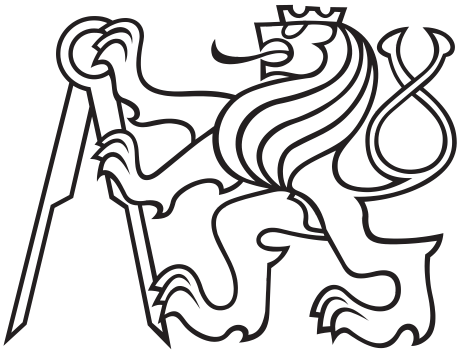
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Experiment Design Methods for Development of Simplified Plasma Boundary Model

Bachelor's Thesis

Šimon Soldát

Study program:	Open Informatics
Specialization:	Artificial Intelligence and Computer Science
Supervisor:	Doc. Ing. Václav Šmídl, Ph.D.

Prague, May 2022

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Soldát Šimon** Personal ID number: **492290**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Experiment Design Methods for Development of Simplified Plasma Boundary Model

Bachelor's thesis title in Czech:

Metody návrhu experimentů pro tvorbu zjednodušeného modelu okraje plasmatu

Guidelines:

1. Create a survey of methods that are used for development of surrogate (simplified) models approximating functions that can be evaluated only point-vise using computationally expensive simulations. Pay special attention to Bayesian methods that minimize the number of runs of the simulation and discuss properties of various building blocks within the methodology (e.g. class of function approximators).
2. Prepare a short description of challenging plasma edge modeling problems for future thermonuclear reactors. Pay attention especially to the power exhaust problem and the plasma detachment problem. Define its specifics and challenges for surrogate model development.
3. Based on theoretical analysis, choose at least two distinct methods of surrogate modeling that would be appropriate for the selected plasma modeling problem. Prepare a flexible software implementation of these methods that allows unified evaluation of their performance and sensitivity studies.
4. Apply the selected methodology to data provided by the institute of plasma physics (in the form of output of the SOLPS-ITER software). Perform a comparative analysis of the selected approaches. Pay special attention to: i) computational complexity of the function approximator, ii) the number of runs of the simulator needed to obtain a predefined precision, and iii) reliability of the result with respect to different choices on the initial subset of the data.

Bibliography / sources:

[1] Shahriari, B., Swersky, K., Wang, Z., Adams, R.P. and De Freitas, N., 2015. Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1), pp.148-175.
[2] Snoek, J., Larochelle, H. and Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.
[3] Bonnin, X., Dekeyser, W., Pitts, R., Coster, D., Voskoboynikov, S. and Wiesen, S., 2016. Presentation of the new SOLPS-ITER code package for tokamak plasma edge modelling. Plasma and Fusion Research, 11, pp.1403102-1403102.
[4] Lore, J., De Pascuale, S., Laiu, P., Phathanapirom, B., Brunton, S., Canik, J., Cetiner, S., Kutz, N. and Stangeby, P., 2021. Model predictive control of boundary plasmas using reduced models derived from SOLPS-ITER. Bulletin of the American Physical Society, 66.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Václav Šmídl, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Mgr. Michael Komm, Ph.D. Institute of Plasma Physics, Academy of Sciences of the Czech Republic

Date of bachelor's thesis assignment: **27.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

doc. Ing. Václav Šmídl, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date ....

Signature

Acknowledgements

First and foremost, I would like to thank my supervisor Doc. Ing. Václav Šmídl, Ph.D. for his guidance, the provided insight into the Bayesian methods, and the time dedicated to the consultations concerning the thesis.

I would also like to express my gratitude toward the physicists at the Institute of Plasma Physics of the CAS for the provided insight into the issues of today's tokamaks. Namely, I would like to thank Dr. David Tskhakaya and Dr. Oleg Shyshkin who dedicated their time to help me get a better understanding of the underlying plasma physics.

Abstract

Many physical phenomena in various applications can be computed using the laws of physics. However, these calculations can be computationally expensive. High computational demands are limiting for example when optimizing a physical process and it is needed to be evaluated many times. The goal of this thesis is to find a surrogate model approximating a complex numerical program requiring the least possible amount of evaluations. The main tool used to train the model is active learning in the form of Bayesian optimization. In the first chapters of the thesis, some of the commonly used surrogate models and experiment design methods are introduced. Afterward, plasma boundary modeling is presented as an example of a practical application of surrogate modeling. The result of the thesis is a comparison of selected methods on modeling of plasma momentum loss in tokamak.

Keywords: experiment design, surrogate model, active learning, Bayesian regression, tokamak, plasma modeling

Abstrakt

Mnoho fyzikálních jevů v různých aplikacích lze spočítat pomocí zákonů fyziky. Takové výpočty mohou ale být výpočetně drahé. Vysoké nároky na výpočetní sílu jsou limitující například při optimalizaci fyzikálního procesu, který je nutné mnohokrát evaluovat. Cílem této práce je najít náhradní model aproximující komplexní numerický program vyžadující co nejmenší počet evaluací. Hlavním nástrojem použitým pro trénování modelu je aktivní učení ve formě Bayesovské optimalizace. V prvních kapitolách práce jsou představeny některé z často používaných náhradních modelů a metod pro návrh experimentů. Poté je představeno modelování okraje plasmatu jako příklad praktického využití náhradních modelů. Výsledkem práce je porovnání vybraných metod na modelování ztráty hybnosti plasmatu v tokamaku.

Klíčová slova: návrh experimentů, náhradní model, aktivní učení, Bayesovská regrese, tokamak, modelování plasmatu

Contents

Chapter 1

Introduction

This thesis concerns surrogate modeling of functions, which are expensive to evaluate. The most common example of such functions are high-fidelity physical simulations which can sometimes take days to complete. Another example are real-world experiments that are monetarily expensive or time-demanding. When training models of such functions it is important to minimize the number of their evaluations. Because of that, the used experiment design methods have to select inputs for the evaluations carefully to minimize the data required to train the model.

Usages

One of the common usages of surrogate modeling is function optimization. During optimization, the objective function usually needs to be queried many times making optimization of expensive-to-evaluate functions problematic. In these cases, having a cheap surrogate model can be extremely advantageous as it can guide the optimization into the areas of the objective function domain which are more likely to yield an improvement according to the surrogate. This can make the optimization process faster, save resources and also lead to better results as the objective function domain can be searched more effectively [1, 2, 3, 4].

Surrogate modeling also allows for black-box function optimization. A black-box function can be queried for output values given an arbitrary input but no other knowledge about its properties is available. That means the gradient of the objective function is also unavailable which can be problematic for some optimization methods. Optimization methods using surrogate models can be used in this case as they usually require only the ability to query the objective function for data and can infer some information about the objective function from the surrogate model.

Another situation where a fast surrogate model can be advantageous is decision-making in real-time systems [5]. In cases where complex decisions must be made quickly, training a surrogate model can be a good solution.

Surrogate models can also be used to gain insight into the behavior of black-box functions or poorly understood physical models. A well-designed surrogate model can provide information about which input variables are the most impactful to different outputs or even provide an interpretable analytical approximation of the true model [6].

Method Selection

Surrogate modeling is a very broad topic with a large number of different methods varying in both the model and experiment design method used. The individual models differ mainly in the class of function approximators they use to model the original function and the general structure of the model. (Here as well as in the remainder of the document, the *original function* refers to the function being approximated by the surrogate model.)

The selection of the model and experiment design depends on the problem at hand. If any information about the function being approximated is available, it can be used to improve model selection. For example, it could be known that some variables are always positive, the function is periodic, the limit of the function approaches zero at one side of the domain, etc. Any additional information such as these examples can help in the selection of a model suitable for the problem.

The experiment design defines how the inputs for future evaluations of the original function are selected to get more valuable data. In active learning, the experiment design usually utilizes the surrogate model to determine the optimal next evaluation point in each iteration. The selection of a suitable experiment design depends largely on the objective. For example, if the goal is to minimize the original function, the experiment design should disregard the areas where the surrogate model predicts high function values. On the other hand, if the goal is to train a precise surrogate model, the experiment design should in general focus on the areas of the function domain where the surrogate model reports high uncertainty.

Thesis Structure

In the first two chapters of the thesis, the basic concepts of surrogate model development and experiment design are introduced and a selection of some of the commonly used methods is presented. In the third chapter, the challenging simulation of plasma edge in tokamaks is briefly introduced. In the next chapter, selected methods are employed in the modeling of plasma momentum loss, their performance is compared, and the results are discussed. Follows a short summary in the last chapter.

Chapter 2

Surrogate Modeling Methods

Introduction

Generally, a surrogate model is an approximation of some original function. The true function can be represented by a numerical program, simulation, or physical phenomena. The goal of surrogate modeling is to create an approximation that behaves as close as possible to the true function given the available data, but is usually less computationally demanding, more interpretable, or has some other desired properties.

The surrogate model is usually represented with an analytical definition of a function containing some free parameters. These are the parameters of the model. The model is trained by finding the best parameter values to fit the available data from previous evaluations of the original function. The exact way this is achieved depends on the surrogate model and methods used. The training can also be iterative, where an experiment design method is used to select the next point to query the original function for new data in each iteration. This is commonly known as *active learning*. Examples of experiment design methods will be discussed in the next chapter. When trained, the model can be used by the experiment design method to guide further evaluations or predict the outcome of the original function given some input.

The data used for the training are in most cases just a set of pairs of inputs and outputs of the true model. Additional information or expert knowledge about the true model can be incorporated into the surrogate model as well but it is often a non-trivial task [4, 7].

Notation

Here is a brief overview of the notation used in this chapter. The data-point generated from the i-th evaluation of the original function is labeled (x_i, y_i) where x_i is the input and y_i is the output of the function. A set of all data gathered by evaluating the original function is labeled D_N, where $N = |D_N|$ is the data set size. The symbol θ is used to represent the model parameters. The function $\phi(x)$ represents an arbitrary parameter-less transformation of the input vector x. Bold vectors or matrices emphasize that they include values of all data points. This includes the vector/matrix $\mathbf{Y}$ of all function values, the matrix $\mathbf{X}$ of all evaluation points, and the matrix $\mathbf{\Phi}$ of all evaluation points lifted by the function ϕ.

A survey of selected surrogate models and surrogate modeling methods are introduced in the remainder of this chapter.

1 Response Surface Model

Response surface models (RSM) are used to approximate a function via polynomials. RSM can be generalized with the following equation [8]:

$$\begin{aligned} y &= \phi(x)^T\theta + \epsilon \\ \phi(x) &= (\phi_1(x), ..., \phi_p(x))^T, \end{aligned} \tag{2.1}$$

where the functions $\phi_1, ..., \phi_p$ are **polynomial**, $\phi : \mathbb{R}^n \to \mathbb{R}^p$, $x \in \mathbb{R}^n$ is the input, $\theta \in \mathbb{R}^p$ is a vector of the model parameters, $y \in \mathbb{R}$ is the output, and ϵ represents a random zero-mean expriment evaluation noise. The function ϕ could for example be defined as $\phi(x) = (1, x_1, x_1x_2, x_2^2)^T$ where $x = (x_1, x_2)^T$.

RSM model can be expressed in a matrix form for N data-points $(x_1, y_1), ..., (x_N, y_N)$ as

$$\mathbf{Y} = \mathbf{\Phi}\theta + \epsilon, \tag{2.2}$$

where $\mathbf{Y} \in \mathbb{R}^N$ is a vector of values $y_1, ..., y_N$, the rows of the matrix $\mathbf{\Phi} \in \mathbb{R}^{N\times p}$ are calculated as $\phi(x_1), ..., \phi(x_N)$, and $\theta \in \mathbb{R}^p$ remains the same.

Finally, we can also expand the model for $y \in \mathbb{R}^m$. Then $\mathbf{Y} \in \mathbb{R}^{N\times m}$ and $\theta \in \mathbb{R}^{p\times m}$ is the parameter matrix.

The model parameters θ can be estimated using the least squares method as [8]

$$\theta = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Y}. \tag{2.3}$$

The most commonly used special cases of RSM are the *first-degree RSM* and the *second-degree RSM* [8]:

$$y = \theta_0 + x^T\theta_1 + \epsilon, \tag{2.4}$$

$$y = \theta_0 + x^T\theta_1 + x^T\theta_2 x + \epsilon, \tag{2.5}$$

where $\theta_0 \in \mathbb{R}$, $\theta_1 \in \mathbb{R}^n$, and the symmetrical matrix $\theta_2 \in \mathbb{R}^{n\times n}$ are the model parameters.

Experiment Design

The experiment design of RSM is described by a so-called *design matrix* $\mathbf{X}$. The rows of the design matrix represent the different points for experimental evaluation called *design points*. Many different designs for the first and second-order RSM have been proposed with different properties and demands on the number of experiments. The choice of a design model depends on the objective of the modeling as well as the experiment cost. Different objectives include training as precise model as possible, finding which input variables $x_1, ..., x_n$ have the most significant effect on the output value y, or finding an optimal setting of the input variables to minimize/maximize some utility function $U(y) : \mathbb{R}^m \to \mathbb{R}$. Applying multiple designs in succession can also be an effective strategy.

Some of the most commonly used designs and their properties are discussed in detail by A. I. Khuri and S. Mukhopadhyay in [8].

2 Radial Basis Function Model

Radial basis functions (RBF) are another class of functions commonly used to construct surrogate models. A radial basis function is a function whose output depends solely on the distance between the input point and some fixed center point. A radial basis function ϕ_i can be generalized as

$$\phi_i(x) = \bar{\phi}_i(||x - c||), \tag{2.6}$$

where the center point c is a hyperparameter of ϕ_i and $\bar{\phi}_i$ is an arbitrary function. The Euclidean distance is used most often but any other distance measure can be used as well.

A RBF surrogate model can be defined in the following form [9]:

$$\begin{aligned} y &= \phi(x)^T \theta_r + \epsilon \\ \phi(x) &= (\phi_1(x), ..., \phi_p(x))^T, \end{aligned} \tag{2.7}$$

where the functions $\phi_1, ..., \phi_p$ are **radial**, $\phi : \mathbb{R}^n \to \mathbb{R}^p$, $x \in \mathbb{R}^n$ is the input, $\theta_r \in \mathbb{R}^p$ is a vector of the model parameters, $y \in \mathbb{R}$ is the output, and ϵ represents a random zero-mean expriment evaluation noise.

Some examples of commonly used basis functions include [9]:

$$\begin{aligned} \text{Linear:} \quad & \bar{\phi}_i(d) = d, \\ \text{Quadratic:} \quad & \bar{\phi}_i(d) = \sqrt{d^2 + \alpha^2}, \\ \text{Gaussian:} \quad & \bar{\phi}_i(d) = exp(-\alpha d^2), \end{aligned} \tag{2.8}$$

where α is a hyperparameter which changes the "area of influence" of the basis function.

The RBF model is often extended by a bias term to get the form [9]

$$\begin{aligned} y &= \phi(x)^T \theta_r + \psi(x)^T \theta_b + \epsilon \\ \phi(x) &= (\phi_1(x), ..., \phi_p(x))^T \\ \psi(x) &= (\psi_1(x), ..., \psi_q()), \end{aligned} \tag{2.9}$$

where $\psi : \mathbb{R}^n \to \mathbb{R}^q$, $\theta_b \in \mathbb{R}^q$. The functions $\psi_1, ..., \psi_q$ are most commonly polynomial. In that case, we are in a way using the RSM as the bias.

Similarly to the RSM, the RBF model can also be written in a matrix form and extended for $y \in \mathbb{R}^m$ resulting in the form

$$\mathbf{Y} = \mathbf{\Phi}\theta_r + \mathbf{\Psi}\theta_b + \epsilon, \tag{2.10}$$

where $\mathbf{Y} \in \mathbb{R}^{N \times m}$, the rows of the matrix $\mathbf{\Phi} \in \mathbb{R}^{N \times p}$ are $\phi(x_1), ..., \phi(x_N)$, $\theta_r \in \mathbb{R}^{p \times m}$, the rows of the matrix $\mathbf{\Psi} \in \mathbb{R}^{N \times q}$ are $\psi(x_1), ..., \psi(x_N)$, $\theta_b \in \mathbb{R}^{q \times m}$, and N is the size of the data set.

The model parameters θ_r can be estimated using the least squares method as [9]

$$\theta_r = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T (\mathbf{Y} - \mathbf{\Psi}\theta_b), \tag{2.11}$$

where the bias parameters θ_b are estimated a priori as

$$\theta_b = (\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T \mathbf{Y}. \tag{2.12}$$

It is also possible to consider the parameters θ_b not known a priori. Then the estimation of parameters θ_r and θ_b becomes slightly more complicated. If you wish, refer to [9] for a detailed description of this approach.

3 Support Vector Regression

Support vector regression (SVR) is a method used for surrogate model training. SVR can be used to train any model with linear parameters [10] given by the equation

$$y = \phi(x)^T\theta + \epsilon, \tag{2.13}$$

where $y \in \mathbb{R}$ is the output, $x \in \mathbb{R}^n$ is the input, $\theta \in \mathbb{R}^p$ is the vector of model parameters, $\phi : \mathbb{R}^n \to \mathbb{R}^p$ is a parameter-free transformation of the input vector x, and ϵ is a random zero-mean experiment noise.

The parameters θ are learned by solving the following optimization problem [10]:

$$\hat{\theta} \in \arg\min_{\theta}\ C\frac{1}{N}\sum_{i=1}^{N}\max(0, |y_i - \phi(x_i)^T\theta| - \varepsilon) + \frac{1}{2}||\theta||^2, \tag{2.14}$$

where ε is a small positive value describing the maximum tolerated model error and the positive constant C controls the balance between maximizing the model precision and minimizing its complexity.

To solve this optimization problem we first rewrite it without the absolute value and max functions as [11]

$$\begin{aligned} \min \quad & \frac{1}{2}||\theta||^2 + C\frac{1}{N}\sum_{i=1}^{N}(\xi_i^- + \xi_i^+) \\ \text{s.t.} \quad & \begin{cases} y_i - \phi(x_i)^T\theta & \leq \varepsilon + \xi_i^- \\ \phi(x_i)^T\theta - y_i & \leq \varepsilon + \xi_i^+ \\ \xi_i^-, \xi_i^+ & \geq 0 \end{cases} \quad \forall i \in \{1, ..., N\}. \end{aligned} \tag{2.15}$$

Then we can construct the dual optimization problem [11]:

$$\begin{aligned} \max \quad & -\frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+)\,\phi(x_i)^T\phi(x_j) \\ & -\varepsilon\frac{1}{N}\sum_{i=1}^{N}(\alpha_i^- + \alpha_i^+) + \frac{1}{N}\sum_{i=1}^{N} y_i\,(\alpha_i^- - \alpha_i^+) \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^{N}(\alpha_i^- - \alpha_i^+) & = 0 \\ \alpha_i^-, \alpha_i^+ & \in [0, C] \quad \forall i \in \{1, ..., N\}. \end{cases} \end{aligned} \tag{2.16}$$

By solving this quadratic programming problem we get $\hat{\alpha_1}^-, \hat{\alpha_1^+}, ..., \hat{\alpha_N^-}, \hat{\alpha_N^+}$, which can be used to calculate the model parameters as [11]

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N}(\hat{\alpha_i^-} - \hat{\alpha_i^+})\,x_i. \tag{2.17}$$

4 Gaussian Process

4.1 Introduction

Gaussian process (GP) is arguably the most commonly used surrogate model for Bayesian optimization [12, 1, 2]. (Bayesian optimization is discussed in the following chapter in section 2.) For every point in the original function domain, a Gaussian process approximates its function value as a one-dimensional normal distribution. Therefore, a Gaussian process approximates the original function as an infinite-dimensional normal distribution over all possible forms the original function could have. The infinite-dimensionality comes from the fact, that the original function is continuous and thus there are infinitely many points in its domain each approximated with a single-dimensional normal distribution.

Gaussian processes have been described in detail by Carl Edward Rasmussen in the book *Gaussian Processes for Machine Learning* [13]. For an introduction to the fundamentals of GP, I also recommend the lectures by Nando de Freitas [14, 15].

4.2 Model Equations

A Gaussian process is determined by two parameters; a prior belief about the original function μ_0 and the kernel function k. Assume the original function $f : \mathbb{R}^n \to \mathbb{R}$. Then, the prior belief about the original function is a function $\mu_0 : \mathbb{R}^n \to \mathbb{R}$ and the kernel function is a positive-definite function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

The prior belief describes our expert knowledge about what the original function might look like. It is often omitted as its impact on the model diminishes with growing data. It can for example be set to a constant $\mu_0(x) = 0$.

The kernel function is also called the covariance function. It describes how the approximation of the original function at a given point is affected by the available data from previous evaluations of the original function.

The GP model approximates the original function value at a given point $\bar{x}$ by the normal predictive distribution [16]

$$\bar{x} \sim \mathcal{N}(\mu_N(\bar{x}), \sigma_N^2(\bar{x})), \tag{2.18}$$

where the mean function $\mu_N(\bar{x})$ and the variance function $\sigma_N^2(\bar{x})$ are given by

$$\mu_N(\bar{x}) = \mu_0(\bar{x}) + \mathbf{k}(\bar{x})^T(\mathbf{K} + \sigma_e^2\mathbf{I})^{-1}(\mathbf{Y} - \mathbf{m}), \tag{2.19}$$

$$\sigma_N^2(\bar{x}) = k(\bar{x}, \bar{x}) - \mathbf{k}(\bar{x})^T(\mathbf{K} + \sigma_e^2\mathbf{I})^{-1}\mathbf{k}(\bar{x}), \tag{2.20}$$

where the vector $\mathbf{k}(\bar{x}) \in \mathbb{R}^N$ contains the covariance values of the point $\bar{x}$ and each individual point x_i from the data set. The covariance matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ contains the covariance values of each pair of points x_i, x_j from the data set. The term $\mathbf{m}$ is a vector of the prior beliefs about the function value for the points from the data set. Therefore the aforementioned terms can be defined as

$$\begin{aligned} \mathbf{m} &= (\mu_0(x_1), ..., \mu_0(x_N))^T, \\ \mathbf{k}(\bar{x}) &= (k(x_1, \bar{x}), ..., k(x_N, \bar{x}))^T, \\ \mathbf{K}_{i,j} &= k(x_i, x_j). \end{aligned}$$

The subscript N of the mean and variance functions emphasizes that these terms depend on the data D_N. Using this set of equations, one can get both the approximation of the

original function at a given point $\bar{x}$ of the model domain by calculating the mean $\mu_N(\bar{x})$ and the uncertainty of the model at that point by calculating the variance $\sigma_N^2(\bar{x})$ based on the available data D_N.

4.3 Kernel Functions

The most commonly used stationary kernel functions are the **matérn** kernels. The three most common matérn kernels are [16]:

$$k_{matérn_1}(x_1, x_2) = \sigma_k^2 \, exp(-\, d), \tag{2.21}$$

$$k_{matérn_3}(x_1, x_2) = \sigma_k^2 \, exp(-\sqrt{3}\, d) \, (1 + \sqrt{3}\, d), \tag{2.22}$$

$$k_{matérn_5}(x_1, x_2) = \sigma_k^2 \, exp(-\sqrt{5}\, d) \, (1 + \sqrt{5}\, d + \frac{5}{3}\, d^2), \tag{2.23}$$

where d is the distance between the two given points x_1, x_2 given by

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \, \Lambda \, (x_1 - x_2)}, \tag{2.24}$$

where Λ is a diagonal matrix of length scales $\lambda_1, \lambda_2, ..., \lambda_n$.

The constants σ_k^2 and $\lambda_1, \lambda_2, ..., \lambda_n$ are hyperparameters of the matérn kernel functions. The general matérn kernel function $matérn_\nu$ is parametrized by the smoothness parameter ν. A special case of a matérn kernel with $\nu \to \infty$ is the **squared exponential kernel**

$$k_{sqexp}(x_1, x_2) = k_{matérn_\infty}(x_1, x_2) = \sigma_k^2 \, exp(-\frac{1}{2}\, d^2). \tag{2.25}$$

Many other kernel functions exist other than the matérn kernels. For example, the previously mentioned radial basis functions are also commonly used as kernels for Gaussian processes. The choice of a kernel function depends largely on the available information about the original function. For more information on different GP kernels and their selection see [17].

4.4 Hyperparameter Optimization

The GP model contains some hyperparameters. Namely the experiment noise σ_e^2, the prior belief μ_0, and potentionally some hyperparameters of the kernel function. The hyperparameters can be estimated by maximizing the marginial likelihood of the data rather than setting them manually. The marginal likelihood can be directly analytically expressed as [16]

$$\begin{aligned} &\log \, p(\mathbf{Y}|\mathbf{X}, \theta_h) = \\ &-\frac{1}{2}\left[(\mathbf{Y} - \mathbf{m}^{(\theta_h)})^T (\mathbf{K}^{(\theta_h)} + \sigma_e^2 \mathbf{I})^{-1} (\mathbf{Y} - \mathbf{m}^{(\theta_h)}) + \log \, |\mathbf{K}^{(\theta_h)} + \sigma_e^2 \mathbf{I}| + N \, log(2\pi)\right], \end{aligned} \tag{2.26}$$

where $\theta_h = (\mu_0, \sigma_e^2, \theta_k)$ is a tuple of model hyperparameters, where θ_k contains the hyperparameters of the kernel function. The superscript (θ_h) highlights the dependence of the term on the hyperparameters. The prior belief μ_0 is often ignored during the hyperparameter optimization as its effect on the model is not substantial.

Other methods for optimizing the hyperparameters of a Gaussian process exist as well [18].

4.5 Benefits and Limitations

A big advantage of Gaussian processes is that the model uncertainty at any point of the domain is very easily extracted from the model using the equation 2.20. Another benefit is the analytical expression for the marginal data likelihood 2.26, which allows for simple hyperparameter optimization.

On the other hand, the main limitation of Gaussian processes is the scalability with growing data. Once we calculate the matrix $(\mathbf{K} + \sigma_e^2\mathbf{I})^{-1}$, the model approximations and uncertainty for different points of the domain are readily available. However, the calculation of this matrix is computationally demanding with large data because of the matrix inversion. Some methods partially mitigating this issue have been introduced [19, 20, 21].

5 Equation Learner

5.1 Introduction

Equation learner (EQL) is an example of a method using neural networks for surrogate modeling. Neural networks are in their essence a type of surrogate model as they are generally trained with data composed of inputs and outputs of some original function and then used to predict future outputs. The topic of neural networks is a mature one with a plethora of literature dedicated to it. The EQL method will serve as a representative of this class of surrogate models in this work.

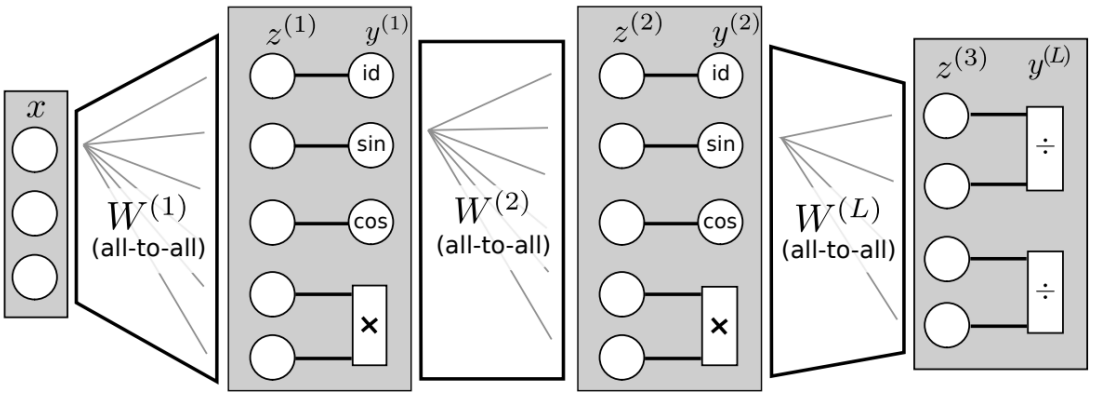
The EQL model constructs a surrogate model represented by a set of equations composed of provided algebraic operators. It was introduced by G. Martius and Ch. H. Lampert in 2016 [22]. The main difference of EQL to other models including other neural networks is that EQL strongly promotes model simplicity and interpretability instead of focusing solely on model precision.

EQL uses a shallow neural network consisting of several linear layers. Each layer except the last is followed by function nodes implementing different base functions which will be used to construct the model. The selection of the base functions depends on the original function being modeled and can be used to limit the model only to plausible functions.

Division cannot be included in the classic EQL as the limits at zero are problematic for the gradient-based optimization of the parameters of the neural network. An improved variant of the method labeled ($\text{EQL}^{\div}$) has been proposed in 2018 by S. S. Sahoo, Ch. H. Lampert and G. Martius [6], which adds a division node after the last layer as well as making the training more stable. The $\text{EQL}^{\div}$ and EQL methods are mostly similar. The $\text{EQL}^{\div}$ variant will be assumed for the rest of the section but the main differences from the classic EQL method will be mentioned when they come up.

The equations and hyperparameter settings presented in this section have been taken from the aforementioned works of S. S. Sahoo, Ch. H. Lampert and G. Martius [6, 22].

Figure 2.1: Example EQL$^{\div}$ NN architecture [6]



5.2 Neural Network Model

An example of an EQL$^{\div}$ neural network architecture is shown on the figure 2.1. The output vector of the l-th layer is denoted $y^{(l)}$ and it is computed as

$$\begin{aligned} y^{(l)} &= F^{(l)}(z^{(l)}) \\ F^{(l)}(z^{(l)}) &= (f_1^{(l)}(z_1^{(l)})) \\ z^{(l)} &= W^{(l)} y^{(l-1)} + w_0^{(l)}. \end{aligned} \tag{2.27}$$

Firstly, the simple linear layer with the matrix W and the vector w_0 as its parameters is applied to get the vector $z^{(l)}$ and then the function nodes are applied (denoted by the non-linear transformation $F^{(l)}$) to get $y^{(l)}$ - the output vector of the whole layer. The transformation $F^{(l)}$ applies different base functions $f_i^{(l)}$ to each element of the vector $z^{(l)}$.

Note that $y^{(0)} = x$ is the input of the neural network and $y^{(L)}$ is the output of the network, where L is the total number of layers. In the case of the classic EQL, it holds that $y^{(L)} = z^{(L)}$ as there is no function node after the last linear layer. In the EQL$^{\div}$ variant, the last transformation $F^{(L)}$ contains the added division.

5.3 Training

Stochasitc gradient descent with the *Adam* algorithm are usually used to train the neural network. The loss function used during the training is defined as

$$L := \frac{1}{N} \sum_{i=1}^{N} \|\psi(x_i) - y_i\|_2^2 + \lambda \sum_{l=1}^{L} \|W^{(l)}\|_1 + P_\tau, \tag{2.28}$$

where N is size of the data set and ψ denotes the neural network. The first two terms of the loss equation are the standard L_2 loss followed by the L_1 regularization. The last term is the penalty only present in the EQL$^{\div}$ variant which is used to steer the denominators of the divisions away from negative values. The penalty term is calculated as

$$\begin{aligned} P_\tau &:= \sum_{i=1}^{N} \sum_{j=1}^{m} p_\tau(z_{2j}^{(L)}(x_i)) \\ p_\tau(d) &:= max(\tau - d, 0), \end{aligned} \tag{2.29}$$

where m is the length of the output vector $y^{(L)}$ and the term $z_{2j}^{(L)}(x_i)$ denotes the value of the $2j$-th element of the vector $z^{(L)}$ given the network input vector x_i. Note that the length of the vector $z^{(L)}$ is equal to $2m$ and by iterating over every second element we select the denominators. The τ threshold is used to avoid large gradients in the division function. Its value is dependent on the current epoch t and is defined as

$$\tau(t) := 1/\sqrt{t+1}. \tag{2.30}$$

The training of the EQL neural network is split into three phases. The first phase is un-regularized as λ is set to zero. In the second phase, λ is set to a non-zero value for the regularization to take place. In the final phase, the model complexity is fixed and the exact parameter values are learned by setting λ to zero again but enforcing an unchanging L_1 norm. The L_1 norm can be fixed by resetting all weights close to zero back to zero (e.g. if $|w| < 0.001$ then $w = 0$) in each iteration.

Consider T as the total number of training epochs and t as the current epoch. Then the first phase takes place while $t \leq \frac{1}{4}T$ then the second phase takes place while $t \leq \frac{19}{20}T$ and the third phase takes place in the remaining epochs.

Additionally, *penalty epochs* are introduced in the EQL$^{\div}$ variant to prevent overfitting. Penalty epochs are inserted in regular intervals (e.g. every 50 epochs) which use a different loss function

$$L_P = P_\tau + P_b, \tag{2.31}$$

$$P_b := \sum_{i=1}^{N}\sum_{j=1}^{m}\left[max(y_j^{(L)}(x_i) - b, 0) + max(-y_j^{(L)}(x_i) - b, 0)\right], \tag{2.32}$$

where b is a hyperparameter constant dependent on the problem but easily estimated from the observed data. (It is reasonable to set b to around 3 times the maximum expected output value.) This custom loss function helps to keep the output values from having vastly different magnitudes than the observed outputs.

5.4 Benefits and Limitations

The advantages of the EQL modeling method include the ability to incorporate expert knowledge into the model via the base functions, the fact that EQL favors simple models which can be more interpretable, and if suitable base functions and NN structure are selected the model can extrapolate surprisingly well in contrast to most other conventional models which behave very poorly outside of the training domain. (See the results in the original articles.)

One of the limitations of the EQL modeling is the added hyperparameters in the form of the base functions which the researcher has to provide and which can be non-trivial to select without any insight into the original model. Another disadvantage is that this model does not provide an uncertainty estimation which is necessary if the model is to be used in active learning. This issue can be solved for example by implementing Bayesian ridge regression into the method which allows for model uncertainty estimation. However, using Bayesian ridge regression with an EQL model with more than one layer introduces some complications as the model parameters are not linear in that case. (Bayesian ridge regression is discussed in the next chapter in section 3.)

Chapter 3

Experiment Design

Introduction

Experiment design in surrogate modeling describes how to select the inputs for the evaluations of the original function to generate data for the training of the surrogate model. In active learning, the experiment design method selects a single input for evaluation in each iteration. Some methods can also be modified to select multiple evaluation points in each iteration, which can be advantageous when more than one experiment can be performed in parallel generating multiple data points at the same time. The experiment design often utilizes the surrogate model to determine the optimal next evaluation point. It can for example use the prediction given by the model to estimate the outputs of the original function or use the uncertainty of the model in different parts of the domain to determine which regions should be explored more. Some experiment design methods also try to estimate the effect new data at certain points of the domain will have on the surrogate model.

Different methods for expriment design for surrogate model development are introduced in the following sections. The first two methods - space mapping and Bayesian optimization - are mainly used for function optimization whereas the third method - Bayesian ridge regression - is used for model development.

1 Space Mapping

Space mapping is an experiment design method used for optimization of an expensive-to-evaluate high fidelity model referred to as the *fine model* using a cheaper less precise model referred to as the *coarse model*. An important prerequisite of the method is that the coarse and fine models describe the same phenomena. For this reason, the method is usually used with physically based models where the fine model is often some complex simulation and some basic equations neglecting losses or finer physical processes are used as the coarse model.

The equations in this section have been taken from [23].

Let $R_f(x), R_c(x) \in \mathbb{R}^m$ be the response vectors (i.e. the output) of the fine and coarse physical models given some input vector $x \in \mathbb{R}^n$. Let $R_s(x, \theta) \in \mathbb{R}^m$ be the response vector of the surrogate model given some input vector $x \in \mathbb{R}^n$ and model parameters $\theta \in \mathbb{R}^p$. The surrogate model is composed of the coarse model and some transformation of x dependent

on the parameters θ. For instance, a linear transformation could be used resulting in the following surrogate model: $R_s(x, \theta) = R_c(\theta^T x)$. In that case, p would equal n.

Our goal is to find some input $\hat{x}$ for the fine model yielding an optimal response vecor in the sense of some objective function $U : \mathbb{R}^m \to \mathbb{R}$. More precisely

$$\hat{x} = \arg\min_x U(R_f(x)). \tag{3.1}$$

Direct optimization of the fine model would be too computationally expensive. For that reason, we will employ the surrogate model to aid with the optimization. The optimization algorithm consists of two main steps;
1) Fit the surrogate model to the fine model:

$$\theta_k = \arg\min_\theta \sum_{i=0}^{k} w_i ||R_s(x_i, \theta) - R_f(x_i)||, \tag{3.2}$$

2) Find the optimal solution using the cheap surrogate model:

$$x_{k+1} = \arg\min_x U(R_s(x, \theta_k)), \tag{3.3}$$

where the index of the current iteration k is also the size of the current data set $D_k = \{(x_1, R_f(x_1)), ..., (x_k, R_f(x_k))\}$. The weights $w_1, ..., w_k \in \mathbb{R}$ can be used to control the contribution of different data points to the training. These two steps are repeated until a satisfactory solution is found.

Notice that the fine model only needs to be evaluated once in each iteration as a single new point x_{k+1} is added to the data set whereas the surrogate model is evaluated many times during the optimization process. This way a lot of resources can be saved as the fine model is usually much more computationally demanding.

2 Bayesian Optimization

2.1 Introduction

Bayesian optimization has been coined by Jonas Mockus in 1970s [24] and since been studied by him [25] and many others [16, 12, 26].

Bayesian optimization is a powerful tool for global optimization. Most commonly, it is used to approximate the global optimum of black-box functions while evaluating them as few times as possible. (Black-box function is a function with an unknown analytical form and with no information about its behavior available. It can only be queried for function values at specific points of its domain.)

Complicated physical simulations are a perfect example of functions that can be effectively optimized using Bayesian optimization [1, 2, 3]. They can be very expensive and slow to evaluate limiting the ability to query them for large amounts of data. They are also usually fairly complicated and the effect of different parameters on the result is hard to predict making them difficult to optimize.

2.2 Basic Concept

Bayesian optimization creates and successively updates a probabilistic surrogate model approximating the objective function which is then used to guide further optimization of the

objective function. The surrogate model is created using the data obtained from previous objective function evaluations. The method is similar in principle to space mapping. However, a big advantage of the probabilistic approach is that Bayesian optimization can use the uncertainty of the surrogate model to improve the experiment design.

In each iteration of the Bayesian optimization, firstly, the next point to evaluate the objective function at is selected by maximizing the so-called *acquisition function*. Then the objective function is evaluated, the data set is augmented with the new data point, and the surrogate model is updated according to the new data. The point of the next evaluation of the objective function is usually selected in order to lower the uncertainty of the surrogate model and/or to try to improve the best global optimum of the objective function found so far. The balance between these two objectives is controlled by the selection of the acquisition function.

Algorithm 1 Bayesian Optimization

procedure OPTIMIZE
  while not term_cond() **do**
    $\bar{x} \leftarrow \arg\max_x \alpha(x, \mathbb{S})$
    $\bar{y} \leftarrow f(\bar{x})$
    $D_i \leftarrow D_{i-1} \cup \{(\bar{x}, \bar{y})\}$
    $\mathbb{S} \leftarrow$ update_model(D_i)
  end while
end procedure

The general algorithm of Bayesian optimization is described by the algorithm 1. The function α is the *acquisition function*, $\mathbb{S}$ is the surrogate model, f is the original function, also called the *objective function* as it is the objective of the optimization, and D_i is the data set after the i-th iteration.

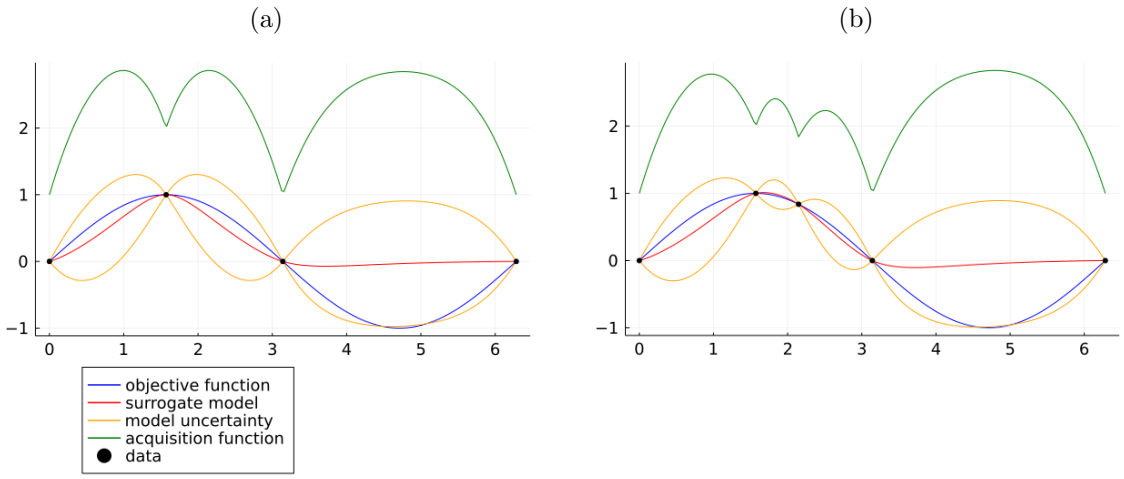
The acquisition function and the surrogate model are selected based on the problem. Any additional information about the objective function properties can help in the selection of a suitable surrogate model. The acquisition function selection depends mainly on the goal. For example, a different acquisition function will be suitable for function maximization and cumulative reward maximization.

In the figure 3.1, there is an example of what an iteration of Bayesian optimization can look like. You can see how the surrogate model changes during the optimization in the plots (a) and (b) which depict two successive iterations.

The blue line represents the objective function. In this example, the objective function is the sinus function. In a real-world application, the objective function is unknown. The only information about the objective function available to the model are the black data points obtained from previous objective function evaluations. (There is no evaluation noise in this example, therefore the data points lie exactly on the graph of the objective function.)

The red and yellow lines represent the surrogate model. The red line is the prediction of the objective function values given by the surrogate model and the yellow lines represent the uncertainty of the model. The bigger the interval between the two yellow lines is, the bigger the uncertainty of the approximation at that point of the domain is. The surrogate model used here is the Gaussian process with the $matérn_3$ kernel.

Figure 3.1: Example of Bayesian Optimization



Finally, the green graph represents the acquisition function. You can see that the value of the acquisition function is high where the uncertainty of the surrogate model is high while also being higher close to data points with high function values. These are generally the areas where the objective function is likely to yield high values. The acquisition function used here is the upper confidence bound which will be introduced later.

In the two plots, we can see how the model changed after a single iteration of the Bayesian optimization. In plot (b), a new data point was added exactly where the acquisition function was maximized in plot (a). By updating the model with new data, its approximation of the objective function improved and the uncertainty around the new data point dropped drastically. (In this case, it drops to zero as a zero experiment noise is assumed by the model.) Because of that, the acquisition function drops there as well and the next evaluation point would be selected elsewhere.

2.3 Acquisition Functions

Introduction

The acquisition function is used to determine the next point of evaluation of the objective function during Bayesian optimization. The acquisition function has the same domain as the objective function and given a point from it returns a value describing its suitability as the next point of evaluation. Thus maximization of this function results in the point in some way most suitable for the next evaluation.

The shape of the acquisition function changes with the updating surrogate model. In general, the acquisition function returns high values in two different areas of the domain. Firstly, in the areas where the uncertainty of the surrogate model is high - where there is a lack of data. Secondly, in the areas where the surrogate model predicts high objective function values - in the proximity of previous objective function evaluations which yielded high values. This corresponds to the well-known problem of exploration versus exploitation. The choice of the acquisition function determines how these two objectives will be kept in balance.

The acquisition function should be computationally cheap to evaluate in comparison to the objective function so it is easily optimized.

Below is an example of a very simplistic acquisition function, which would never be used in practice, but will be useful to get a basic understanding of how acquisition functions work.

Assume an extremely greedy policy trying to maximize the expected outcome of the objective function evaluation in each iteration of the Bayesian optimization. Such a policy could compare the suitability of different points of the objective function domain simply by the objective function values predicted by the surrogate model always selecting the point with the highest predicted value. The acquisition function α_{greedy} of such policy would be

$$\alpha_{greedy}(x) = \mu_y(x), \tag{3.4}$$

where $\mu_y(x)$ is the mean of the predictive distribution for the point x of the function domain. (The predictive distribution is defined by the probabilistic surrogate model.)

This policy would in practice perform very poorly as it completely disregards exploration in favor of exploitation. Below are introduced some commonly used policies which balance exploration and exploitation better. The equations of the acquisition functions discussed below have been taken from [16].

Probability of Improvement (PI)

This policy selects the next objective function evaluation point based on the probability that the evaluation yields a higher value than some fixed value τ. This makes it suitable for objective function optimization. The hyperparameter τ is usually set to the highest objective function value found yet but can be set to different values as well.

Assuming the predictive distribution of the surrogate model is Gaussian, the PI acquisition function can be calculated as

$$\alpha_{PI}(x) = \Phi\left(\frac{\mu_y(x) - \tau}{\sigma_y(x)}\right), \tag{3.5}$$

where $\mu_y(x)$ and $\sigma_y^2(x)$ are the mean and variance of the normal predictive disrtibution for the point x and Φ is the cumulative distribution function of the standard normal distribution.

Note that the predictive distribution of the surrogate model changes with the growing data set and so the acquisition function will change with new data as well.

Expected Improvement (EI)

This is a similar policy to the PI policy. However, rather than considering just the probability of improvement, it considers the amount of improvement as well. More precisely, the candidate points are compared based on the average expected improvement over some value τ. This policy is also used for objective function optimization.

Assuming a Gaussian predictive distribution of the surrogate model, the EI acquisition function can be calculated as

$$\alpha_{EI}(x) = (\mu_y(x) - \tau)\ \Phi\left(\frac{\mu_y(x) - \tau}{\sigma_y(x)}\right) + \sigma_y(x)\ \phi\left(\frac{\mu_y(x) - \tau}{\sigma_y(x)}\right), \tag{3.6}$$

where the terms τ, Φ, $\mu_y(x)$ and $\sigma_y^2(x)$ are the same as in the previous case and ϕ is the probability density function of the standard normal distribution. You can compare the equation with the previous one.

Upper Confidence Bound (UCB)

The upper confidence bound acquisition function balances exploration and exploitation by considering the surrogate model's prediction as well as the uncertainty of that prediction for each evaluated point. Using the surrogate model, the objective function value at a given point x can be predicted to be within some interval I centered around $\mu_y(x)$ with some fixed confidence. The upper confidence bound acquisition function computes the suitability of the point x as the upper bound of such confidence interval.

When the uncertainty of the surrogate model is high, the confidence interval I will be larger pushing its upper bound higher. When the predicted objective function value $\mu_y(x)$ is high, the upper bound of the interval I will also be higher as the whole interval is pushed higher. This way, exploration and exploitation are balanced.

This policy is suitable for both objective function optimization and cumulative reward maximization.

Again, assuming a normal predictive distribution, the UCB acquisition function is defined as

$$\alpha_{UCB}(x) = \mu_n(x) + \beta\, \sigma_n(x), \tag{3.7}$$

where the fixed confidence used to calculate the confidence intervals can be altered by changing the β hyperparameter.

Thompson Sampling (TS)

Thompson sampling is a bit different from the previous two experiment design methods. It is not defined with an analytical acquisition function which would be maximized in each iteration of the Bayesian optimization. Instead, a point x from the objective function domain is selected as the point of the next objective function evaluation with a probability proportionate to the probability that x is the optimum of the objective function. This can be done very easily without the need to directly calculate this probability simply by taking a single sample of the model parameters θ from the posterior distribution $p(\theta|D)$ and then maximizing surrogate model prediction with the sampled parameters. The selection of the next evaluation point $\bar{x}$ is therefore performed by solving

$$\begin{aligned} \bar{x} &= \arg\max_x m_{\theta_s}(x) \\ \theta_s &\sim p(D|\theta)\, p(\theta) \propto p(\theta|D), \end{aligned} \tag{3.8}$$

where θ_s is drawn from $p(D|\theta)\, p(\theta)$. (This distribution is proportional to the parameter posterior $p(\theta|D)$ because of the Bayes' theorem.)

The Thompson sampling is most commonly used for cumulative reward maximization.

The sampling makes the method stochastic but the introduced randomness can be advantageous for example for problems where the objective function is evaluated in batches in parallel. In that case, multiple promising evaluation points can be selected easily by simply evaluating the experiment design method multiple times.

3 Bayesian Ridge Regression

Bayesian ridge regression is an experiment design method used for model development. The basic algorithm of Bayesian regression is identical to Bayesian optimization. It also selects

a point for the next evaluation of the true model in each iteration and then updates the surrogate model according to the result. However, the main difference is in the evaluation point selection. In Bayesian optimization, the acquisition function was used to balance exploration and exploitation to direct the search for the optimum of the objective function. In Bayesian regression, a so-called *utility function* is used instead which selects such evaluation points which will improve the surrogate model the most.

The equations in this section have been taken (with slight alterations in notation) from the *Pattern Recognition and Machine Learning* book written by Ch. M. Bishop [27].

3.1 Linear Regression

The Bayesian regression is discussed for a linear model below. Laplace approximation is introduced later which is one of the common methods for dealing with non-linear models.

Experiment and Model

Assume we have an experiment that can be run with different input parameters x and some values y are observed. The experiment behaves as a black-box function. Now assume we would like to create some model which would approximate the results $\bar{y}$ of the experiment for new inputs $\bar{x}$. This can be motivated by the experiment being expensive to run or by the wish to gain some insight into the dependence of y on x.

Assume the experiment

$$y = \xi(x) + \epsilon, \tag{3.9}$$

where $y \in \mathbb{R}$, $x \in \mathbb{R}^n$, and the measured values y are given by a deterministic function ξ and some noise variable $\epsilon \sim \mathcal{N}(0, \omega^{-1}I)$.

Assume the linear model

$$y = \theta^T \phi(x), \tag{3.10}$$

where y and x are the outputs and inputs of the model, $\theta \in \mathbb{R}^p$ are the model parameters and $\phi : \mathbb{R}^n \to \mathbb{R}^p$ is a function composed from so-called *basis functions* $\phi_1, ..., \phi_p : \mathbb{R}^n \to \mathbb{R}$.

Least Squares

The solution $\hat{\theta}$ for the model parameters in the sense of least squares is presented below.

Assume some data $D_N = \{(x_1, y_1), ..., (x_N, y_N)\}$ from previous runs of the experiment. Let $\mathbf{X} \in \mathbb{R}^{N \times n}$ be a matrix of input vectors $x_1, ..., x_N$ and $\mathbf{Y} \in \mathbb{R}^N$ be a vector of output values $y_1, ..., y_N$. We wish to learn the parameters $\hat{\theta}$ which maximize the likelihood of the observed data D_N. We will start by expressing the log-likelihood of the observed data

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X}, \theta) &= \prod_{i=1}^{N} \mathcal{N}(y_i|\theta^T \phi(x_i), \omega^{-1}I) \\ \log\ p(\mathbf{Y}|\mathbf{X}, \theta) &= \sum_{i=1}^{N} \mathcal{N}(y_i|\theta^T \phi(x_i), \omega^{-1}I) \\ &= \frac{Nm}{2} \log(\frac{\omega}{2\pi}) - \omega\ Err(\theta), \end{aligned} \tag{3.11}$$

where the least squares error $Err(\theta)$ of the model is defined as

$$Err(\theta) = \frac{1}{2} \sum_{n=1}^{N} (y_i - \theta^T \phi(x_i))^2. \tag{3.12}$$

The gradient of the log-likelihood of the data can be expressed as

$$\nabla ln\, p(\mathbf{Y}|\mathbf{X},\theta) = \sum_{n=1}^{N}(y_i - \theta^T\phi(x_i))\phi(x_i)^T. \tag{3.13}$$

Finally, by setting the gradient to zero we can derive the equation for the optimal model patameters

$$\hat{\theta} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Y}, \tag{3.14}$$

where the rows of the matrix $\mathbf{\Phi}$ are equal to $\phi(x_1), ..., \phi(x_N)$. (See [27] for a more detailed derivation.)

3.2 Posterior Parameter Distribution

The posterior model parameter distribution is defined as

$$p(\theta|D) = \frac{p(D|\theta)\, p(\theta)}{p(D)} \propto p(D|\theta)\, p(\theta). \tag{3.15}$$

In the case of Gaussian prior and a linear model, an analytical form exists. Assume the prior parameter distribution $p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$. If we assume the linear model from the first subsection, then the posterior distribution $p(\theta|D)$ is also Gaussian and can be calculated as

$$\begin{aligned} p(\theta|D) &= \mathcal{N}(\theta|\bar{\mu_\theta}, \bar{\Sigma_\theta}) \\ \bar{\mu_\theta} &= \bar{\Sigma_\theta}(\Sigma_\theta^{-1}\mu_\theta + \omega\mathbf{\Phi}^T\mathbf{Y}) \\ \bar{\Sigma_\theta} &= (\Sigma_\theta^{-1} + \omega\mathbf{\Phi}^T\mathbf{\Phi})^{-1}. \end{aligned} \tag{3.16}$$

Usually, a zero-mean prior with covariance matrix $(\alpha^{-1}I)$ is considered where α is a positive number. Then we get

$$\begin{aligned} \bar{\mu_\theta} &= \omega\bar{\Sigma_\theta}\mathbf{\Phi}^T\mathbf{Y} \\ \bar{\Sigma_\theta} &= (\alpha I + \omega\mathbf{\Phi}^T\mathbf{\Phi})^{-1}. \end{aligned} \tag{3.17}$$

Using either of the equations above, we do not only get the highest likelihood estimation of θ (as the $\bar{\mu_\theta}$) but the covariance of the θ distribution as well, which will be useful in determining the utility of input points for potential future experiments as we will see later.

3.3 Approximating Integrals over Distributions

Before we continue with the Bayesian regression, let us have a look at a simple trick which can be used to approximate an analytically intractable integral over a distribution from which we can draw samples.

An integral over a distribution $p(x)$ can be approximated by drawing samples $x_1, ..., x_S$ from $p(x)$ as

$$\begin{aligned} \int g(x)p(x)dx &\approx \frac{1}{S}\sum_{i=1}^{S} g(x_i) \\ x_1, ..., x_S &\overset{\text{iid}}{\sim} p(x), \end{aligned} \tag{3.18}$$

where $g(x)$ is some arbitrary function of x.

3.4 Posterior Predictive Distribution

The model predictions are given by the posterior predictive distribution defined as

$$p(y|x, D) = \int p(y|x, \theta)\, p(\theta|D)\, d\theta. \tag{3.19}$$

The prediction $\tilde{y}$ of the model for a new point x can be estimated using the equation 3.18 as

$$\begin{aligned} \tilde{y} &\approx \frac{1}{S} \sum_{i=1}^{S} \theta_i^T \phi(x) \\ \theta_1, ..., \theta_S &\overset{\text{iid}}{\sim} p(\theta|D). \end{aligned} \tag{3.20}$$

In the case the model is linear and the posterior parameter distribution is Gaussian, the predictive distribution will also be Gaussian and can be calculated analytically as

$$\begin{aligned} p(y|x, D) &= \mathcal{N}(\mu_y, \sigma_y) \\ \mu_y &= \mu_\theta^T \phi(x) \\ \sigma_y^2 &= \sigma_e^2 + \phi(x)^T \Sigma_\theta \phi(x), \end{aligned} \tag{3.21}$$

where $p(\theta|D) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$ and σ_e^2 is the assumed experiment noise.

3.5 Utility Functions

In this subsection, the two most commonly used utility functions for experiment design in Bayesian ridge regression are presented.

Information Gain (IG)

The information gain method aims to maximize the gain in Shannon information of the next experiment. The utility of a potential new data point $(\bar{x}, \bar{y})$ is defined as the difference in the expected entropy of the posterior parameter distribution after the data set is augmented by the new data point and its current entropy [28]

$$\begin{aligned} U_{IG}(\bar{x}, \bar{y}) = &\ H(\theta|D) - H(\theta|\bar{D}) = \\ &= \int \log(p(\theta|\bar{D}))\, p(\theta|\bar{D})\, d\theta \ - \int \log(p(\theta|D))\, p(\theta|D)\, d\theta, \end{aligned} \tag{3.22}$$

where $\bar{D} := D \cup \{(\bar{x}, \bar{y})\}$ are the current data D augmented by the new data point.

When evaluating the utility of a potential next evaluation point $\bar{x}$, the value $\bar{y}$ is unknown. The utility is therefore calculated by integrating over $\bar{y}$

$$U_{IG}(\bar{x}) = \int U_{IG}(\bar{x}, \bar{y})\, p(\bar{y}|\bar{x}, D)\, d\bar{y}, \tag{3.23}$$

where $p(\bar{y}|\bar{x}, D)$ is the posterior predictive distribution of the model for the point $\bar{x}$.

By combining the previous two equations we get the equation

$$\begin{aligned} U_{IG}(\bar{x}) = &\int \int \log(p(\theta|\bar{D}))\, p(\theta|\bar{D})\, d\theta\, p(\bar{y}|\bar{x}, D)\, d\bar{y} \\ &- \int \log(p(\theta|D))\, p(\theta|D)\, d\theta. \end{aligned} \tag{3.24}$$

Note that the second integral does not have to be evaluated when maximizing the utility over $\bar{x}$ as it is not dependent on $\bar{x}$.

The double integral over $\bar{y}$ and θ can be aproximated by sampling the variables from $p(\bar{y}|\bar{x}, D)$ and $p(\theta|\bar{D})$ respectively by using the equation 3.18.

In case the posterior model parameter distribution $p(\theta|D)$ is Gaussian, we can calculate its entropy $H(\theta)$ analytically as [29]

$$H(\theta|\bar{D}) = \frac{1}{2}\log(|\bar{\Sigma_\theta}|) + \frac{p}{2}(1 + \log(2\pi)), \tag{3.25}$$

where $p(\theta|\bar{D}) = \mathcal{N}(\bar{\mu_\theta}, \bar{\Sigma_\theta})$. Therefore, considering a Gaussian parameter distribution, the information gain utility for a point $\bar{x}$ can be calculated analytically as

$$\begin{aligned} U_{IG}(\bar{x}) &= H(\theta|D) - \int H(\theta|\bar{D})\, p(\bar{y}|\bar{x}, D)\, d\bar{y}, \\ U_{IG}(\bar{x}) &= H(\theta|D) - \int \left[\frac{1}{2}\log(|\bar{\Sigma_\theta}|) + \frac{p}{2}(1 + \log(2\pi))\right] p(\bar{y}|\bar{x}, D)\, d\bar{y}, \\ U_{IG}(\bar{x}) &= H(\theta|D) - \left[\frac{1}{2}\log(|\bar{\Sigma_\theta}|) + \frac{p}{2}(1 + \log(2\pi))\right], \end{aligned} \tag{3.26}$$

where the entropy $H(\theta|D)$ of the current data set does not have to be evaluated when maximizing $U_{IG}(\bar{x})$ over $\bar{x}$. Note that we can leave out the integral over $\bar{y}$ because the entropy $H(\theta|\bar{D})$ does not depend on $\bar{y}$ and integral over any probability distribution equals 1. (The entropy $H(\theta|\bar{D})$ only depends on $\bar{x}$. See the section 3.2.)

Kullback-Leibler Divergence (KL)

The aim of this method is to choose a new evaluation point $\bar{x}$ such that the Kullback-Leibler divergence of the new posterior $p(\theta|\bar{D})$ from the previous distribution $p(\theta|D)$ is maximized. The utility of a potential new data point $(\bar{x}, \bar{y})$ is defined as [30]

$$\begin{aligned} U_{KL}(\bar{x}, \bar{y}) &= D_{KL}(p(\theta|\bar{D}) \,||\, p(\theta|D)) \\ D_{KL}(p(\theta|\bar{D}) \,||\, p(\theta|D)) &= \int \log(p(\theta|\bar{D}))\, p(\theta|\bar{D})\, d\theta \\ &\quad - \int \log(p(\theta|D))\, p(\theta|\bar{D}) d\theta. \end{aligned} \tag{3.27}$$

The value of the utility of $\bar{x}$ is again calculated by integrating over $\bar{y}$

$$U_{KL}(\bar{x}) = \int U_{KL}(\bar{x}, \bar{y})\, p(\bar{y}|\bar{x}, D)\, d\bar{y}. \tag{3.28}$$

And by combining the previous two equations we get

$$\begin{aligned} U_{KL}(\bar{x}) = &\int\int \log(p(\theta|\bar{D}))\, p(\theta|\bar{D})\, d\theta\, p(\bar{y}|\bar{x}, D)\, d\bar{y} \\ &- \int\int \log(p(\theta|D))\, p(\theta|\bar{D})\, d\theta\, p(\bar{y}|\bar{x}, D)\, d\bar{y}. \end{aligned} \tag{3.29}$$

This equation can again be approximated by sampling from the distributions $p(\bar{y}|\bar{x}, D)$ and $p(\theta|\bar{D})$ and using the equation 3.18.

In case the posterior parameter distribution is Gaussian, the Kullback-Leibler divergence can be calculated analytically as [31]

$$D_{KL}(p(\theta|\bar{D}) \,||\, p(\theta|D)) = \frac{1}{2}\left(tr(\Sigma_\theta^{-1}\bar{\Sigma}_\theta) + (\mu_\theta - \bar{\mu_\theta})^T\Sigma_\theta^{-1}(\mu_\theta - \bar{\mu_\theta}) - p - \log\frac{|\bar{\Sigma}_\theta|}{|\Sigma_\theta|}\right), \tag{3.30}$$

where $p(\theta|D) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$, $p(\theta|\bar{D}) = \mathcal{N}(\bar{\mu_\theta}, \bar{\Sigma}_\theta)$.

3.6 Model Selection

Bayesian regression can also be used to select the best surrogate model between multiple considered models. To do this we need to modify the predictive distribution and utility calculation slightly. Otherwise, the method remains largely the same.

Mixed Predictive Distribution

The predictive distribution is replaced with the mixed predictive distribution. For K considered models, the mixed predictive distribution is expressed as

$$p(y|x, D) = \sum_{i=1}^{K} p(y|x, M_i, D)\, p(M_i|D), \tag{3.31}$$

where the predictive distribution of an individual model $p(y|x, M, D)$ has been introduced in the equation 3.19.

Model Evidence

The probability of the observed data $p(D|M)$ is called the *model evidence*. It is calculated by integrating over the parameters θ_M of the model M as

$$p(D|M) = \int p(D|M, \theta_M)\, p(\theta_M)\, d\theta_M. \tag{3.32}$$

Considering a linear model and a Gaussian parameter prior, the evidence can be calculated as

$$p(D|M) = (\frac{\omega}{2\pi})^{\frac{N}{2}}(\frac{\alpha}{2\pi})^{\frac{L}{2}} \int \exp(-E(\theta_M))\, d\theta_M, \tag{3.33}$$

where $E(\theta_M)$ is defined as

$$\begin{aligned} E(\theta_M) &= \frac{\omega}{2}||Y - \Phi_M\theta_M||^2 + \frac{\alpha}{2}||\theta_M||^2 \\ E(\theta_M) &= E(\mu_{\theta_M}) + \frac{1}{2}(\theta_M - \mu_{\theta_M})^T(\Sigma_{\theta_M})^{-1}(\theta_M - \mu_{\theta_M}), \end{aligned} \tag{3.34}$$

where $p(\theta_M|D) = \mathcal{N}(\mu_{\theta_M}, \Sigma_{\theta_M})$, $p(\theta_M) = \mathcal{N}(0, \alpha^{-1}I)$, $\epsilon \sim \mathcal{N}(0, \omega^{-1}I)$ and $\theta_M \in \mathbb{R}^L$.

Using this equation, the integral from the equation 3.33 can be expressed as

$$\int \exp(-E(\theta_M))\, d\theta_M = \exp(-E(\mu_{\theta_M}))\, (2\pi)^{\frac{L}{2}}\, |\Sigma_{\theta_M}^{-1}|^{-\frac{1}{2}}. \tag{3.35}$$

By substituting this expression into the equation 3.33 we get

$$\begin{aligned} p(D|M) &= (\frac{\omega}{2\pi})^{\frac{N}{2}} (\frac{\alpha}{2\pi})^{\frac{L}{2}} \exp(-E(\mu_{\theta_M})) \, (2\pi)^{\frac{L}{2}} \, |\Sigma_{\theta_M}^{-1}|^{-\frac{1}{2}} \\ \log \, p(D|M) &= \frac{N}{2} \log \, \omega + \frac{L}{2} \log \, \alpha - E(\mu_{\theta_M}) - \frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_{\theta_M}^{-1}|, \end{aligned} \tag{3.36}$$

where N is the size of the data set.

Model evidence is used to calculate the posterior model probability in the next section.

Posterior Model Probability

Using the Bayes' theorem, we see that the posterior probability of one of the considered models is proportional to the likelihood of the observed data multiplied by the model prior probability:

$$p(M_i|D) \propto p(D|M_i) \, p(M_i). \tag{3.37}$$

Therefore, using the model evidence we can calculate the posterior model probability as

$$p(M_i|D) = \frac{p(D|M_i) \, p(M_i)}{\sum\limits_{j=1}^{K} p(D|M_i) \, p(M_i)}, \tag{3.38}$$

where K is the number of considered models.

Additionally, if we assume the same prior pribability for all models, then $p(M_i|D) \propto p(D|M_i)$ and the previous equation simplifies to

$$p(M_i|D) = \frac{p(D|M_i)}{\sum\limits_{j=1}^{K} p(D|M_i)}. \tag{3.39}$$

Utility Modification

To select new points for evaluation which will help the most in determining the correct model, we can again use the information gain or the Kullback-Leibler divergence. However, the methods need to be modified to express the divergence of the model distribution instead of the parameter distribution.

The **information gain utility** will transform to

$$\begin{aligned} U_{IG}(\bar{x}) =& H(M|D) - \int H(M|\bar{D}) \, d\bar{y} \\ U_{IG}(\bar{x}) =& H(M|D) + \int \sum_{i=1}^{K} \left[\log(p(M_i|\bar{D})) \, p(M_i|\bar{D})\right] \, p(\bar{y}|\bar{x}, D) \, d\bar{y}, \end{aligned} \tag{3.40}$$

where the current entropy $H(M|D)$ does not have to be evaluated when maximizing the utility over $\bar{x}$. The calculation of the posterior model probability $p(M|\bar{D})$ and the mixed predictive distribution $p(\bar{y}|\bar{x}, D)$ have been described in previous sections.

The **Kullback-Leibler divergence utility** will transform to

$$\begin{aligned} U(\bar{x}) &= \int D_{KL}(p(M|\bar{D})||p(M|D))\, d\bar{y} \\ U(\bar{x}) &= \int \sum_{i=1}^{K} \left[\log(p(M_i|\bar{D}))\, p(M_i|\bar{D})\right]\; p(\bar{y}|\bar{x}, D)\, d\bar{y} \\ &\quad - \int \sum_{i=1}^{K} \left[\log(p(M_i|D))\, p(M_i|\bar{D})\right]\; p(\bar{y}|\bar{x}, D)\, d\bar{y}. \end{aligned} \tag{3.41}$$

In both cases, the integrals over $\bar{y}$ can be approximated by sampling from the posterior mixed predictive distribution $p(\bar{y}|\bar{x}, D)$ again.

3.7 Laplace Approximation

Laplace approximation is one of the simplest methods which can be used to deal with non-linear models.

If the model is non-linear, the posterior distribution $p(\theta|D)$ will not necessarily be Gaussian anymore. In that case, we can use the Laplace approximation to get an approximation of $p(\theta|D)$ in the form of a Gaussian distribution. The methodology is presented below.

Assume a non-Gaussian distribution $p(\theta|D)$ given by

$$\begin{aligned} p(\theta|D) &= \frac{1}{Z} f(\theta) \\ Z &= \int f(\theta) d\theta, \end{aligned} \tag{3.42}$$

where f is an arbitrary function of θ and Z is an unknown normalization constant.

Firstly, we find the mode of $p(\theta|D)$ which we will denote μ_θ. We can do this either by finding a stationary point of $f(\theta)$:

$$\left.\frac{df(\theta)}{d\theta}\right|_{\theta=\mu_\theta} = 0, \tag{3.43}$$

or directly by maximizing the probability $p(\theta|D)$:

$$\mu_\theta = \arg\max_\theta \log f(\theta). \tag{3.44}$$

Now we calculate Σ_θ using the Hessian matrix of $\log f(\theta)$ as

$$\Sigma_\theta = \left(-\nabla\nabla log f(\theta)|_{\theta=\mu_\theta}\right)^{-1}. \tag{3.45}$$

It is important to check that the matrix Σ_θ is positive-definite, which implies that the mode μ_θ is indeed a local maximum.

Using Σ_θ, we can construct a second-order Taylor expansion of $\log f(\theta)$ at μ_θ:

$$\log f(\theta) \approx \log f(\mu_\theta) - \frac{1}{2}(\theta - \mu_\theta)^T (\Sigma_\theta)^{-1} (\theta - \mu_\theta). \tag{3.46}$$

By taking the exponential of the previous equation we get

$$f(\theta) \approx f(\mu_\theta) \exp\left(-\frac{1}{2}(\theta - \mu_\theta)^T (\Sigma_\theta)^{-1} (\theta - \mu_\theta)\right). \tag{3.47}$$

Finally, we can approximate the distribution $p(\theta|D)$ as

$$p(\theta|D) \approx \mathcal{N}(\theta|\mu_\theta, \Sigma_\theta) = \frac{|\Sigma_\theta|^{-1/2}}{(2\pi)^{p/2}} \exp\left(-\frac{1}{2}(\theta - \mu_\theta)^T (\Sigma_\theta)^{-1} (\theta - \mu_\theta)\right) \tag{3.48}$$

by adding a normalization constant to the previous equation.

3.8 Importance Sampling

In this subsection, we will briefly discuss how to approximate integrals over distributions from which we cannot sample directly. In that case, we can use importance sampling together with some approximation method (e.g. the Laplace approximation) and sample from the approximate distribution instead.

Assume we have some distribution $p(\theta)$ and we want to calculate

$$\int g(\theta)\, p(\theta)\, d\theta, \tag{3.49}$$

where $g(\theta)$ is an arbitrary function of θ.

Assume that the integral is analytically intractable and we cannot sample from $p(\theta)$. If some approximation $q(\theta)$ is available from which we can sample, then the integral can be approximated as [27]

$$\begin{aligned}
\int g(\theta)\, p(\theta)\, d\theta &= \int g(\theta)\, \frac{p(\theta)}{q(\theta)}\, q(\theta)\, d\theta \\
\int g(\theta)\, p(\theta)\, d\theta &\approx \mathcal{C} \sum_{i=1}^{S} g(\theta_i)\, \frac{r(\theta_i)}{q(\theta_i)} \\
\mathcal{C} &= 1/\sum_{i=1}^{S} \frac{r(\theta_i)}{q(\theta_i)} \\
\theta_1, ..., \theta_S &\overset{\text{iid}}{\sim} q(\theta),
\end{aligned} \tag{3.50}$$

where $q(\theta) \approx p(\theta)$ and $r(\theta) \propto p(\theta)$.

Chapter 4

Plasma edge modeling

1 Motivation: Tokamaks

1.1 Introduction

The first documented attempt to construct a functioning thermonuclear fusion reactor was made in 1938 by Kantrowitz and Jacobs [32]. The first tokamak was built in 1957 [33]. Yet thermonuclear fusion power plants remain "just a few decades away" for many decades now. The main challenge of fusion reactors is maintaining the plasma stable and at a sufficient temperature while keeping the reactor itself intact. This has proven to be very difficult as the deuterium-tritium fusion, which powers most of the thermonuclear reactors being developed today, only starts to occur at around 100 million degrees kelvin [34] which results in heat fluxes far beyond what any available material can withstand. This is known as the *power exhaust problem*.

Different reactor designs are being developed including tokamaks, spherical tokamaks, and stellarators. They all share the same basic design of plasma held by a magnetic field inside a vacuum chamber. This design minimizes contact of the plasma with the plasma-facing components to reduce the heat flux into them. The main differences between the designs are in the geometry of the magnetic field and the chamber. The topics covered in the remainder of this chapter are discussed from the perspective of conventional tokamaks. The specifics of the other reactor designs will not be discussed as the focus of this thesis is the simulation of plasma in a conventional tokamak reactor. In the remainder of this document, if a *tokamak* is mentioned assume the conventional design.

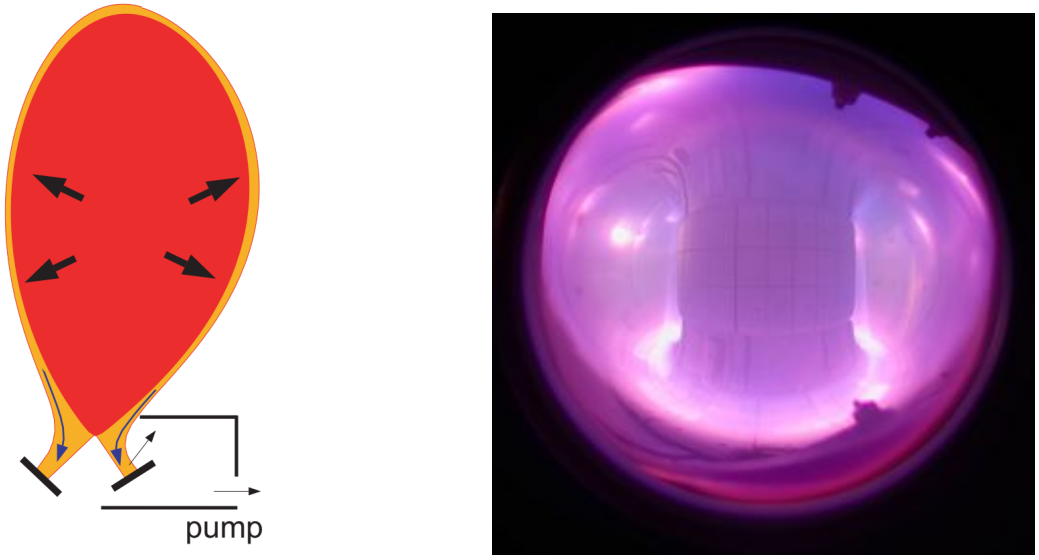
The basic concepts and terms concerning nuclear fusion reactors explained in this chapter have been taken mainly from [34, 35, 36]. If you wish for more detailed information about any topic discussed here, refer to these sources.

1.2 Power Exhaust Problem

In tokamaks, the magnetic field holding the plasma in the vacuum chamber forms a separatrix which divides it into two parts; the closed magnetic field lines holding most of the plasma in the middle of the chamber and the open magnetic field lines on the edges diverting the so-called *scrape-off layer (SOL)* of the plasma into the divertor plates at the bottom of the chamber [34]. You can see a schema of the separatrix in the figure 4.1.

Most of the heat from the plasma is diverted into the divertors at the bottom of the chamber. You can see glowing divertor plates in the figure 4.1. The heat flux to the divertors can reach tens or hundreds of $\mathrm{MW}^2\,\mathrm{m}^{-1}$ [36]. Reducing this value is an

Figure 4.1: **Left:** Cross-section of the tokamak chamber with the divertors depicted as black lines at the bottom [37]. The orange SOL and red inner plasma are divided by the separatrix. **Right:** View into the COMPASS tokamak with the divertor plates glowing [38].



essential step toward thermonuclear fusion power plants. For the reactor to be operational for prolonged periods of time this value has to be reduced to $\sim 5\,\mathrm{MW}^2\,\mathrm{m}^{-1}$ [39]. The amount of the heat flux directed at the divertor plates depends greatly on the properties of the SOL which change with many parameters of the reactor like plasma density, initial energy inputted into the system, geometry of the magnetic field, etc. Correctly selecting these parameters to reduce the maximum power flux to the plasma-facing components is unfortunately exceedingly difficult as the behavior of the plasma is complex and difficult to simulate.

1.3 Plasma Detachment

Three SOL particle transport regimes are distinguished in tokamaks; the sheath-limited regime, the conduction-limited (or high-recycling) regime, and the detached regime [35]. The three regimes are defined by the dominant physical processes involved.

The temperature gradient f_T and the plasma collisionality v^* are defined as [40]

$$\begin{aligned} f_T &= \frac{T_u}{T_t} \\ v^* &= 10^{-16}\frac{n_u\,L}{T_u^2}. \end{aligned} \tag{4.1}$$

These terms are useful for the definition of the transport regimes below.

The *sheath-limited regime* occurs when $f_T < 1.5$ and $v^* < 10$. In this regime, the particle transport in the SOL is affected mostly by the electric field of the sheath, which is a thin region above the divertors [40].

The *conduction-limited regime* occurs when $f_T > 3$ and $v^* > 15$. In this regime, the plasma collsionality is high and so the transport is predominantly affected by the heat conduction [40].

The *detached regime* is defined by the low plasma temperature at the target caused by the detachment of the plasma from the divertor plates [41]. This regime is dominated by the volumetric loss processes and it occurs when the temperature at the target decreases to about $T_{et} < 10$. Furthermore, two detached sub-regimes are sometimes distinguished by the dominant volumetric loss process; the *momentum-loss detachment* which occurs at $1 < T_{et} < 10$ and the *particle-loss detachment* which occurs when $T_{et} < 1$ [35]. (All temperatures are in electronvolts.)

Plasma detachment has to be achieved in order to reduce the power flux into the divertors and solve the power exhaust problem in future tokamaks. However, to control and optimize the plasma detachment a sufficiently good model of the SOL is required [41].

2 SOLPS-ITER

SOLPS-ITER is a complex code designed for high fidelity simulations of the plasma boundary in tokamaks [42]. It can be used to test experiment configurations to find which ones yield promising results - for example, a low peak temperature of the plasma-facing components.

The issue with the SOLPS-ITER is the immense computational demands of the software as well as the complexity of its usage. This makes it unsuitable for optimization of the experiment parameters. However, a computationally cheaper surrogate model approximating the high fidelity simulation could be used to guide the optimization of some experiment parameters. This could make the optimization process more efficient, yielding better results and saving resources and time.

3 Two-Point Model

The equations in this subsection have been taken from [35].

A so-called two-point model (*2PM*) exists which relates plasma density and temperature at the target (the divertor) to the plasma pressure and parallel power flux in the upstream (the outer mid-plane) in an individual flux tube. The most complete version of the 2PM by V. Kotov and D. Reiter [43] is presented below:

$$T_{et} = \frac{8m_f}{e\gamma^2}\frac{q_{||u}^2}{p_u^2}\left[\frac{(1-f_{cooling})^2}{(1-f_{mom-loss})^2}\left(\frac{R_u}{R_t}\right)^2\right]\left[\frac{(1+\tau_t/z_t)}{2}\frac{(1+M_t^2)^2}{4M_t^2}\right] \text{ (eV)}, \tag{4.2}$$

$$n_{et} = \frac{\gamma^2}{32m_f}\frac{p_u^3}{q_{||u}^2}\left[\frac{(1-f_{mom-loss})^3}{(1-f_{cooling})^2}\left(\frac{R_t}{R_u}\right)^2\right]\left[\frac{4}{(1+\tau_t/z_t)^2}\frac{8M_t^2}{(1+M_t^2)^3}\right] \text{ (m}^{-3}\text{)}, \tag{4.3}$$

$$\Gamma_{e||t} = \frac{\gamma}{8m_f}\frac{p_u^2}{q_{||u}}\left[\frac{(1-f_{mom-loss})^2}{(1-f_{cooling})}\left(\frac{R_t}{R_u}\right)\right]\left[\frac{2}{(1+\tau_t/z_t)}\frac{4M_t^2}{(1+M_t^2)^2}\right] \text{ (m}^{-2}\,\text{s}^{-1}\text{)}. \tag{4.4}$$

A brief explanation of the equations follows.

First of all, the subscripts i and e denote whether the variable is associated with *ions* or *electrons* in the plasma. The subscripts u and t denote whether the variable is associated with the *upstream* or the *target*. The upstream variables relate to the outer mid-plane region outside of the separatrix. The target variables are measured just above the outer divertor plate.

The main variables are: temperature T, plasma density n, plasma pressure p, parallel power flux $q_{||}$. In the classic "forward" version of 2PM, p_u and $q_{||u}$ are considered the independent variables and T_{et} and n_{et} are considered the dependent ones.

The constants are fuel ion mass m_f, the elementary charge e, and the sheath heat transmission coefficient γ.

The terms in the two square brackets are later extensions of the original basic model which included only the terms outside of the square brackets. The first extension of the model added terms describing different volumetric losses; the momentum loss $f_{mom-loss}$, the energy loss $f_{cooling}$ and the losses caused by the toroidal flux tube expansion described by the ratio $\left(\frac{R_t}{R_u}\right)$ of the radii of the flux tube at the target and the outer mid-plane. These losses are not negligible and incorporating them into the model is important for it to be realistic. However, no universal models exist for the momentum and energy losses. The second brackets contain additional correction terms. M is the plasma flow Mach number, $\tau \equiv T_i/T_e$ is the ratio of ion and electron temperatures and $z \equiv n_e/n_i$ is the ratio of electron and ion densities.

For simplicity, we will assume $z_t = \tau_t = M_t = 1$ in the rest of the paper as it is an acceptable approximation [35]. With these approximations, we get the following simplified 2PM equations:

$$T_{et} = \frac{8m_f}{e\gamma^2}\frac{q_{||u}^2}{p_u^2}\frac{(1-f_{cooling})^2}{(1-f_{mom-loss})^2}\left(\frac{R_u}{R_t}\right)^2 \text{ (eV)}, \tag{4.5}$$

$$n_{et} = \frac{\gamma^2}{32m_f}\frac{p_u^3}{q_{||u}^2}\frac{(1-f_{mom-loss})^3}{(1-f_{cooling})^2}\left(\frac{R_t}{R_u}\right)^2 \text{ (m}^{-3}\text{)}, \tag{4.6}$$

$$\Gamma_{e||t} = \frac{\gamma}{8m_f}\frac{p_u^2}{q_{||u}}\frac{(1-f_{mom-loss})^2}{(1-f_{cooling})}\left(\frac{R_t}{R_u}\right) \text{ (m}^{-2}\,\text{s}^{-1}\text{)}. \tag{4.7}$$

The 2PM equations have been derived using the following equations:

$$(1-f_{mom-loss})\,p_u = p_t, \tag{4.8}$$

$$(1-f_{cooling})\,q_{||u}R_u = q_{||t}R_t, \tag{4.9}$$

$$p_u = 4\,n_{eu}\,kT_{eu}, \tag{4.10}$$

$$q_{||t} \approx 7.5\,n_t\,k\,T_{et}\,c_{st}, \tag{4.11}$$

$$c_{st} = \sqrt{2\,e\,T_{et}/m_f}, \tag{4.12}$$

where c_{st} is the isothermal plasma sound speed and $k = 1.6 * 10^{-19}$ is a constant to convert the temperature from electronvolts to joules. Note that the equations for p_u and $q_{||t}$ have been simplified by the assumed approximations.

Finally, below are values of the physical constants appearing in the equations:

$$\begin{aligned} m_f &= 2.5 * 1.67 * 10^{-27}\ (kg), \\ e &= 1.60 * 10^{-19}\ (C). \end{aligned} \tag{4.13}$$

The two-point model could be a great tool for better control of the plasma detachment and reduction of the power flux to the divertors. However, development of models for the volumetric losses is necessary for the 2PM to be calibrated accurately.

3.1 Momentum Loss Surrogate Model

It has been observed that the following model relating the momentum loss to the electron temperature at the target can be fit well to experimental data [35, Figure 7]:

$$(1 - f_{mom-loss}) = \frac{n_{et}T_{et}}{n_{eu}T_{eu}} \approx a(1 - exp(b\,T_{et}))^c. \tag{4.14}$$

The model has an arbitrarily chosen form. The symbols a, b, c are model parameters. The model is not general, so the parameters need to be trained for each reactor.

Chapter 5

Experiment Design for Momentum Loss Surrogate Model Training

To calibrate the 2PM it is crucial to also be able to estimate the momentum and energy losses as they are important terms in the 2PM equations. We would like to attempt to train a surrogate model for estimation of the *momentum loss* for the COMPASS tokamak at IPP CAS.

1 Model and Training

The model from the equation 4.14 will be used. The model is defined as

$$\begin{aligned} y &= m_\theta(x) = a(1 - \exp(-x/b))^c \\ \theta &:= (a, b, c)^T \\ x &:= T_{et} \\ y &:= (1 - f_{mom-loss}) = \frac{n_{et}T_{et}}{n_{eu}T_{eu}}. \end{aligned} \tag{5.1}$$

(See the section 3 in the previous chapter for a description of the variables.)

We would like to train this model with data from the SOLPS-ITER simulations of the COMPASS tokamak. This simulation can provide T_{et} and n_{et} given values of T_{eu} and n_{eu} as input. The momentum loss model takes T_{et} as input and therefore the experiment design methods will provide us with values of T_{et} which are in some way optimal for the next data-point to train the model efficiently. However, to query the simulation for new data we have to provide the input variables T_{eu} and n_{eu}. We will estimate the optimal values of T_{eu} and n_{eu} from the T_{et} value provided by the experiment design method using the 2PM with volumetric losses:

$$T_{et} = \frac{8m_f}{e\gamma^2}\frac{q_{||u}^2}{p_u^2}\frac{(1 - f_{cooling})^2}{(1 - f_{mom-loss})^2}\left(\frac{R_u}{R_t}\right)^2, \tag{5.2}$$

where the momentum loss term $(1 - f_{mom-loss})$ will be aproximated using the model we are currently training and the following approximations will be used for the remaining

terms:

$$\begin{aligned} p_u &= 4\, n_{eu}\, k\, T_{eu}, \\ f_{cooling} &= 0, \\ \frac{R_u}{R_t} &= 1. \end{aligned} \tag{5.3}$$

This set of equations has two degrees of freedom; n_{eu} and T_{eu}. We will set n_{eu} to a constant value and adjust the value of T_{et} by changing T_{eu} only. Then we can solve the equations numerically to find the optimal value of T_{eu}, which when given as input to the SOLPS-ITER simulation together with the constant n_{eu} will yield a value of T_{et} close to the one requested by the experiment design method.

By using this approach the model will of course receive slightly different data points from the simulation than the ones the experiment design requested, but this should have a minimal effect on the performance of the method as long as the difference is not too large.

The momentum loss model training procedure is summarized by the algorithm 2. The term $D_N = (x_1, y_1), ..., (x_N, y_N)$ represents the data after the N-th iteration and n_{eu} is set to a constant value.

Algorithm 2 Model Training

procedure TRAIN_MODEL
  while not term_cond() **do**
    update_model!(D_{N-1})
    $\widehat{T_{et}} \leftarrow \arg\max_{x_1,...,x_S} \text{utility}(x)$
    $T_{eu} \leftarrow \text{2PM}(\widehat{T_{et}}, n_{eu})$
    $n_{et}, T_{et} \leftarrow \text{SOLPS-ITER}(n_{eu}, T_{eu})$
    $(x_N, y_N) := (T_{et}, \frac{n_{et} T_{et}}{n_{eu} T_{eu}})$
  end while
end procedure

In the N-th iteration of the training procedure, the following is performed. Firstly, the model is updated according to the current data D_{N-1}. Then the selected experiment design method is used to estimate the optimal value $\widehat{T_{et}}$ of the next data-point, which will improve the model the most. Afterward, the 2PM is used to approximate the optimal value of T_{eu}. Finally, the SOLPS-ITER simulation is queried with the input (n_{eu}, T_{eu}) and the data set is augmented by the new data point.

2 Synthetic Experiment

As a proof of concept, we tested our method with synthetically generated data instead of using the SOLPS-ITER simulation for now. We replaced the SOLPS-ITER simulation in the training procedure with an approximate calculation using the 2PM with volumetric losses with the same approximations as the ones used in the training except this time the term $(1 - f_{mom-loss})$ was computed with predefined parameters $\theta_t := (a_t, b_t, c_t)^T$, which we attempted to learn with the trained model.

This substitute simulation of course provides somewhat unrealistic data in comparison to real data from SOLPS-ITER, but it is sufficient for the proof of concept as long as the

following property of the generated data is not violated:

$$\frac{n_{et}T_{et}}{n_{eu}T_{eu}} = (1 - f_{mom-loss}) \approx m_{\theta_t}(T_{et}). \tag{5.4}$$

It is apparent that in case this property would not hold for the generated data, it would be impossible to train the momentum loss surrogate model.

If one simply used the 2PM equations for n_{et} and T_{et} to calculate the output of the substitute simulation and substituted the momentum loss model with the predefined *true* parameters for the term $(1 - f_{mom-loss})$, the equation above would not hold for the generated data. This is caused by the inaccuracy of the 2PM model with the assumed approximations. For this reason, the calculation of the synthetic data was slightly modified as described below.

The substitue simulation receives values (n_{eu}, T_{eu}) as input. In the first step, the following set of equations from the 2PM is solved numerically to get the value of T_{et}:

$$\begin{aligned} T_{et} &= \frac{8m_f}{e\gamma^2}\frac{q_{||u}^2}{p_u^2}\frac{(1 - f_{cooling})^2}{(1 - f_{mom-loss})^2}\left(\frac{R_u}{R_t}\right)^2, \\ p_u &= 4\, n_{eu}\, k\, T_{eu}, \\ (1 - f_{mom-loss}) &= m_{\theta_t}(T_{et}), \end{aligned} \tag{5.5}$$

where $f_{cooling} = 0$, $\frac{R_u}{R_t} = 1$ and θ_t is the true value of the model parameters which we are trying to learn. The parallel power flux $q_{||u}$ was set to a constant value.

In the second step, the value of n_{et} is calculated as

$$n_{et} = m_{\theta_t}(T_{et})\frac{n_{eu}T_{eu}}{T_{et}}. \tag{5.6}$$

By calculating the output of the substitute simulation this way, we sacrifice the accuracy of n_{et} to make sure that 5.4 holds. We do not need the data to be realistic as long as the momentum loss model can be trained from them so this is a reasonable trade-off.

The whole synthetic data generation procedure is shown in the algorithm 3. This procedure replaces the SOLPS-ITER simulation in the model training (algorithm 2). The computation using the 2PM is performed as has been described above. After the new data-point is computed by solving the 2PM equations a predefined data noise σ_d^2 is applied to the result.

Algorithm 3 Synthetic Data Generation

procedure GENERATE_DATAPOINT(n_{eu}, T_{eu})
  $n_{et}, T_{et} \leftarrow$ 2PM(n_{eu}, T_{eu})
  $x \leftarrow T_{et}$
  $y \sim \mathcal{N}(\frac{n_{et}T_{et}}{n_{eu},T_{eu}}, \sigma_d^2)$
  return (x, y)
end procedure

3 Bayesian Model

Let us now define the surrogate model in Bayesian terms. We will assume the following **prior distributions** for the model parameters and data:

$$\begin{aligned} p(\theta_i) &= \text{LogNormal}(\mu_\theta, \sigma_\theta^2) \propto \theta_i^{-1} \exp\left(-\frac{1}{2}\sigma_\theta^{-2}(\log(\theta_i) - \mu_\theta)^2\right) \\ p(y_i|x_i, \theta) &= \mathcal{N}(m_\theta(x_i), \sigma_y^2) \propto \exp\left(-\frac{1}{2}\sigma_y^{-2}(y_i - m_\theta(x_i))^2\right), \end{aligned} \tag{5.7}$$

where $\mu_\theta, \sigma_\theta, \sigma_y$ are hyperparameters of the model.

The **posterior parameter distribution** can be expressed in the following form by using the Bayes' theorem (assuming the data are independent):

$$\begin{aligned} p(\theta|D) &\propto p(D|\theta)\, p(\theta) \\ p(\theta|D) &\propto \prod_{i=1}^{N} \left[p(y_i|x_i, \theta)\right] \prod_{i=1}^{3} \left[p(\theta_i)\right] \\ p(\theta|D) &\propto \prod_{i=1}^{N} \left[\exp\left(-\frac{1}{2}\sigma_\theta^{-2}(\log(\theta_i) - \mu_\theta)^2\right)\right] \prod_{i=1}^{3} \left[\theta_i^{-1} \exp\left(-\frac{1}{2}\sigma_\theta^{-2}(\log(\theta_i) - \mu_\theta)^2\right)\right]. \end{aligned} \tag{5.8}$$

We will need to integrate over this distribution, which would be analytically intractable. For this reason, we will use the **Laplace approximation** and later integrate over it using importance sampling. The Laplace approximation is performed as follows. First we express the logarithm of the function f_D proportionate to the posterior parameter distribution $p(\theta|D)$:

$$\begin{aligned} p(\theta|D) &\propto f_D(\theta) \\ f_D(\theta) &= \prod_{i=1}^{N} \left[\exp\left(-\frac{1}{2}\sigma_y^{-2}(y_i - \hat{y}_i)^2\right)\right] \prod_{i=1}^{3} \left[\theta_i^{-1} \exp\left(-\frac{1}{2}\sigma_\theta^{-2}(\log(\theta_i) - \mu_\theta)^2\right)\right] \\ \log f_D(\theta) &= -\frac{1}{2}\sigma_y^{-2} \sum_{i=1}^{N} \left[(y_i - \hat{y}_i)^2\right] - \sum_{i=1}^{3} \left[\theta_i\right] - \frac{1}{2}\sigma_\theta^{-2} \sum_{i=1}^{3} \left[(\log(\theta_i) - \mu_\theta)^2\right], \end{aligned} \tag{5.9}$$

where $|D| = N$. Then we compute the Laplace approximation $\mathcal{N}(\mu_\theta, \Sigma_\theta)$ of the parameter posterior as

$$\begin{aligned} p(\theta|D) &\approx \mathcal{N}(\mu_\theta, \Sigma_\theta) \\ \mu_\theta &= \arg\max_\theta \log f_D(\theta) \\ \Sigma_\theta &= \left(-\nabla\nabla \log f_D(\theta)|_{\theta=\mu_\theta}\right)^{-1}. \end{aligned} \tag{5.10}$$

See the *Bayesian Ridge Regression* section for more information on Laplace approximation.

4 Exeperiment Design

We will test two experiment design methods with the model - the information gain utility and the Kullback-Leibler divergence utility - and make a comparison of the two. The analytical

computation of these utilities is intractable as the parameter posterior is too complex. We will use an approximate computation using importance sampling to approximate the utilities.

The utility of a single point $\bar{x}$ is computed as

$$U(\bar{x}) = \frac{1}{\sum_{i=1}^{T} \frac{r(\theta_i|D)}{q(\theta_i|D)}} \sum_{i=1}^{T} u\left(q(\theta|D), q(\theta|\bar{D}_i)\right) \frac{r(\theta_i|D)}{q(\theta_i|D)}, \tag{5.11}$$

where

$$\begin{aligned} r(\theta|D) &\propto p(D|\theta)\, p(\theta) \propto p(\theta|D), \\ q(\theta|D) &= \text{Laplace\_approx}(p(\theta|D)) \approx p(\theta|D), \\ \bar{D}_i &= D \cup \{(\bar{x}, \bar{y}_i)\}, \\ \bar{y}_i &\sim \mathcal{N}(m_{\theta_i}(\bar{x}), \sigma_y^2), \\ \theta_1, ..., \theta_T &\overset{\text{iid}}{\sim} q(\theta|D), \end{aligned} \tag{5.12}$$

where T samples are drawn from $q(\theta|D)$ and for each sample θ_i a single sample y_i is drawn from $\mathcal{N}(m_{\theta_i}(\bar{x}), \sigma_y^2)$. See section the *Bayesian Ridge Regression* section for more information on importance sampling.

The function $u(s,t)$ takes two probability distributions and returns a utility value from $\mathbb{R}$. This function is what differs between the information gain and Kullback-Leibler divergence utilities. The corresponding function are described below:

$$u_{IG}\left(q(\theta|D), q(\theta|\bar{D}_i)\right) = -H\left(q(\theta|\bar{D}_i)\right), \tag{5.13}$$

$$u_{KL}\left(q(\theta|D), q(\theta|\bar{D}_i)\right) = D_{KL}\left(q(\theta|\bar{D}_i) \,||\, q(\theta|D)\right). \tag{5.14}$$

Note that we skipped the term $H\left(q(\theta|D)\right)$ in the information gain utility as it does not depend on $\bar{x}$ and so it does not have to be evaluated when maximizing the utility over $\bar{x}$.

The computation of the utilities for a set of S distinct input values $\bar{x}_1, ..., \bar{x}_S$ is presented in the algorithm 4. The variables $q\theta$, $\overline{q\theta}_j$, $r\theta$ represent the distributions $q(\theta|D)$, $q(\theta|\bar{D}_j)$, $r(\theta|D)$ respectively and one of the functions from equations 5.13 or 5.14 is to be used in place of the *util* function in the algorithm. The notation $r\theta(\theta_i)$ denotes the evaluation of the probability density function of the distribution $r\theta$ at θ_i.

The computation starts by computing the Laplace approximation $q\theta$ of the posterior parameter distribution $p(\theta|D)$ given the current data D and then drawing T samples of θ (the model parameters) from $q\theta$.

Then the increment of utilities of all points $\bar{x}_j$ is calculated for each sampled θ_i successively. The utility increment for a sample θ_i is calculated as follows. Firstly, a single sample of $\bar{y}_j$ is drawn for each point $\bar{x}_j$ from the predictive distribution $p(m_{\theta_i}(\bar{x}_j), \sigma_y^2)$ where the mean is the prediction of the model with parameter values θ_i. Secondly, a Laplace approximation $\overline{q\theta}_j$ of the posterior parameter distribution $p(\theta|\bar{D}_j)$ given the data augmented by the potential new data-point $\bar{D}_j = D \cup \{(\bar{x}_j, \bar{y}_j)\}$ is calculated for each point $\bar{x}_j$. Finally, the utility of each point $\bar{x}_j$ is incremented by the value given by the selected utility function (which takes the distributions $q\theta$ and $\overline{q\theta}_j$ as input) weighted by the importance weight w. (See the section *Bayesian Ridge Regression* for more information on the importance weights.)

Algorithm 4 Utility Calculation

function CALCULATE_UTILITY($\bar{x}_1, ..., \bar{x}_S,\ D$)
  $q\theta \leftarrow$ Laplace_approx(D)
  thetas $\leftarrow \theta_1, ..., \theta_T \overset{\text{iid}}{\sim} q\theta$
  $U_1, ..., U_S \leftarrow 0, ..., 0$
  $weight\_sum \leftarrow 0$

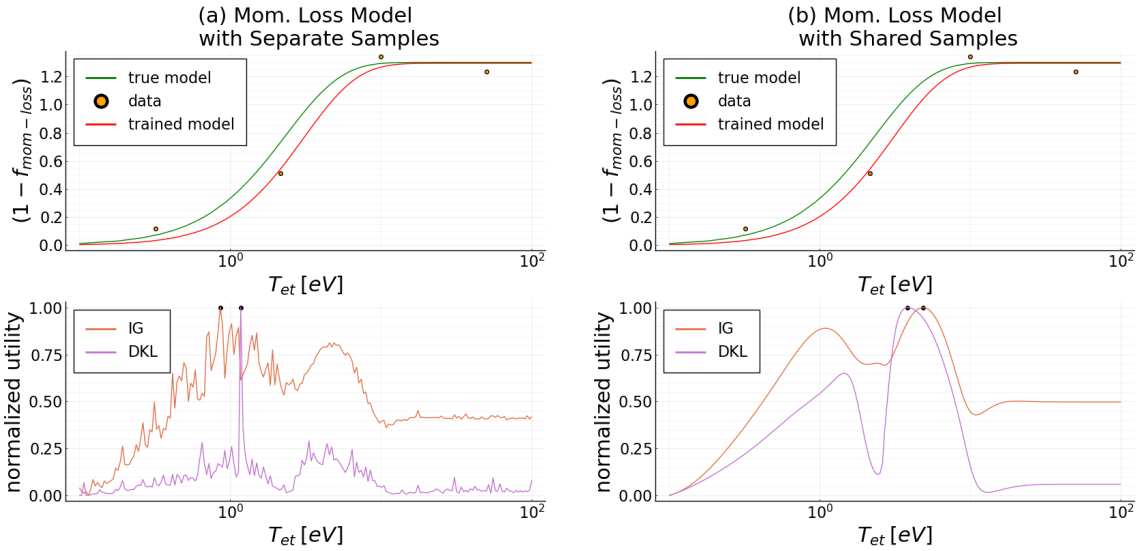
  for each θ_i in $\theta_1, ..., \theta_T$ **do**
    $y_{dev} \leftarrow$ sample($\mathcal{N}(0, \sigma_y^2)$)
    $\bar{y}_j \leftarrow m_{\theta_i}(\bar{x}_j) + y_{dev}$ **for each** j **in** $1, ..., S$
    $\overline{q\theta}_j \leftarrow$ Laplace_approx($D \cup \{(\bar{x}_j, \bar{y}_j)\}$) **for each** j **in** $1, ..., S$
    $w \leftarrow \frac{r\theta(\theta_i)}{q\theta(\theta_i)}$
    U_j += $w *$ util($q\theta, \overline{q\theta}_j$) **for each** j **in** $1, ..., S$
    $weight\_sum$ += w
  end for
  return U / $weight\_sum$
end function

To partially mitigate the noise of the utility function caused by the random sampling the samples $\bar{y}_j$ for a given θ_i share their deviation from the mean. That is implemented by first sampling their shared deviation y_{dev} from a normal zero-mean distribution with variance σ_y^2 and than calculating the values $\bar{y}_j$ by adding this shared deviation to the corresponding means $m_{\theta_i}(\bar{x}_j)$. This trick results in smooth utility functions even with low parameter sample sizes. This does not mean that the utility functions are completely relieved of the noise caused by sampling. The utility functions will still differ when calculated multiple times with different samples. However, the trick eliminates large value spikes in the utility functions making the methods more stable. The effect of the shared sampling trick is illustrated in the figure 5.1.

After the utility increments are calculated for all samples θ_i, the utility values of all points $\bar{x}_j$ are divided by the sum of the importance weights and returned. Then the point x_j with the highest utility is selected as the next evaluation point. Note that the division by the weight sum has no effect on the maximization of the utility over $\bar{x}$.

Occasionally, the calculation of some Laplace approximation $\overline{q\theta_j}$ of some distribution $p(\theta|\bar{D})$ may fail during the utility computation because the covariance matrix Σ_θ is not positive definite. This usually happens when the data set $\bar{D}$ has been augmented by some extreme outlier. Then the optimal fit to the data can degenerate to some extreme shape (like an almost constant model) and the posterior parameter distribution also degenerates and cannot be reasonably approximated by a Gaussian distribution. In this case, the sample θ_i which caused the problem is skipped for all points $x_1, ..., x_S$. (Even though likely only a few of the approximations $\overline{q\theta_1}, ..., \overline{q\theta_S}$ could not be computed for the sample θ_i.) Skipping the θ_i sample for all points keeps the method more stable.

Figure 5.1: Shared Sampling Trick



(a) The IG and DKL utilities calculated with $\bar{y}_j$ sampled separately for each point $\bar{x}_j$.
(b) The IG and DKL utilities calculated with shared deviation sample y_{dev}.
The utilities on both plots have been calculated with 200 model parameter samples.

5 Method Performance Analysis

In this section, the performance of the two selected experiment design methods - information gain maximization and Kullback-Leibler divergence maximization - is compared with data from the synthetic experiments. Additionally, the effect of the parameter sample size T on the performance of both methods is tested.

5.1 Method Performance Metric

The main metric used for the comparison of the methods was the number of iterations it took the method to train the model to a predefined precision. The number of iterations also corresponds to the number of evaluations of the simulation. Thus it is a good metric of the method performance as the goal of the experiment design methods is to minimize the number of simulations needed to train the model.

The model precision has been measured using the *root mean square error (RMS)* of the model on a pre-generated test data set. This data set has been generated by selecting R data points from the model domain spaced evenly according to a logarithmic scale. The reason behind using a logarithmic scale and thus giving more importance to the region near zero is that a large portion of the domain on the side far from zero is almost constant. This constant value is defined by a single model parameter a, whereas the more complex shape of the model near zero is defined by the remaining two parameters b and c. If a linear scale was used instead of a logarithmic one, only learning the parameter a would result in a very low model error. Another motivation behind using a logarithmic scale is that the same absolute model error results in a substantially larger relative error in the region close to zero so it is reasonable to give that area higher importance.

5.2 Experiment Settings

The values of all hyperparameters and constants used during the synthetic experiments are presented below.

The following values have been used for the **hyperparameters of the 2PM**:

$$\begin{aligned} \gamma &:= 8, \\ q_{||u} &:= 10\,\mathrm{MW}. \end{aligned} \tag{5.15}$$

The same values have been used in the 2PM during the synthetic data generation (algorithm 3) and when estimating the values n_{eu}, T_{eu} by the experiment design method (algorithm 2). The exact values of these constants are not very important. Changing them to different values could result in unrealistic data, but it would have minimal effect on the results of the synthetic experiments.

The following values have been used for the **hyperparameters of the model**:

$$\begin{aligned} mode_\theta &:= (1., 1.5, 1.5)^T, \\ var_\theta &:= (0.2, 0.5, 0.5)^T, \\ \sigma_y^2 &:= 0.01. \end{aligned} \tag{5.16}$$

The hyperparameter σ_y is the expected variance of the data. The standard parameters μ_θ and σ_θ^2 of the log-normal prior parameter distribution $p(\theta) \sim LogNormal(\mu_\theta, \sigma_\theta^2)$ have been calculated from the defined values of the mode and the variance of the distribution.

The **hyperparameters of the experiment design methods** have been set as follows. The evaluation point sample size S has been set to 200 in all experiments as this results in a sufficiently fine resolution of the utility function. The samples have been spread out evenly according to a logarithmic scale to allow for a better resolution for smaller values of T_{et}. The methods have been tested with parameter sample sizes T equal to 2000, 200, and 20 as presented in the results.

Finally, the **parameters of the synthetic experiments** were set as follows. The test data set size has been set to 10 thousand as the model error is still computed easily with this data set size and a larger size does not bring any significant improvement to the error estimation. The data noise variance σ_d^2 has been set to 0.01. The target model RMS error ϵ_t was set to values slightly higher than the data noise, this included values 0.05 and 0.035 as presented in the results.

The experiment domain on which the model was trained was defined as $T_{et} \in [10^{-1}, 10^2]$. Different initial data sets D_0 have been used in each experiment. The data set was defined by a list of points $X_0 = [x_1, ..., x_I]$ where $|D_0| = I$. The intial data set was generated as

$$\begin{aligned} y_1, ...y_I &= m_{\theta_t}(x_1), ..., m_{\theta_t}(x_I) \\ \tilde{x_i} &\sim \mathcal{N}(x_i, \sigma_d^2)\ \forall i \in \{1, ..., I\} \\ \tilde{y_i} &\sim \mathcal{N}(y_i, \sigma_d^2)\ \forall i \in \{1, ..., I\} \\ D_0 &= \{(\tilde{x_1}, \tilde{y_1}), ..., (\tilde{x_I}, \tilde{y_I})\}. \end{aligned} \tag{5.17}$$

The used set X_0 of initial points is presented in the results with each experiment.

In all experiments, there was a maximum iteration limit set to 50. This was implemented to avoid wasting computational resources on rare experiment runs where the data were very skewed at the beginning and a large amount of data would be necessary to correct for it.

5.3 The Experiments

In all experiments, a simple experiment design labeled *random* was tested alongside the IG and DKL methods as a control method. This method selected the next evaluation point in each iteration by sampling from a log-uniform distribution over the experiment domain $[10^{-1}, 10^2]$. This way the method performs significantly better than if it was completely randomized but the IG and DKL methods should still outperform it.

Experiment 1

In this experiment, the initial data set was defined by $X_0 = [1.]$ and the target model error was set to $\epsilon_t = 0.05$. Both methods were tested with sample sizes $T \in \{20, 200, 2000\}$. The experiment was repeated 20 times with each method variant.

Experiment 2

This experiment was very similar to the first one the only difference being the initial data set which was defined by $X_0 = [10.]$ this time to test if the methods' performance changes with different initial data-point. The experiment was again repeated 20 times with each method variant.

Experiment 3

In this experiment, the target model error was decreased to $\epsilon_t = 0.035$. The initial data set was larger this time and was spread over most of the experiment domain. It was defined by $X_0 = [0.4, 2., 10., 50.]$. The methods were tested with sample sizes $T \in \{20, 200\}$. The number of runs with each method variant was increased to 40.

Experiment 4

The initial data in this experiment consisted of 4 data points condensed very close together. The initial data set was defined by $X_0 = [4., 4., 4., 4.]$. The model error was again set to $\epsilon_t = 0.035$ and the used parameter sample sizes and the number of runs remained the same as in experiment 3.

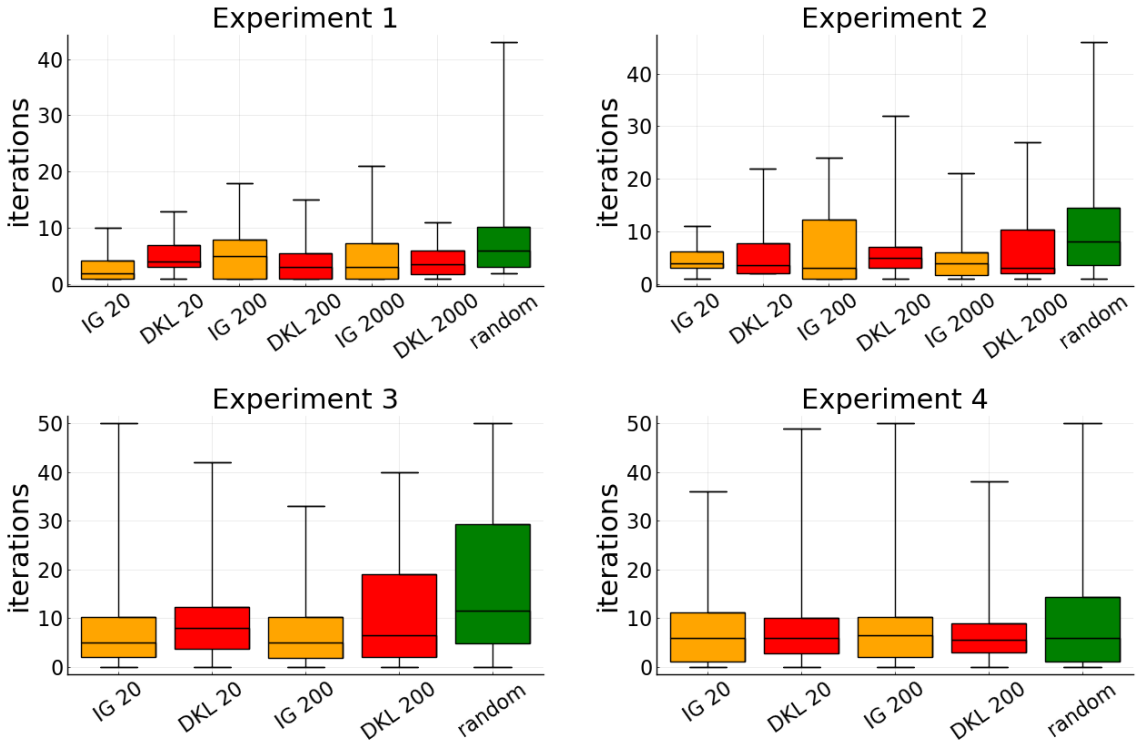
5.4 Results and Discussion

The results of the experiments are presented in the figure 5.2. The boxplots show the median, lower and upper quartiles and the minimum and maximum of iterations it took the respective method to reach the target model error. The labels *IG*, *DKL* and *random* indicate the used experiment design method - information gain, Kullback-Leibler divergence, or the control random method. The numbers $20, 200, 2000$ in the labels denote the parameter sample size T used for the utility calculation. The median and lower and upper quartile values from the boxplots are presented in the table under the plots as well.

Parameter Sample Size

From the results of experiments 1 and 2, it is apparent that the slightly increased precision of the methods caused by increasing the sample size by an order of magnitude seems to have no significant effect on the performance of the methods. The median values change very little with different sample sizes and a decreasing median value with an increasing

Figure 5.2: Experiment Results



Median and Lower and Upper Quartiles of Iterations							
Experiment	IG 20	DKL 20	IG 200	DKL 200	IG 2000	DKL 2000	random
1	$2_1^{4.25}$	4_3^7	5_1^8	$3_1^{5.5}$	$3_1^{7.25}$	$3.5_{1.75}^6$	$6_3^{10.25}$
2	$4_3^{6.25}$	$3.5_2^{7.75}$	$3_1^{12.25}$	5_3^7	$4_{1.75}^6$	$3_2^{10.25}$	$8_{3.5}^{14.5}$
3	$5_2^{10.25}$	$8_{3.75}^{12.25}$	$5_{1.75}^{10.25}$	6.5_2^{19}	/	/	$11.5_{4.75}^{29.25}$
4	$6_1^{11.25}$	$6_{2.75}^{10}$	$6.5_2^{10.25}$	5.5_3^9	/	/	$6_1^{14.25}$

The values in the table are formatted as med_{low}^{upp} where *med* is the median and *low*, *upp* are the lower and upper quartiles of the number of iterations the method required to reach the target model error.

sample size is *not* generally observed. For this reason, it seems that the approximation of the integral over the posterior parameter distribution calculated during the computation of the utility is not improved substantially with larger sample sizes.

The values of the first and third quartiles fluctuate significantly with different sample sizes but again seemingly randomly. I ascribe this fluctuation mainly to the random data noise and the "luck" of different experiment runs for data with low or high errors. If the method receives multiple data points in the same region of the model domain with similar deviation to one side, it can take a large amount of additional data to correct for it. Similarly, if the first few received data points are all with extremely low noise, the model can converge extremely fast. The number of such runs can affect the values of the quartiles significantly. The values of the minimum and maximum are affected even more by this effect as a single such run can change them drastically. For this reason, the values of the minimum and maximum can be somewhat unreliable.

Reliability

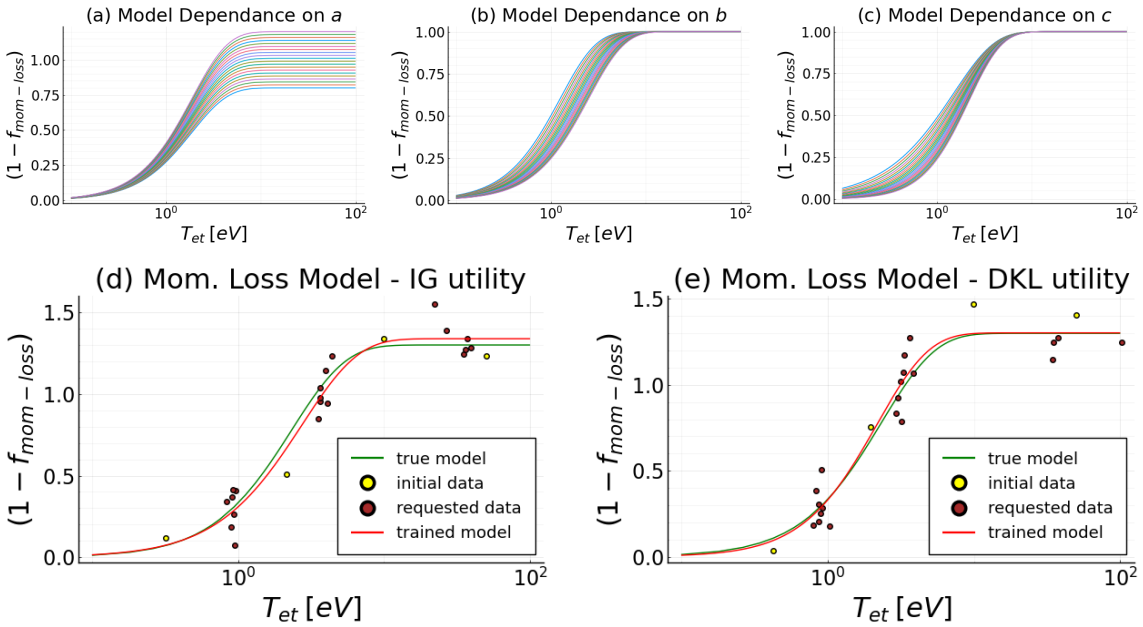
Different choices of the initial data set seem to have no major effect on the performance of either of the Bayesian methods. Experiments 1 and 2 differed only in the initial data set. Equally, experiments 3 and 4 also differed only in the initial data set. The results of the Bayesian methods do not show any significant change in the number of iterations required to train the model which could be interpreted as dependent on the initial data. On the other hand, the log-uniform random sampling method performed significantly worse in experiment 3 than in experiment 4. This is caused by the random method not being able to take advantage of the condensed initial data in experiment 3 and continuing to sample the whole domain, whereas the Bayesian methods will avoid the already well-explored area and shift the focus of subsequent experiments elsewhere.

Method Comparison

Both IG and DKL methods performed perceptibly better than the random method. The median, as well as the upper quartile of iterations of the random method, were consistently higher than the corresponding values of the IG and DKL methods. The performance of the random method was however not as inferior as one would expect. In experiment 4, the median value of the random method even equalized with the medians of the Bayesian methods. I ascribe this to the random method being better than expected rather than the IG and DKL methods performing poorly. The random method was not truly random as it implemented some expert knowledge by sampling from the log-uniform distribution over the model domain to get more data in the important regions near zero, which I believe helped substantially in the training of this particular model. Also, if the model selection was required for the problem the Bayesian methods would be more advantageous. Nevertheless, the IG and DKL methods performed well in comparison to the random one.

Neither of the two Bayesian methods can be considered better suited for the training of the momentum loss model based on the results. The median values of the number of iterations required by the methods to train the model were similar in all experiments and neither method performed better overall. The upper quartiles of both methods reached a slightly higher in some experiments, but that is ascribed to the random data noise as these deviations were inconsistent among all experiments.

Figure 5.3: Example Experiment Runs and Model-Parameter Dependance



(a,b,c) The mom. loss model plotted with parameter values:
(a) $a \in (0.8, 1.2), b = 1.5, c = 1.5$; **(b)** $a = 1, b \in (1, 2), c = 1.5$; **(c)** $a = 1, b = 1.5, c \in (1, 2)$.
(d) The mom. loss model trained with data requested by the IG method.
(e) The mom. loss model trained with data requested by the DKL method.

Method Properties

Both methods opted to deposit data in three specific regions of the model domain and only rarely requested data outside of these areas. Two example runs - one with each of the methods - are shown in the plots (d,e) in the figure 5.3. The yellow data points are the ones from the initial data set while the brown ones are the ones the methods requested successively in each iteration to improve the model. It is apparent that the requested data-points form three distinct groups; the first group located in the flat region of the model curve around $T_{et} \in (10\,eV, 100\,eV)$, the second group roughly around $T_{et} = 4\,eV$ and the third group roughly around $T_{et} = 1\,eV$. These three regions of the model domain correspond roughly to the regions where the model curve is affected the most by the three model parameters a, b, c respectively as shown in the plots (a,b,c) in the figure 5.3.

Conclusion and Limitations

A conclusion of the synthetic experiments and a discussion of their limitations follow.

The information gain maximization and Kullback-Leibler divergence maximization experiment design methods have been applied and compared in a synthetic experiment simulating the problem of training the momentum loss model for plasma in the SOL in tokamaks. The results show that neither of the methods is significantly better suited for the problem. Both of the methods performed well and it was shown that in comparison to a naive method (the log-uniform random sampling) they required fewer data to train the model thus potentially saving time and resources.

The proof of concept in the form of the synthetic experiments does have some limitations. The most significant one is that once either of the methods is used in conjunction with the SOLPS-ITER simulations, the received data from the simulations will have different T_{et} values (the input of the momentum loss model) than the ones which the experiment design method requested. The first reason for this data deviation is that the momentum loss model that is being trained is simultaneously being used in the 2PM equations when estimating the (n_{eu}, T_{eu}) inputs of the SOLPS-ITER simulation. The model is however not fully trained at that moment and thus inherently unprecise. The second reason is that the other terms and constants in the 2PM equations (e.g. $(1 - f_{cooling})$, $q_{||u}$, γ) also have to be approximated or estimated in some way and these approximations will cause further data deviation depending on how precise they are. The first reason for the data deviation is contained in the synthetic experiment the same way it will be with real data. However, the second reason is not as in the synthetic experiment the 2PM equations with the same approximations were used both for the experiment design and in the substitute simulation replacing the SOLPS-ITER simulation and thus the approximations were precise by default.

If the approximations used for the 2PM equation terms will be too inaccurate when applying the method to real data, it could perform worse than in the syntetic experiments as in the most extreme case the method is practically reduced to random data sampling. However, as long as the approximations are decent, the method will perform well as has been shown.

Chapter 6

Summary

In this thesis, the basic concepts of surrogate modeling and its usages have been introduced. Some of the most commonly used surrogate models have been presented and a selection of experiment design methods used for surrogate model training have been discussed with a focus on probabilistic Bayesian methods. A brief introduction to the problems of today's tokamaks was presented including a definition of the *power exhaust problem* and the concept of *plasma detachment*. Toady's surrogate models and simulations of the plasma SOL in tokamaks have been introduced. In the last chapters, two selected experiment design methods were adapted for training the plasma momentum loss model, a synthetic experiment simulating the data from SOLPS-ITER was proposed as a proof of concept, and both methods were applied in the experiment. It has been shown that both methods are suitable for the training of the plasma momentum loss model and could save resources in comparison to a more naive approach. Neither of the two methods could be conclusively deemed as superior to the other.

References

[1] Alonso Marco et al. "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1557–1563. DOI: 10.1109/ICRA.2017.7989186.

[2] Yichi Zhang, Daniel W Apley, and Wei Chen. "Bayesian optimization for materials design with mixed quantitative and qualitative variables". In: *Scientific reports* 10.1 (2020), pp. 1–13.

[3] Shenghong Ju et al. "Designing Nanostructures for Phonon Transport via Bayesian Optimization". In: *Phys. Rev. X* 7 (2 2017), p. 021024. DOI: 10.1103/PhysRevX.7.021024. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.021024.

[4] Joseph Duris et al. "Bayesian optimization of a free-electron laser". In: *Physical review letters* 124.12 (2020), p. 124801.

[5] Raul Yondo, Esther Andrés, and Eusebio Valero. "A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses". In: *Progress in aerospace sciences* 96 (2018), pp. 23–61.

[6] Subham Sahoo, Christoph Lampert, and Georg Martius. "Learning equations for extrapolation and control". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4442–4450.

[7] Kevin Swersky, Jasper Snoek, and Ryan P Adams. "Multi-task bayesian optimization". In: *Advances in neural information processing systems* 26 (2013).

[8] André I Khuri and Siuli Mukhopadhyay. "Response surface methodology". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.2 (2010), pp. 128–149.

[9] Kaveh Amouzgar and Niclas Strömberg. "Radial basis functions as surrogate models with a priori bias in comparison with a posteriori bias". In: *Structural and Multidisciplinary Optimization* 55.4 (2017), pp. 1453–1469.

[10] Alvin Lal and Bithin Datta. "Development and implementation of support vector machine regression surrogate models for predicting groundwater pumping-induced saltwater intrusion into coastal aquifers". In: *Water Resources Management* 32.7 (2018), pp. 2405–2419.

[11] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[13] Carl Edward Rasmussen. “Gaussian processes in machine learning”. In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[14] Nando de Freitas. *Machine learning - Introduction to Gaussian processes*. YouTube. 2013. URL: https://www.youtube.com/watch?v=4vGiHC35j9s&ab_channel=NandodeFreitas.

[15] Nando de Freitas. *Machine learning - Gaussian processes*. YouTube. 2013. URL: https://www.youtube.com/watch?v=MfHKW5z-OOA&ab_channel=NandodeFreitas.

[16] 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. DOI: 10.1109/JPROC.2015.2494218.

[17] David Duvenaud. “Automatic model construction with Gaussian processes”. PhD thesis. University of Cambridge, 2014.

[18] Manuel Blum and Martin A Riedmiller. “Optimization of Gaussian process hyperparameters using Rprop.” In: *ESANN*. Citeseer. 2013, pp. 339–344.

[19] Santu Rana et al. “High dimensional Bayesian optimization with elastic Gaussian process”. In: *International conference on machine learning*. PMLR. 2017, pp. 2883–2891.

[20] Zi Wang et al. “Batched large-scale Bayesian optimization in high-dimensional spaces”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 745–754.

[21] David Eriksson et al. “Scaling Gaussian process regression with derivatives”. In: *arXiv preprint arXiv:1810.12283* (2018).

[22] Georg Martius and Christoph H Lampert. “Extrapolation and learning equations”. In: *arXiv preprint arXiv:1610.02995* (2016).

[23] Slawomir Koziel, Qingsha S Cheng, and John W Bandler. “Space mapping”. In: *IEEE Microwave Magazine* 9.6 (2008), pp. 105–122.

[24] Jonas Močkus. “On Bayesian methods for seeking the extremum”. In: *Optimization techniques IFIP technical conference*. Springer. 1975, pp. 400–404.

[25] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*. Vol. 37. Springer Science & Business Media, 2012.

[26] Peter I Frazier. “A tutorial on Bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).

[27] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[28] Roland Lvovich Dobrushin and Vyacheslav Valer’evich Prelov. *Differential entropy*. 2020. URL: https://encyclopediaofmath.org/index.php?title=Differential_entropy.

[29] Joram Soch. *Differential entropy of the Multivariate Normal Distribution*. 2020. URL: https://statproofbook.github.io/P/mvn-dent.

[30] Michiel Hazewinkel. *Kullback-Leibler information*. 2020. URL: https://encyclopediaofmath.org/wiki/Kullback-Leibler_information.

[31] Joram Soch. *Kullback-Leibler divergence for the Multivariate Normal Distribution*. 2020. URL: https://statproofbook.github.io/P/mvn-kl.

[32] National Academy of Engineering. “Arthur R. Kantrowitz”. In: *Memorial tributes volume 16*. National Academies Press, 2012.

[33] Robert Arnoux. 2008. URL: https://www.iter.org/newsline/55/1194.

[34] R Bilato and R Kleiber. “IPP Summer University for Plasma Physics, September 17-21, 2012, Garching”. In: *IPP Summer University for Plasma Physics and Fusion Research*. Max-Planck-Institut für Plasmaphysik. 2012.

[35] PC Stangeby. “Basic physical processes and reduced models for plasma detachment”. In: *Plasma Physics and Controlled Fusion* 60.4 (2018), p. 044022.

[36] A Loarte and R Neu. “Power exhaust in tokamaks and scenario integration issues”. In: *Fusion Engineering and Design* 122 (2017), pp. 256–273.

[37] R Schneider et al. “Plasma edge physics with B2-eirene”. In: *Contributions to Plasma Physics* 46.1-2 (2006), pp. 3–191.

[38] IPP. *Amazing plasma pictures from COMPASS*. URL: http://www.ipp.cas.cz/sd/novinky/hlavni-stranka/160719_foto_compass.html.

[39] RA Pitts et al. “Physics basis and design of the ITER plasma-facing components”. In: *Journal of Nuclear Materials* 415.1 (2011), S957–S964.

[40] Kateřina Jiráková. “Study of edge plasma of tokamak COMPASS and its poloidal variations”. MA thesis. Czech Technical University in Prague, 2018.

[41] AW Leonard. “Plasma detachment in divertor tokamaks”. In: *Plasma Physics and Controlled Fusion* 60.4 (2018), p. 044001.

[42] Xavier Bonnin et al. “Presentation of the new SOLPS-ITER code package for tokamak plasma edge modelling”. In: *Plasma and Fusion Research* 11 (2016), pp. 1403102–1403102.

[43] V Kotov and D Reiter. “Two-point analysis of the numerical modelling of detached divertor plasmas”. In: *Plasma physics and controlled fusion* 51.11 (2009), p. 115002.