
Initialize $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$ and $\mathbf{y} = [y_0, \dots, y_i]$
while $n \leq N$ **do**
 Fit the hyperparameters $\boldsymbol{\theta}$ of \mathcal{GP}_s on X .
 Update the predictive posterior of \mathcal{GP}_s , using X, \mathbf{y} .
 Set up a_{PGEI} with the current predictive posterior of \mathcal{GP}_s .
 Select the next point \mathbf{x}_{n+1} by optimizing the acquisition function.
 Add \mathbf{x}_{n+1} to X .
 Evaluate the objective function at \mathbf{x}_{n+1} , observe y_{n+1}
 Add y_{n+1} to \mathbf{y} .
 Increment n .
end
Return parameter vector \mathbf{x}^* that yields best observed value \mathbf{y}^* .

3.6.3 Hybrid Algorithms

So far, only the standard GP model was used within the prior-guided acquisition function. As explained in section 3.4, the prior-guided acquisition function only influences the predictive posterior mean within the acquisition function. However, it could also be useful to additionally integrate prior information into the predictive posterior variance. To achieve this, the prior-guided acquisition function can be used with a GP model with prior-informed kernel. Therefore, the prior-guided acquisition function is combined with the approach of employing either a normal prior-informed kernel function, or a WMK. This yields a hybrid algorithm, which can be found in alg. 6.

Here, $n_s + 1$ GP models for n_s sets of simulation data are initialized, as already seen in algs. 2 to 4. Then, some additional steps are included, before setting up the prior-informed acquisition function. Firstly, The Monte Carlo CV is computed for all GP models and the best performing GP model is chosen. In case of deploying a composed kernel structure, the WMK is created and used within an additional GP, named \mathcal{GP}_c . Finally, the prior-informed acquisition function is set up, based on the overall best performing model \mathcal{GP}^{**} , and the algorithm proceeds selecting the next promising point and repeating the optimization loop until termination.

Algorithm 6: Bayesian Optimization with Prior-Guided Expected Improvement and prior-informed kernels

```

Choose composed kernel structure  $c = \{\text{True}, \text{False}\}$ .
Initialize  $\mathcal{GP}_s$  with mean  $m(\mathbf{x}) = c$  and kernel  $k_{SE}(\mathbf{x}, \mathbf{x}')$ .
Initialize  $\mathcal{GP}_j$  with mean  $m(\mathbf{x}) = c_j$  and prior-informed kernel  $k_{NN_j}(\mathbf{x}, \mathbf{x}')$ 
  for  $j = 1, \dots, n_s$ .
Perform  $i$  objective function evaluations. Set counter  $n = i$ .
Initialize  $X = [\mathbf{x}_0, \dots, \mathbf{x}_i]$  and  $\mathbf{y} = [y_0, \dots, y_i]$ 
while  $n \leq N$  do
  Fit the hyperparameters  $\theta_j$  of  $\mathcal{GP}_j$  on  $X, \mathbf{y}$  for all  $j = 1, \dots, n_s + 1$ .
  Compute  $CV_{CV}(\mathcal{GP}_j)$  for all  $j = 1, \dots, n_s + 1$ .
  if  $c = \text{True}$  then
    Choose prior-informed  $\mathcal{GP}^*$ , that yields the highest  $CV_{CV}$  score.
    Initialize  $\mathcal{GP}_c$  with composed kernel function from  $\mathcal{GP}^*$  and  $\mathcal{GP}_s$ .
    Compute  $CV_{CV}(\mathcal{GP}_c)$ .
  end
  Choose  $\mathcal{GP}^{**}$ , that yields the highest  $CV_{CV}$  score.
  Update the predictive posterior of  $\mathcal{GP}^{**}$ , using  $X, \mathbf{y}$ .
  Set up  $a_{PGEI}$  with the current predictive posterior of  $\mathcal{GP}^{**}$ .
  Select the next point  $\mathbf{x}_{n+1}$  by optimizing the acquisition function.
  Add  $\mathbf{x}_{n+1}$  to  $X$ .
  Evaluate the objective function at  $\mathbf{x}_{n+1}$ , observe  $y_{n+1}$ 
  Add  $y_{n+1}$  to  $\mathbf{y}$ .
  Increment  $n$ .
end
Return parameter vector  $\mathbf{x}^*$  that yields best observed value  $\mathbf{y}^*$ .

```

3.6.4 Algorithm Overview

The prior-informed approaches, presented in this chapter, result in a total number of 16 different algorithms, which are listed in table 3.1.

Firstly, there are six options for algorithms with prior-informed kernel (PIK), without composed kernel structure. Namely, approaches with PIK and either a standard EI acquisition function, the MLEI acquisition function, or the WMEI acquisition function. Additionally, every option can be performed either with or without correction of the mismatches between simulation data and observed objective function values.

Secondly, there are four algorithms with prior-informed kernels that have a composed kernel structures, called Weighted Mixture Kernels (WMK). These approaches can also be used with or without mismatch correction and combined with either the standard EI acquisition function or the MLEI acquisition function, respectively.

Lastly, there are six algorithms with Prior-Guided Expected Improvement (PGEI) acquisition function. It is possible to combine the PGEI either with the PIK, the WMK, or a kernel without prior information, where in this case a standard squared exponential kernel (SEK) will be used. Again, for every algorithm, there is the option of using it with or without mismatch correction. In the following table table 3.1, the use of a mismatch correction will be denoted with the abbreviation "m".

Algorithm Overview		
algorithm abbreviation	kernel	acquisition function
PIK, EI	prior-informed kernel	EI
PIK, EI, m	prior-informed kernel	EI
PIK, WMEI	prior-informed kernel	Weighted Mixture EI
PIK, WMEI, m	prior-informed kernel	Weighted Mixture EI
PIK, MLEI	prior-informed kernel	Most-Likely EI
PIK, MLEI, m	prior-informed kernel	Most-Likely EI
PIK, PGEI	prior-informed kernel	Prior-Guided EI
PIK, PGEI, m	prior-informed kernel	Prior-Guided EI
WMK, EI	Weighted Mixture Kernel	EI
WMK, EI, m	Weighted Mixture Kernel	EI
WMK, MLEI,	Weighted Mixture Kernel	Most-Likely EI
WMK, MLEI, m	Weighted Mixture Kernel	Most-Likely EI
WMK, PGEI	Weighted Mixture Kernel	Prior-Guided EI
WMK, PGEI, m	Weighted Mixture Kernel	Prior-Guided EI
SEK, PGEI	SE kernel	Prior-Guided EI
SEK, PGEI, m	SE kernel	Prior-Guided EI

Table 3.1: Overview on resulting prior-informed algorithms for leveraging multiple sets of simulation data

Chapter 4

Algorithm Evaluations and Performance Analysis

In this chapter, the prior-informed algorithms, which were presented in chapter 3, are tested, evaluated and compared to each other. Therefore, a meaningful test scenario has to be constructed. Firstly, a multidimensional test function with known optimum has to be chosen as an objective function, which imitates the objective function of an aim point controller with multiple hyperparameters. Secondly, multiple sets of simulation data are generated for the test function, which ideally cover realistic scenarios, like being a medium or possibly even a bad fit for the objective function. Moreover, a suitable reference has to be chosen to assess the performance of the prior-informed approaches and identify possible improvements in sample efficiency, compared to a commonly used baseline approach. Lastly, some evaluation criteria have to be defined, which facilitate the comparison of the different prior-informed algorithms and the baseline. Eventually, they should help to identify the best performing algorithms, which fulfill the requirements, imposed by the objective of this thesis and the application.

To perform the tests and evaluate the results, all algorithms, were implemented in Python with Ax [7], an open-source platform for sequential experimentation, where Ax relies on BoTorch [13], which is a python library for Bayesian Optimization.

4.1 Hartmann6 Test Function

As a test function, which imitates the objective function of an aim point controller, the six-dimensional Hartmann function was chosen, also referred to as Hartmann6, with known optimal function value as well as respective parameter values. Hartmann6 is a commonly used test function for optimization problems, with the objective to find the global minimum of the function. It is usually evaluated on the hypercube $x_i \in [0, 1]$, for all $i = 1, \dots, 6$. Moreover, the function possesses two local minima, stated in tab. 4.1. The global minimum is highlighted.

Local Minima of Hartmann6

#	x_1	x_2	x_3	x_4	x_5	x_6	y
1	0.405	0.882	0.846	0.574	0.139	0.038	-3.203
2	0.202	0.150	0.477	0.275	0.312	0.657	-3.322

Table 4.1: Local minima of 6-dimensional Hartmann6 test function

4.2 Simulation Data

Four sets of simulation data were created for the Hartmann6 objective function. To imitate a real use case with possibly inaccurately estimated simulation parameters, the simulation data was drawn from shifted or noisy versions of the real Hartmann6 function. To consider the case of completely unsuitable simulation data, one set resembles a six dimensional polynomial function. Consequently, none of the simulation data perfectly resembles the true function in the area of the global optimum. To interpolate the data points, a regression NN was used. More details about the employed NN and the training procedure can be found in appendix A.1.1.

The plots in fig. 4.1 compare real Hartmann6 function values with predicted values from the interpolated simulation data for several parameter configurations, where the true minimum of the Hartmann6 function is highlighted by a red marker, respectively. Fig. 4.1a shows the case where simulation data was drawn from a noisy version of the Hartmann6 function. The result is simulation data, which roughly resembles the underlying function. However, the true minimum of the function is underestimated by the simulation data, whereas values in the area of the local minimum seem to be overestimated. In fig. 4.1b and fig. 4.1c, the case was imitated, where simulation hyperparameters, like the mirror error, were estimated incorrectly. In fig. 4.1b, the data points were drawn from the Hartmann6 function, shifted for 0.1 points on the x_1 -axis and for 0.2 points on the x_2 -axis. In fig. 4.1c, the data was drawn from the Hartmann6 function, shifted for 0.5 points on the x_1 -axis. Lastly, in fig. 4.1d, the simulation data does not resemble the Hartmann6 function at all, since the data points were collected from a six-dimensional polynomial function.

In general, none of the shown simulation data would directly lead to the true objective function's optimum. Transferred to the problem of optimizing an aim point controller of a solar power tower plant, the parameter configuration, obtained from simulation data alone, wouldn't lead to optimal control behavior and would have to be readjusted on the real system. However, the following sections will evaluate, to which extent it is still beneficial to use those corrupted sets of simulation data in order to find optimal parameters on the real system.

4.3 Baseline

A suitable reference approach for hyperparameter optimization is needed to assess the performance of the prior-informed approaches and identify possible improvements in sample efficiency, compared to the baseline. For this purpose, a standard BO algorithm with SE kernel and EI acquisition function is chosen as a reference. As explained in section 2.2, BO already outperforms other commonly used hyper-

parameter optimization approaches in terms of sample efficiency and consequently also in terms of number of required function evaluation to reach well performing parameter configurations. Therefore, the standard BO approach is a suitable choice to assess the prior-informed algorithms regarding possible further improvement in sample efficiency.

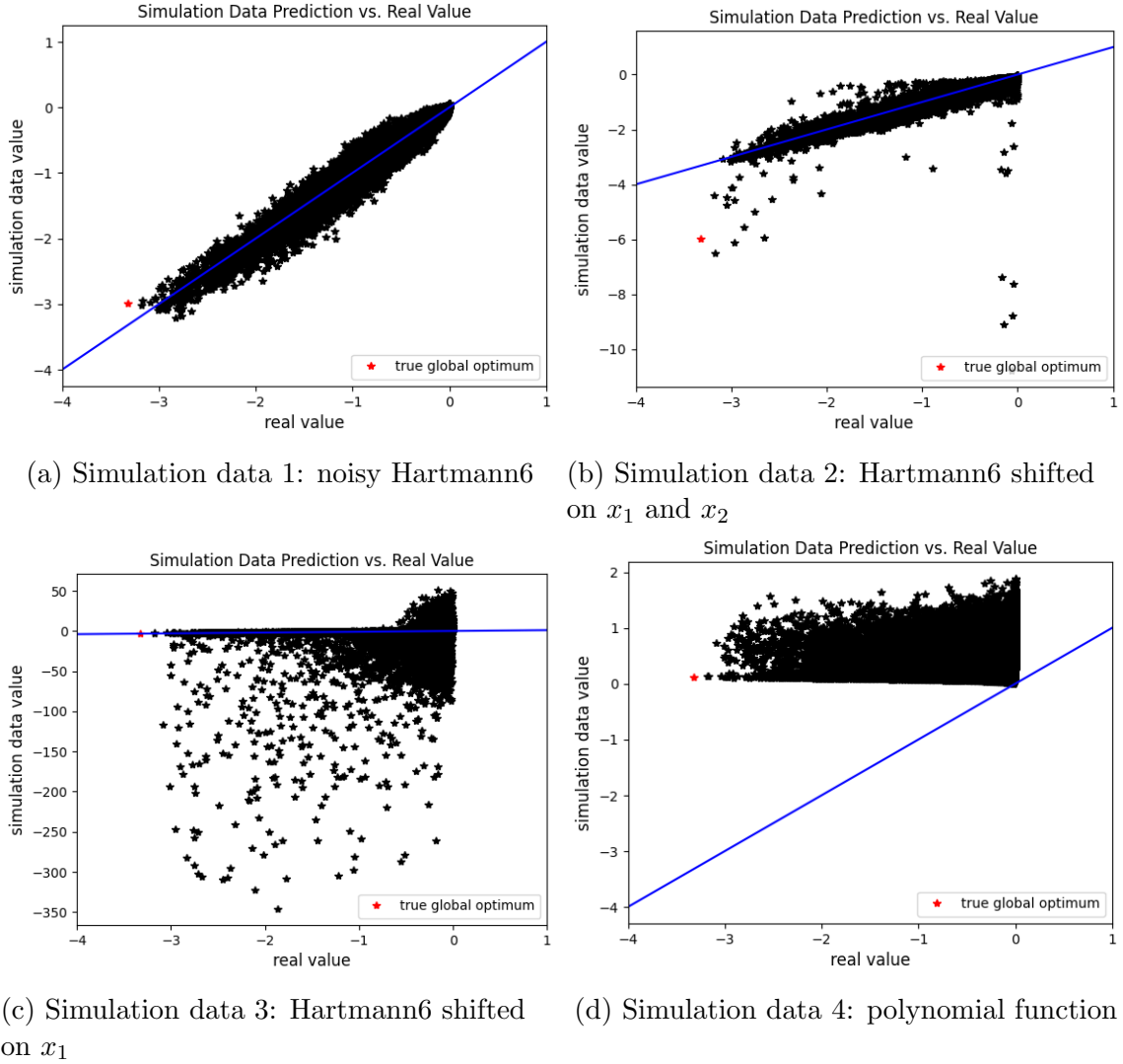


Figure 4.1: Hartmann6 function values vs. simulation data predictions for four sets of simulation data

4.4 Evaluation Criteria

To evaluate the proposed algorithms and compare them with the baseline, as well as with each other, evaluation criteria have to be developed. The goal is to identify if one or more prior-informed algorithms fulfill the following requirements. Firstly, an increase in sample efficiency should be assessed, which should be clearly recognizable by a reduced number of required objective function evaluations to find well

performing parameters, compared to the baseline. Secondly, well performing parameters should not only be located fast, but also as close to the global optimum as possible. Thirdly, an algorithm, which uses prior-information, should not be more prone to get stuck in a local optimum, as the baseline. Ideally, the prior information should even help to overcome a local extreme value. Lastly, the performance of the prior-informed algorithms, regarding the evaluation criteria, should not notably decrease when they are exposed to simulation data, which does not resemble the true objective function. In this case, the performance should be similar to the standard BO's performance.

Within the following subsections, these criteria are explained in more detail and applied to the standard BO, which will be used as the baseline performance for the assessment of the prior-informed algorithms. For the evaluation of standard BO in this section and all following evaluations of prior-informed algorithms within this chapter, the determined results are rounded to integers.

4.4.1 Number of Function Evaluations

The first criterion, which is used to assess the algorithms, is the number of required function evaluations to obtain well performing parameters. More precisely, it will be determined how many function evaluations have to be performed to reach a certain accuracy of the optimum in objective function space. In this context, the accuracy is calculated in percentage and defined as

$$acc = \left(1 - \frac{|f_{\text{opt}} - f^+|}{d_{\text{max}}}\right) \cdot 100 \%, \quad (4.1)$$

with $|f_{\text{opt}} - f^+|$ denoting the absolute difference between the optimal function value (maximum value of the function for maximization problems or minimum value of the function for minimization tasks) and the best observed objective function value f^+ so far. d_{max} stands for the maximum difference between values in function output space.

To gain an insight into the whole optimization process, it will be determined how many function evaluations are required to reach an accuracy of 50 %, 80 %, 90 % and 95 %, respectively. Especially when comparing standard BO to proposed prior-informed algorithms, the amount of required function evaluations to reach a certain accuracy can be used as the indicator for an increase in sample efficiency.

However, to obtain meaningful results, the optimization should be performed several times and the number of required function evaluations determined on average. This is necessary, firstly, since one or more randomly chosen initial parameter configurations can considerably affect the optimization process. Secondly, BoTorch uses a gradient-based optimizer for optimizing the acquisition function. This optimizer makes use of multiple random restarts to improve optimization quality, which can lead to deviations in selected parameter configurations from one optimization run to another.

It was decided to perform an overall number of 30 optimization runs, where this guiding value was taken from literature with comparable test scenarios [48], [45]. The 30 optimization runs were performed with five randomly chosen initial parameter configurations, respectively, which represent the first five function evaluations.

In general, the number of initial function evaluations can be chosen freely. However, it should trade off the preferably widespread initial discovery of the search space, while not wasting costly function evaluations. For a six-dimensional test function, five initial function evaluations seemed reasonable.

Results for Standard BO In tab. 4.2, the standard BO approach for finding the minimum of the Hartmann6 function was evaluated regarding the introduced criterion, where the abbreviation "eval." is used for number of function evaluations, "acc." for the achieved accuracy and "BO" for the standard Bayesian Optimization approach. For the six-dimensional Hartmann function, the standard BO without simulation data needs on average 12 function evaluations to find parameters, which yield a 50 % accuracy of the objective function's optimum. Respectively, accuracies of 80 %, 90 % and 95 %, can be reached on average after 23, 31, and 43 function evaluations.

Number of Function Evaluations				
algorithm	50 % acc.	80 % acc.	90 % acc.	95 % acc.
BO	12 eval.	23 eval.	31 eval.	43 eval.

Table 4.2: Hartmann6 test: number of required function evaluations (eval.) of standard Bayesian Optimization to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum, averaged over 30 optimization runs

4.4.2 Accuracy of the Located Optimum

As the second criterion, the accuracy, introduced in eq. (4.1), is determined, which an algorithm is able to reach after a certain amount of function evaluations. Intuitively speaking, the first criterion can be used to assess how fast an algorithm finds well performing parameter configurations, the second one indicates more how close to optimal and thus how accurate the results are.

The accuracy is determined after the maximum number of function evaluations, denoted as the budget N , which consists of the number of initial parameter configurations and the amount of optimization iterations. For the following tests, the number of optimization iterations was chosen to 70. With the five initial parameter evaluations, this results in a budget of $N = 75$ maximum function evaluations. This number was chosen, since standard BO does not show noteworthy improvement in objective function observations after more than 75 function evaluations. A prior-informed algorithm, which outperforms standard BO, will therefore show improvements within this fixed budget.

In addition to the accuracy after the maximum number of function evaluations, some intermediate results are determined, to gain a deeper insight into the performance during the whole optimization process. Therefore, the accuracy is also calculated after 25 %, 50 %, 75% in addition to 100 % of the number of maximum function evaluations N , which corresponds to 19, 38, 57 and 75 objective function evaluations, respectively.

Primarily, the reachable accuracy in objective function space is of main importance, since it will be the indicator for the controller's performance. However, some objective functions, including Hartmann6, have local optima closely located to the global optimum in function output space, as stated in tab. 4.1. Therefore, it may not be clear if an algorithm is prone to get stuck in a local optimum, from only looking at the yielded accuracy of the objective function's output. For Hartmann6, local and global optimum lie in different parts of the parameter space. Thus, for a comprehensive performance analysis of the algorithms on the Hartmann6 test function, it is helpful to also consider the accuracy of the algorithms in parameter space. For this purpose, a second accuracy metric is defined as

$$acc_{\text{param}} = \left(1 - \frac{\|\mathbf{x}_{\text{opt}} - \mathbf{x}^+\|}{d_{\text{max,p}}}\right) \cdot 100 \%, \quad (4.2)$$

which is defined analogously to eq. (4.1), but considers the euclidean distance between parameter vectors instead of objective function output values. Consequently, $d_{\text{max,p}}$ denotes the maximum euclidean distance between points in parameter space.

Results for Standard BO In tab. 4.3 and tab. 4.4, the standard BO approach for finding the minimum of the Hartmann6 function, was evaluated regarding the accuracy criterion in function output and parameter space, respectively. Again, 30 optimization runs were performed and averaged to obtain meaningful results. To additionally give an indication of the confidence interval (CI), fig. 4.2a visualizes the mean distances of the so far best observed value to the global optimum in objective function space, as well as the CI 95 %. Analogously, fig. 4.2b depicts the mean distance of the parameter vector, which leads to the so far best observed objective function value, to the true function optimum in parameter space and CI 95 %. Here, the so-called model change denotes the iteration where the regression model is adapted to observation data for the first time. This usually happens after randomly collecting and evaluating the initial parameter configurations within the initial iterations.

From tab. 4.3, it can be seen that the standard BO approach reaches on average 98 % accuracy of the true optimum in function output space. After at least 57 function evaluations, the accuracy does not show any noteworthy improvements. However, in parameter space, on average only 84 % accuracy is reached after budget N is exhausted. Again, there is only small improvement during the last quarter of the optimization iterations. Additionally, in fig. 4.2b a relatively large CI can be observed, which indicates that optimization runs with different initial parameter configurations lead to highly varying results in parameter space. Thus, the presumption is, that for some initial parameter configurations, the optimization moves towards the local extreme value and is not able to overcome it in order to find the global optimum. For other initial parameter configurations, the global optimum is discovered. The next evaluation criterion will further investigate this assumption.

Accuracy in Function Output Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO	72 % acc.	94 % acc.	98 % acc.	98 % acc.

Table 4.3: Hartmann6 test: accuracy (acc.) of standard Bayesian Optimization after 25 % (= 19 eval.), 50 % (= 38 eval.), 75 % (= 57 eval.) and 100 % of maximum function evaluations (eval.) (= 75 eval.), averaged over 30 optimization runs

Accuracy in Parameter Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO	75 % acc.	82 % acc.	83 % acc.	84 % acc.

Table 4.4: Hartmann6 test: accuracy of standard Bayesian Optimization in parameter space after 25 % (= 19 eval.), 50 % (= 38 eval.), 75 % (= 57 eval.) and 100 % of maximum function evaluations (eval.) (= 75 eval.), averaged over 30 optimization runs

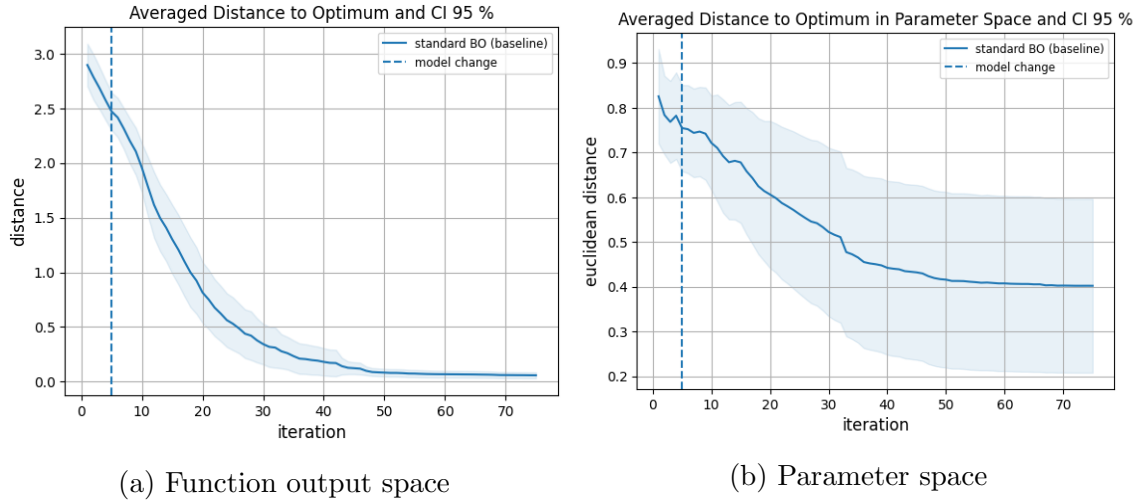


Figure 4.2: Hartmann6 test: Mean distance to the global optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 30 optimization runs

4.4.3 Overcoming Local Optima

As already explained, it may be useful to determine the accuracy of the algorithms in parameter space, to achieve a comprehensive performance analysis of the algorithms, when testing them on objective functions with local extreme values, like the used Hartmann6 function.

In tab. 4.4, it was observed, that standard BO only achieves an averaged accuracy of 84% in parameter space, within the budget of maximum function evaluations $N = 75$, while yielding 98 % accuracy in function output space. Therefore, the

presumption was made, that certain initial parameter configurations lead towards the local minimum of Hartmann6, which the standard BO is not able to overcome within a reasonable number of function evaluations.

To support this statement, a set of five fixed initial parameter configurations was created, where one parameter value was included, which lies in the area of the local optimum in parameter space. The standard BO approach with these fixed initial parameters was tested for optimizing Hartmann6 and averaged over five optimization runs. In this context, averaging over a smaller number of optimization runs seems reasonable, since the initial parameter configuration remains the same, which reduces the deviation between optimization runs.

Results for Standard BO The accuracies in tab. 4.5 show the consequences of the fixed initial parameter configurations. They immediately lead the optimization close to the local optimum, which in case of Hartmann6 yields good accuracies in function output space, since the local extreme value is located close to the global optimum in function output space. However, from the accuracies in parameter space in tab. 4.6, it becomes apparent, that the optimization at most moves deeper into the local optimum and resigns there, while it is not able to discover parameters with an averaged accuracy greater than 56 % of the global optimum in parameter space, within the budget of $N = 75$ function evaluations.

Again, the averaged distances to the global optimum and respective CIs are plotted in function output space, as well as parameter space in fig. 4.3a. In addition, this time also the mean distances to the local optimum and respective CIs are plotted in function output space, as well as parameter space in fig. 4.4a. The plots support previous observations. Moreover, it can be seen, that the optimization on average exactly reaches the local optimum in function output space with a very small confidence interval, while on average not completely reaching it in parameter space with a larger confidence interval. This is an indicator, that the local optimum of Hartmann6 is a rather broad plateau in parameter space.

Accuracy in Function Output Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO	93 % acc.	94 % acc.	96 % acc.	96 % acc.

Table 4.5: Hartmann6 test: accuracy (acc.) of standard Bayesian Optimization in function output space after 19, 38, 57 and 75 evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

Accuracy in Parameter Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO	55 % acc. p.	57 % acc. p.	57 % acc. p.	56 % acc. p.

Table 4.6: Hartmann6 test: accuracy of standard Bayesian Optimization in parameter space (acc. p.) after 19, 38, 57 and 75 evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

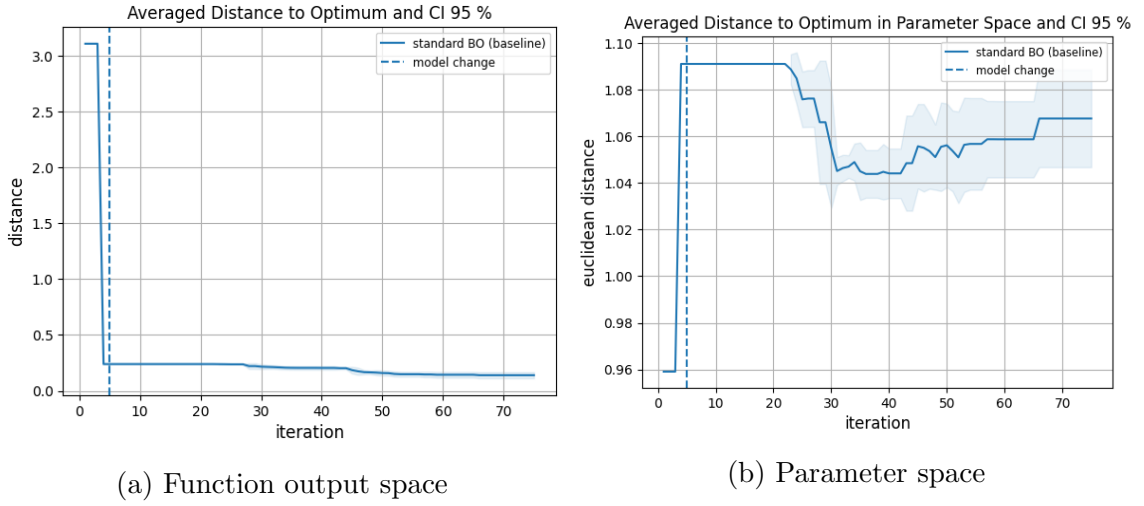


Figure 4.3: Hartmann6 test: mean distance to the global optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 5 optimization runs with fixed initial parameter configurations

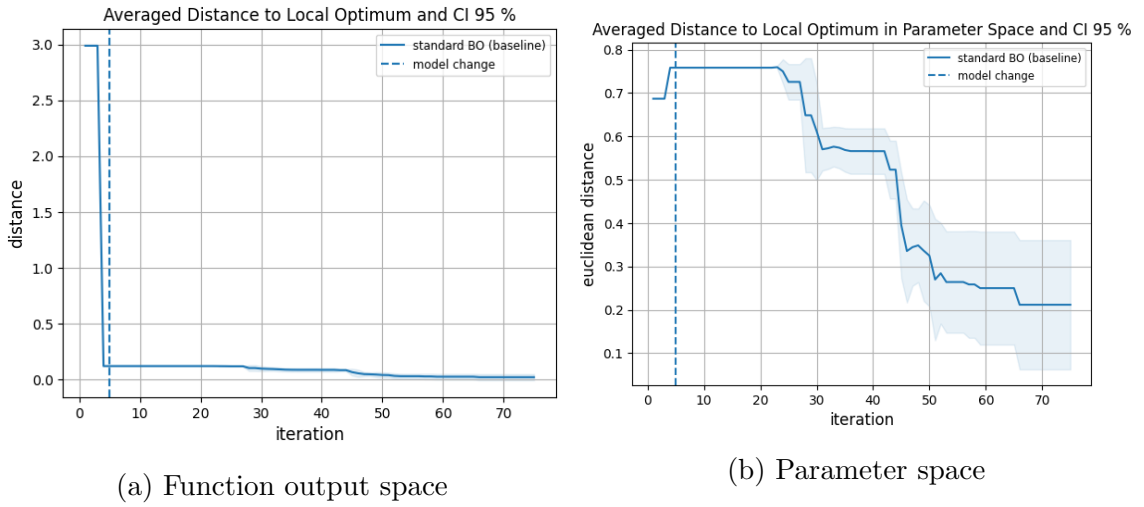


Figure 4.4: Hartmann6 test: mean distance to the local optimum and confidence interval 95 % in function output space (a) and parameter space (b) for standard Bayesian Optimization, averaged over 5 optimization runs with fixed initial parameter configurations

4.4.4 Dealing with Unsuitable Simulation Data

Lastly, the prior-informed algorithms are analyzed regarding their exposure to simulation data, which does not resemble the objective function. This is an essential aspect, since it has to be secured, that possibly unsuitable simulation data does not worsen the performance of a prior-informed algorithm, compared to standard BO.

4.5 Algorithm Evaluations

In the following, the 16 prior-informed algorithms, which were summarized in section 3.6.4, are tested, evaluated regarding introduced evaluation criteria and compared to the standard BO, which is used as the performance baseline.

Recalling the different approaches, there are, firstly, six options for algorithms with prior-informed kernel (PIK), without composed kernel structure. Namely, approaches with PIK and either the standard EI acquisition function, the Most-Likely Expected Improvement (MLEI) acquisition function, or the Weighted Mixture Expected Improvement (WMEI) acquisition function. Additionally, every option can be performed either with or without correction of the mismatches between simulation data and observed objective function values. Secondly, there are four algorithms with prior-informed kernels and composed kernel structures, the so-called Weighted Mixture Kernels (WMK). These approaches can also be used with or without mismatch correction and combined with either the standard EI acquisition function or the MLEI acquisition function, respectively. Lastly, there are six algorithms with Prior-Guided Expected Improvement (PGEI) acquisition function. It is possible to combine the PGEI either with the PIK, the WMK or a kernel without prior information, where in this case a standard squared exponential kernel (SEK) will be used. Again, for every algorithm, there is the option of using it with or without mismatch correction. In the following tables and figures, the use of a mismatch correction will be shortened as "m".

4.5.1 Test Case

If not stated otherwise, the test scenario is considered, where all available sets of simulation data from section 4.2 are used as prior information for the algorithms. Moreover, for each algorithm, 30 optimization runs were performed, and their results averaged for the individual evaluation criteria. Each optimization run again starts with evaluating five randomly chosen initial parameter configurations, respectively. However, for better comparability, it was ensured that all algorithms start with the same initial parameter configuration for a certain optimization run.

To additionally analyze the algorithms' behaviors under exceptional circumstances, further test cases will be introduced and explained when assessing the if the algorithms are prone to get stuck in local optima and their behavior when being exposed to unsuitable simulation data.

4.5.2 Number of Function Evaluations

Firstly, the mentioned algorithms are evaluated regarding the criterion which assesses the number of function evaluations. Additionally, their improvement or decline in comparison to the baseline is determined in percentage. It should be noted, that since this criterion evaluates the number of function evaluations, an improvement is indicated by a negative percentage number.

In tab. 4.7, the results are stated. For easier assessment, well performing algorithms, which outperform the baseline, are marked in light gray. Algorithms, which yield

the best performance, regarding the considered criterion, are highlighted in a darker shade of gray.

Results Generally, it can be noticed that all prior-informed algorithms reach accuracies of 50 %, 80 % and 90 % on average within 30 to almost 70 % less function evaluations, compared to standard BO. For reaching 95 % accuracy, especially the algorithms with PIK in combination with WMEI acquisition function, as well as the PGEI approaches with WMK or SEK stand out with up to 58 % less required function evaluations, compared to the baseline. The two WMEI approaches, with and without mismatch correction, show similarly well averaged performance throughout the optimization procedure, compared to each other. The same observation holds for both approaches with SEK and PGEI. Additionally, the algorithm with PIK and MLEI, with mismatch correction shows good performance, whereas PIK and MLEI without mismatch correction reaches 95 % accuracy within an increased, but still reasonable number of evaluations, compared to standard BO.

On the contrary, all algorithms, which utilize the standard EI acquisition function, are on average either not able to make it to 95 % accuracy or reach 95 % accuracy much slower than the standard BO. Likewise, algorithms with PIK in combination with PGEI and approaches with WMK in combination with MLEI are outperformed by standard BO for reaching 95 % accuracy. Here, the performance of algorithms with mismatch correction turns out worse than without correcting mismatches.

Number of Function Evaluations

algorithm	50 % acc.	80 % acc.	90 % acc.	95 % acc.
BO (baseline)	12 eval.	23 eval.	33 eval.	43 eval.
PIK, EI	8 (-33 %)	12 (-48 %)	16 (-48 %)	- (-)
PIK, EI, m	8 (-33 %)	13 (-43 %)	17 (-45 %)	- (-)
PIK, WMEI	7 (-42 %)	12 (-48 %)	16 (-48 %)	29 (-33 %)
PIK, WMEI, m	8 (-33 %)	13 (-43 %)	17 (-45 %)	29 (-33 %)
PIK, MLEI	7 (-42 %)	11 (-52 %)	14 (-55 %)	46 (+7 %)
PIK, MLEI, m	8 (-33 %)	12 (-48 %)	17 (-45 %)	37 (-14 %)
PIK, PGEI	6 (-50 %)	9 (-61 %)	12 (-61 %)	48 (+12 %)
PIK, PGEI, m	6 (-50 %)	9 (-61 %)	11 (-65 %)	70 (+63 %)
WMK, EI	8 (-33 %)	15 (-35 %)	21 (-32 %)	- (-)
WMK, EI, m	8 (-33 %)	13 (-43 %)	20 (-35 %)	73 (+70 %)
WMK, MLEI	8 (-33 %)	13 (-43 %)	16 (-48 %)	52 (+21 %)
WMK, MLEI, m	9 (-25 %)	13 (-43 %)	17 (-45 %)	54 (+26 %)
WMK, PGEI	6 (-50 %)	9 (-61 %)	11 (-65 %)	26 (-40 %)
WMK, PGEI, m	6 (-50 %)	9 (-61 %)	11 (-65 %)	34 (-21 %)
SEK, PGEI	6 (-50 %)	8 (-65 %)	10 (-68 %)	19 (-56 %)
SEK, PGEI, m	6 (-50 %)	8 (-65 %)	11 (-65 %)	18 (-58 %)

Table 4.7: Hartmann6 tests with all simulation data sets: number of required function evaluations (eval.) to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum in comparison to the baseline, averaged over 30 optimization runs

4.5.3 Accuracy of the Located Optimum

The algorithms are evaluated regarding the accuracy criterion in function output, as well as in parameter space. Again, their improvement or decline in comparison to the baseline's performance is determined in percentage. However, in comparison to the first criterion, an improvement is now denoted by a positive percentage number, since it indicates an increase in accuracy.

The results are stated in tab. 4.8 and tab. 4.9 for the function output space and parameter space, respectively. Algorithms, that outperform the baseline are marked in light gray, whereas algorithms which yield the highest accuracies in function output space, are highlighted in a darker shade of gray.

Results in Function Output Space In function output space, almost all prior-informed algorithms reach accuracies of at least 90 % within only 19 function evaluations, which is a quarter of the maximum number of evaluations. Compared to the baseline, this is an improvement of up to 33 %.

Especially well performances show algorithms with PIK in combination with either WMEI or MLEI, as well as the Prior-Guided EI approach with SEK. After 38 and 57 function evaluations, these algorithms mostly achieve better averaged accuracies than the baseline. After 75 function evaluations, they outperform standard BO with one to two percentage points improvement. Here, algorithms with mismatch correction seem to yield similar to slightly better performance compared to their counterpart without mismatch correction.

On the contrary, the performance of algorithms with a standard EI acquisition function declines for the second half of the optimization, compared to standard BO and all other approaches. Likewise, the PGEI approaches with either PIK or WMK, as well as approaches with WMK with MLEI acquisition function can not reach the baseline's performance for the second half of the optimization process.

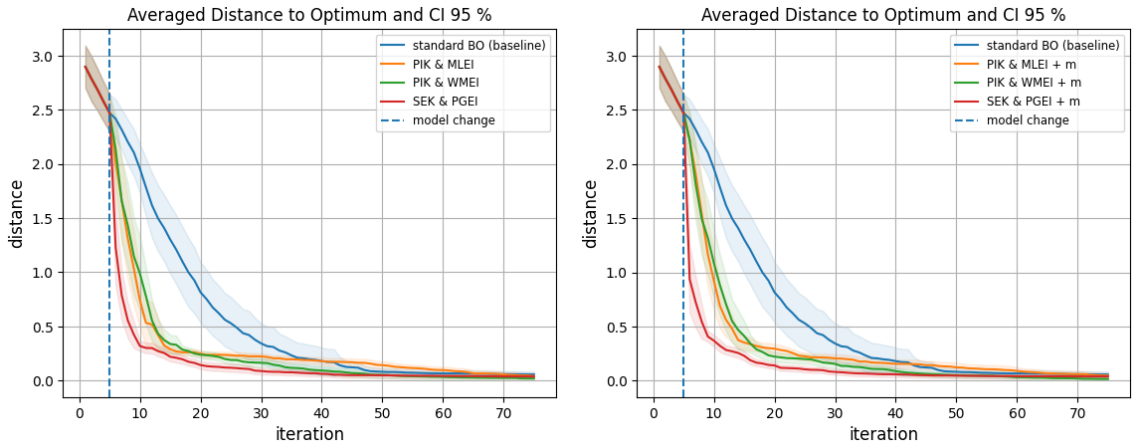
The observations for the best performing algorithms can be visually assessed in fig. 4.5, where the mean distances to the optimum are plotted for well performing algorithms in function output space. To keep the plots clear, the algorithms were divided in approaches with and without mismatch correction. The mean distances are complemented with the respective 95 % CI, which provides information about the divergence between optimization runs. These turned out especially small for the prior-informed algorithms, which indicates that the performance in function output space barely differs between optimization runs.

The plots for the remaining algorithms can be found as supplementary material in appendix A.2, in fig. A.1.

Results in Parameter Space From the averaged accuracies in parameter space, it can be seen, that many prior-informed algorithms move away from the true optimum in parameter space, throughout the first half of the optimization. In general, this is not critical, as long as the accuracy in function output space does not suffer. This is not the case, as long as the optimization does not get stuck in a local optimum and eventually discovers values close to the global optimum.

Accuracy in Function Output Space				
algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	72 % acc.	94 % acc.	98 % acc.	98 % acc.
PIK, EI	92 % (+28 %)	93 % (-1 %)	94 % (-4 %)	94 % (-4 %)
PIK, EI, m	92 % (+28 %)	93 % (-1 %)	94 % (-4 %)	95 % (-3 %)
PIK, WMEI	92 % (+28 %)	97 % (+3 %)	99 % (+1 %)	99 % (+1 %)
PIK, WMEI, m	93 % (+29 %)	97 % (+3 %)	99 % (+1 %)	100 % (+2 %)
PIK, MLEI	92 % (+28 %)	94 % (+0 %)	97 % (-1 %)	99 % (+1 %)
PIK, MLEI, m	91 % (+26 %)	95 % (+1 %)	97 % (-1 %)	99 % (+1 %)
PIK, PGEI	94 % (+31 %)	95 % (+1 %)	95 % (-3 %)	96 % (-2 %)
PIK, PGEI, m	93 % (+29 %)	94 % (+0 %)	94 % (-4 %)	95 % (-3 %)
WMK, EI	88 % (+22 %)	94 % (+0 %)	94 % (-4 %)	95 % (-3 %)
WMK, EI, m	90 % (+25 %)	94 % (+0 %)	94 % (-4 %)	95 % (-3 %)
WMK, MLEI	92 % (+28 %)	94 % (+0 %)	96 % (-2 %)	98 % (+0 %)
WMK, MLEI, m	92 % (+28 %)	94 % (+0 %)	96 % (-2 %)	97 % (-1 %)
WMK, PGEI	94 % (+31 %)	96 % (+2 %)	96 % (-2 %)	97 % (-1 %)
WMK, PGEI, m	93 % (+29 %)	95 % (+1 %)	96 % (-2 %)	96 % (-2 %)
SEK, PGEI	95 % (+32 %)	98 % (+4 %)	99 % (+1 %)	99 % (+1 %)
SEK, PGEI, m	96 % (+33 %)	98 % (+4 %)	99 % (+1 %)	99 % (+1 %)

Table 4.8: Hartmann6 tests with all simulation data sets: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

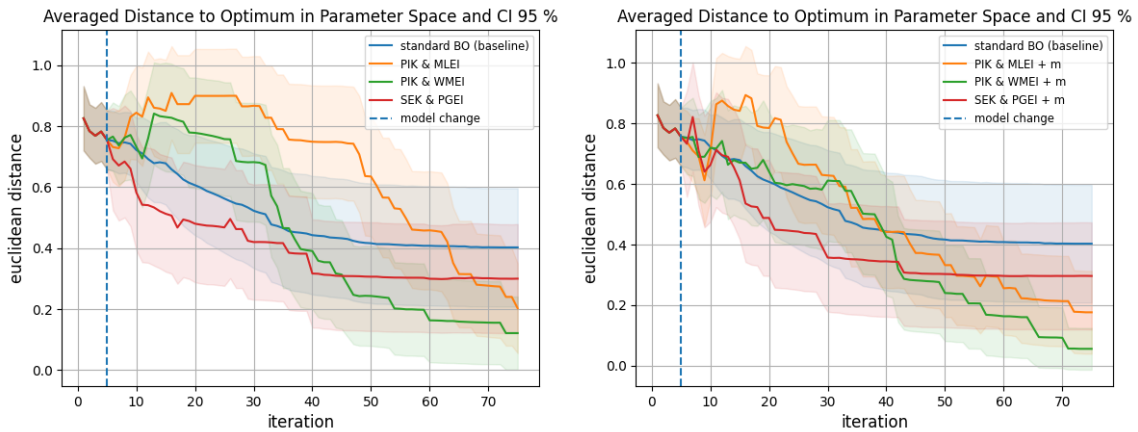
Figure 4.5: Hartmann6 tests with all simulation data sets for best performing algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

The algorithms with PIK in combination with WMEI and MLEI acquisition function, initially move away from the true optimum, but overcome the local extreme value and eventually outperform the accuracy of the baseline in parameter space. The algorithms with PGEI even directly approach the global optimum from the beginning. However, as previously stated, only the SEK with PGEI yields convincing performance in function output space, compared to standard BO.

In contrast to the well performing algorithms, approaches with EI acquisition function show the worst performance in parameter space, compared to the baseline and all other algorithms. One explanation for this behavior could be, that they are likely to get stuck at values, that look locally promising, but are located in another area of the parameter space, compared to the globally optimal parameters. However, this will be separately examined with the next evaluation criterion. Lastly, the approaches with WMK in combination with MLEI seem to overcome local extreme values in parameter space and approach the true optimum, however, in function output space it can not surpass the baseline's performance.

The plots in fig. 4.6 show the mean euclidean distances for the well performing algorithms in parameter space and additionally visualize the CIs, which are larger than in function output space. This indicates, that the history of evaluated parameters can differ between optimization runs, while the performance in function output space remains more or less stable. For Hartmann6 this makes sense, since the global and the local optimum lie in different parts of the parameter space, while yielding similar results in function output space. Additionally, for PIK with WMEI and MLEI, the trend becomes visible, that using a mismatch correction seems to accelerate overcoming local extreme values and finding the true optimum.

The plots for all remaining algorithms in parameter space can be found in fig. A.2.



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

Figure 4.6: Hartmann6 tests with all simulation data sets for best performing algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

Accuracy in Parameter Space				
algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	75 % acc. p.	82 % acc. p.	83 % acc. p.	84 % acc. p.
PIK, EI	67 % (-11 %)	73 % (-11 %)	77 % (-7 %)	80 % (-5 %)
PIK, EI, m	70 % (-7 %)	77 % (-6 %)	77 % (-7 %)	81 % (-4 %)
PIK, WMEI	68 % (-9 %)	84 % (+2 %)	92 % (+11 %)	95 % (+13 %)
PIK, WMEI, m	72 % (-4 %)	80 % (-2 %)	93 % (+12 %)	98 % (+17 %)
PIK, MLEI	64 % (-15 %)	69 % (-16 %)	81 % (-2 %)	92 % (+10 %)
PIK, MLEI, m	68 % (-9 %)	80 % (-2 %)	88 % (+6 %)	93 % (+11 %)
PIK, PGEI	75 % (+0 %)	82 % (+0 %)	85 % (+2 %)	83 % (-1 %)
PIK, PGEI, m	80 % (+7 %)	85 % (+4 %)	84 % (+1 %)	87 % (+4 %)
WMK, EI	65 % (-13 %)	76 % (-7 %)	74 % (-11 %)	77 % (-8 %)
WMK, EI, m	63 % (-16 %)	66 % (-20 %)	67 % (-19 %)	76 % (-10 %)
WMK, MLEI	70 % (-7 %)	70 % (-15 %)	82 % (-1 %)	90 % (+7 %)
WMK, MLEI, m	64 % (-15 %)	69 % (-16 %)	82 % (-1 %)	89 % (+6 %)
WMK, PGEI	78 % (+4 %)	87 % (+6 %)	87 % (+5 %)	87 % (+4 %)
WMK, PGEI, m	71 % (-5 %)	80 % (-2 %)	84 % (+1 %)	84 % (+0 %)
SEK, PGEI	80 % (+7 %)	84 % (+2 %)	88 % (+6 %)	88 % (+5 %)
SEK, PGEI, m	80 % (+7 %)	86 % (+5 %)	88 % (+6 %)	88 % (+5 %)

Table 4.9: Hartmann6 tests with all simulation data sets: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs

4.5.4 Overcoming Local Optima

In this passage, the ability to overcome local extreme values is assessed more accurately. Therefore, the algorithms are firstly tested with the set of fixed initial parameter configurations, introduced in section 4.4.3. These lead towards the area of the local extreme value of Hartmann6.

However, when evaluating if the prior-informed algorithms are likely to get stuck in local optima, an additional test case should be considered. Not only disadvantageous initial parameters are able to lead the optimization towards local extreme values. In case of prior-informed algorithms, logically also unsuitable simulation data, which overestimate local extreme values, can guide the search into a local optimum. Recalling section 4.2, where simulation data for Hartmann6 was introduced, the simulation data set 1 in fig. 4.1a, exactly reproduces this case. Simulation data set 1 is a noisy version of the six-dimensional Hartmann function and in general resembles the true objective function. However, in the area of the objective function's optimum there are mismatches between simulation data and true objective function. Since for Hartmann6 the local and global optimum lie close together in the function's output space, adding noise had the effect of underestimating the global extreme values, while overestimating the local ones. As a consequence for using only this set of misleading simulation data, the prior-informed algorithms will be directly led to the local optimum. Therefore, evaluations are made for a second test case, where only simulation data set 1 is used as prior information for the algorithms.

Test Case 1: All Simulation Data Sets with Fixed Initial Parameters

In tab. 4.10 and tab. 4.11 the results for testing the algorithms with the previously introduced fixed initial parameter configuration can be found in function output space and parameter space, respectively.

Results in Function Output Space In function output space, the accuracies of the prior-informed approaches mainly resemble the baseline.

One to two percent improvement can be achieved by the PIK with MLEI, WMK with MLEI and mismatch correction and the SEK with PGEI and mismatch correction. An improvement of up to four percent is visible for the PIK with the WMEI acquisition function.

The remaining algorithms yield either similar performances as the baseline or slightly worse, with a maximum decline in accuracy of two percent for approaches with EI.

Accuracy in Function Output Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	93 % acc.	94 % acc.	96 % acc.	96 % acc.
PIK, EI	93 % (+0 %)	94 % (+0 %)	94 % (-2 %)	95 % (-1 %)
PIK, EI, m	94 % (+1 %)	94 % (+0 %)	94 % (-2 %)	94 % (-2 %)
PIK, WMEI	95 % (+2 %)	100 % (+6 %)	100 % (+4 %)	100 % (+4 %)
PIK, WMEI, m	95 % (+2 %)	97 % (+3 %)	99 % (+3 %)	99 % (+3 %)
PIK, MLEI	93 % (+0 %)	96 % (+2 %)	97 % (+1 %)	98 % (+2 %)
PIK, MLEI, m	94 % (+1 %)	95 % (+1 %)	98 % (+2 %)	98 % (+2 %)
PIK, PGEI	94 % (+1 %)	94 % (+0 %)	95 % (-1 %)	96 % (+0 %)
PIK, PGEI, m	95 % (+2 %)	95 % (+1 %)	95 % (-1 %)	95 % (-1 %)
WMK, EI	94 % (+1 %)	94 % (+0 %)	94 % (-2 %)	95 % (-1 %)
WMK, EI, m	93 % (+0 %)	93 % (-1 %)	94 % (-2 %)	95 % (-1 %)
WMK, MLEI	93 % (+0 %)	94 % (+0 %)	94 % (-2 %)	95 % (-1 %)
WMK, MLEI, m	93 % (+0 %)	95 % (+1 %)	97 % (+1 %)	98 % (+2 %)
WMK, PGEI	94 % (+1 %)	96 % (+2 %)	96 % (+0 %)	96 % (+0 %)
WMK, PGEI, m	94 % (+1 %)	95 % (+1 %)	95 % (-1 %)	95 % (-1 %)
SEK, PGEI	93 % (+0 %)	95 % (+1 %)	95 % (-1 %)	96 % (+0 %)
SEK, PGEI, m	94 % (+1 %)	96 % (+2 %)	96 % (+0 %)	97 % (+1 %)

Table 4.10: Hartmann6 tests with all simulation data sets and fixed initial parameters: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

Results in Parameter Space In parameter space, the true differences between standard BO and prior-informed algorithms become apparent. Obviously, the four sets of utilized simulation data help the optimization to overcome the local extreme value in parameter space and expand the search more into the direction of the true optimum, even if it is not directly contained in the simulated data.

Those prior-informed algorithms, which showed slightly better performance than the

baseline in function output space, yield much higher averaged accuracies in parameter space with improvements up to 77 % compared to the baseline and therefore are more likely to find the true optimum than standard BO for initial parameters, leading to the local optimum of the objective function.

The remaining algorithms also yield better accuracies than the baseline in parameter space, which indicates that the simulation data guides them away from the local extreme value, closer to the global one. However, apparently they are on average neither able to exactly locate the true optimum of the function, nor the local optimum within the predefined budget of 75 function evaluations, since their performance in function output space declines compared to the baseline.

An outlier among the best performing algorithms is the approach with SEK and PGEI, which yields a small improvement in parameter space of only 16 %, which is even smaller than the improvement of algorithms, which did not yield convincing results in function output space. A possible explanation is, that in some cases the global optimum is discovered, which boosts the accuracy in function output space, while in other cases the algorithm however still finds the local extreme value.

In fig. 4.7, the mean distances to the global optimum of the objective function are plotted in function output and parameter space. To clearly identify the CIs, only the two best performing approaches were selected, namely PIK with WMEI with and without mismatch correction, in comparison to the baseline. The plots for the remaining algorithms can again be found in the supplementary material in fig. A.3 and fig. A.4.

From the little CI of the standard BO approach in fig. 4.7b, it can be clearly seen, that the baseline with a high probability never overcomes the local optimum in parameter space. With a similarly little CI, the WMEI without mismatch correction, is very likely to always reach parameters close to the global optimum for given sets of simulation data. On the other side, indicated by a rather large CI, WMEI with mismatch correction apparently in some cases gets close to the global optimum, while in other cases not. A possible explanation for this difference between these approaches with and without mismatch correction is, that without mismatch correction some or all simulation data are discarded at some point of the optimization, since they are assessed as unsuitable by the model selection criterion, however, after already guiding the search into the right direction of the search space. In contrast, for the approach with mismatch correction, multiple corrected sets of simulation data could be assessed as trustworthy throughout the whole optimization, depending on the exact values of previously evaluated parameters per optimization run and guiding the search into different directions of the search space, for different optimization runs, respectively. However, these differences between approaches with and without mismatch correction will be examined in more detail with the criterion of how prior-informed algorithms deal with unsuitable simulation data.

Accuracy in Parameter Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	55 % acc. p.	57 % acc. p.	57 % acc. p.	56 % acc. p.
PIK, EI	56 % (+2 %)	64 % (+12 %)	64 % (+12 %)	80 % (+43 %)
PIK, EI, m	71 % (+29 %)	79 % (+39 %)	79 % (+39 %)	79 % (+41 %)
PIK, WMEI	88 % (+60 %)	99 % (+74 %)	99 % (+74 %)	99 % (+77 %)
PIK, WMEI, m	71 % (+29 %)	89 % (+56 %)	90 % (+58 %)	90 % (+61 %)
PIK, MLEI	73 % (+33 %)	81 % (+42 %)	81 % (+42 %)	82 % (+46 %)
PIK, MLEI, m	63 % (+15 %)	88 % (+54 %)	90 % (+58 %)	91 % (+62 %)
PIK, PGEI	64 % (+16 %)	72 % (+26 %)	80 % (+40 %)	88 % (+57 %)
PIK, PGEI, m	79 % (+44 %)	79 % (+39 %)	79 % (+39 %)	79 % (+41 %)
WMK, EI	55 % (+0 %)	53 % (-7 %)	54 % (-5 %)	79 % (+41 %)
WMK, EI, m	55 % (+0 %)	64 % (+12 %)	72 % (+26 %)	88 % (+57 %)
WMK, MLEI	54 % (-2 %)	55 % (-4 %)	56 % (-2 %)	65 % (+16 %)
WMK, MLEI, m	63 % (+15 %)	80 % (+40 %)	89 % (+56 %)	98 % (+75 %)
WMK, PGEI	71 % (+29 %)	71 % (+25 %)	71 % (+25 %)	71 % (+27 %)
WMK, PGEI, m	71 % (+29 %)	71 % (+25 %)	80 % (+40 %)	81 % (+45 %)
SEK, PGEI	56 % (+2 %)	65 % (+14 %)	65 % (+14 %)	64 % (+14 %)
SEK, PGEI, m	64 % (+16 %)	66 % (+16 %)	66 % (+16 %)	65 % (+16 %)

Table 4.11: Hartmann6 tests with all simulation data sets and fixed initial parameters: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

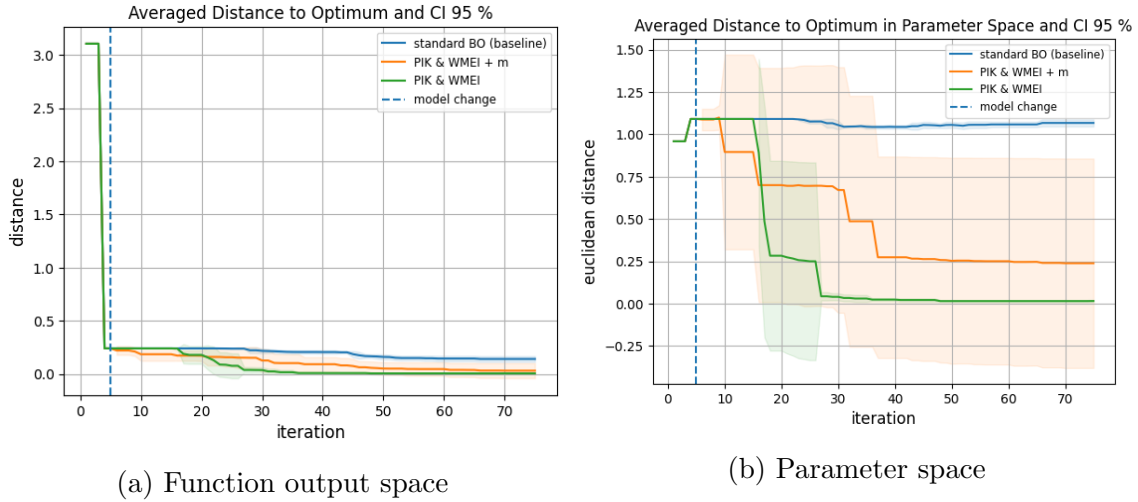


Figure 4.7: Hartmann6 tests with all simulation data sets and fixed initial parameters for best performing algorithms: mean distance to the global optimum and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, in function output space (a) and parameter space (b)

Test Case 2: Simulation Data Set 1

In tab. 4.12 and tab. 4.13 the test results are stated for the test case of utilizing only simulation data set 1, in function output space and parameter space, respectively. In this test case, the goal is to identify prior-informed algorithms, which are not more prone to get stuck in local optima, than the standard BO itself, when being exposed to respective simulation data. Thus, as well performing will be considered those algorithms, which yield accuracies at least as good as the baseline in function output space, as well as in parameter space, which is the better indicator for local extreme values.

Results in Function Output Space In function output space, the averaged accuracies for all prior-informed algorithms increase up to 93 % within only few function evaluations, following the guidance of the noisy Hartmann6 simulation data. After the maximum number of function evaluations, approaches with WMEI and MLEI reach similar performance as the baseline, while the other algorithms yield less accuracy.

Accuracy in Function Output Space				
algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	72 % acc.	94 % acc.	98 % acc.	98 % acc.
PIK, EI	91 % (+26 %)	92 % (-2 %)	93 % (-5 %)	94 % (-4 %)
PIK, EI, m	90 % (+25 %)	93 % (-1 %)	93 % (-5 %)	94 % (-4 %)
PIK, WMEI	91 % (+26 %)	95 % (+1 %)	97 % (-1 %)	98 % (+0 %)
PIK, WMEI, m	89 % (+24 %)	94 % (+0 %)	96 % (-2 %)	98 % (+0 %)
PIK, MLEI	92 % (+28 %)	95 % (+1 %)	97 % (-1 %)	98 % (+0 %)
PIK, MLEI, m	91 % (+26 %)	95 % (+1 %)	97 % (-1 %)	99 % (+1 %)
PIK, PGEI	92 % (+28 %)	93 % (-1 %)	93 % (-5 %)	94 % (-4 %)
PIK, PGEI, m	91 % (+26 %)	93 % (-1 %)	94 % (-4 %)	94 % (-4 %)
WMK, EI	87 % (+21 %)	94 % (+0 %)	95 % (-3 %)	95 % (-3 %)
WMK, EI, m	88 % (+22 %)	92 % (-2 %)	93 % (-5 %)	94 % (-4 %)
WMK, MLEI	86 % (+19 %)	95 % (+1 %)	97 % (-1 %)	99 % (+1 %)
WMK, MLEI, m	88 % (+22 %)	93 % (-1 %)	96 % (-2 %)	98 % (+0 %)
WMK, PGEI	91 % (+26 %)	93 % (-1 %)	94 % (-4 %)	94 % (-4 %)
WMK, PGEI, m	92 % (+28 %)	93 % (-1 %)	94 % (-4 %)	95 % (-3 %)
SEK, PGEI	93 % (+29 %)	96 % (+2 %)	96 % (-2 %)	97 % (-1 %)
SEK, PGEI, m	93 % (+29 %)	96 % (+2 %)	96 % (-2 %)	97 % (-1 %)

Table 4.12: Hartmann6 tests with simulation data set 1: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs

Results in Parameter Space In parameter space, it becomes clear that all prior-informed algorithms move away from the true optimum in parameter space during the half of the optimization procedure, following the simulation data to the local extreme value.

However, within the second half of function evaluations, some algorithms manage to increase the accuracy in parameter space and thus overcome the local optimum. After 75 function evaluations, the algorithms with WMEI and MLEI and mismatch correction yield results similar to the baseline. Approaches with WMK and MLEI even outperform the baseline, with both mismatch correction options.

In contrast, PIK without mismatch correction in combination with WMEI or MLEI is not able to reach a similar accuracy as the baseline in parameter space, even if the accuracy in function output space was similar to the baseline. Therefore, it is likely that these approaches are slightly more prone to resign at a local extreme value, than their counterparts with mismatch correction.

The remaining algorithms can not convince in this test scenario. Especially sensitive to the local optimum, which is overestimated by the used simulation data, seems to be the algorithm with SEK and PGEI. It yields the smallest accuracies in parameter space, while being only slightly worse than the baseline in function output space. For the Hartmann6 test function, this is a strong indicator that the algorithm mostly discovers parameters close to the local optimum.

In fig. 4.8, previous analysis for the best performing algorithms is complemented with the visual presentation of the average mean distances of the best observed values so far to the function's global optimum, as well as confidence intervals. Additionally, in fig. 4.9 the respective mean distances to the local optimum are plotted in parameter space for the best performing algorithms, as well as the worst performing ones. From the parameter space of the best performing approaches in fig. 4.9a, it can be seen that the optimizations at first follow the simulation data into the direction of the local optimum, before discovering the area of the global optimum. Especially, the algorithm with WMK and MLEI without mismatch correction stands out in quickly overcoming local extreme values. On the contrary, fig. 4.9b shows the respective plot for the worst performing approaches regarding this evaluation criterion, namely the algorithms with PGEI acquisition function, which is especially sensitive to get stuck in local optima.

The plots of the mean distances to the global optimum for the remaining algorithms can be found in fig. A.5 for the function output space and in fig. A.6 for the parameter space.

4.5.5 Dealing with Unsuitable Simulation Data

This passage will analyze how the individual algorithms deal with the exposure to unsuitable simulation data. Especially, the differences between approaches with mismatch correction and without mismatch correction will be examined.

To facilitate the analysis, a new test scenario will be introduced. The tests will be performed on only one set of unsuitable simulation data, namely the simulation data set 4, introduced in section 4.2, fig. 4.1d. This simulation data set resembles a six-dimensional polynomial function, which is not related to Hartmann6. Assessed will be the performance of the algorithms in combination with the amount of prior information, used from the simulation data. Since the used simulation data set does not resemble the true objective function and thus most likely does not contain

beneficial prior information, the goal is to identify those prior-informed algorithms which are not impeded by unsuitable simulation data and yield performances at least as good as standard BO in function output space.

Accuracy in Parameter Space

algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	75 % acc. p.	82 % acc. p.	83 % acc. p.	84 % acc. p.
PIK, EI	59 % (-21 %)	66 % (-20 %)	73 % (-12 %)	74 % (-12 %)
PIK, EI, m	65 % (-13 %)	76 % (-7 %)	81 % (-2 %)	82 % (-2 %)
PIK, WMEI	65 % (-13 %)	68 % (-17 %)	72 % (-13 %)	76 % (-10 %)
PIK, WMEI, m	62 % (-17 %)	67 % (-18 %)	76 % (-8 %)	83 % (-1 %)
PIK, MLEI	59 % (-21 %)	68 % (-17 %)	75 % (-10 %)	79 % (-6 %)
PIK, MLEI, m	66 % (-12 %)	70 % (-15 %)	77 % (-7 %)	89 % (+6 %)
PIK, PGEI	67 % (-11 %)	74 % (-10 %)	77 % (-7 %)	77 % (-8 %)
PIK, PGEI, m	70 % (-7 %)	72 % (-12 %)	75 % (-10 %)	75 % (-11 %)
WMK, EI	70 % (-7 %)	74 % (-10 %)	84 % (+1 %)	85 % (+1 %)
WMK, EI, m	64 % (-15 %)	67 % (-18 %)	68 % (-18 %)	68 % (-19 %)
WMK, MLEI	73 % (-3 %)	81 % (-1 %)	86 % (+4 %)	97 % (+15 %)
WMK, MLEI, m	64 % (-15 %)	72 % (-12 %)	82 % (-1 %)	90 % (+7 %)
WMK, PGEI	61 % (-19 %)	64 % (-22 %)	70 % (-16 %)	73 % (-13 %)
WMK, PGEI, m	64 % (-15 %)	72 % (-12 %)	75 % (-10 %)	77 % (-8 %)
SEK, PGEI	57 % (-24 %)	62 % (-24 %)	62 % (-25 %)	63 % (-25 %)
SEK, PGEI, m	60 % (-20 %)	62 % (-24 %)	63 % (-24 %)	63 % (-25 %)

Table 4.13: Hartmann6 tests with simulation data set 1: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 30 optimization runs

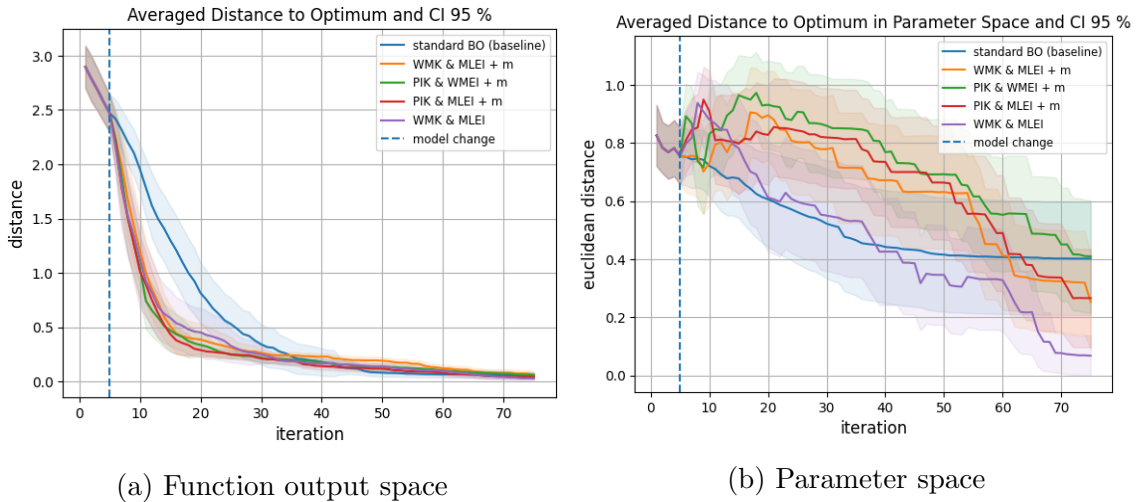


Figure 4.8: Hartmann6 tests with simulation data set 1 for best performing algorithms: mean distance to the global optimum and confidence interval 95 %, averaged over 30 optimization runs, in function output space (a) and parameter space (b)

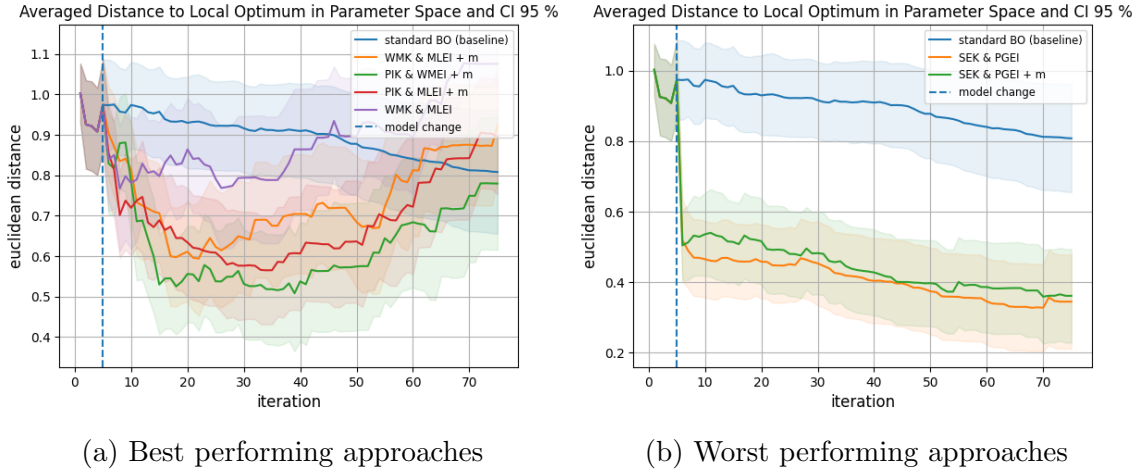


Figure 4.9: Hartmann6 tests with simulation data set 1 for selected algorithms: mean distance to the local optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for best performing approaches (a) and worst performing approaches (b)

Test Case 3: Simulation Data Set 4

Accuracy In tab. 4.14 and tab. 4.15 the accuracies in function output and parameter space can be found.

The results show, that for most algorithms the accuracies resemble the baseline. However, some algorithms with mismatch correction are outperformed by standard BO in function output, as well as in parameter space. These are approaches with PIK or WMK in combination with EI, as well as the approaches with PGEI in combination with PIK and WMK. An additional accuracy drop can be observed for the WMK and EI without mismatch correction during the first half of the optimization. However, for the second half, the accuracy again resembles the baseline.

The remaining algorithms with mismatch correction, which yield good performance in function output space, still show a slight decrease in accuracy in parameter space. An example are the approaches with PIK and WMEI or MLEI acquisition function. Since the performance drop of those algorithms does not exceed more than two percent, compared to the baseline, and does not noticeably affect the accuracy in function output space, the approaches can still be considered as well performing.

The corresponding plots for this test case, which show the mean distances to the true optimum of Hartmann6 and respective CIs, can be found in fig. A.7 and fig. A.8.

To better understand the observed performance differences between algorithms with and without mismatch correction, the following sections examine the behavior of the algorithms by firstly having a closer look at the amount of prior information, which is used throughout the optimization, and secondly explaining the reasons and consequences of observed behavior.

Accuracy in Function Output Space				
algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	72 % acc.	94 % acc.	98 % acc.	98 % acc.
PIK, EI	69 % (-4 %)	95 % (+1 %)	98 % (+0 %)	99 % (+1 %)
PIK, EI, m	53 % (-26 %)	75 % (-20 %)	85 % (-13 %)	87 % (-11 %)
PIK, WMEI	72 % (+0 %)	95 % (+1 %)	98 % (+0 %)	98 % (+0 %)
PIK, WMEI, m	76 % (+6 %)	97 % (+3 %)	98 % (+0 %)	98 % (+0 %)
PIK, MLEI	73 % (+1 %)	94 % (+0 %)	98 % (+0 %)	98 % (+0 %)
PIK, MLEI, m	70 % (-3 %)	96 % (+2 %)	98 % (+0 %)	98 % (+0 %)
PIK, PGEI	72 % (+0 %)	98 % (+4 %)	99 % (+1 %)	99 % (+1 %)
PIK, PGEI, m	60 % (-17 %)	80 % (-15 %)	86 % (-12 %)	88 % (-10 %)
WMK, EI	59 % (-18 %)	89 % (-5 %)	97 % (-1 %)	98 % (+0 %)
WMK, EI, m	73 % (+1 %)	91 % (-3 %)	93 % (-5 %)	93 % (-5 %)
WMK, MLEI	74 % (+3 %)	95 % (+1 %)	98 % (+0 %)	98 % (+0 %)
WMK, MLEI, m	75 % (+4 %)	95 % (+1 %)	98 % (+0 %)	98 % (+0 %)
WMK, PGEI	63 % (-12 %)	98 % (+4 %)	98 % (+0 %)	99 % (+1 %)
WMK, PGEI, m	76 % (+6 %)	92 % (-2 %)	94 % (-4 %)	94 % (-4 %)
SEK, PGEI	75 % (+4 %)	96 % (+2 %)	98 % (+0 %)	99 % (+1 %)
SEK, PGEI, m	74 % (+3 %)	96 % (+2 %)	98 % (+0 %)	98 % (+0 %)

Table 4.14: Hartmann6 tests with simulation data set 4: accuracy (acc.) in function output space in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

Used Prior Information In fig. 4.11, a visualization is given for assessing which models are predominantly chosen by the model selection criterion and used in each iteration within the acquisition function for 30 optimization runs. Therefore, one plot shows in total 30 markers at each iteration step, located at the y-location of the respective used model. To make differences better visible, the marker sizes enlarge with the quantity of markers, gathered at the respective model for at a certain iteration. This representation was chosen to make assessments also possible for algorithms, which choose between more than two models. In this case, determining the mean between used models would lead to wrong results. For example, the algorithm with WMK chooses between the standard GP, the GP with prior-informed kernel and the GP with composed kernel. Here, determining a simple mean between models with y-location $y = 0$ and $y = 2$, would yield the model at $y = 1$ as the model, which was used on average, although this model was not used at all.

In fig. 4.11a and fig. 4.11b the used models are plotted for the PIK in combination with EI, without and with mismatch correction, respectively. For the EI acquisition function, simply the model with the highest model selection score is selected in each iteration and used within the EI acquisition function. The plots make the difference between the options for mismatch correction clearly visible. Without correcting the mismatches between simulation data and function observations, mainly the stan-

Accuracy in Parameter Space				
algorithm	19 eval.	38 eval.	57 eval.	75 eval.
BO (baseline)	75 % acc. p.	82 % acc. p.	83 % acc. p.	84 % acc. p.
PIK, EI	76 % (+1 %)	85 % (+4 %)	88 % (+6 %)	88 % (+5 %)
PIK, EI, m	72 % (-4 %)	72 % (-12 %)	73 % (-12 %)	70 % (-17 %)
PIK, WMEI	75 % (+0 %)	82 % (+0 %)	84 % (+1 %)	84 % (+0 %)
PIK, WMEI, m	75 % (+0 %)	81 % (-1 %)	82 % (-1 %)	82 % (-2 %)
PIK, MLEI	77 % (+3 %)	83 % (+1 %)	85 % (+2 %)	85 % (+1 %)
PIK, MLEI, m	74 % (-1 %)	81 % (-1 %)	82 % (-1 %)	82 % (-2 %)
PIK, PGEI	80 % (+7 %)	88 % (+7 %)	89 % (+7 %)	89 % (+6 %)
PIK, PGEI, m	73 % (-3 %)	73 % (-11 %)	74 % (-11 %)	75 % (-11 %)
WMK, EI	73 % (-3 %)	83 % (+1 %)	86 % (+4 %)	86 % (+2 %)
WMK, EI, m	74 % (-1 %)	77 % (-6 %)	78 % (-6 %)	78 % (-7 %)
WMK, MLEI	76 % (+1 %)	84 % (+2 %)	85 % (+2 %)	85 % (+1 %)
WMK, MLEI, m	72 % (-4 %)	78 % (-5 %)	80 % (-4 %)	81 % (-4 %)
WMK, PGEI	78 % (+4 %)	87 % (+6 %)	87 % (+5 %)	88 % (+5 %)
WMK, PGEI, m	75 % (+0 %)	80 % (-2 %)	81 % (-2 %)	81 % (-4 %)
SEK, PGEI	79 % (+5 %)	86 % (+5 %)	89 % (+7 %)	89 % (+6 %)
SEK, PGEI, m	74 % (-1 %)	80 % (-2 %)	82 % (-1 %)	82 % (-2 %)

Table 4.15: Hartmann6 tests with simulation data set 4: accuracy in parameter space (acc. p.) in comparison to the baseline after 19, 38, 57 and 75 function evaluations (eval.), averaged over 5 optimization runs with fixed initial parameter configurations

standard model is used throughout the whole optimization. The simulation data of the polynomial function is slightly utilized for some optimization runs, only during the first iterations, but then discarded. On the contrary, for the algorithm with mismatch correction, mainly the model with prior-information from the simulation data is chosen, whereas the standard model is rarely chosen from time to time. Obviously, the prior-informed model with unsuitable simulation data and corrected mismatch seems more suitable to the model selection criterion, than the standard GP. However, apparently this is a misconception, since the accuracies in function output- and parameter space noticeable drops for this approach in comparison to standard BO.

In case of the WMEI, a weighted mixture acquisition function is created, between the EI scores of the standard GP and the best performing prior-informed model. The weighted mixture is only created if any prior-informed model yields a higher model selection score, than the standard GP. If this is not the case, only the standard model is used to not possibly worsen the performance. The chosen models in fig. 4.11c and fig. 4.11d look similar to the results of EI. This makes sense, because just like before, the model with the highest model selection score is chosen. However, this time, the selected prior-informed GP is not solely used within the acquisition function. Whenever the prior-informed model is chosen, it is used within the weighted mixture of the models' EIs. Therefore, the weights, used within the composed WMEI acquisition function are averaged over 30 optimization runs and

depicted in fig. 4.10a and fig. 4.10b for approaches without and with mismatch correction, respectively. For the WMEI, the weights themselves are dependent on the similarity of model selection scores of the considered models. For instance, identical performance of the models would lead to a 50/50 weighting. Only if one prior-informed model would outperform the standard GP by far, the standard model's weight would approach zero and thus, would not be considered. From fig. 4.10a, it is noticeable, that the algorithm without mismatch correction on average only uses a small share of maximum 20 % of the prior-informed model's information during the first iterations of the optimization. On the contrary, in fig. 4.10b, it can be seen, that even if the prior-informed model was selected by the model selection criterion, still 20 % to 40 % of the information, used within the acquisition function, comes from the standard model. By also leveraging the standard GP, the PIK with WMEI and mismatch correction yields higher accuracies than the PIK with EI and mismatch correction.

For MLEI, the model with the highest product of model selection score and EI was plotted, indicating from which model the respective point in parameter space was chosen. Again, two separate plots show the algorithms without and with mismatch correction in fig. 4.11e and fig. 4.11f, respectively. Here, the difference between the approach with and without mismatch correction is less apparent, which also explains their minor difference in yielded accuracies. Both algorithms mainly use the standard GP throughout the optimization. The approach without mismatch correction slightly increases the use of the prior-informed model in the second half of the optimization process, whereas the approach with mismatch correction utilizes the prior-informed GP only within the first quarter of the procedure with a noteworthy occurrence. Apparently, even if the model selection criterion would predominantly choose a prior-informed model with corrected mismatch over the standard model, as seen for the case of EI, the MLEI acquisition function still mainly chooses points, proposed by the standard GP, where EI scores are high enough to overrule the model selection criterion.

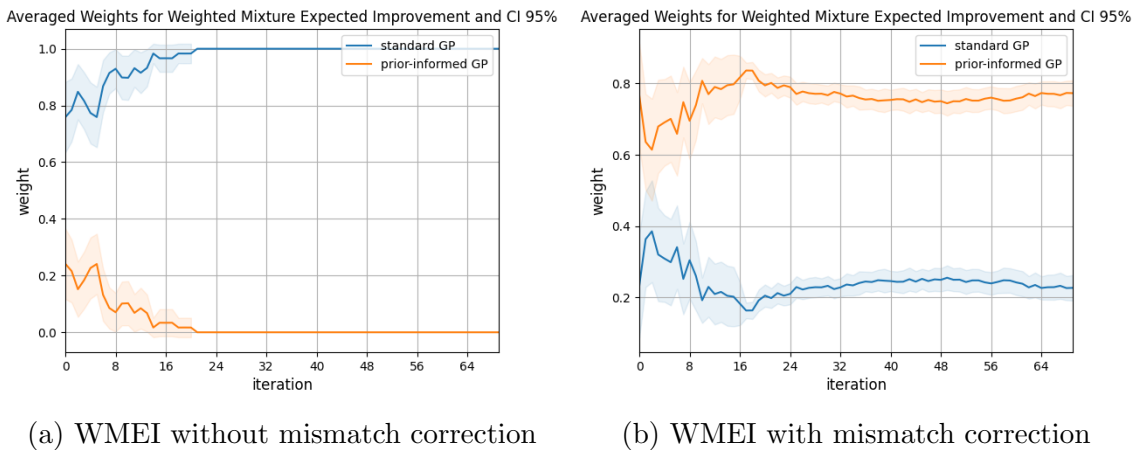
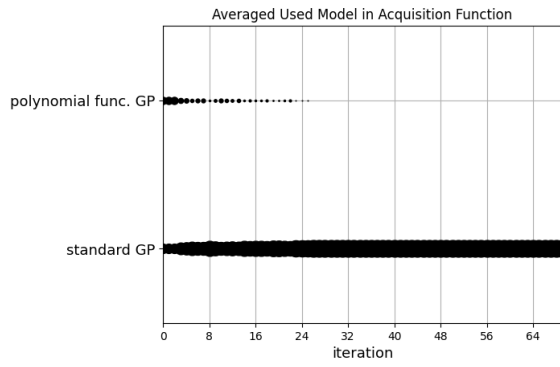
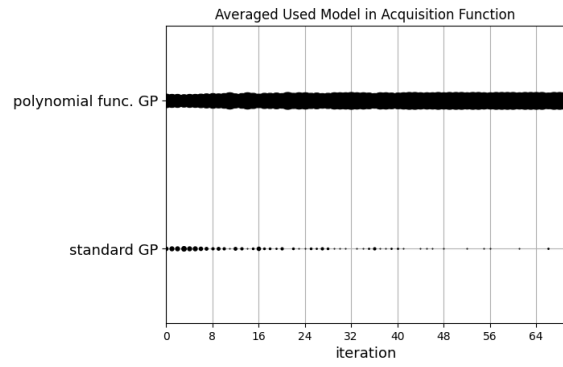


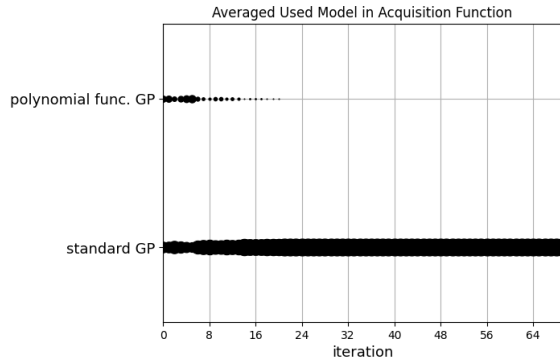
Figure 4.10: Hartmann6 tests with simulation data set 4: weights of used models in the Weighted Mixture Expected Improvement, averaged over 30 optimization runs, without mismatch correction (a) and with mismatch correction (b)



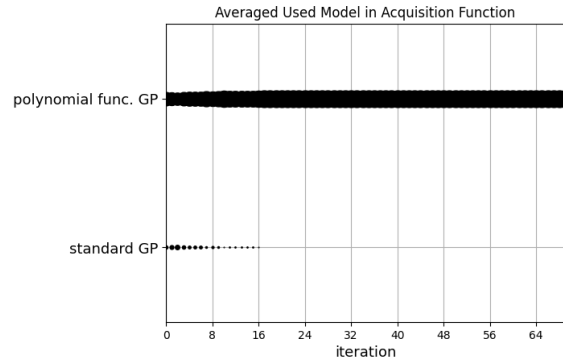
(a) PIK, EI, no mismatch correction



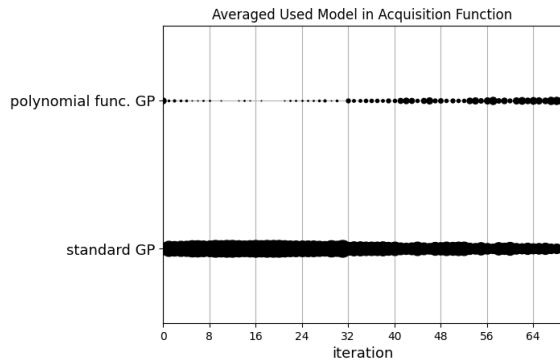
(b) PIK, EI, with mismatch correction



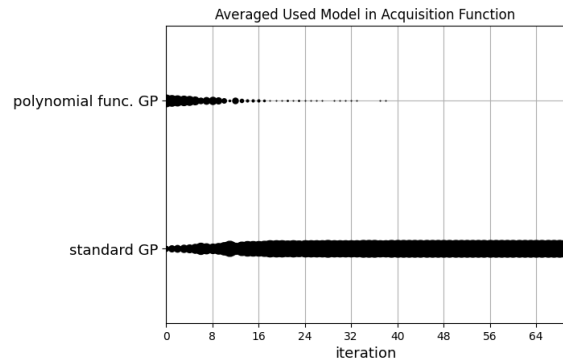
(c) PIK, WMEI, no mismatch correction



(d) PIK, WMEI, with mismatch correction



(e) PIK, MLEI, no mismatch correction



(f) PIK, MLEI, with mismatch correction

Figure 4.11: Hartmann6 tests with simulation data set 4 for algorithms with prior-informed kernel: used model per iteration for 30 optimization runs, for algorithms with and without mismatch correction and different acquisition functions

Next, it will be assessed which models are used within the acquisition function for the case of choosing between three model, namely, the standard GP, the prior-informed GP and an additional model with WMK. In fig. 4.12 selected models are visualized for EI and MLEI without and with mismatch correction.

In fig. 4.12a and fig. 4.12b, again, the difference can be clearly seen between the model choices for an approach without mismatch and an approach with mismatch correction, both with EI acquisition function. The weights for the sub-kernels of the WMK can be found in fig. 4.13. Without mismatch correction, the model selection criterion mainly chooses the GP with WMK. From the weighting in fig. 4.13a it can be seen, that information from the standard model significantly prevails throughout the optimization, except for the first optimization iterations, which explains the observed accuracy drop during the first iterations. In the last half of the optimization procedure, where the accuracy again resembles the baseline, the prior-informed sub-kernel is indeed only rarely used. On the contrary, for the algorithm with mismatch correction the weighted mixture GP is predominantly used during the first half of the optimization. From fig. 4.13b, it can be seen that the prior-informed sub-kernel dominates the WMK throughout the optimization, but still uses information from the standard kernel. For the second half of optimization iterations, the prior-informed GP, without information from the standard model, is chosen by the model selection criterion. From this analysis it is again obvious, that the model selection criterion tends to choose a model with unsuitable prior information and corrected mismatch over the standard model. Because the accuracy again decreases for this algorithm with mismatch correction, in comparison to the baseline, the model selection criterion again seems to overestimate the model with corrected simulation data.

For the WMK with MLEI, the weighting of the sub-kernels looks similar to the previous case of EI. The exact curves can be found in fig. A.9. The approach with WMK and MLEI without corrected mismatch again mostly uses the weighted mixture model with prevailing information from the standard GP, as plotted in fig. 4.12c, which explains its accuracies, similar to the baseline. However, in some optimization runs also the standard GP alone, as well as the prior-informed GP are used.

In fig. 4.12d the model choice for MLEI with mismatch correction is shown. MLEI still selects mainly points, proposed by the standard model, which explains why the accuracies do not drop as much as for the approach with mismatch correction and EI. The standard model apparently again yields EI scores, which are high enough to overrule the overly optimistic model selection score, which would favor the model with predominant information from corrected simulation data.

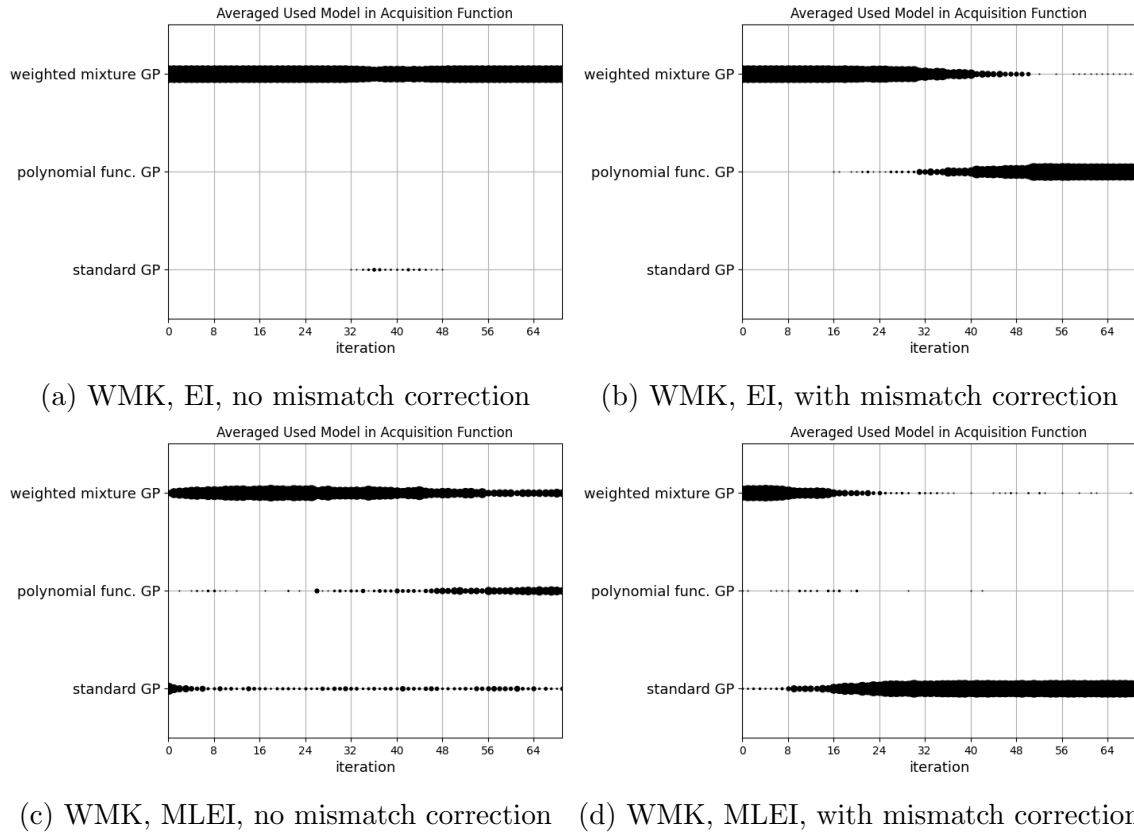


Figure 4.12: Hartmann6 tests with simulation data set 4 for algorithms with Weighted Mixture Kernel: used model per iteration for 30 optimization runs, for algorithms with and without mismatch correction and different acquisition functions

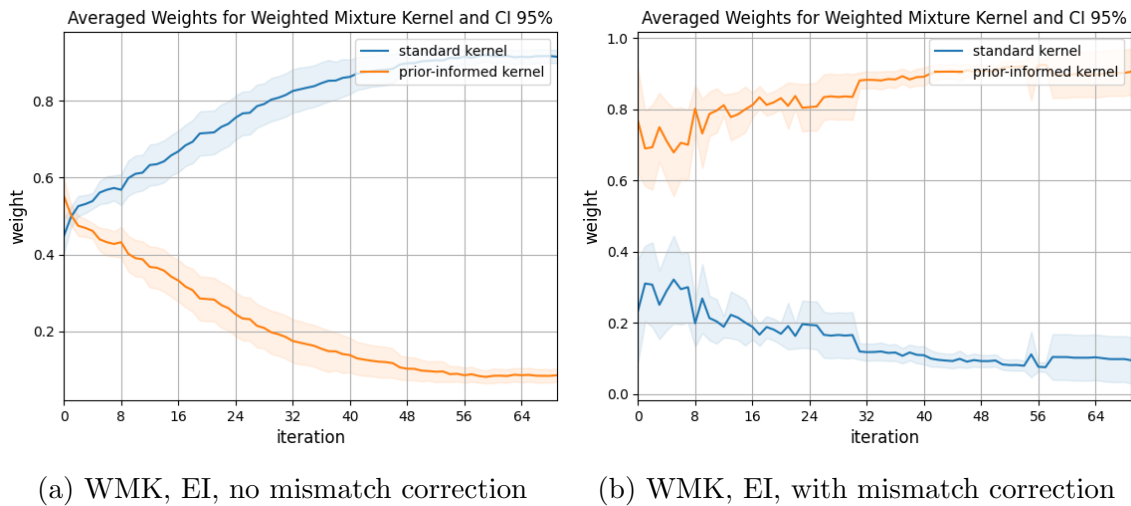


Figure 4.13: Hartmann6 tests with simulation data set 4: weights of used sub-kernels in the Weighted Mixture Kernel, averaged over 30 optimization runs with Expected Improvement, without mismatch correction (a) and with mismatch correction (b)

Lastly, one algorithm with PGEI will be examined, where the approach with standard SEK was chosen. The plots in fig. 4.14 show the weights of the prior information within the PGEI for an approach without and with mismatch correction, respectively. For both cases, the prior information is only used during the first half of the optimization procedure, indicated by a weight unequal to zero. This means that for the second half of the optimization iterations, the simulation data information is suppressed and the standard EI acquisition function is used. For the case with mismatch correction, the prior information is suppressed slightly faster, and takes values of exactly zero during the second half of the iterations, while the weight without mismatch correction still stays slightly positive.

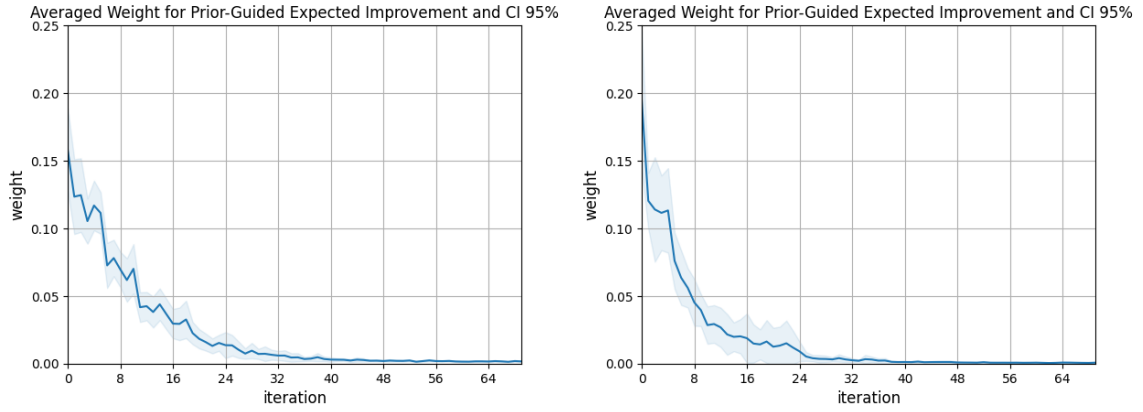


Figure 4.14: Hartmann6 tests with simulation data set 4: weighting of the simulation data in the Prior-Guided Expected Improvement, averaged over 30 optimization runs, without mismatch correction (a) and with mismatch correction (b)

Reasons and Consequences In the previous analysis, it was observed that the model selection criterion discards unsuitable simulation data after a few optimization iterations for algorithms without mismatch correction. After gathering some function observations, a meaningful standard GP model can be built, which yields higher model selection scores than the prior-informed GP model with unsuitable simulation data. Consequently, this leads to performance similar to standard BO for all algorithms without mismatch correction.

However, in case of correcting mismatches between simulation data and function observations, the model selection criterion predominantly chooses the prior-informed model. Here, the predictive performance of the corrected prior-informed GP is obviously overestimated, since the accuracy in function output, as well as parameter space considerably drops, when using only this model in the EI acquisition function. A possible reason for this behavior is the way of calculating the model selection criterion for a GP model with mismatch correction, which is included in the kernel. As explained in section 3.3.2, the mismatch correction is performed by creating an additional GP, denoted as the mismatch GP. A mismatch regression model is created by conditioning the mismatch GP on the deviations between observed function evaluations and respective simulation data values, to obtain its predictive posterior

mean. This predictive posterior mean is used to predict the mismatches of arbitrary points within the simulation data. The mismatch correction is included into the kernel of the prior-informed GP. To determine the predictive performance of the prior-informed GP, the Monte Carlo CV is computed with the predictive posterior probability. To calculate this predictive score, the observation data is separated into a training data set and a set of withheld data points for testing the predictive performance of the model. The prior-informed GP is conditioned on the training data set, and the predictive posterior probability is determined on the withheld data points. However, the predictive posterior mean of the mismatch GP, which is incorporated in the prior-informed GP's kernel function, is still conditioned on all deviations, also those, which belong to withheld observations. Therefore, the Monte Carlo CV uses information about the mismatches of data points, which are actually considered as unseen to the model. As a result, the fit of the prior-informed GP with mismatch correction is possibly overestimated for withheld data points.

Luckily, there are alternative acquisition functions to the EI acquisition function, which not solely rely on the previous choice of the model selection criterion. One opportunity is to use the MLEI acquisition function, which includes the EI scores of the models into the model selection criterion. Thus, high EI scores of the standard GP have the effect of balancing out the overly optimistic choice of the model selection criterion, which overestimates the fit of the corrected prior-informed model on the observations. The performance of the algorithm with PIK and MLEI and mismatch correction is therefore comparable to the baseline. The WMK with MLEI shows similar behavior regarding the utilized information, however in parameter space a slightly higher accuracy decrease was visible. This could be explained by the fact, that on average the simulation data was slightly more used, than for the PIK with MLEI.

Similarly, the WMEI acquisition function is able to deal with a corrected prior-informed model, which is overestimated by the model selection criterion. WMEI uses a weighted mixture of the EI scores of the prior informed model and the standard model. Therefore, high EI scores of the standard GP, again have the effect of balancing out poor choices, made by the model selection criterion. This is still dependent on the weight of the EI scores of the standard GP, which itself is dependent on the similarity of model selection scores. However, even if the model selection score of the prior-informed model is higher than the model selection score of the standard GP, there is still a high chance of considering the standard model's information.

Lastly, the PGEI acquisition function with standard kernel seems to be less sensitive to corrected simulation data, which is overestimated by the predictive posterior probability. The PGEI discards the unsuitable simulation data in both cases without and with mismatch correction, after slightly using it for the first quarter to half of optimization iterations. The reason is, that for the PGEI, in any case, the simulation data information is usually not used during the entire optimization process. It should be recalled, that the weight, which tunes the influence of the simulation data, is determined by maximizing the predictive posterior probability. During the phase where only a few function observations are gathered, and the standard model is not yet a meaningful representative of the true objective function, adding weighted

simulation data to the posterior mean, can possibly increase the predictive posterior probability on withheld data and help the optimization to quickly find the optimum in simulation data space. However, as soon as the standard GP is conditioned on enough function observations, the predictive posterior mean usually fits the objective function well. Then, an additive term to the predictive posterior mean is unnecessary and even worsens the predictive probability, which leads to discarding the simulation data in any case, with or without mismatch correction. This behavior of the PGEI also explains its liability to local optima, which is overestimated within the simulation data, as previously observed for the criterion of overcoming local extreme values. If the simulation data is not completely unsuitable on the initial function observations, the PGEI will immediately lead the optimization to the local extreme values. Before the simulation data is significantly corrected by gathered observations from the local optimum, it is already discarded and not further used by the PGEI. Without the use of simulation data, the PGEI transitions into the standard EI acquisition function. As shown in section 4.4.3, the BO algorithm with EI acquisition function alone is not able to overcome the local extreme value.

4.5.6 Comparison of the Algorithms

Based on previous assessments, the algorithms can now be compared regarding the individual evaluation criteria. In tab. 4.16, the algorithms are listed, together with the four evaluation criteria. The standard BO approach is listed as the reference and rated as reasonable ("r") regarding required number of function evaluations ("no. eval.") and achieved accuracy. However, for the criterion of overcoming a local optimum, the standard BO does not show convincing results for initial parameters, guiding towards the local extreme value. Therefore, it is rated with "x", which denotes bad performance. How the algorithm deals with an exposure to unsuitable simulation data is logically not assessed for the standard BO.

Regarding the criterion of finding well performing parameters with an accuracy in function output space of 90 % and higher within a few function evaluations, all prior-informed algorithms are able to outperform the baseline for the case of four utilized simulation data sets, which do not contain the global optimum of the test function. Therefore, all algorithms are at least rated with "g", which denotes good performance. Especially, algorithms with PGEI acquisition function are able to find well performing parameters outstandingly fast. Therefore, their performance is rated as "vg", denoting very good results.

However, for the criterion of reaching a high accuracy of the true optimum in function output, as well as in parameter space, most algorithms with PGEI fall back behind the baseline. Only the PGEI with SEK yields very good performance in function output space and good performance in parameter space, which was overall rated as good performance. From the approaches with WMK, the MLEI acquisition function shows reasonable performance, similar to the baseline. Good and very good accuracies in function output, as well as in parameter space, are observed for approaches with PIK and MLEI or WMEI acquisition function, respectively.

Moreover, the algorithms were assessed in two test scenarios with circumstances, which are likely to guide them to local extreme values. The WMK with MLEI con-

vinced with reasonable to good performance for the first test case and very good performance for the second test case, which was overall ranked as a good performance. Additionally, the PIK with either WMEI or MLEI, both without correcting mismatches, provided reasonable performance, by outperforming the baseline in the first test case and reaching only slightly lower accuracies as the reference in the second test case. However, their counterparts with mismatch correction could convince in both test cases with very good performance, compared to the baseline.

Subsequently, the algorithms were exposed to unsuitable simulation data and analyzed regarding the amount of utilized prior information, used throughout the optimization procedure. All algorithms, without mismatch correction, yielded good results, resembling the baselines performance or even slightly outperforming it. However, algorithms with mismatch correction mostly overestimated the fit of the prior-informed model on observed data and mostly showed a decrease in accuracy compared to the standard BO. The PIK in combination with WMEI or MLEI can still be ranked as good, since there was no noteworthy accuracy decline in function output space and only a small accuracy drop in parameter space. Likewise, WMEI with mismatch correction and MLEI acquisition function still showed reasonable performance, in comparison to the baseline and other prior-informed approaches. Lastly, the PGEI approach with standard kernel, turned out as robust to unsuitable simulation data and therefore is also ranked as well performing.

Algorithm Comparison

algorithm	1. no. eval.	2. accuracy	3. loc. optima	4. bad sim. data
BO (baseline)	r	r	x	-
PIK, EI	g	x	x	g
PIK, EI, m	g	x	x	x
PIK, WMEI	g	vg	r	g
PIK, WMEI, m	g	vg	vg	g
PIK, MLEI	g	g	r	g
PIK, MLEI, m	g	g	vg	g
PIK, PGEI	vg	x	x	g
PIK, PGEI, m	vg	x	x	x
WMK, EI	g	x	x	g
WMK, EI, m	g	x	x	x
WMK, MLEI	g	r	g	g
WMK, MLEI, m	g	r	g	r
WMK, PGEI	vg	x	x	g
WMK, PGEI, m	vg	x	x	r
SEK, PGEI	vg	g	x	g
SEK, PGEI, m	vg	g	x	g

Table 4.16: Comparison of the algorithms regarding the evaluation criteria 1. number of required function evaluations, 2. accuracy of the located optimum, 3. overcoming local optima and 4. dealing with unsuitable or "bad" simulation data, where "r" denotes reasonable performance, "g" denotes good performance, "vg" denotes very good performance and "x" denotes bad performance

4.5.7 Conclusion

As a conclusion it can be stated, that some prior-informed algorithms are able to fulfill all evaluation criteria with a good to very good performance, compared to the baseline. Namely, the approaches using a PIK with mismatch correction and WMEI or MLEI acquisition function can be recommended for optimization problems with multiple sets of simulation data, which are possibly unsuitable for the objective function to be optimized. Even for unfavorable prior information, these algorithms yield similar performance as standard BO, while clearly outperforming standard BO with helpful prior information. Furthermore, for the case that there is knowledge about having an objective function without local optima or having simulation data which most likely fits the objective function well, also the PGEI with SEK can be a very efficient choice, by yielding nearly optimal parameters within a minimum number of required function evaluations.

Chapter 5

Application: Vant-Hull-Controller

In this chapter, the best performing algorithms from previous evaluations will optimize a Vant-Hull controller with two controller parameters, which is used as an aim point controller for a solar power tower plant. The goal is to find a parameter configuration, which yields the maximum of the objective function, indicating the performance of the controller. Again, a comparison to standard BO will be given, in order to precisely assess the improvements regarding number of required function evaluations to find well performing controller parameters and achieved accuracy of the located parameter configuration.

5.1 Vant-Hull Control Strategy with Two Controller Parameters

As already mentioned in section 2.1.2, the Vant-Hull controller is a simple control method to prevent exceeding the overflux conditions of the receiver. When overflux is measured, the respective heliostat aim points are moved away from the receiver's equator in a vertical direction and are located in alternate rows at the upper and lower receiver edges. The distance of the new aim point to the edge of the receiver is determined by taking the product of the heliostat beam radius and a parameter k , which controls the aiming process for the entire heliostat field.

This single-parameter Vant-Hull controller is able to decrease an exceeding flux density to allowable flux values. However, it can result in two flux peaks at the receiver edges with a gap in the middle, which leads to high spillage and thus decreases the efficiency. To yield a more efficient aim point control strategy, it is desirable to fill the gap between those flux peaks and make the flux profile more homogenous. Therefore, the Vant-Hull controller was extended to a two-parameter control strategy by Collado et al. [17]. Here, the heliostat field is divided into three zones, where zone one contains heliostats which are close to the tower, zone two is further away and zone three denotes the furthest zone from the tower. Consequently, heliostats in zone three produce the largest heliostat beam radius on the receiver. Every zone is allocated with a parameter k_i , with i denoting the respective zone $i = 1, \dots, 3$. However, it is assumed, that $k_1 = k_2$, which results in two selectable controller hyperparameters k_1 and k_3 , which are aimed to be optimized in this test.

Compared to the single-parameter approach, the additional parameter results in a more homogeneous flux profile on the receiver, by redirecting aim points from zone three closer to the receiver’s equator. Thus, the spillage is reduced, which increases the efficiency, while not exceeding allowable flux values.

5.2 Simulation Data

The simulation data was generated by simulating the solar power tower plant in Jülich. Here, the receiver is a rectangular receiver, which consists of ceramic absorber structures and works with air as heat transfer medium. The 2153 heliostats are arranged in a so-called northern field [9] with respect to the tower, which is depicted in fig. 5.1.

In total 14 sets of simulation data were generated for different values of certain simulation hyperparameters. Namely, the maximum allowable flux density of the receiver, as well as the mirror error and the mirror reflectivity, where the latter are usually difficult to estimate. The mirror reflectivity is directly proportional to the direct normal irradiation (DNI), which denotes the amount of solar radiation, received per unit area by the heliostat surface, that is always held perpendicular to the rays, which are coming in a straight line from the direction of the sun at its current position in the sky [16]. Therefore, changing the simulated DNI, has the effect of varying the simulated mirror reflectivity factor. Here, a simulated DNI of 1000 was assumed as a mirror reflectivity factor of 0.9. The exact values of the simulation hyperparameters for all simulation data sets are listed in tab. 5.1.

Subsequently, the generated sets of simulation data were again interpolated by a regression NN. Further details on the NN architecture and the training process can be found in appendix A.1.2.

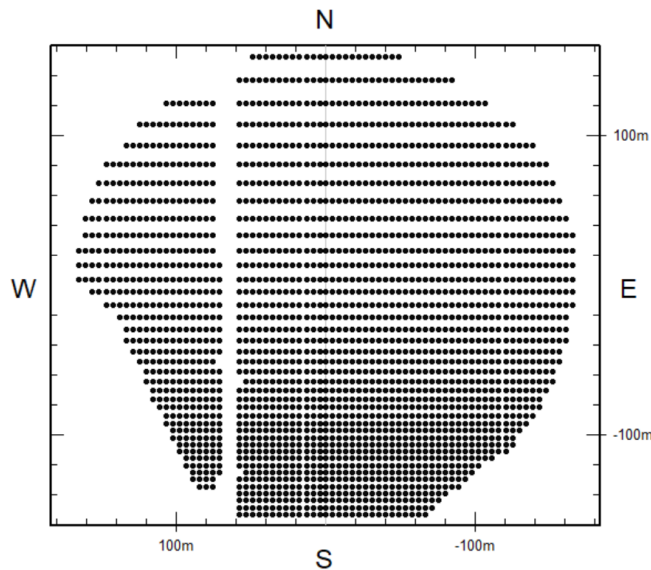


Figure 5.1: Heliostat field layout: northern field of the Jülich plant

Overview of Simulation Data Sets for the Vant-Hull Test

data set	max. allowable flux [W m^{-2}]	mirror error [mrad]	mirror reflectivity
1	750000	0.002	0.72
2	750000	0.002	0.81
3	750000	0.002	0.9
4	750000	0.0015	0.72
5	750000	0.0015	0.81
6	750000	0.0015	0.9
7	775000	0.002	0.81
8	800000	0.002	0.72
9	800000	0.002	0.765
10	800000	0.002	0.81
11	800000	0.002	0.9
12	800000	0.0015	0.72
13	800000	0.0015	0.81
14	800000	0.0015	0.9

Table 5.1: Generating 14 sets of simulation data of the solar power tower plant in Jülich, by varying the following simulation hyperparameters: maximum allowable flux density [W m^{-2}], mirror error [mrad] and mirror reflectivity factor

5.3 Test Case

To avoid running time-consuming tests on the real plant, one set of interpolated simulation data was chosen to imitate the true objective function of the Vant-Hull controller, which indicates the controller’s performance, depending on the controller parameters k_1 and k_3 . Namely, the simulation data set 7 in tab. 5.1 was chosen for this purpose. To evaluate the accuracy of parameters, selected by the optimization, the global optimum of the objective function, and the respective optimal controller parameters, have to be known. Therefore, the optimum of the NN function, taken as the Vant-Hull controller’s objective function, and respective parameters were determined by a grid search procedure.

Consequently, the tests were performed on all remaining sets of simulation data. By plotting the interpolated values of the remaining simulation data sets against the respective Vant-Hull objective function values, it was assessed that none of the simulation data sets exactly resembles the objective function. To give an example for the fit of the simulation data, in fig. 5.2 the plots are shown for two different sets of simulation data, namely data set 3 and data set 4 from the simulation data sets, listed in tab. 5.1. In the plots, the true optimum of the objective function is highlighted by a red marker.

From the evaluations in chapter 4, the best performing algorithms were selected to optimize the Vant-Hull objective function, which are namely the algorithms with prior-informed kernel function (PIK) and Most-Likely Expected Improvement (MLEI), as well as Weighted Mixture Expected Improvement (WMEI) acquisition function. Additionally, the approach with a standard squared exponential kernel (SEK) and Prior-Guided Expected Improvement (PGEI) was tested. As a refer-

ence, again the performance of the standard BO is stated, regarding number of function evaluations and accuracy.

Like in the previous chapter, again 30 optimization runs were performed and averaged, to obtain meaningful results. In every optimization run, 50 optimization iterations are executed. In the same way as for the Hartmann6 test case, the number of optimization iterations was set to the number of iterations, after which the standard BO stopped showing any noteworthy improvement in objective function observations. In contrast to the tests on Hartmann6, only two initial parameters are randomly chosen and evaluated, which in total yields a budget of $N = 52$ maximum function evaluations. Choosing less initial parameter configurations, than for the six-dimensional Hartmann function is reasonable, since the objective function of the Vant-Hull controller is only two-dimensional.

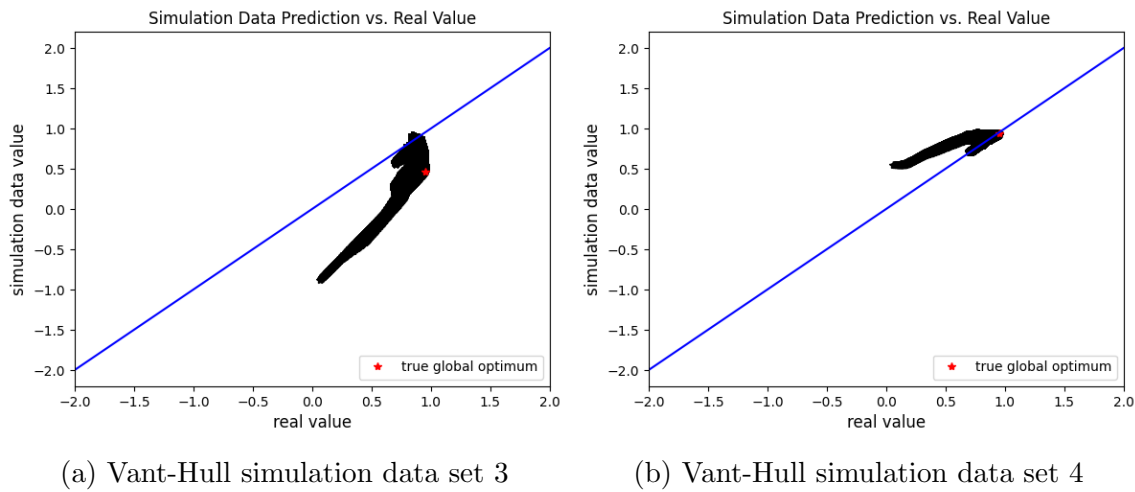


Figure 5.2: Vant-Hull objective function values vs. simulation data predictions for two sets of simulation data

5.4 Results

To assess the performance of the algorithms, again the average number of function evaluations was determined, which is required to reach accuracies of 50 %, 80 %, 90 % and 95 % of the global optimum in objective function space, where the accuracy is determined by eq. (4.1). Moreover, the averaged accuracies, reached after 25 %, 50 %, 75 % and 100% of the maximum number of function evaluations $N = 52$, were calculated to gain a deeper insight into the performance of the algorithms throughout the optimization procedure. Analogously to the evaluations in chapter 4, the accuracies are again stated in objective function output space and in parameter space, where the accuracy in parameter space is calculated from eq. (4.2).

All resulting averaged function evaluations and accuracies, stated in the following tables, are rounded to integers.

Number of Function Evaluations From the number of required function evaluations, stated in tab. 5.2, it is clearly visible, that standard BO itself yields high

accuracies within a very little number of function evaluations.

For the prior-informed algorithms, the number of required evaluations to reach 50 % and 80 % accuracy, resemble the baseline. For 90 % accuracy, the PIK with WMEI requires one evaluation less than the baseline, which is an improvement of 25 % compared to the baseline’s function evaluations. The same observation holds for the SEK with PGEI and mismatch correction. The other prior-informed algorithms either yield the same result as the reference or need one function evaluation more. However, to reach an accuracy of 95 %, almost all prior-informed algorithms require between 15 % and 23 % less function evaluations than standard BO. Only the approach with SEK and PGEI (without mismatch correction) yields the same results as the baseline.

Number of Function Evaluations				
algorithm	50 % acc.	80 % acc.	90 % acc.	95 % acc.
BO (baseline)	1 eval.	2 eval.	4 eval.	13 eval.
PIK, WMEI	1 (+0 %)	2 (+0 %)	3 (-25 %)	10 (-23 %)
PIK, WMEI, m	1 (+0 %)	2 (+0 %)	5 (+25 %)	10 (-23 %)
PIK, MLEI	1 (+0 %)	2 (+0 %)	4 (+0 %)	10 (-23 %)
PIK, MLEI, m	1 (+0 %)	2 (+0 %)	5 (+25 %)	12 (-8 %)
SEK, PGEI	1 (+0 %)	2 (+0 %)	4 (+0 %)	13 (+0 %)
SEK, PGEI, m	1 (+0 %)	2 (+0 %)	3 (-25 %)	11 (-15 %)

Table 5.2: Vant-Hull tests with all simulation data sets: number of required function evaluations (eval.) to reach 50 %, 80 %, 90 % and 95 % accuracy (acc.) of the optimum in comparison to the baseline, averaged over 30 optimization runs

Accuracy in Function Output Space The accuracies in objective function output space are calculated after 25 %, 50 %, 75% and 100 % of the number of maximum function evaluations $N = 52$, which corresponds to 14, 27, 39 and 52 objective function evaluations, respectively.

From the results, stated in tab. 5.3, it can be seen that the accuracies mostly resemble the baseline’s performance. After 14 function evaluations, all algorithms already reach at least 96 % accuracy. The maximum accuracy of 99 % is reached by all algorithms after at most 39 function evaluations. Because there was no further change in accuracies, the column for $N = 52$ function evaluations was omitted in tab. 5.3.

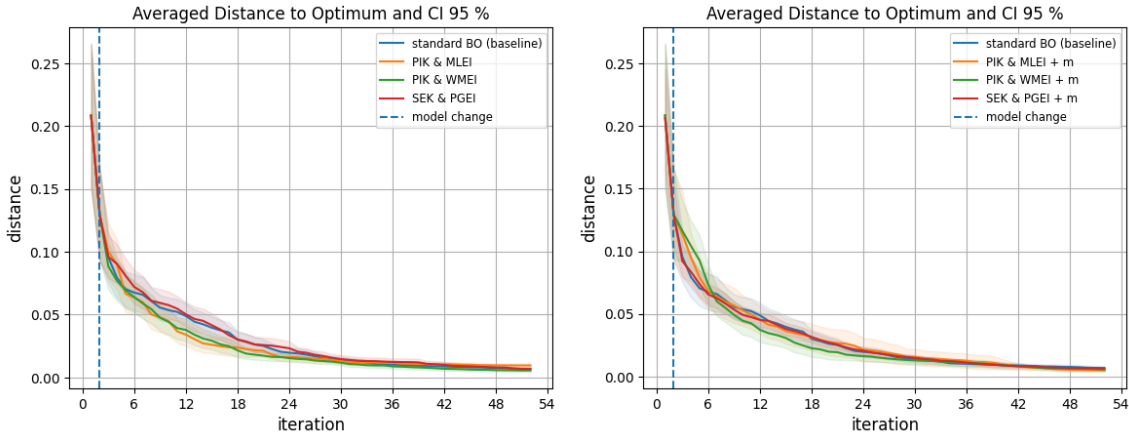
For a visual assessment of the results, the plots in fig. 5.3 show the mean distances of the best found values so far to the objective function’s optimum and the confidence interval of 95 %. To keep the plots clear and easy to distinguish, separate plots were generated for algorithms without and with mismatch correction in comparison to the baseline. From fig. 5.3a, it can be seen that the PIK with WMEI and the PIK with MLEI without mismatch correction, on average mostly outperform the baseline throughout the optimization. However, compared to the PIK with MLEI, the baseline seems to reach slightly better performing parameters during the last quarter of the optimization procedure. The differences are yet small enough to not be unnoticeable from the rounded numbers in tab. 5.3.

In fig. 5.4, the algorithms' counterparts with mismatch correction are plotted. While the PIK with MLEI and the SEK with PGEI show performances similar to the baseline, the PIK with WMEI slightly outperforms the baseline for several optimization steps.

Accuracy in Function Output Space

algorithm	14 eval.	27 eval.	39 eval.
BO (baseline)	96 % acc.	98 % acc.	99 % acc.
PIK, WMEI	97 % (+1 %)	99 % (+1 %)	99 % (+0 %)
PIK, WMEI, m	97 % (+1 %)	98 % (+0 %)	99 % (+0 %)
PIK, MLEI	97 % (+1 %)	98 % (+0 %)	99 % (+0 %)
PIK, MLEI, m	96 % (+0 %)	98 % (+0 %)	99 % (+0 %)
SEK, PGEI	95 % (-1 %)	98 % (+0 %)	99 % (+0 %)
SEK, PGEI, m	96 % (+0 %)	98 % (+0 %)	99 % (+0 %)

Table 5.3: Vant-Hull tests on all simulation data sets: accuracy (acc.) in function output space after 25 % (= 14 eval.), 50 % (= 27 eval.) and 75 % (= 39 eval.) of maximum function evaluations (eval.) (= 52 eval.), averaged over 30 optimization runs



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

Figure 5.3: Vant-Hull tests on all simulation data sets: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

Accuracy in Parameter Space In tab. 5.4, the accuracies are stated in parameter space. Some prior-informed approaches show a slight improvement or decrease throughout the optimization process, however in total the accuracies of all prior-informed algorithms mostly resemble the baseline. The maximum accuracy, reached by most algorithms in parameter space, comprises 95 %.

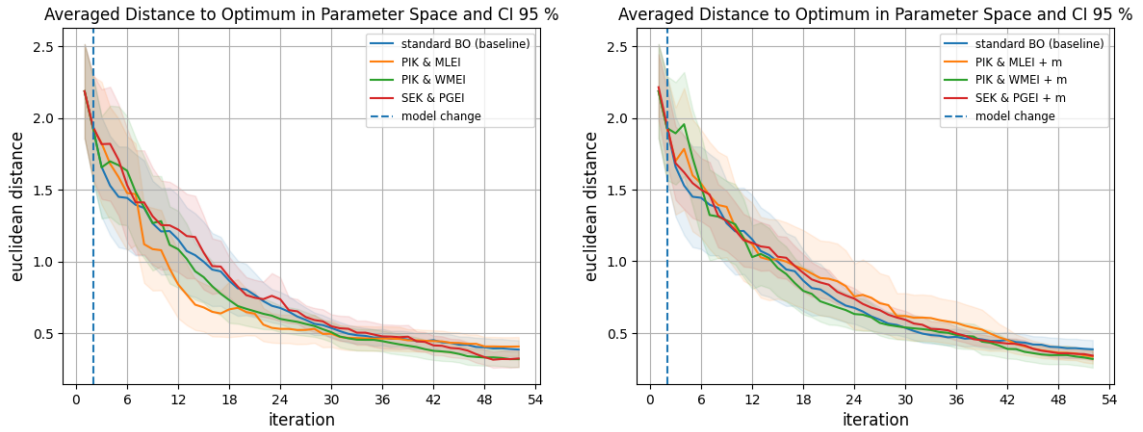
The results are again complemented with plots in fig. 5.4a and fig. 5.4b, which

Accuracy in Parameter Space

algorithm	14 eval.	27 eval.	39 eval.	52 eval.
BO (baseline)	85 % acc. p.	92 % acc. p.	94 % acc. p.	95 % acc. p.
PIK, WMEI	87 % (+2 %)	92 % (+0 %)	94 % (+0 %)	95 % (+0 %)
PIK, WMEI, m	86 % (+1 %)	92 % (+0 %)	94 % (+0 %)	95 % (+0 %)
PIK, MLEI	90 % (+6 %)	93 % (+1 %)	94 % (+0 %)	94 % (-1 %)
PIK, MLEI, m	86 % (+1 %)	90 % (-2 %)	93 % (-1 %)	95 % (+0 %)
SEK, PGEI	83 % (-2 %)	91 % (-1 %)	93 % (-1 %)	95 % (+0 %)
SEK, PGEI, m	85 % (+0 %)	91 % (-1 %)	94 % (+0 %)	95 % (+0 %)

Table 5.4: Vant-Hull tests on all simulation data sets: accuracy in parameter space (acc. p.) after 25 % (= 14 eval.), 50 % (= 27 eval.), 75 % (= 39 eval.) and 100 % of maximum function evaluations (eval.) (= 52 eval.), averaged over 30 optimization runs

show the mean distances to the optimum in parameter space and the confidence intervals of 95 % for algorithms without and with mismatch correction, respectively. After the maximum number of $N = 52$ function evaluations, all prior-informed algorithms, except approach with PIK and MLEI (without mismatch correction), seem to outperform the baseline from the reached accuracy in parameter space. However, the improvements are again small enough to not be noticeable from the rounded numbers in tab. 5.4.



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

Figure 5.4: Vant-Hull tests on all simulation data sets: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

5.5 Conclusion

From previously stated results, it can be seen, that most of the considered prior-informed algorithms slightly outperform the standard BO approach regarding the number of required function evaluations. However, the improvements are much smaller compared to improvements, observed for the Hartmann6 test function.

From the reached accuracies it can be seen, that for most parts of the optimization process, the algorithm with PIK and WMEI slightly outperforms the baseline, while the PIK with MLEI is slightly outperformed by the standard BO. Additionally, the approach with PGEI and mismatch correction could yield slight improvements regarding the number of required function evaluations, compared to the baseline. In total, however, the results of all prior-informed algorithms are quite similar to the baseline, which already performs well without leveraging simulation data.

The reason for the hardly recognizable differences in observed performances is, that the Vant-Hull controller possesses only two parameters, which yields a two-dimensional objective function. In comparison to the six-dimensional Hartmann function, a two-dimensional function is less complex to optimize. Therefore, the standard BO algorithm without simulation data already yields well performance. For simulation data sets, which do not exactly represent the true objective function, it is hardly possible for the prior-informed algorithms to outperform the reference performance.

To support this explanation, in fig. 5.5, the Vant-Hull objective function observations, gathered throughout one BO optimization run, are plotted in parameter space and constitute a prediction of the underlying objective function, given as a mean prediction and the standard error. Here, it is clearly visible that the area of bad performing parameter values is rather small, compared to a rather large the area of well performing parameter values, which is easy to find within a few function evaluations.

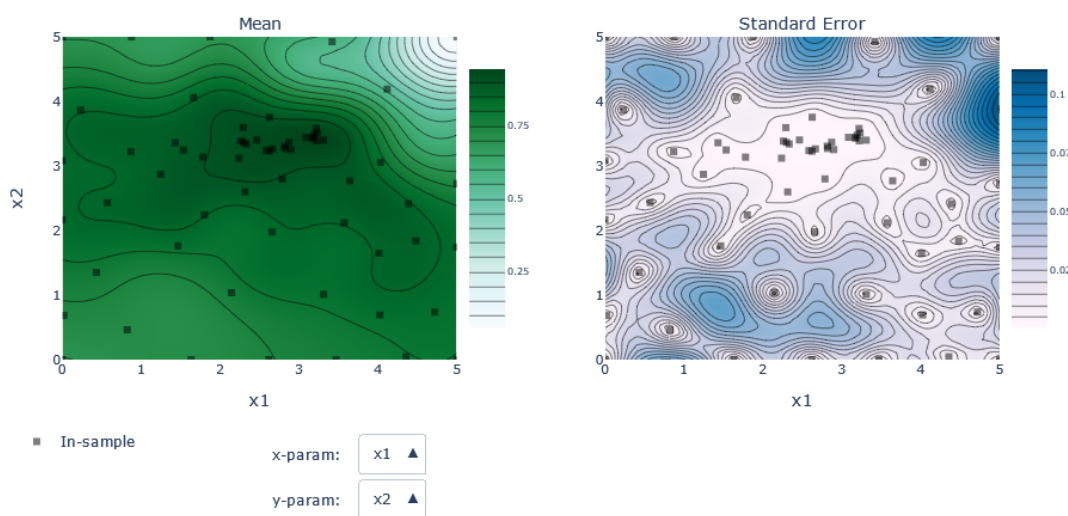


Figure 5.5: Mean prediction and standard error of the Vant-Hull objective function, based on objective function observations

To conclude the test of optimizing a Vant-Hull aim point controller, it can be stated, that using the prior-informed approach with PIK and WMEI can decrease the number of required objective function evaluations by a couple. When testing a parameter configuration on the real system, this is still more time efficient than using the standard BO approach and therefore beneficial. However, bigger improvements are only noticeable when having a controller with more than two hyperparameters, which would increase the complexity of the objective function and make it more challenging to be optimized.

Chapter 6

Summary and Outlook

In this work, novel approaches were proposed for sample-efficient hyperparameter optimization of an arbitrary aim point controller for solar power tower plants, by leveraging multiple sets of simulation data as prior information in Bayesian Optimization. In total, three classes of prior-informed algorithms were developed. Namely, approaches with prior-informed kernels, approaches with a novel acquisition function, called Prior-Guided Expected Improvement, and hybrid algorithms, which utilize both mentioned concepts. In order to utilize information from multiple sets of simulation data, the algorithms deploy a model selection criterion, which determines the prior information to be used in each optimization iteration. Moreover, methods were developed to make use of more than one source of prior information simultaneously within one optimization iteration, by combining multiple prior-informed models. These are, namely, the Most Likely Expected Improvement and the Weighted Mixture Expected Improvement acquisition function, as well as the Weighted Mixture Kernel. To further increase the performance with possibly unsuitable simulation data, the algorithms were extended with a strategy for correcting mismatches between simulation data and objective function observations. The approaches were tested and evaluated for all possible combinations of proposed kernel types, acquisition functions and mismatch correction options, where the six-dimensional Hartmann test function was used as an objective function. The prior-informed algorithms utilized multiple sets of simulation data, that do not contain the true optimum of the objective function. For comparison, a standard Bayesian Optimization approach was used as a baseline. In total, four test cases were analyzed. Firstly, a general test was performed on all tests of available simulation data with randomly selected initial parameter configurations. Secondly, the test was repeated for a set of fixed initial parameters, which lead the optimization into the direction of a local optimum. Thirdly, the proposed approaches were tested with only one set of simulation data, which underestimated the global optimum of the objective function, while overestimating the local extreme values of the function. Lastly, it was assessed how the prior-informed algorithms deal with only one set of simulation data, which does not resemble the objective function. From the test results, the drawbacks of some developed algorithms became apparent. However, especially the approaches with prior-informed kernel and either Weighted Mixture Expected Improvement acquisition function or Most Likely Expected Improvement acquisition

tion function could convince in all test cases and mostly outperform the baseline for considered evaluation criteria. For instance, the algorithm with prior-informed kernel and Weighted Mixture Expected Improvement, tested with all sets of simulation data, managed to reach parameters, which yielded an accuracy of 95 % of the objective function's global optimum, within 33 % less function evaluations, than the baseline. This was an absolute difference of 14 function evaluations. Moreover, for the remaining test cases, this approach was able to quickly overcome local optima and was not negatively affected by unsuitable simulation data.

To give a second test scenario, the best performing algorithms were selected and used to optimize a Vant-Hull aim point controller with two controller hyperparameters. Here, it was noticed, that for a two-dimensional objective function, the improvements from using a prior-informed algorithm compared to a standard Bayesian Optimization approach, become less apparent, than for Hartmann6. The algorithm with prior-informed kernel and Weighted Mixture Expected Improvement still outperformed the baseline. For finding controller parameters, which yield an accuracy of 95 % of the objective function's global optimum, the prior-informed algorithm needed 23 % less function evaluations compared to the baseline. However, this resulted in the absolute difference of only three function evaluations. Therefore, it was concluded, that using a prior-informed algorithm for controller hyperparameter optimization is especially profitable when having more complex objective functions, than for the Vant-Hull controller with two controller hyperparameters.

Overall, at least two of the prior-informed algorithms, could fulfill all objectives of this thesis. These were, firstly, to increase the sample-efficiency and thus reduce the number of required objective function evaluations to find well performing controller parameters, compared to a standard Bayesian Optimization algorithm, by leveraging multiple sets of simulation data. Secondly, the proposed approaches should either take advantage of, or discard simulation data, that is (partly) unsuitable for the objective function or shifted on the x-axis, which imitates the realistic case of having inaccurately estimated simulation hyperparameters. In fact, the best-performing approaches could meet these requirements by outperforming the baseline regarding required number of function evaluations and accuracy of the located parameters, in most test cases. Simultaneously, they yielded performances, similar to the baseline for the tested worst case scenarios, like having unsuitable simulation data.

In future work, the algorithms could be evaluated on other aim point control strategies, to validate their performance for arbitrary objective functions. Furthermore, as explained in section 4.5.5, the strategy used for calculating the model selection score, tends to overestimate the predictive performance of a prior-informed model with unsuitable simulation data and mismatch correction between simulation data and objective function observations. The reason was, that the mismatch correction is calculated only one time for all objective function observations. The model selection criterion then determines a predictive score of the used model on withheld data points after being fitted to training data points. Consequently, the predictive score uses information about the mismatch of withheld data points, which leads to overly optimistic results. Respective algorithms could be further modified and improved by adjusting the way of calculating the model selection criterion. A new mismatch correction could be determined on every share of observation data, which is used

as training data within the calculation of the model selection criterion. However, it would be necessary to validate, if possible improvements in performance would be big enough to justify the additional computational burden, caused by this modification.

Lastly, an additional modification or extension of the prior-informed algorithms would be necessary to test them on a real plant. As mentioned in section 2.1.2, the objective of the aim point controller is to maximize the power on the receiver. However, at the same time the allowed flux density must not be exceeded to prevent damage to the receiver, caused by overheating. Running the plant with an arbitrary configuration of controller hyperparameters, carries the risk, that the controller is not able to meet these objectives due to bad performance. In the worst case, damage to the receiver could be caused. To prevent this, the prior-informed algorithms could be further modified by introducing safety constraints, which limit BO to areas of the parameter space, which were estimated as "safe".

Appendix A

Supplementary Material

A.1 Neural Networks for Interpolating Simulation Data

The regression NNs for interpolating the sets of simulation data points, were implemented and trained in Python with the open-source machine learning framework PyTorch [44].

A.1.1 Regressing Hartmann6 Simulation Data

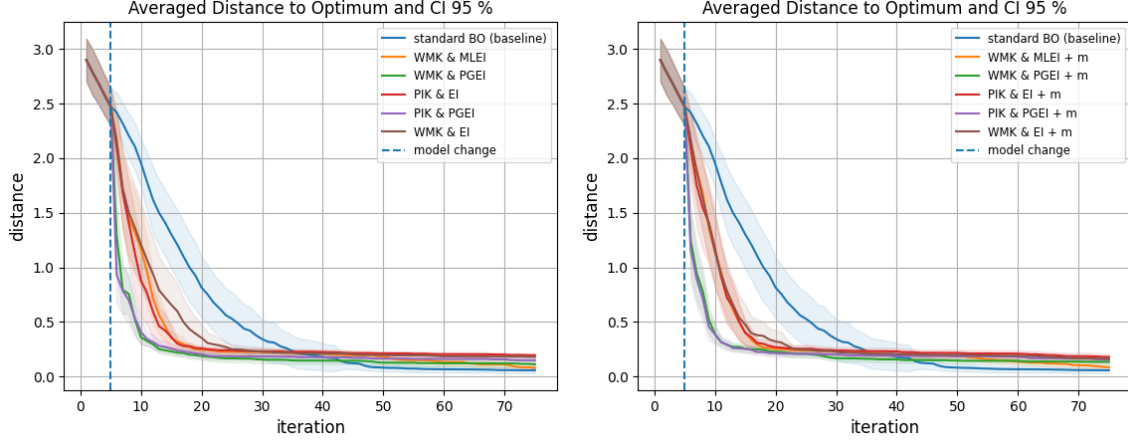
To interpolate the simulation data sets, generated for the six-dimensional Hartmann test function, a regression NN with five neuron layers was used, based on the proposed NN in [39]. The training was performed in 250 epochs with a learning rate of 0.005 and a batch size of 32.

A.1.2 Regressing Vant-Hull Simulation Data

To interpolate the simulation data sets, generated for the two-dimensional Vant-Hull controller objective function, a regression NN was used, again based on the proposed NN in [39], however, with only three neuron layers. This choice was made, since a two-dimensional function requires less complexity than the six-dimensional Hartmann function. Since the amount of available simulation data points was much more limited, than for the Hartmann6 function, the number of training epochs and the learning rate were reduced to 50 and 0.001, respectively. This prevents the undesired behavior of overfitting the function, which usually leads to poor generalization to unseen data points.

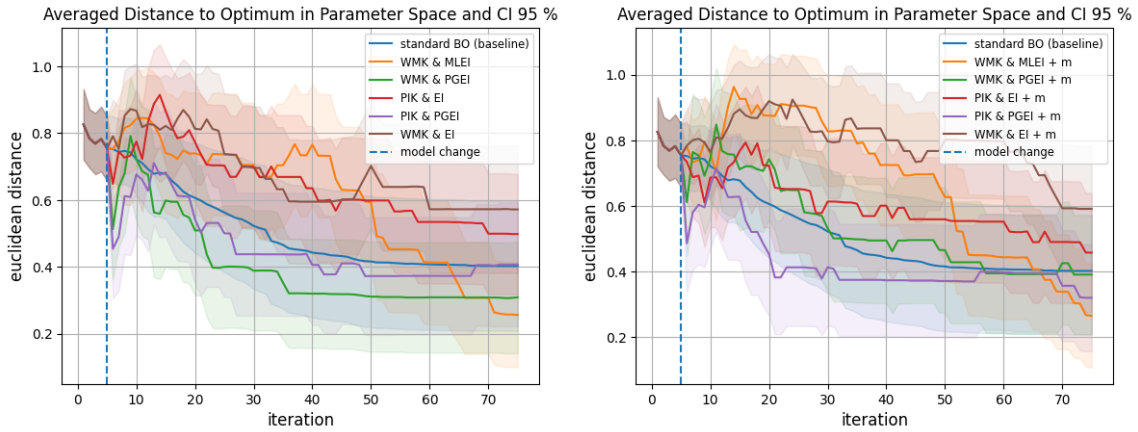
A.2 Supplementary Test Results for Hartmann6

A.2.1 All Simulation Data Sets



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

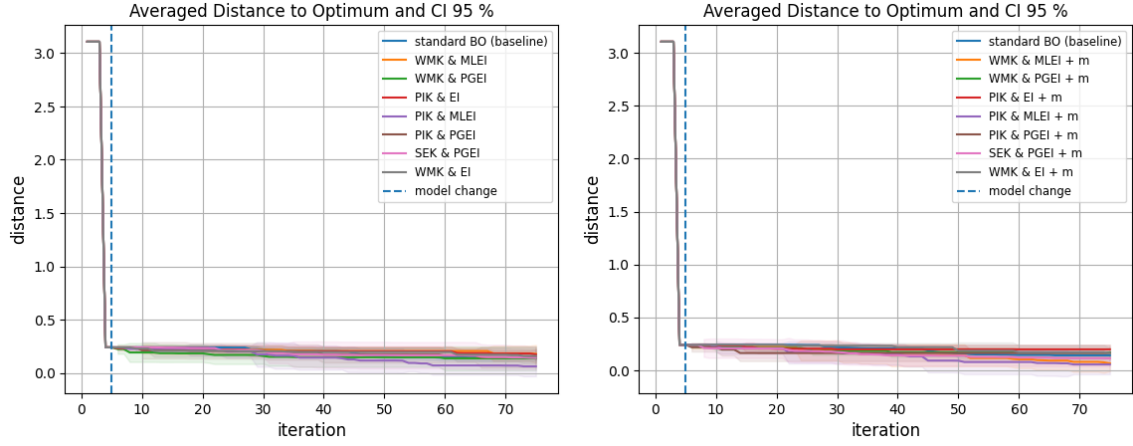
Figure A.1: Hartmann6 tests with all simulation data sets for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

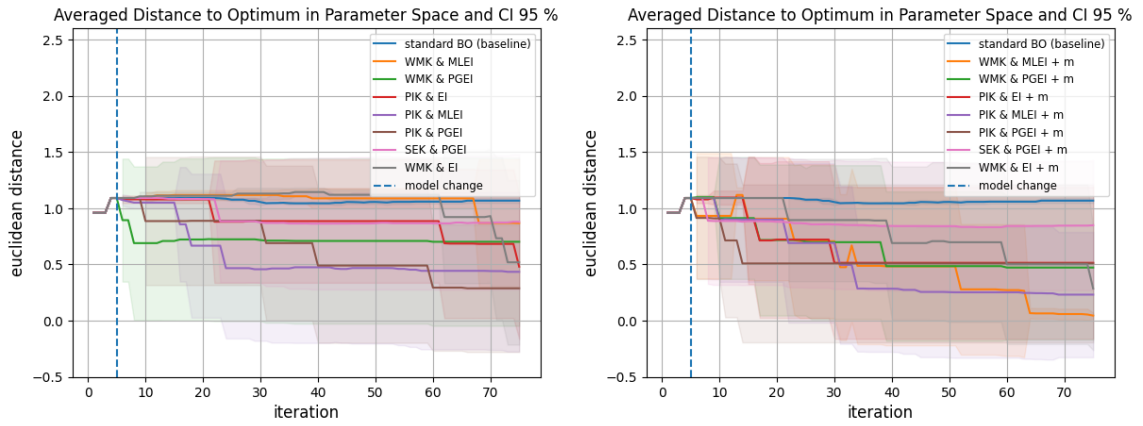
Figure A.2: Hartmann6 tests with all simulation data sets for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

A.2.2 All Simulation Data Sets with Fixed Initial Parameters



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

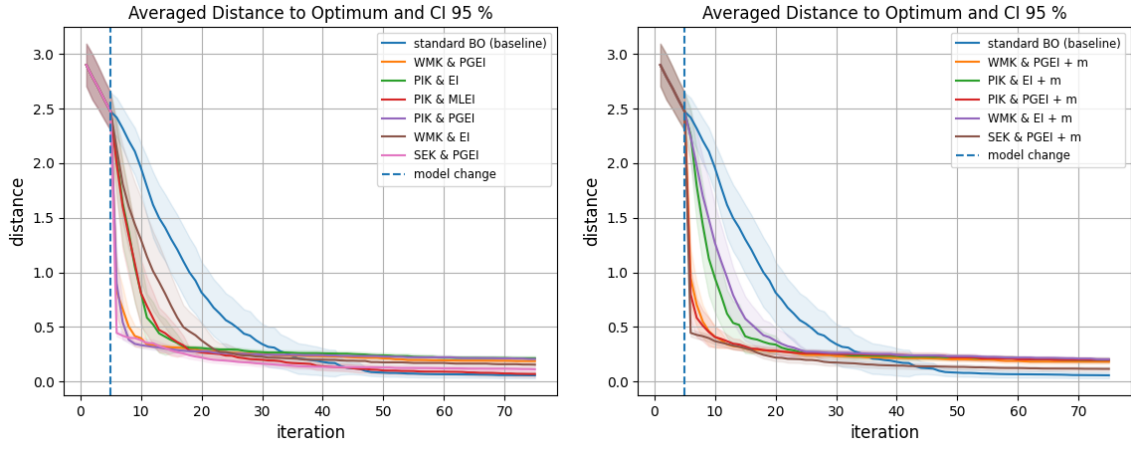
Figure A.3: Hartmann6 tests with all simulation data sets and fixed initial parameters for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

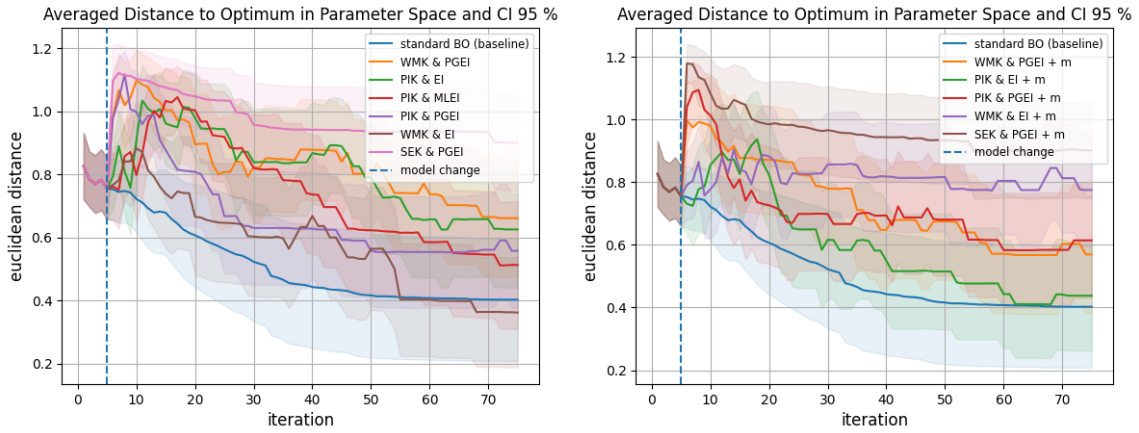
Figure A.4: Hartmann6 tests with all simulation data sets and fixed initial parameters for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 5 optimization runs with fixed initial parameter configurations, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

A.2.3 Simulation Data Set 1



(a) Algorithms without mismatch correction (b) Algorithms without mismatch correction

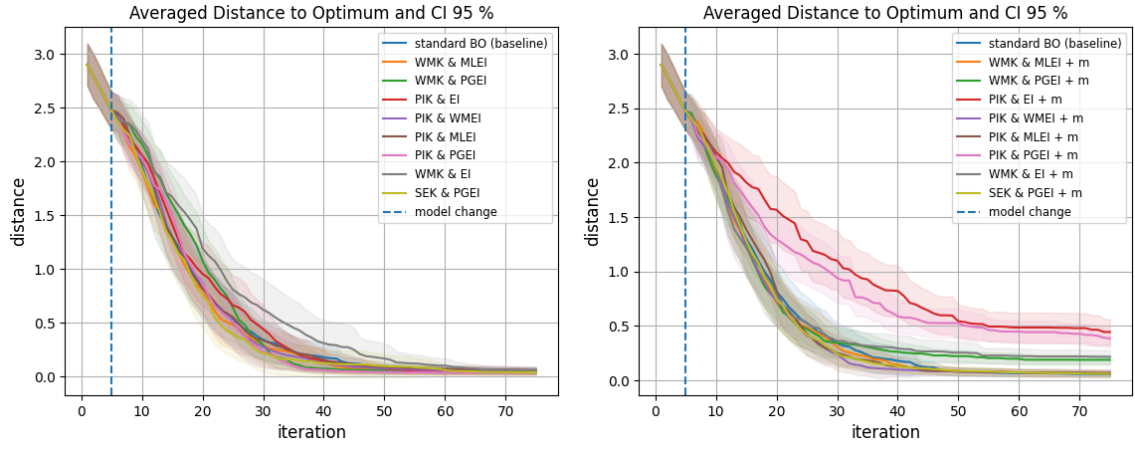
Figure A.5: Hartmann6 tests with simulation data set 1 for remaining algorithms: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)



(a) Algorithms without mismatch correction (b) Algorithms without mismatch correction

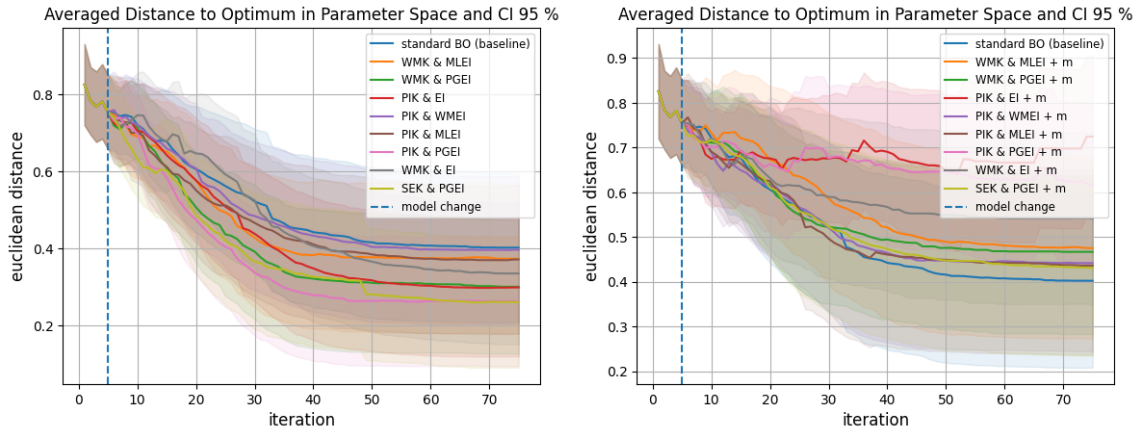
Figure A.6: Hartmann6 tests with simulation data set 1 for remaining algorithms: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)

A.2.4 Simulation Data Set 4



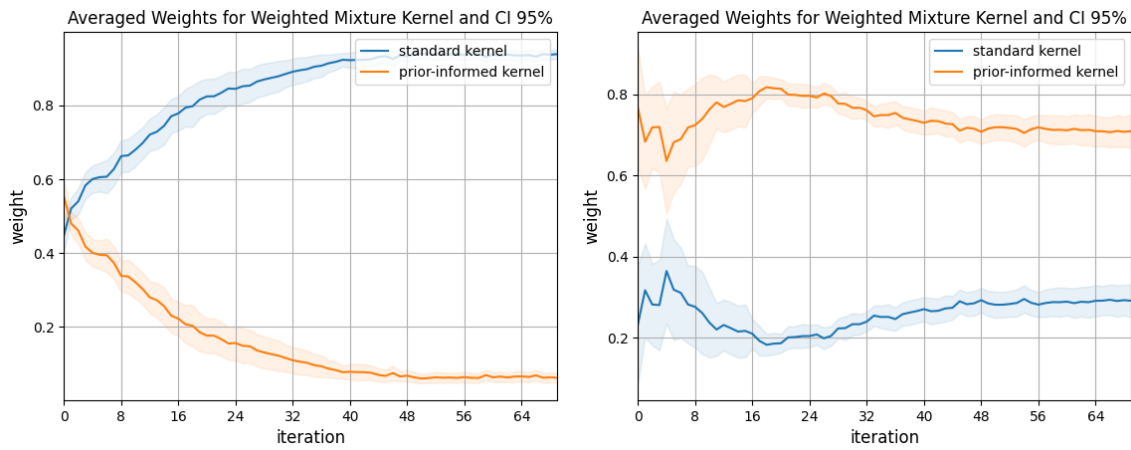
(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

Figure A.7: Hartmann6 tests with simulation data set 4: mean distance to the global optimum in function output space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)



(a) Algorithms without mismatch correction (b) Algorithms with mismatch correction

Figure A.8: Hartmann6 tests with simulation data set 4: mean distance to the global optimum in parameter space and confidence interval 95 %, averaged over 30 optimization runs, for algorithms without mismatch correction (a) and algorithms with mismatch correction (b)



(a) WMK, MLEI, no mismatch correction (b) WMK, MLEI, with mismatch correction

Figure A.9: Hartmann6 tests with simulation data set 4: weights of used sub-kernels in the Weighted Mixture Kernel, averaged over 30 optimization runs, with Most Likely Expected Improvement acquisition function without mismatch correction (a) and with mismatch correction (b)

Bibliography

- [1] *2030 Climate Target Plan*. 2022. URL: https://ec.europa.eu/clima/eu-action/european-green-deal/2030-climate-target-plan_en (visited on 06/29/2022).
- [2] Aaron Wilson, Alan Fern, and Prasad Tadepalli. “Using Trajectory Data to Improve Bayesian Optimization for Reinforcement Learning”. In: *Journal of Machine Learning Research* 15.8 (2014), pp. 253–282. ISSN: 1533-7928.
- [3] International Energy Agency. *Renewable Energy Market Update*. 2022, p. 29. DOI: <https://doi.org/https://doi.org/10.1787/faf30e5a-en>. URL: <https://www.oecd-ilibrary.org/content/publication/faf30e5a-en>.
- [4] Rika Antonova, Akshara Rai, and Christopher G. Atkeson. “Deep Kernels for Optimizing Locomotion Controllers”. In: *PMLR 78:47-56* ((2017). URL: <https://arxiv.org/pdf/1707.09062>.
- [5] Marco Astolfi et al. “Heliostat aiming point optimization for external tower receiver”. In: *Solar Energy* 157 (2017), pp. 1114–1129. ISSN: 0038-092X. DOI: 10.1016/j.solener.2016.03.042.
- [6] George de Ath, Jonathan E. Fieldsend, and Richard M. Everson. *What do you Mean? The Role of the Mean Function in Bayesian Optimisation*. 2020. DOI: 10.48550/arXiv.2004.08349. URL: <https://arxiv.org/pdf/2004.08349>.
- [7] *Ax*. 2022. URL: <https://ax.dev/> (visited on 07/09/2022).
- [8] David Barlev, Ruxandra Vidu, and Pieter Stroeve. “Innovation in concentrated solar power”. In: *Solar Energy Materials and Solar Cells* 95.10 (2011), pp. 2703–2725. ISSN: 09270248. DOI: 10.1016/J.SOLMAT.2011.05.020.
- [9] Boris Belhomme and Robert Pitz-Paal. “Bewertung und Optimierung von Zielpunktstrategien für solare Turmkraftwerke: Fakultät für Maschinenwesen”. PhD thesis. Shaker and Zugl.: Aachen, Techn. Hochsch., Diss., 2010.
- [10] Boris Belhomme, Robert Pitz-Paal, and Peter Schwarzbözl. “Optimization of Heliostat Aim Point Selection for Central Receiver Systems Based on the Ant Colony Optimization Metaheuristic”. In: *Journal of Solar Energy Engineering* 136.1 (2014). ISSN: 0199-6231. DOI: 10.1115/1.4024738.
- [11] James Bergstra et al. “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems* 24 (2011). ISSN: 1049-5258. URL: <https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619b/Paper.pdf>.

- [12] Alois Bissuel. *Hyper-parameter optimization algorithms: a short review*. 2019. URL: <https://medium.com/criteo-engineering/hyper-parameter-optimization-algorithms-2fe447525903> (visited on 05/23/2022).
- [13] *BoTorch*. URL: <https://botorch.org/> (visited on 07/09/2022).
- [14] Konstantinos Chatzilygeroudis and Jean-Baptiste Mouret. *Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics*. 2017. URL: <https://arxiv.org/pdf/1709.06917>.
- [15] Shi Cheng et al. “Survey on data science with population-based algorithms”. In: *Big Data Analytics* 1.1 (2016), pp. 1–20. ISSN: 2058-6345. DOI: 10.1186/s41044-016-0003-3. URL: <https://bdataanalytics.biomedcentral.com/track/pdf/10.1186/s41044-016-0003-3>.
- [16] “Section 10 - Solar”. In: *Handbook of energy, Volume I*. Ed. by Cutler J. Cleveland, Christopher Morris, and Christopher G. Morris. Oxford: Elsevier Science, 2013, pp. 405–450. ISBN: 978-0-08-046405-3. DOI: 10.1016/B978-0-08-046405-3.00010-3. URL: <https://www.sciencedirect.com/science/article/pii/B9780080464053000103>.
- [17] Francisco J. Collado and Jesus Guallar. “A two-parameter aiming strategy to reduce and flatten the flux map in solar power tower plants”. In: *Solar Energy* 188 (2019), pp. 185–189. ISSN: 0038-092X. DOI: 10.1016/j.solener.2019.06.001. URL: <https://www.sciencedirect.com/science/article/pii/S0038092X19305663>.
- [18] *Concentrating Solar Power Projects*. 2022. URL: <https://solarpaces.nrel.gov/by-status/operational> (visited on 06/29/2022).
- [19] *Concentrating Solar Power Projects*. 2022. URL: <https://solarpaces.nrel.gov/by-status/under-construction> (visited on 06/29/2022).
- [20] Antoine Cully et al. *Robots that can adapt like animals*. 2015. DOI: 10.1038/nature14422. URL: <https://arxiv.org/pdf/1407.3501>.
- [21] M. Cutler and J. How. “Efficient reinforcement learning for robots using informative simulated priors”. In: *undefined* (2015). URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7139550>.
- [22] David Duvenaud. “Automatic model construction with Gaussian processes”. In: *undefined* (2014). DOI: 10.17863/CAM.14087.
- [23] David Duvenaud et al. *Structure Discovery in Nonparametric Regression through Compositional Kernel Search*. 2013. URL: <http://arxiv.org/pdf/1302.4922v4>.
- [24] E. Fong and C. C. Holmes. “On the marginal likelihood and cross-validation”. In: *Biometrika* 107.2 (2020), pp. 489–496. ISSN: 0006-3444. DOI: 10.1093/biomet/asz077. URL: <https://academic.oup.com/biomet/article-pdf/107/2/489/33217997/asz077.pdf>.
- [25] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. URL: <https://arxiv.org/pdf/1807.02811>.

- [26] Jesús García et al. “Heat Flux Distribution Over a Solar Central Receiver Using an Aiming Strategy Based on a Conventional Closed Control Loop”. In: American Society of Mechanical Engineers Digital Collection, 2017. DOI: 10.1115/ES2017-3615. URL: <https://asmedigitalcollection.asme.org/ES/proceedings-pdf/ES2017/57595/V001T05A011/2379246/v001t05a011-es2017-3615.pdf>.
- [27] Jesús García et al. “Multivariable Closed Control Loop Methodology for Heliostat Aiming Manipulation in Solar Central Receiver Systems”. In: *Journal of Solar Energy Engineering* 140.3 (2018). ISSN: 0199-6231. DOI: 10.1115/1.4039255. URL: https://asmedigitalcollection.asme.org/solarenergyengineering/article-pdf/140/3/031010/6330634/sol_140_03_031010.pdf.
- [28] Donald R. Jones, Matthias Schonlau, and William J. Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492. ISSN: 1573-2916. DOI: 10.1023/A:1008306431147. URL: <https://link.springer.com/content/pdf/10.1023/A:1008306431147.pdf>.
- [29] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *undefined* (1995). URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=488968>.
- [30] Brent Komer, J. Bergstra, and C. Eliasmith. “Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn”. In: *undefined* (2014). URL: <https://pdfs.semanticscholar.org/9078/7a8ebbebcba951fa9ba6579f2f7e221e4522.pdf>.
- [31] H. J. Kushner. “A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise”. In: *Journal of Basic Engineering* 86.1 (1964), pp. 97–106. ISSN: 0021-9223. DOI: 10.1115/1.3653121. URL: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/86/1/97/5763745/97_1.pdf.
- [32] Vidhi Lalchand and Carl Edward Rasmussen. *Approximate Inference for Fully Bayesian Gaussian Process Regression*. 2019. DOI: 10.48550/arXiv.1912.13440. URL: <https://arxiv.org/pdf/1912.13440>.
- [33] George Lindfield and John Penny. *Introduction to nature-inspired optimization*. London, San Diego, and Cambridge, MA: Academic Press an imprint of Elsevier, 2017. ISBN: 9780128036365.
- [34] Sanae Lotfi et al. *Bayesian Model Selection, the Marginal Likelihood, and Generalization*. 2022. DOI: 10.48550/arXiv.2202.11678. URL: <https://arxiv.org/pdf/2202.11678>.
- [35] Daniel Maldonado, Robert Flesch, and Peter Schwarzbözl. “Hybridization of Aim Point Optimization Methods for Solar Tower Power Plants”. In: *MATHMOD 2018 extended abstract volume*. Ed. by Felix Breitenecker et al. Argesim Report. Vienna: ARGESIM Publisher, 2018, pp. 39–40. ISBN: 9783901608919. DOI: 10.11128/arep.55.a55230.

- [36] Daniel Maldonado et al. “Evaluation of aim point optimization methods”. In: *AIP Conference Proceedings* 2033.1 (2018), p. 040025. ISSN: 0094-243X. DOI: 10.1063/1.5067061. URL: <https://aip.scitation.org/doi/pdf/10.1063/1.5067061>.
- [37] Alonso Marco et al. “Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ: IEEE, 2017, pp. 1557–1563. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989186.
- [38] Melanie Mitchell. “Genetic algorithms: An overview”. In: *Complexity* 1.1 (1995), pp. 31–39. ISSN: 1099-0526. DOI: 10.1002/cplx.6130010108. URL: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cplx.6130010108>.
- [39] *MLP for regression model*. 2022. URL: <https://github.com/wlemusl/MLP-for-regression-model/blob/main/MLP%5C%20for%5C%20regression%5C%20model.ipynb> (visited on 07/12/2022).
- [40] J. Mockus. “On Bayesian Methods for Seeking the Extremum and their Application”. In: *undefined* (1977).
- [41] Jonas Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*. Vol. v.37. Mathematics and Its Applications Ser. Dordrecht: Springer Netherlands, 1989. ISBN: 9789400909090. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6556877>.
- [42] J. Moćkus. *On Bayesian Methods for Seeking the Extremum*. 1975. DOI: 10.1007/978-3-662-38527-2_{\text{underscore}}55. URL: https://link.springer.com/content/pdf/10.1007/978-3-662-38527-2_55.pdf.
- [43] Rémi Pautrat, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. *Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search*. 2017. DOI: 10.48550/arXiv.1709.06919. URL: <https://arxiv.org/pdf/1709.06919>.
- [44] *PyTorch*. 2022. URL: <https://pytorch.org/> (visited on 07/12/2022).
- [45] Akshara Rai et al. *Using Simulation to Improve Sample-Efficiency of Bayesian Optimization for Bipedal Robots*. 2018. DOI: 10.48550/arXiv.1805.02732. URL: <https://arxiv.org/pdf/1805.02732>.
- [46] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge Mass.: MIT Press, 2006. ISBN: 026218253X.
- [47] Pascal Richter et al. “Optimization of robust aiming strategies in solar tower power plants”. In: *SOLARPACES 2018: International Conference on Concentrating Solar Power and Chemical Energy Systems*. AIP Conference Proceedings. AIP Publishing, 2019, p. 030045. DOI: 10.1063/1.5117557.
- [48] Ibai Roman et al. “An Experimental Study in Adaptive Kernel Selection for Bayesian Optimization”. In: *IEEE Access* 7 (2019), pp. 184294–184302. DOI: 10.1109/ACCESS.2019.2960498.

- [49] Alberto Sánchez González, Rodríguez Sánchez, María de los Reyes, and Domingo José Santana Santana. “Allowable solar flux densities for molten-salt receivers: Input to the aiming strategy”. In: *2590-1230* (2020). ISSN: 2590-1230. URL: https://e-archivo.uc3m.es/bitstream/10016/32589/1/Solar-flux_RE_2020.pdf.
- [50] Matteo Saveriano et al. “Data-efficient control policy search using residual dynamics learning”. In: *undefined* (2017). URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8206343>.
- [51] Bobak Shahriari et al. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2494218.
- [52] Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems* (2012). ISSN: 1049-5258. URL: <https://dash.harvard.edu/bitstream/1/11708816/1/snoek-bayesopt-nips-2012.pdf>.
- [53] *Statistical factor analysis and related methods: Theory and applications*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Hoboken, N.J.: Wiley InterScience, 2008, pp. 80–82. ISBN: 9780470316894. DOI: 10.1002/9780470316894. URL: <https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=294328>.
- [54] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. 1st ed. Springer Series in Statistics Ser. New York, NY: Springer New York, 1999. ISBN: 9781461214946. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=3074879>.
- [55] David Stenger, Muzaffer Ay, and Dirk Abel. “Robust Parametrization of a Model Predictive Controller for a CNC Machining Center Using Bayesian Optimization”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 10388–10394. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2020.12.2778.
- [56] Carolin Strobl et al. “Conditional variable importance for random forests”. In: *BMC Bioinformatics* 9.1 (2008), p. 307. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-307. URL: <https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/1471-2105-9-307>.
- [57] Lorin L. Vant-Hull. “The Role of “Allowable Flux Density” in the Design and Operation of Molten-Salt Solar Central Receivers”. In: *Journal of Solar Energy Engineering* 124.2 (2002), pp. 165–169. ISSN: 0199-6231. DOI: 10.1115/1.1464124. URL: https://asmedigitalcollection.asme.org/solarenergyengineering/article-pdf/124/2/165/5700741/165_1.pdf.
- [58] Jia Wu et al. “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization”. In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40. ISSN: 1674-862X. DOI: 10.11989/JEST.1674-862X.80904120. URL: <https://www.sciencedirect.com/science/article/pii/S1674862X19300047>.

-
- [59] Tong Yu and Hong Zhu. *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. 2020. DOI: 10.48550/arXiv.2003.05689. URL: <https://arxiv.org/pdf/2003.05689>.
 - [60] Yongli Zhang and Yuhong Yang. “Cross-validation for selecting a model selection procedure”. In: *Journal of Econometrics* 187.1 (2015), pp. 95–112. ISSN: 0304-4076. DOI: 10.1016/j.jeconom.2015.02.006.
 - [61] J. G. Ziegler and N. B. Nichols. “Optimum Settings for Automatic Controllers”. In: *Journal of Dynamic Systems, Measurement, and Control* 115.2B (1993), pp. 220–222. ISSN: 0022-0434. DOI: 10.1115/1.2899060.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

30.07.2022

München, date



Barbara Marie Anna Lenz

Name