國立臺灣大學工學院工業工程學研究所

碩士論文

Institute of Industrial Engineering

College of Engineering

National Taiwan University

Master Thesis

以高斯過程及啟發演算法的離線最佳化模擬系統參數
—海底渦輪機應用

Simulation parameters optimization based on Gaussian process and metaheuristics - An application of marine turbine

秦柔

Jou Chin

指導教授：洪一薰博士

Advisor: I-Hsuan Hong, Ph.D.

中華民國 108 年 7 月

July 2019

誌謝

在工工所的這兩年裡, 非常感謝 409 的各位、Annie、楊 G、剛 G 的好夥伴們, 還有一些曾與我一起修課寫作業的同學, 沒有你們就不會有今天即將要畢業的我, 真的很感謝你們的照顧。

謝謝一薰老師這兩年的教導和願意收我進洪 G, 讓我能與這群人很好的男生們走完這兩年既辛苦又美好的時光。這期間我們一起經歷過柔性期末報告寫不完看日出的故事, 一起壓線趕各種大大小小的作業, 和一起吃遍 118 巷的小吃還吃倒了不少家店。很感謝治華的熱心幫助、子堯的搞笑緩解嚴肅氣氛、翌軒的數學教學、有為的溫馨提醒、沛恩不時提出的理論讓我腦力激盪、Annie 的陪伴和梓育的程式教學。因為有你們才有這多采多姿的兩年生活!

也非常感謝爸爸媽媽, 您們每天等我回家吃飯、洗衣服, 還要聽我分享在學校或是實習公司遇到的大事小事或是不順利的事, 您們的陪伴和鼓勵是支持我一路走下去的動力。最後, 感謝在德國一直默默支持我的你, 不論是學業上的輔助或是未來人生的規劃, 謝謝你願意放下自己的成敗優先考慮我的未來。

終於要暫別學生生活邁向另一個挑戰, 也希望大家能在下一個人生階段裡平安順利!

摘要

本文建立離線最佳化的架構, 並以此架構來尋找電腦模擬系統中的四個參數最佳組合, 以取代透過不斷測試來調整電腦模擬系統參數的方法。在離線最佳化的架構中, 包含後設模型的預測和啟發式演算法的參數求解。本文主要是以高斯過程當作後設模型, 此模型能有效預測電腦模擬系統的表現, 並以啟發式演算法中的基因演算法和粒子團演算法, 來尋找在電腦模擬系統中參數的最佳解。最後, 本研究訓練出高斯迴歸模型與電腦模擬系統的平均絕對百分比誤差在百分之七以內, 且基因演算法的參數最佳解帶入電腦模擬系統調整得出的結果, 與物理實驗的平均絕對百分比誤差低於百分之十五。

關鍵字：離線最佳化、電腦模擬系統、後設模型、基因演算法、粒子團演算法

Abstract

This study proposes the framework of offline optimization by using surrogate model and metaheuristics to approximate the simulator parameters of a marine turbine. Surrogate models are built to combine information obtained from numerical simulations. The integration of Gaussian surrogate model reduces the number of large-scale numerical simulations needed to find reliable estimates of the system parameters. Through the use of metaheuristics of the Genetic Algorithm and Particle Swarm Optimization, an efficient exploration of the parameter space is performed. The objective is to determine the system parameters to improve the accuracy of numerical simulation. For this purpose, the results of these parameters in surrogate model and metaheuristics are evaluated through a numerical simulation. We compare our resutls with the physical experiment to make sure that the modified simulator can reduce the errors between the simulator and physical experiments. Finally, the mean absolute percentage error between the result of numerical simulation and physical experiment in our case is within 15%.

Keywords: Offline optimization, Numerical simulation, Surrogate model, Genetic Algorithm, Particle Swarm Optimization

Contents

List of Figures

List of Tables

Chapter 1

Introduction

According to the REN21's[1] report, renewables energy contributed 19.3% to humans' global energy consumption and 24.5% to their generation of electricity in 2015 and 2016, respectively. Among sources of renewable energy, wind power and solar power are known to people. But solar power suffers from rain, clouds, and fog. Wind turbines are impacted by calm weather, yet tidal cycles are as reliable as the rising of the moon. Elghali et al. (2007) stated that tidal current flow is very predictable, to within 98% accuracy for decades. Also, tidal current is independent of prevailing weather conditions like rain, fog, wind, and clouds that can impact other renewable generation forecasts. However, tidal energy has traditionally suffered from relatively high cost. The operation of marine turbine will also affect algae growth patterns and concentrations of nutrients, so there is a limited physical data of marine James et al. (2017). To help balance performance of physical experiment and environmental protection, there is a need to develop numerical simulation.

Numerical simulations provide a wealth of physical insight into the system behavior and fill the gap between theory and experiment. Heermann (1990) stated methods of numerical simulation, and he declared that some quantities or behaviors may be impossible or difficult to measure in an experiment. With numerical simulations such quantities can be computed. However, from the design perspective, they provide only pointwise information like information for single combinations of the design parameters. On the other hand, performing numerical simulations for a large set of design alternatives would require

[1] Global Status Report-REN21

considerable resources. The challenge is to gain an understanding of the system behavior through a small number of numerical simulations. The paradigm shift that has been proposed to meet this challenge is to treat numerical simulations as computer experiments and build a surrogate model which is similar to a response surface or interpolation model of the numerical simulation.

Moser et al. (1999) also used numerical simulation to develop turbulent channel flow at three Reynolds numbers. He emphasized on using simulation to find the result of physical experiment, and he clearly described the physical phenomenon. However, we put more efforts on integrating surrogate model and heuristic algorithm to modify simulation's parameters in our paper.

The method of building surrogate model and heuristic algorithm is so called black box optimization which can be viewed in terms of its inputs and outputs, without any knowledge of its internal workings. We can use online optimization or offline optimization to solve this problem. The comparison of these method can also be viewed in Barton and Meckesheimer (2006). Because online optimization need to run simulation from time to time, it is a lack of efficiency. Therefore, We apply offline optimization by building the surrogate from historical data and predict the result at a fast pace.

Surrogate models can typically be used to substitute simulator, which may require computational effort, so it is common to use them to solve the engineering problems. Anand et al. (2011) published paper about developing surrogate model to mimic the characteristics of the target fuel and to predict the burning result of target fuel in advanced combustion engines. Braconnier et al. (2011) provided a dynamic building of surrogate models for aerodynamic flight data generation. The efficiency of this method is assessed on an analytic test case and a classical aerodynamic transonic flow around a 2D profile. Yamazaki et al. (2010) applied the gradient/Hessian-enhanced surrogate models are shown to develop aerodynamic database construction. Kang et al. (2016) provided a system reliability analysis of soil slopes by using ν-support vector machine, particle swarm optimization, and artificial bee colony algorithms. Apart from these papers that we mentioned above, we have tried all the possible regression model. Finally, we choose the best surrogate model which is Gaussian process regression (GPR) Rasmussen (2003) by doing some data preprocessing to have more precisely result.

The initial objective of this paper is to find the parameter values that minimize the error which is measured between the physical experiment and the result of numerical simulation. The surrogate model is built from a few historical data of numerical simulation with different parameter combinations. Then, we use the surrogate model and metaheuristics to find the minimizing parameters which are inputted into the numerical simulation and turn out the result of physical experiment's prediction. The purpose is to illustrate the possibility of finding the best parameter via the surrogate model and metaheuristics.

The problem investigated in this paper is highly complex and nonlinear. Traditional deterministic methods such as linear programming and non-linear programming might lead to local optima and thus become unsuitable to solve such complex problems. A review of the literature reveals of metaheuristics such as rey Horn et al. (1994) who used the Genetic Algorithm (GA) in place of a linear combination for solving multi-objective optimization problems. Van Laarhoven et al. (1992) applied simulated annealing (SA) for job shop scheduling. The result has been examined on the standard IEEE 30-bus test system with different objectives and generator cost curves. Wang and Landau (2001) presented random walk for calculating the density of states and a new Monte Carlo algorithm that produces results of high accuracy with reduced simulation effort. Tseng and Yun (2009) minimized the sum of a smooth function and a separable convex function by gradient decent method. Coello et al. (2004) applied Particle Swarm Optimization (PSO) for handling multiple objectives. Unlike other current proposals to extend PSO to solve multi-objective optimization problems. Their algorithm uses a secondary repository of particles that is later used by other particles to guide their own flight. Above all, they all used different metaheuristics to solve problems, compared with our paper. We have successfully applied GA and PSO on marine turbine problems, and the result can make simulation be improved.

The Genetic Algorithms (GAs) are proposed based on Darwin's principle of survival of the fittest by Holland (1992) to solve larger scale combination optimization problem. It can jump out local search space to achieve optimal solutions in global space. In Genetic Algorithms, it process a population of individuals which represent search space solutions, each individual is candidate solution and population including all individuals are examined simultaneously, and quality of population are improved gradually, at last the best solution or secondary solutions are achieved by repeating employing three GA operations: selection, crossover and mutation. GA are theoretically and empirically proven to provide robust search capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective search.

PSO was originally developed by Kennedy (2010), and was inspired by the social behavior of the flock of birds. In the PSO algorithm, the birds in a flock are symbolically represented as particles. These particles are considered to be “flying” through the problem space searching for optimal solution. A particle’s location in the multidimensional problem space represents one solution for the problem. When a particle moves to a new location, a different solution to the problem is generated.

In this paper, our main goal is to find the best parameters of the simulator in order to predict the result of the physical experiment precisely. Instead of doing the costly physical experiment from time to time, we use simulator to estimate the turbulence intensity and velocity along the centerline of the wake each at two influent. Both of the intensity and velocity are all under ten different distance Figure 1.1. In the next chapter, we will introduce the framework of offline optimization, mathematical symbols of data, and data preprocessing which is a challenge stage to data analysis in the reality. As a result, the way to deal with data preprocessing would be introduced in detail. In the third chapter, since running the simulator by trial and error is time-consuming, we use the Gaussian process regression as surrogate model to predict the value of intensity and velocity by learning the historical result in simulator. Finally, we apply GA and PSO to estimate the best parameters of the simulator which can make all intensity and velocity in the smallest gap compare with the physical experiment. The fouth chapter, we will display the result of two set of best parameters via GA and PSO, respectively, in comparison with the physical experiment. The main contributions of this paper is that we develop a set of data preprocessing to optimize Gaussian process. Also, we have improved the result of simulation by modifying the simulator's parameters.

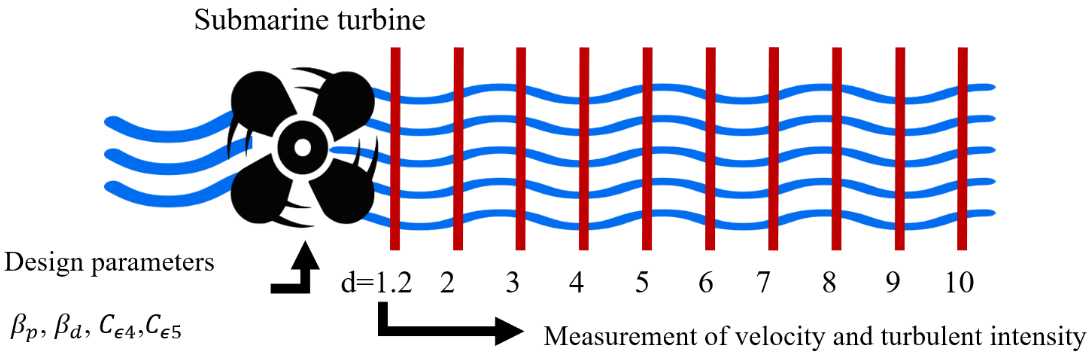

Submarine turbine
Design parameters
β_p, β_d, $C_{\epsilon 4}$,$C_{\epsilon 5}$
d=1.2
2
3
4
5
6
7
8
9
10
Measurement of velocity and turbulent intensity

Figure 1.1: Marine Turbine's process

Chapter 2

Methodology

This chapter contains four parts, including simulation, data preprocessing, surrogate model, and heuristic algorithm in Figure 2.1. We will introduce all of the following sections in details.

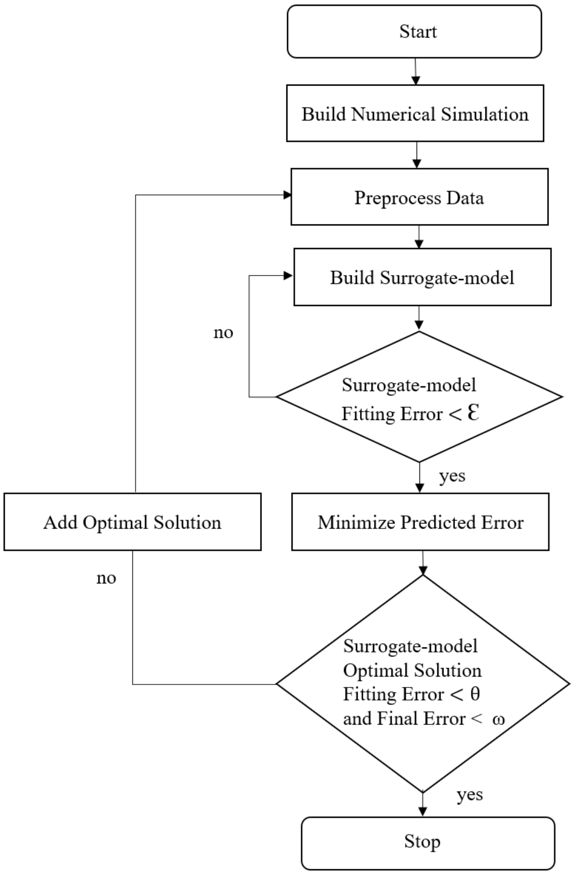

Start
Build Numerical Simulation
Preprocess Data
Build Surrogate-model
no
Surrogate-model Fitting Error < ε
yes
Add Optimal Solution
Minimize Predicted Error
no
Surrogate-model Optimal Solution Fitting Error < θ and Final Error < ω
yes
Stop

Figure 2.1: Flowchart for the surrogate-model framework. (Adapted from Sterling et al., 2019)

2.1 Simulation

In our case, the simulator are used to predict the turbulence intensity (I) and velocity (U) each at two influent 3% and 15% which are written in $I_{3\%}$, $I_{15\%}$,$U_{3\%}$, and $U_{15\%}$ respectively. These quantities of interest (QoI) are all under ten different distance. Furthermore, we need to modify the simulation's parameters, so it is also important to define the design parameters of our simulator.

β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ are the four parameters to be determined in the simulator. These parameters can come to the result of $y^S_{i,j,d}$. Then, $y^T_{i,j,d}$ is the physical experiment estimated by turbine marine. On the other hand, the surrogate model prediction noted as $\hat{y}_{i,j,d}$. All the variables are written in Table 2.1.

Table 2.1: The definition of variables

Variable	Definition
$y^T_{i,j,d}$ i = 3%, 15% $j = U, I$ d = 1.2, 2, 3, 4, 5, 6, 7, 8, 9, 10	$y^T_{i,j,d}$ is the state variables j of physical experiment under the initial condition i at spatial locations d.
$y^S_{i,j,d}$ i = 3%, 15% $j = U, I$ d = 1.2, 2, 3, 4, 5, 6, 7, 8, 9, 10	$y^S_{i,j,d}$ is the state variables j of simulator prediction under the initial condition i at spatial locations d.
$\hat{y}_{i,j,d}$ i = 3%, 15% $j = U, I$ d = 1.2, 2, 3, 4, 5, 6, 7, 8, 9, 10	$\hat{y}_{i,j,d}$ is the state variables j of surrogate model's prediction under the initial condition i at spatial locations d.
β_p	The first parameter of the simulator.
β_d	The second parameter of the simulator.
$C_{\epsilon 4}$	The third parameter of the simulator.
$C_{\epsilon 5}$	The fourth parameter of the simulator.

2.2 Data Preprocessing

We will focus on the theory of clustering, outlier detection, and bootstrap that we have applied in the data preprocessing in this section.

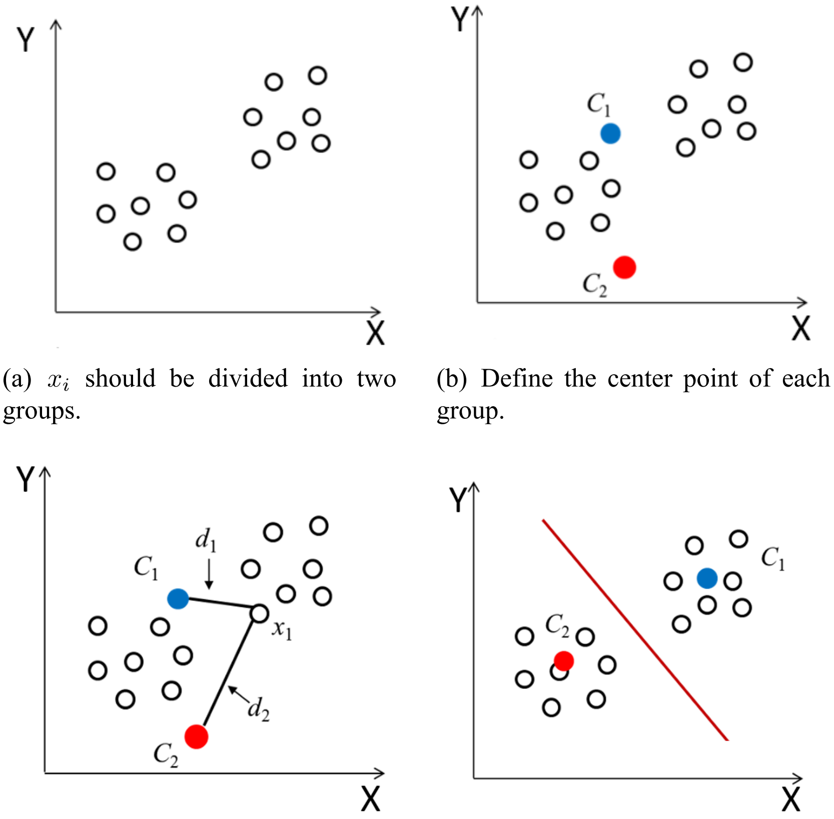
2.2.1 Clustering

In the clustering, suppose that there is a set of d-dimensional data

$$x_i \in R^d, i = 1, 2, ..., n \tag{2.1}$$

Let K be the number of cluster which noted as S_1, S_2,...,S_k. The purpose of K-means clustering is to minimize the Euclidean distance of the number of data (n) and the center point in the group x_c. Then, x_i will belong to the group which is the smallest Euclidean distance between it.The Euclidean equation is written in

$$arg\ min \sum_{c=1}^{K} \sum_{i=1}^{n} \|x_i - x_c\|^2 \tag{2.2}$$



(a) x_i should be divided into two groups.

(b) Define the center point of each group.

(c) Cause d_1<d_2, x_1 belongs to C_1.

(d) Repeating (a) to (c) can finish grouping.

Figure 2.2: The concept of K-means clustering, adapted from http://medium.com/

2.2.2 Univariate Boxplot

A univariate boxplot Frigge et al. (1989) is specified by five parameters: the two extremes, the upper UQ (75th percentile), lower LQ (25th percentile) quartiles, and the median (50th percentile). The lower and upper extremes of a boxplot are defined as

$$x_L = max\left\{x_{(1)}, LQ - \frac{3}{2}IQR\right\}, x_U = min\left\{x_{(n)}, UQ + \frac{3}{2}IQR\right\}. \qquad (2.3)$$

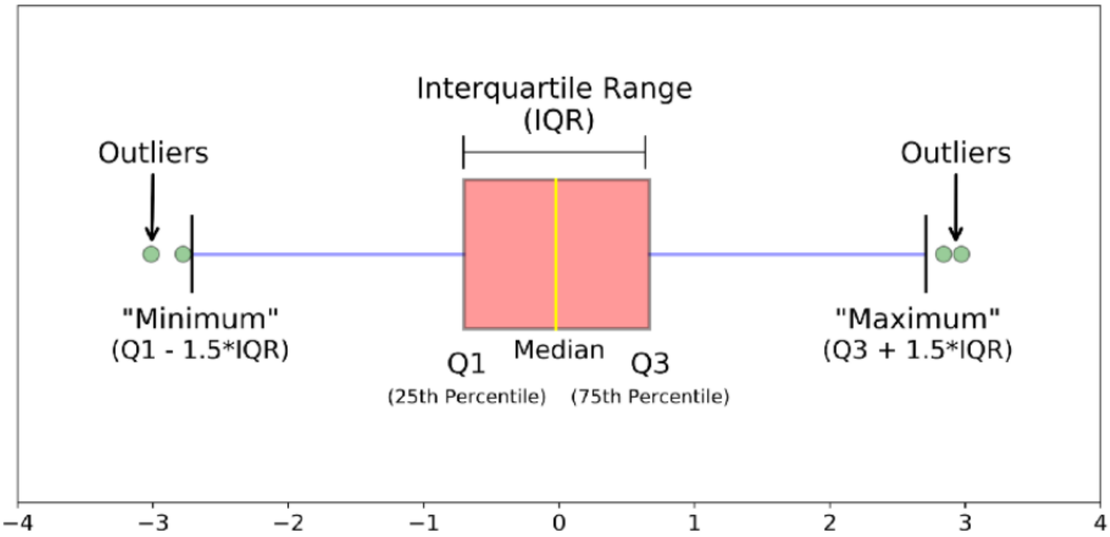


Figure 2.3: Different parts of a boxplot, adapted from http://towardsdatascience.com

In Figure 2.3 shows some outliers in the boxplot. We will remove the data that lays out of these two extremes.

2.2.3 Bootstrap

The basic idea of bootstrap or bootstrapping is that inference about a population from sample data can be modelled by resampling the sample data and performing inference about a sample from resampled data. In short, bootstrap resamples the data with replacement as illustrated in Figure 2.4. In bootstrap method, we keep resample the data in order to find the distribution of data. As we already know the data distribution, we only resample for the sufficient samples that we need in order to have enough datasets to train the model.

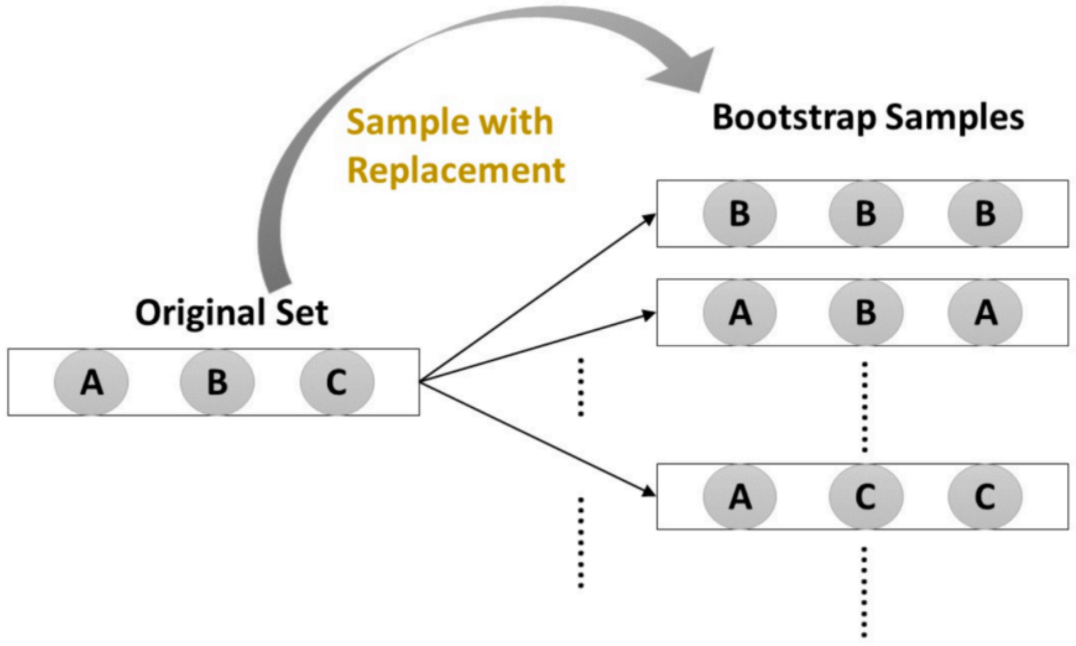


Figure 2.4: Bootstrap, adapted from Chen (2019)

2.3 Surrogate Model

In this section, we will introduce the selection of surrogate model, five-fold cross validation, and the Gaussian Process.

2.3.1 Selection of Surrogate Model

The first time we get the data. It is difficult to know which is the best model to predict all these data. Luckily, scikit-learn, which is a library containing different kinds of package in the machine learning field, provides us with a cheat-sheet that illustrates four aspect of machine learning methodology such as regression, dimensionality reduction, clustering, and classification. In our case, regression is suitable for training the surrogate model. Now we have a set of regression model to be selected. It is necessary to define a quantitative measure, evaluating the error between the surrogate model predictions and numerical simulation. The mean absolute percentage error MAPE is chosen as a metric to compare surrogate model predictions with numerical simulation. In our paper, surrogate model fitting error $\mathrm{MAPE}_{srgt/sim}$ is defined as

$$MAPE_{srgt/sim} = \frac{1}{N_{datasets}} \frac{1}{N_{points}} \sum_{m=1}^{|N_{datasets}|} \sum_{i} \sum_{d} \sum_{j} \left(\frac{\left| \hat{y}_{i,j,d} - y^{S}_{i,j,d} \right|}{y^{S}_{i,j,d}} \right) \tag{2.4}$$

By choosing the smallest surrogate model fitting error MAPE$_{srgt/sim}$, we can make sure that this surrogate model can predict simulation's result well.

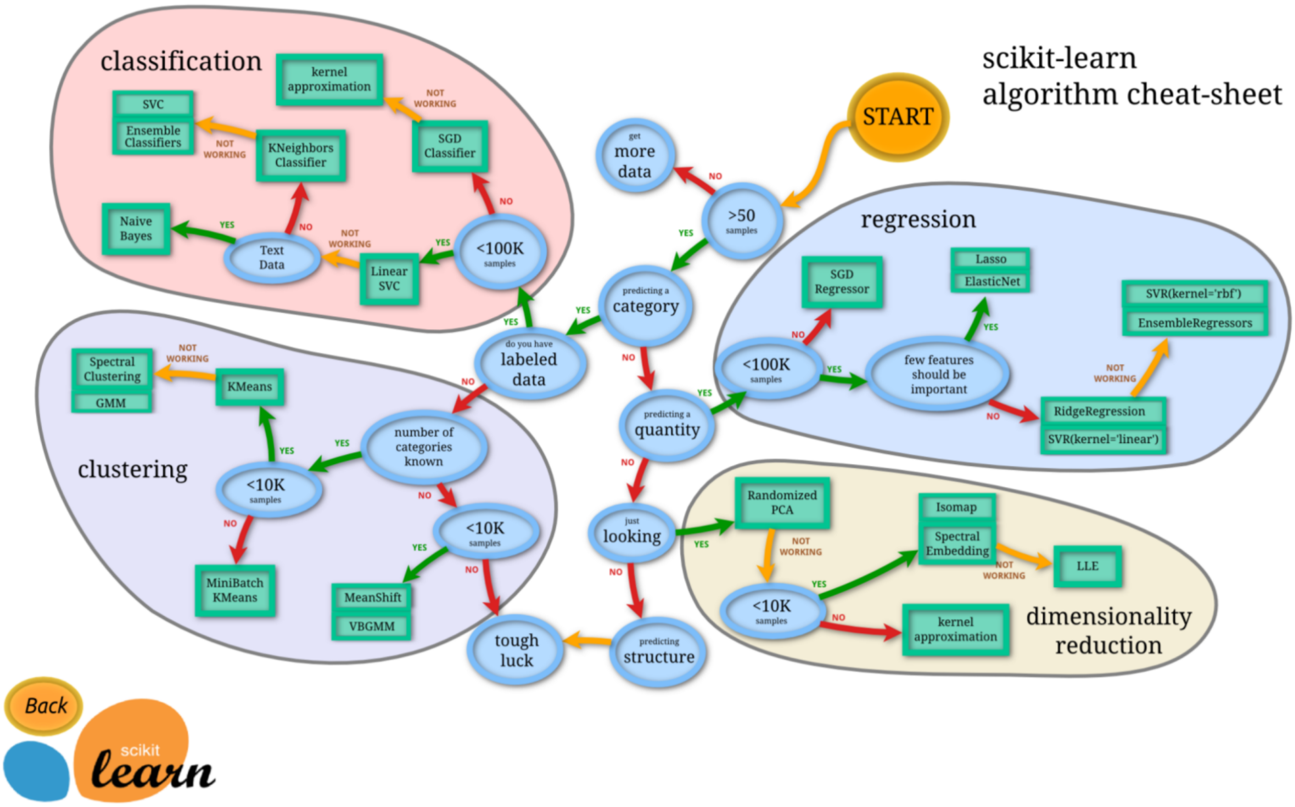


Figure 2.5: Mind map of machine learning, adapted from http://scikit-learn.org

2.3.2 K-fold cross-validation

In order to avoid over-fitting, we often apply K-fold cross-validation. K is the number of group that we split. In Figure 2.6 is the example of five-fold cross-validation—that is, we split the training data into five sets, and we use each in turn as validation set to evaluate the model fit on the other four-fifths of the training set. Repeating the validation across different subsets gives us a better idea of the performance of the algorithm in advance.

After doing five-fold cross-validation in training set, the surrogate model is built, and they should be tested. Therefore, testing set plays an important role to test whether the surrogate model is precise or not.

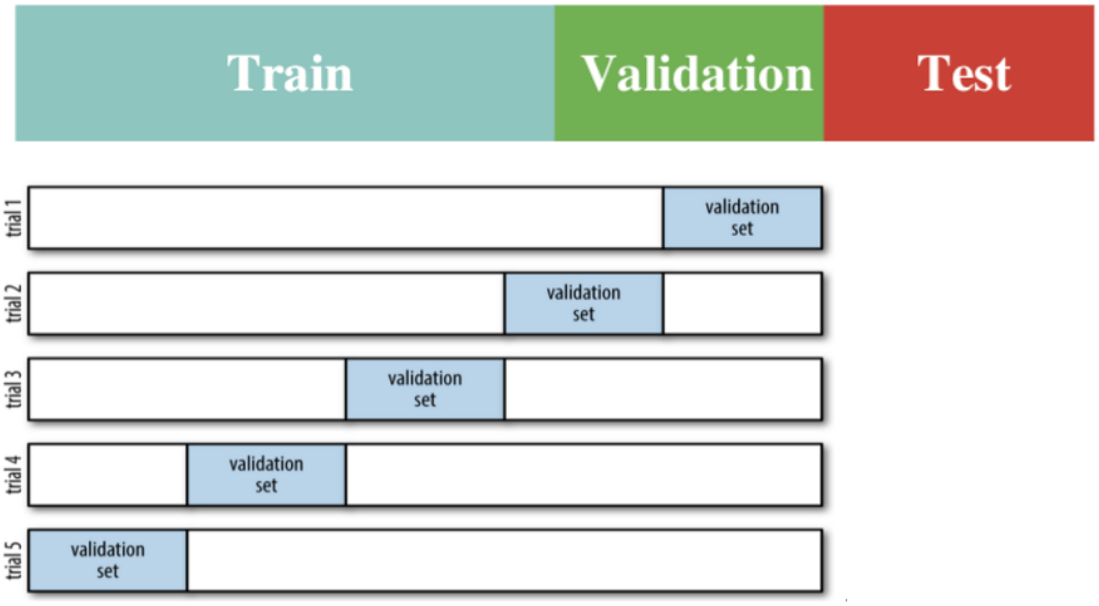


Figure 2.6: Five-fold cross-validation, adapted from http://sebastianraschka.com

2.3.3 Gaussian Process

A Gaussian process is a collection of random variables, any finite set of which have a joint Gaussian distribution. A Gaussian process is completely specified by its mean function $m(x)$ and the covariance function $k(x, x')$:

$$f(x) \sim GP(m(x), k(x, x')). \tag{2.5}$$

There is a training set D of n observations, $D = \{(x_i, y_i) \mid i = 1, \ldots n\}$ where x denotes an input vector like d, β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ in our case, y denotes a scalar output or target such as $\hat{y}_{i,j,d}$ in this paper. The column vector inputs for all n cases are aggregated in the D×n design matrix X and the targets are collected in the vector y.

The goal of Bayesian forecasting is to compute the distribution $p(y_*|x_*, D)$ of output y_* given a test input x_* and a set of training points D. Using Bayesian rule, the posterior distribution for the Gaussian process outputs y_* can be obtained. By conditioning on the observed targets in the training set, the predictive distribution is Gaussian:

$$y_*|x_*, X, y \sim N(\hat{y}(x_*), \hat{\sigma}(x_*)). \tag{2.6}$$

where, the mean and variance are given by

$$\hat{y}(x_*) = k_*^T(K + \sigma_n^2 I)^{-1} y. \tag{2.7}$$

$$\hat{\sigma}(x_*) = k(x_*, x_*) - k_*^T(K + \sigma_n^2 I)^{-1} k_*^T. \tag{2.8}$$

where, a compact form of the notation setting for matrix of the covariance functions are: $k_* = K(X, x_*)$, $K = K(X, X)$, σ_n^2 is the unknown variance of the Gaussian noise. Gaussian process procedure can handle interesting models by simply using a covariance function with an exponential term:

$$k_y(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_p - x_q)^2) + \sigma_n^2 \delta_{pq}. \tag{2.9}$$

In addition to an exponential covariance function, there are five others in Table 2.2

Table 2.2: The definition of covariance functions

Covariance function	Equation
ExpSineSquared	$exp(-2(\frac{sin(\frac{\pi}{p}d(x_i,x_j))}{l})^2))$
DotProduct	$\sigma_0^2 + x_i \cdot x_j$
Radial-basis function(RBF)	$exp(-\frac{1}{2}d(\frac{x_i}{l}, \frac{x_j}{l})^2)$
Rational quadratic(RQ)	$(1 + \frac{d(x_i,x_j)^2}{2\alpha l^2})^{-\alpha}$
Matern	$\frac{1}{2^{v-1}\Gamma(v)} \left(\frac{\sqrt{2v}}{l} r\right)^v K_v \left(\frac{\sqrt{2v}}{l} r\right)$

Equation 2.9 expresses the idea the cases with nearby inputs will have highly correlated outputs. The covariance is denoted k_y as it is for the noisy targets y rather than for the underlying function f GP employs a set of hyperparameters θ including the length-scale l, the signal variance $\sigma_f^2$and the noise variance σ_n^2, θ can be optimized based on log-likelihood framework:

$$\log p(y|X,\theta) = -\frac{1}{2}y^T(K+\sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log 2\pi. \tag{2.10}$$

Hyperparameters θ are initialized to random values (in a reasonable range) and then use an iterative method, for example conjugate gradient, to search for the optimal values.

2.4 Metaheuristics

In this section, we will provide two metaheuristics to minimize our predicted error.After building the suitable surrogate models, we should use metaheuristics to find the best parameters which can get lowest predicted error. In this paper, predicted error MAPE$_{srgt/exp}$ describes the difference between surrogate model prediction and physical experiment. The predicted error MAPE$_{srgt/exp}$ is written in

$$MAPE_{srgt/exp} = \frac{1}{N_{points}}\sum_i\sum_d\sum_j\left(\frac{\left|\hat{y}_{i,j,d} - y^T_{i,j,d}\right|}{y^T_{i,j,d}}\right) \tag{2.11}$$

The parameters predicted by metaheuristics will be set up in the simulator. After doing simulation, we will get N_{points} of simulation result which will be verified with the surrogate model predictions. This optimal solution fitting error is also written in MAPE$_{srgt/sim}$. It is restricted to below θ. If not, these N_{points} points should be added as optimal solution points to the samples, and we should go to step two again in Figure 2.1. The optimal solution fitting error MAPE$_{srgt/sim}$ is formulated as

$$MAPE_{srgt/sim} = \frac{1}{N_{points}}\sum_i\sum_d\sum_j\left(\frac{\left|\hat{y}_{i,j,d} - y^S_{i,j,d}\right|}{y^S_{i,j,d}}\right) \tag{2.12}$$

The difference between N_{points} of simulation result and the physical experiment is so called Final error MAPE$_{sim/exp}$. In our case, final error must be within ω because the smallest MAPE$_{sim/exp}$ of historical simulation data is ω. We should find other parameters which are better than the historical ones; otherwise, the optimal solution points that we estimated will be added to the samples, and the surrogate model should be rebuilt. The final error MAPE$_{sim/exp}$ is the form of

$$MAPE_{sim/exp} = \frac{1}{N_{points}} \sum_i \sum_d \sum_j \left(\frac{\left| y^S_{i,j,d} - y^T_{i,j,d} \right|}{y^T_{i,j,d}} \right) \tag{2.13}$$

2.4.1 Genetic Algorithm

Genetic Algorithm (GA) is a metaheuristic search algorithm based on the evolutionary. The main idea is derived from the behavior of reproduction animal which consists of selection, crossover and mutation. GA represent an intelligent exploitation of a random search. It has been applied to solve optimization problem for many years. The original Genetic Algorithm consists of the following component as Figure 2.8. First, we should generate the initial population. If the chromosome has N_{var} variables (an N_{var}-dimensional optimization problem) given by P1, P2,..., P(t) , then the chromosome is written as an N_{var} element row vector.

$$chromosome = [P_1, P_2, P_3, ..., P_{N_{var}}] \tag{2.14}$$

In our case, we generate some sets of initial populations of chromosome = [P_{β_p},P_{β_d}, $P_{C_{\epsilon 4}}$, $P_{C_{\epsilon 5}}$] randomly. Then, we define the fitness function of the populations as

$$\min MAPE_{srgt/exp} = \frac{1}{N_{points}} \sum_i \sum_d \sum_j \left(\frac{\left| G(\beta_p, \beta_d, C_{\epsilon 4}, C_{\epsilon 5}) - y^T_{i,j,d} \right|}{y^T_{i,j,d}} \right) \tag{2.15}$$

Each set of fitness function will be evaluated. Because we want to get the smallest fitness function by counting down the fitness function. Let a set of variables with the smallest fitness function is prone to be selected by Roulette wheel selection. Next step is doing encoding which can make the chromosome be binary variable in Figure 2.7. As binary code length(n) is defined, the range of upperbound $\bar{x}$ and lowerbound $\underline{x}$ can be divided into 2^n. After we match the variable values to binary variable, the genes can apply one-point crossover operator to obtain a child chromosome C(t) like C1 and C2. Then, we can also apply mutation operator to C1 and C2 with mutation rate Pm(t) to produce D(t) as a child after mutation. Finally, we evaluate chromosome $P_{N_{var}}$, C1, C2, D(t) by decoding these binary variables into value and count their fitness ranking to have the best set of β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ as our solution. At the end, if GA reach the maximum stopping criteria then stop; otherwise, return to Step 1.

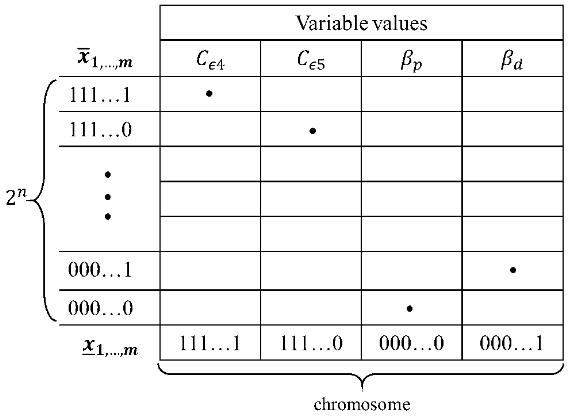


Figure 2.7: Four continuous parameter values are graphed with the quantization levels shown. The corresponding gene or chromosome indicates the quantization level where the parameter value falls.This figure adapted from Haupt and Ellen Haupt (2004)

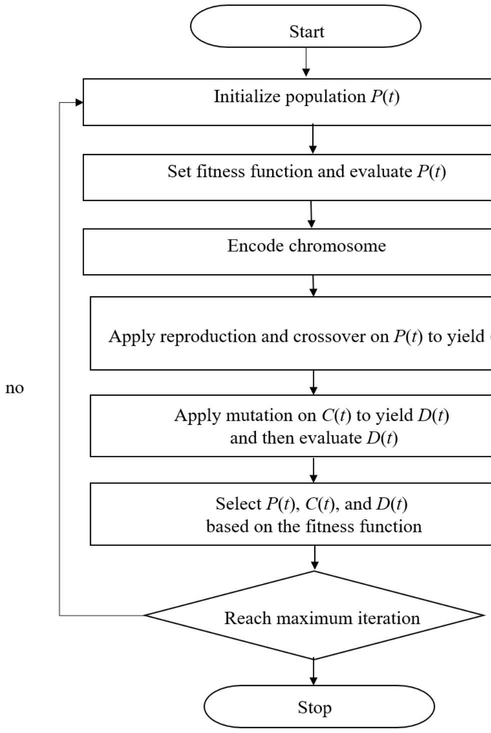

Start
Initialize population $P(t)$
Set fitness function and evaluate $P(t)$
Encode chromosome
Apply reproduction and crossover on $P(t)$ to yield $C(t)$
no
Apply mutation on $C(t)$ to yield $D(t)$ and then evaluate $D(t)$
Select $P(t)$, $C(t)$, and $D(t)$ based on the fitness function
Reach maximum iteration
Stop

Figure 2.8: Flowchart for implementation of Genetic Algorithm

2.4.2 Particle Swarm Optimization Algorithm

PSO is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution it has achieved so far. This value is called p_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. When a particle takes all the population as its topological neighbors, the best value is a global best and is called g_{best}. After finding the two best values, the particle updates its velocity and positions with following equation:

$$\left.\begin{array}{l} v_{id}^{t} = v_{id}^{t-1} + c_1 r_1 (p_{id} - x_{id}^{t-1}) + c_2 r_2 (p_{gd} - x_{id}^{t-1}) \\ x_{id}^{t} = x_{id}^{t-1} + v_{id}^{t} \end{array}\right\} \tag{2.16}$$

v_{id}^{t-1} is velocity of the i^{th} particle in d dimension space at the time of $t-1$; x_{id}^{t-1} is location of the i^{th} particle at the d dimensions at the time of $t-1$; r_1 and r_2 are random number distributed uniformly in (0,1). c_1 and c_2 are learning factors; p_{id} is p_{best} and p_{gd} is g_{best}. As Figure 2.9 shows, each particle represents a set of β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ in our paper. The second step is that we will evaluate the fitness of each particle by Equation 2.11. Then, the particles will also update their positions by the randomizer r_t and learning rate c_t to modify their local best p_{gd} and global best p_{gd}. p_{gd} is the solution of Equation 2.11. p_{gd} with the minimum solution of Equation 2.11 has the best set of parameters. If PSO reach the maximum stopping criteria then stop; otherwise, return to Step 2.

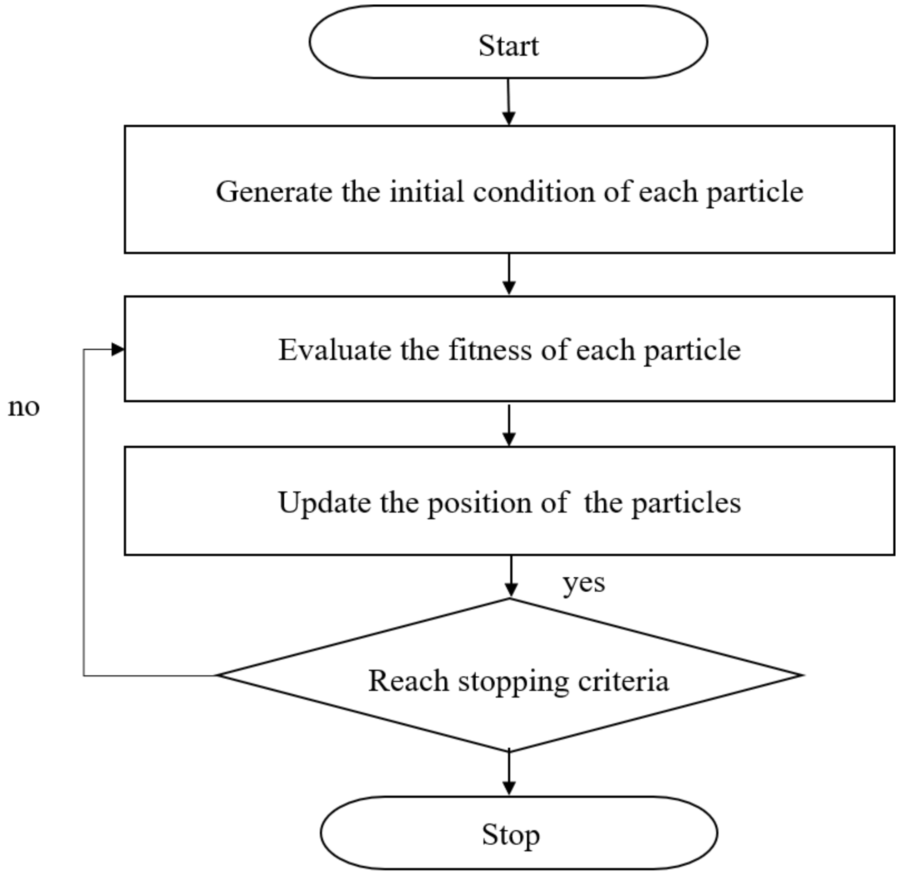

Start
Generate the initial condition of each particle
Evaluate the fitness of each particle
no
Update the position of the particles
yes
Reach stopping criteria
Stop

Figure 2.9: Flowchart for implementation of Particle Swarm Optimization

Chapter 3

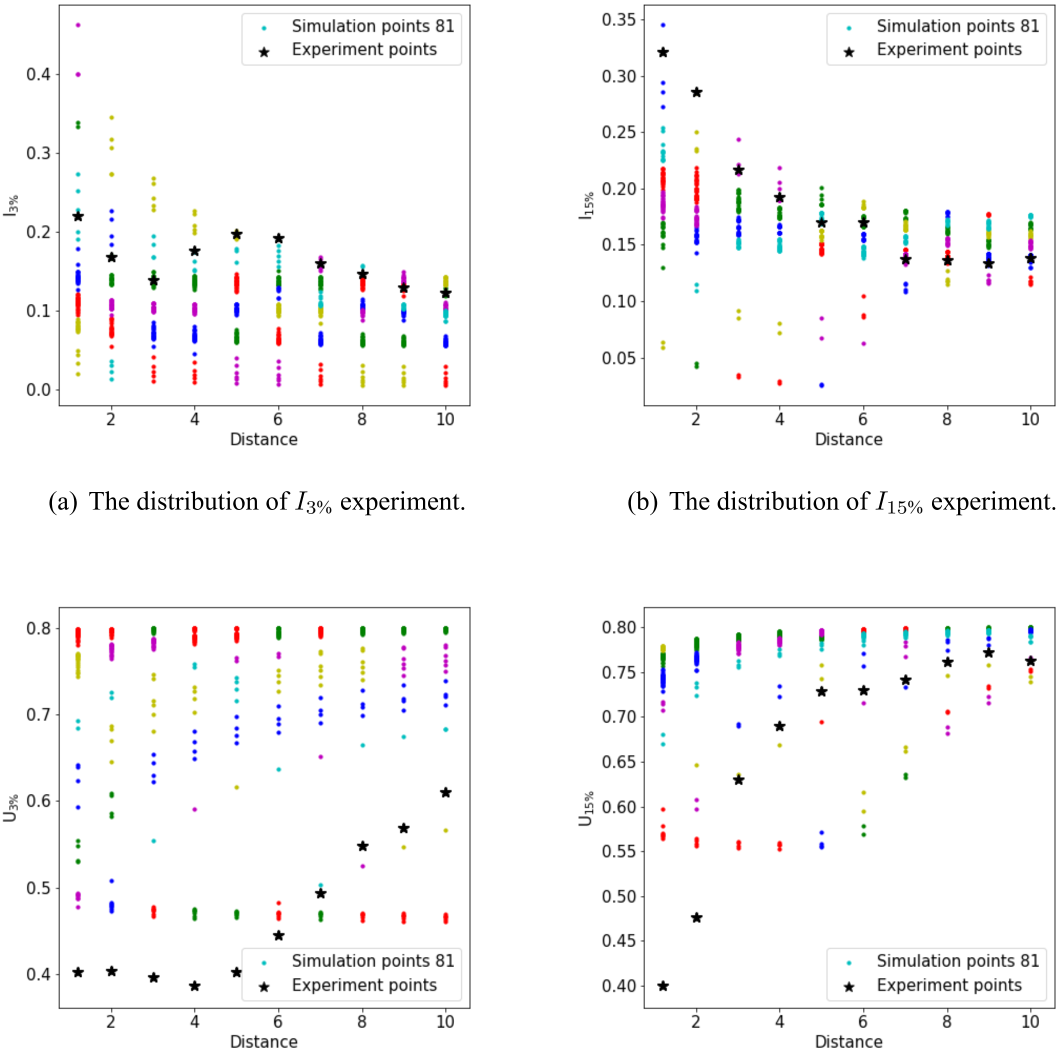
Numerical Analysis

3.1 Data Intergration

According to the physical experiment of Sandia National Laboratories (SNL), there is a set of state variables under ten different spatial distances. Instead of doing physical experiment, which is quite repetitive, time-consuming, expensive and environmentally damaging, we apply computer simulator and surrogate model to predict the result of the physical experiment under different parameters. However, some of the data which are generated from the simulator are not as moderate as our thought, so we should do data preprocessing to enhance the accuracy of the prediction. Figure 3.1 shows the distribution of the physical experiment and simulation, namely numerical experiments before data preprocessing.

As Figure 3.1 shows, the horizontal axis represents ten distance points, while the state variables $U_{3\%}$, $U_{15\%}$, $I_{3\%}$, $I_{15\%}$ are presented in the vertical axis of each sub-graph individually. We apply K-means clustering to the data under different distances by dividing them into six groups. By applying K-means clustering, we set cluster number into 6 and random state number as 42. The group which is the closest to the physical experimental points (marked with black pentagram) is chosen, and our new set of data will be extracted from them.

In order to avoid over-fitting, the whole datasets should be divided into 70% of training set and 30% of testing set. Training set is used to build the surrogate model, and testing set



(a) The distribution of $I_{3\%}$ experiment.

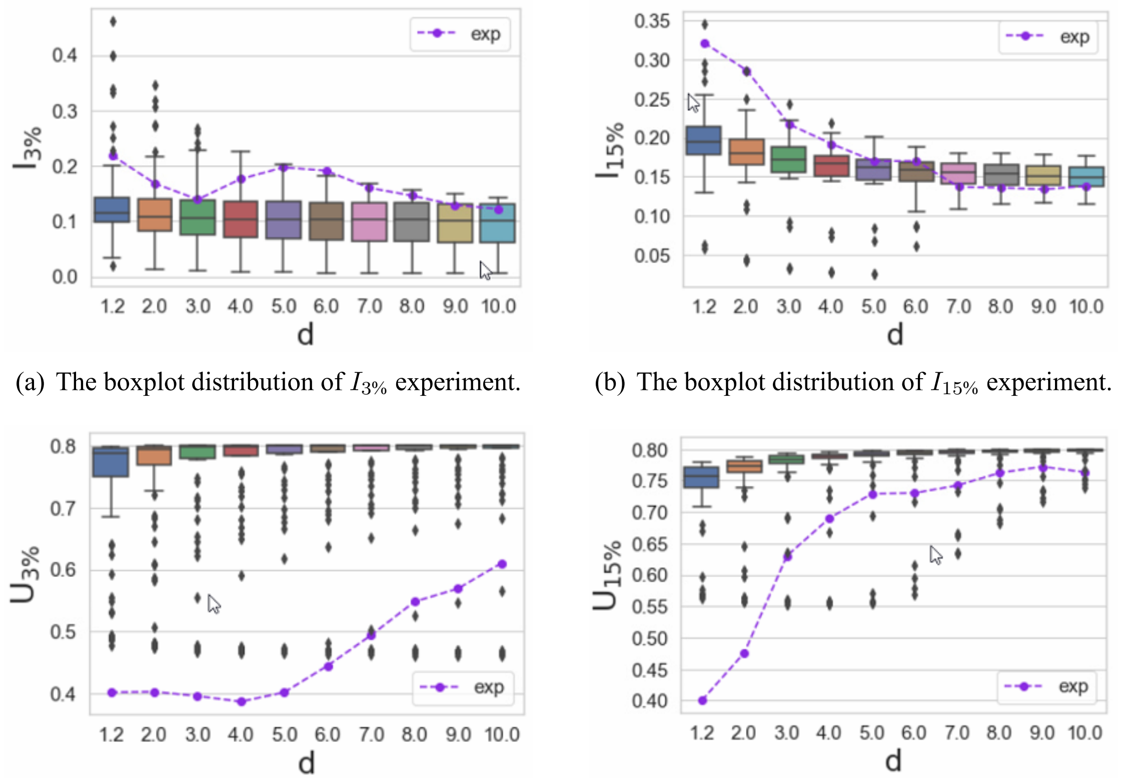
(b) The distribution of $I_{15\%}$ experiment.

(c) The distribution of $U_{3\%}$ experiment.

(d) The distribution of $U_{15\%}$ experiment.

Figure 3.1: The distribution of the physical experiment and the numerical experiment.

is to verify the accuracy of the model. In this paper, the number of 97 datasets are separated into 81 datasets for training and 16 datasets for testing. After K-means clustering, we reduce 81 datasets to 36 datasets and retrieve them to be our surrogate model training dataset.



(a) The boxplot distribution of $I_{3\%}$ experiment.

(b) The boxplot distribution of $I_{15\%}$ experiment.

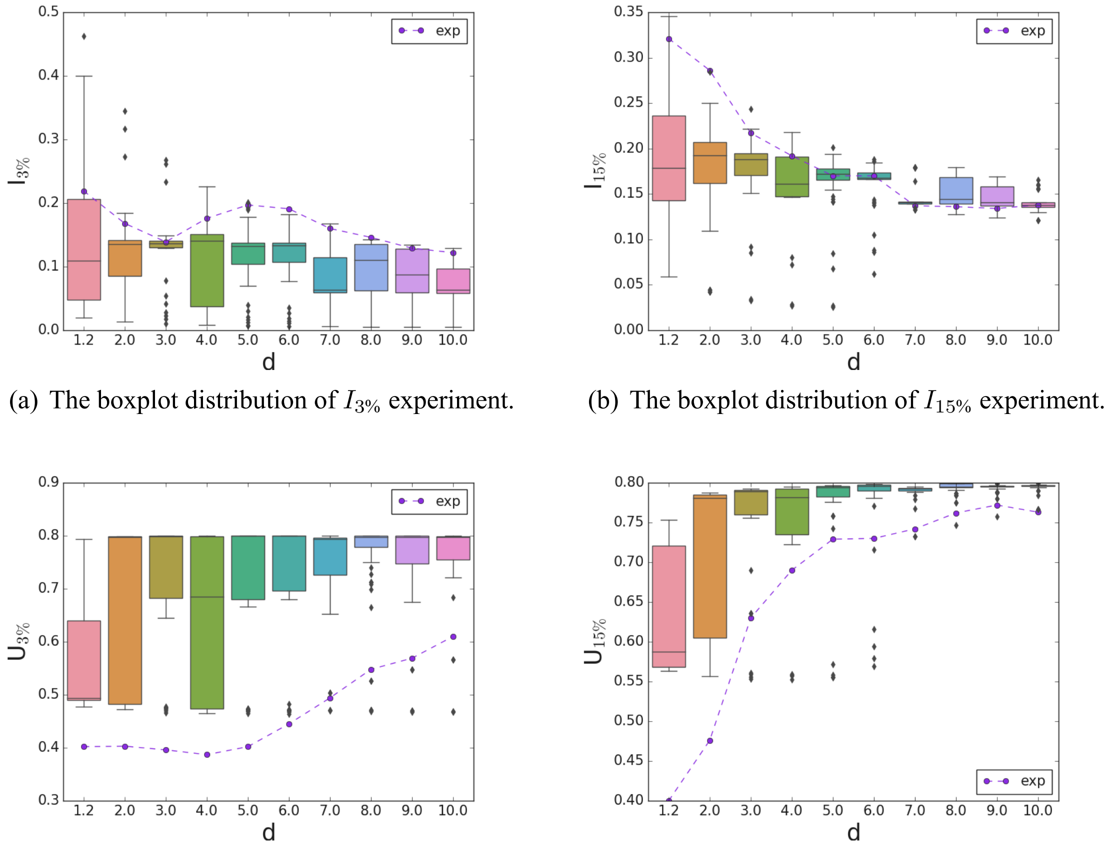
(c) The boxplot distribution of $U_{3\%}$ experiment.

(d) The boxplot distribution of $U_{15\%}$ experiment.

Figure 3.2: The boxplot distribution of the physical experiment and the numerical experiment without clustering.

In this section, we will also compare the Figure 3.2 with Figure 3.3 to show the data distribution after clustering. Also, Figure 3.2 is in comparison to Figure 3.5 to present the data distribution after applying outlier detection.Finally, we will compare Figure 3.3 with Figure 3.6 to verify that there is no change on data distribution after using bootstrap method. We use these four kinds of boxplot to verify each data preprocessing which has the positive effect on surrogate model's prediction.

Figure 3.2 represents 81 datasets without any data preprocessing, whereas Figure 3.3 shows these datasets with clustering. After applying clustering, the historical experiment of $U_{3\%}$ and $U_{15\%}$ gets closer to the physical experiment. Therefore, it is necessary to apply this method.



(a) The boxplot distribution of $I_{3\%}$ experiment.

(b) The boxplot distribution of $I_{15\%}$ experiment.

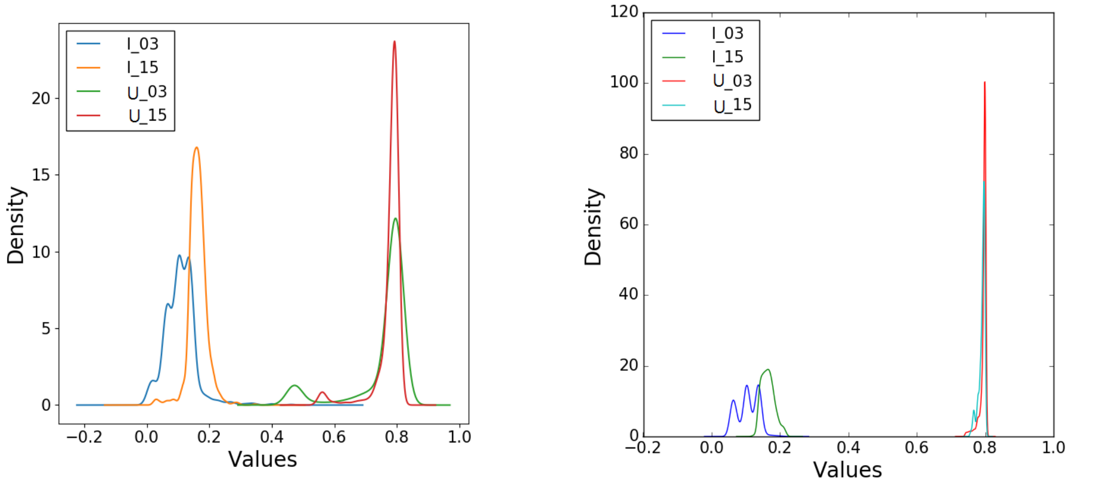
(c) The boxplot distribution of $U_{3\%}$ experiment.

(d) The boxplot distribution of $U_{15\%}$ experiment.

Figure 3.3: The boxplot distribution of the physical experiment and the numerical experiment with clustering.

As Figure 3.4(a) shows, the retrieved dataset doesn't obey Gaussian distribution so it is necessary to use univariate boxplot in order to remove the outliers Shevlyakov et al. (2013). By applying outlier detection, we set a function called detect outliers which contains the calculation of LQ(25th percentile), median(50th percentile), UQ(75th percentile), IQR(Interquartile range), x_L(minimum), and x_U(maximum). If the point , which is below the range of x_L and above the range of x_U, will be regarded as outlier, so we will remove all of them.

As Figure 3.5 shows, we have removed most of the outlier points by comparing with Figure 3.2. But once the points have been removed the UQ, LQ, median and two extremes will be changed. It is difficult to retrieve all the points that are between x_L and x_U so now we still have some outlier points in $U_{3\%}$, $U_{15\%}$, $I_{3\%}$, and $I_{15\%}$.

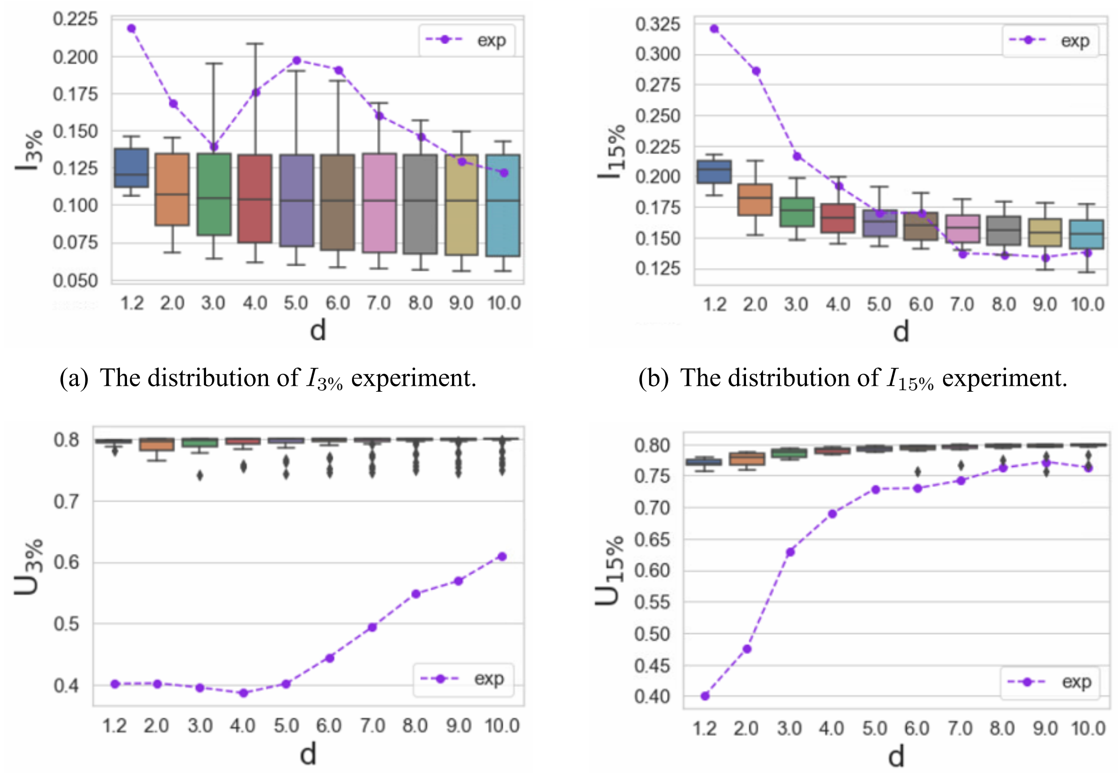


(a) Distribution of state variables with outliers.

(b) Distribution of state variables without outliers.

Figure 3.4: Using boxplot to remove the outliers

Besides, we use bootstrap method that involves iteratively resampling the 27 datasets with replacement. We use for loop to randomly select the 270 points for 1000 times and finally get 100 datasets. After this method, we increase the number of datasets into 100 datasets. Because all of the dataset are based on the original one, Figure 3.6 is the same as Figure 3.3. After these data preprocessing, it is about to train our model with 100 datasets.



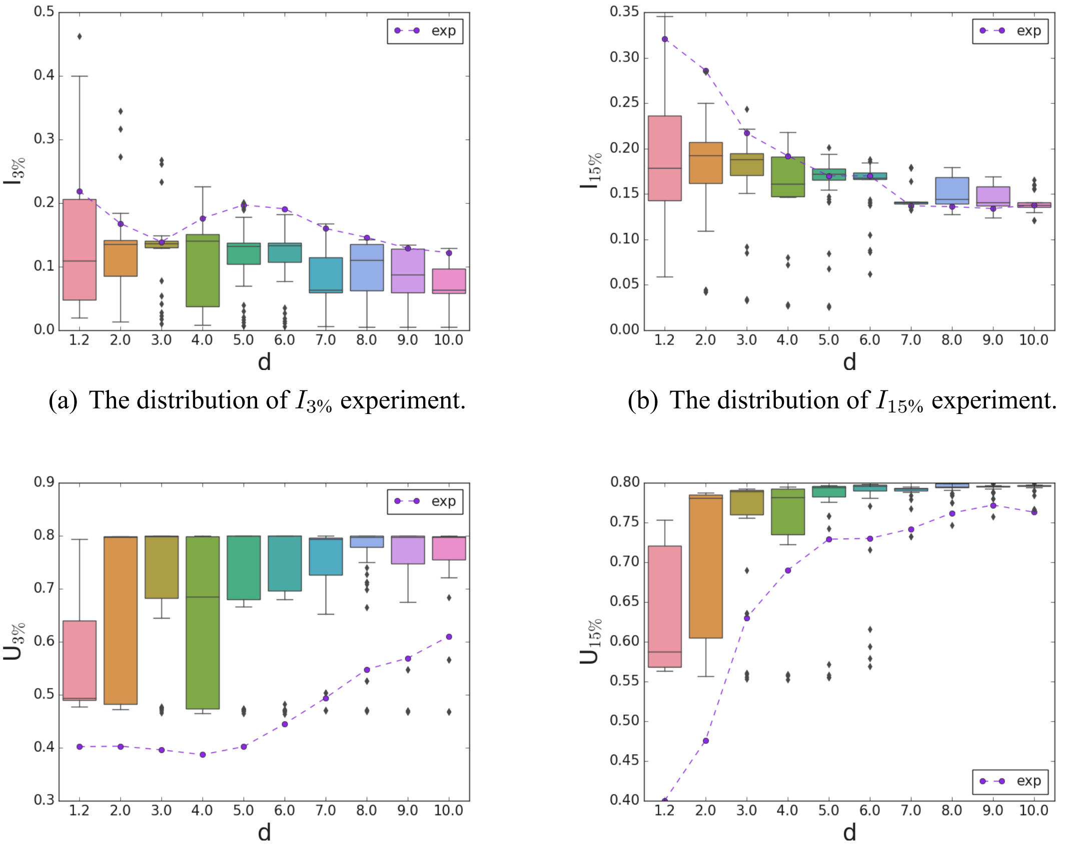
(a) The distribution of $I_{3\%}$ experiment.

(b) The distribution of $I_{15\%}$ experiment.

(c) The distribution of $U_{3\%}$ experiment.

(d) The distribution of $U_{15\%}$ experiment.

Figure 3.5: The distribution of the physical experiment and the numerical experiment with outlier.



(a) The distribution of $I_{3\%}$ experiment.

(b) The distribution of $I_{15\%}$ experiment.

(c) The distribution of $U_{3\%}$ experiment.

(d) The distribution of $U_{15\%}$ experiment.

Figure 3.6: The distribution of the physical experiment and the numerical experiment with bootstrap.

3.2 Surrogate Model Development

In this section, we will use Equation 2.4 as the evaluative criteria to evaluate the performance of the surrogate model. By following Figure 2.5, the samples in our research are only 810 points which are less than the number of 100K so it is reasonable to use Lasso, ElasticNet, SVR, RidgeRegression and EnsembleRegressors to build the surrogate model.

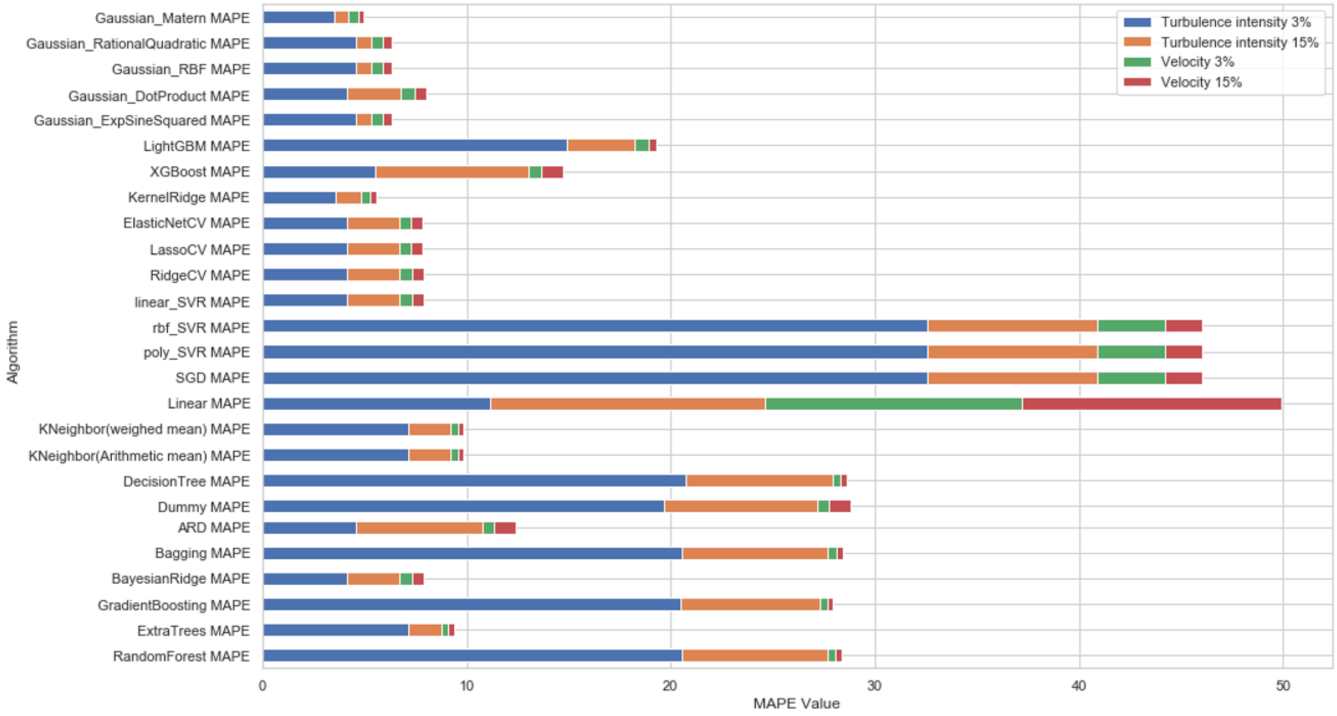


Figure 3.7: The MAPE of different kinds of machine learning regression

The Figure 3.4 presents information about the total percentage of MAPE in $U_{3\%}$, $U_{15\%}$, $I_{3\%}$ and $I_{15\%}$ predicted by different machine learning models.

Before the data is analyzed, it is difficult to know which is the best one from these twenty-six models. Through trial and error method, Gaussian model with five kinds of different kernel functions is the fittest, compared with three types of Support Vector Regression (SVR), Lasso and Ridge Regression, Decision tree and so on. All the total MAPE values of Gaussian models are under $\epsilon = 10\%$. $U_{3\%}$ always takes the lowest percentage within these four state variables.

Overall, the Gaussian model is noticeably lower in the MAPE of $U_{3\%}$, $U_{15\%}$, $I_{3\%}$ and $I_{15\%}$, so the whole dataset will be trained by this model.

3.3 Building Gaussian Process Regression

The training set, which contains 100 datasets and is close to the physical experimental points, is about to build Gaussian model.

The Gaussian Process Regression (GPR) surrogate model is constructed using Python version 3.6.4 with Numpy version 1.14.3 and Pandas version 0.22.0. The GPR is trained by scikit-learn 0.19.1, and the graph is drawn by Matplotlib version 2.1.2. The processor is GPU GeForce GTX 1060 which is extremely faster than the traditional computer equipped with CPU.

The GPR model is trained to minimize the fitting error between the surrogate model and simulator by adjusting kernel and alpha. The kernel with the lowest MAPE should be found by testing one by one. In contrast, alpha can be precisely predicted by grid search method which is tuned in the range 10^{-5} to 10^{5}. Table 3.1 shows five different kernels of GPR in $U_{3\%}$, $U_{15\%}$, and $I_{15\%}$ with alpha 0.00001 and the remaining $I_{3\%}$ with alpha 0.0001.

Surrogate-model fitting error MAPE$_{srgt/sim}$ Equation 2.4 is an indicator of the performance in five kernels. All the MAPE$_{srgt/sim}$ values are estimated by five-fold cross-validation in Figure 2.6. After doing five-fold cross-validation in training set, the GPR model is built, and it need to be tested. Therefore, testing set $N_{datasets}$ = 16 will test whether the GPR model is precise or not.

Table 3.1: Summary of the covariance functions

QoI	ExpSineSquared	DotProduct	RBF	RQ	Matern
$U_{3\%}$	5.6%	3.2%	5.6%	5.6%	2.5%
$U_{15\%}$	0.6%	0.6%	0.6%	0.5%	0.6%
$I_{3\%}$	18.8%	14.3%	18.8%	18.8%	15.9%
$I_{15\%}$	1.2%	2.8%	1.2%	1.2%	1.2%
Mean	6.55%	5.2%	6.55%	6.5%	5%

Table 3.1 presents the result of testing set (30%) in five different Gaussian kernels. It

is shown that the lowest $U_{3\%}$ lays on Matern (2.5%). RationalQuadratic has the lowest number of $U_{15\%}$ (0.5%). DotProduct contributes the lowest $I_{3\%}$ (14.3%). Each of the ExpSineSquared, RBF, RationalQuadratic, and Matern are all responsible for 1.2% in $I_{15\%}$. The average QoI of each kernals is below ϵ = 10%. There is a narrow gap of MAPE between each of the kernels, so we choose Matern, which is the lowest MAPE in the average of Qol, to build surrogate models.

3.4 Finding parameters via Metaheuristics

After building the GPR model, we will provide a set of GA and PSO individually, which can make optimal solution fitting error within θ and final error MAPE$_{sim/exp}$ less than ω.

3.4.1 The performance of Surrogate-model Fitting Error

As we have defined surrogate model fitting error in Equation 2.4, this equation is to evaluate the error between GPR model and the historical simulation data. Table 3.2 shows the performance of the error in Final$_3$.

Table 3.2: The performance of surrogate-model fitting error MAPE$_{srgt/sim}$ in Final$_3$

QoI	GPR and GA	GPR and PSO
$U_{3\%}$	4.21%	5.14%
$U_{15\%}$	1.67%	1.82%
$I_{3\%}$	14.41%	14.38%
$I_{15\%}$	5.02%	3.66%
Mean Fitting Error MAPE$_{srgt/sim}$	6.33%	6.25%

3.4.2 The performance of Predicted Error

Predicted error MAPE$_{srgt/exp}$ in Equation 2.11 is the difference between surrogate model prediction and physical experiment. Table 3.3 shows the comparison of GA and PSO in predicted error.

Table 3.3: The performance of predicted error MAPE$_{srgt/exp}$ in Final$_3$

QoI	GPR and GA	GPR and PSO
$U_{3\%}$	16.48%	16.95%
$U_{15\%}$	9.58%	19.08%
$I_{3\%}$	24.9%	24.8%
$I_{15\%}$	5.58%	19.81%
Mean Predicted Error MAPE$_{srgt/exp}$	14.13%	20.16%

3.4.3 The performance of Optimal Solution Fitting Error

The parameters which is predicted by metaheuristics will be set up in the simulator. After doing the simulation, we will get N_{points} = 40 points of simulation result which will compare with the GPR model. This method is to verify the accuracy of our GPR model's prediction.

Table 3.4: The performance of optimal solution fitting error MAPE$_{srgt/sim}$ in Final$_3$

QoI	GPR and GA	GPR and PSO
$U_{3\%}$	4.84%	9.83%
$U_{15\%}$	0.84%	0.63%
$I_{3\%}$	6.45%	12.24%
$I_{15\%}$	2.95%	2.18%
Mean Optimal Solution Fitting Error MAPE$_{srgt/sim}$	3.77%	6.22%

3.4.4 The performance of Final Error

Final error MAPE$_{sim/exp}$ reveals the difference between 40 points of simulation result and physical experiment. As Table 3.5 shows the final error of GA and PSO, we can conclude that the performance of GA is better than PSO.

Table 3.5: The performance of final error $\text{MAPE}_{sim/exp}$ in Final_3

QoI	GPR and GA	GPR and PSO
$U_{3\%}$	11.83%	12.53%
$U_{15\%}$	9.57%	20.02%
$I_{3\%}$	28.07%	35.71%
$I_{15\%}$	7.44%	17.41%
Mean Final Error $\text{MAPE}_{sim/exp}$	14.23%	21.42%

3.4.5 The performance of Genetic Algorithm and Particle Swarm Optimization

In Genetic Algorithm, 100 set of initial population variables are created randomly, and the length of binary strings is in ten. Then, we create a new population from old population by selecting and reproducing. A set of old strings are selected to reproduce a set of new strings according to the probability determined by the simulated spin of weighted roulette wheel. Because we set the stopping criteria as 1000 generations, these strings will stop their crossover and mutation until 1000 iterations.

Table 3.6 shows figures about the performance of GPR and Genetic Algorithm's estimations. Final_1 contains the 81 datasets which we increase to 100 datasets by datapreprocessing because final error $\text{MAPE}_{sim/exp}$ is not within ω = 23%. There is the second round of simulator which generates a new dataset, so the total dataset increases to 28. After we do the bootstrap, dataset in Final_2 increases to 100 datasets. In spite of the fact that there is a decline of 5.44% in final error $\text{MAPE}_{sim/exp}$ (22.41%) of Final_2 which is under the percentage of 23%. The optimal fitting error $\text{MAPE}_{srgt/sim}$ reaches to 15.04% which is over the range of θ = 10%. There is a need to have the third round. Final_3 provides the information of optimal fitting error $\text{MAPE}_{srgt/sim}$ (3.77%) and final error $\text{MAPE}_{sim/exp}$ (14.23%). As they all below to the standard of θ = 10% and ω = 23%, individually. The estimation of GPR and Genetic Algorithm's parameters comes to an end.

There is a slightly difference in the data-preprocessing of final_1, final_2, and final_3. Clustering, outlier detection, and bootstrap are applied in final_1 in order to retrieve the data which is related to the physical experiment and Gaussian distribution. However, there is no need to do clustering and outlier detection in the final$_2$ and final$_3$, as the data is already cleaned in final$_1$. On the other hand, in order to make final$_1$, final$_2$, and final$_3$ under the same standard, the kernels and the alpha are not changed during these three times estimations. GPR-surrogate is still trained by kernel Matern and the same alpha.

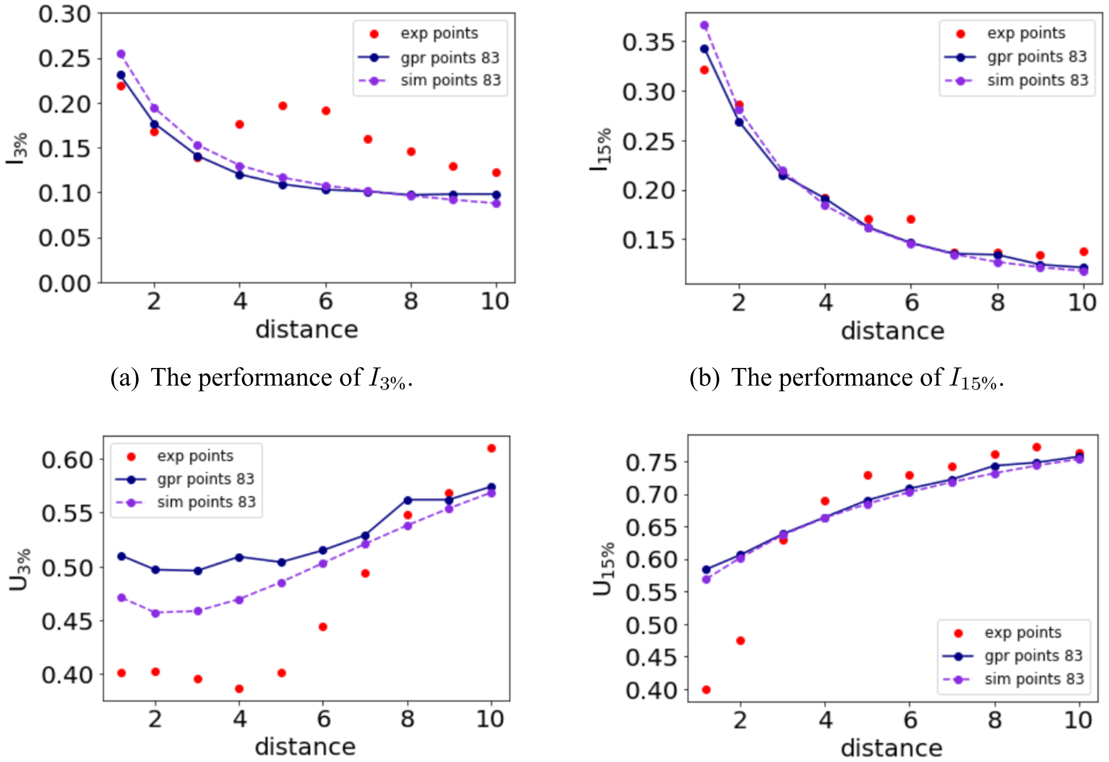
To adjust β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ parameters in simulator is considerably significant. β_p and $C_{\epsilon 4}$ are set between 0.1 to 1. The other two parameters β_d and $C_{\epsilon 5}$ are in the range of 0.1 to 4. As Genetic Algorithm can estimate the parameters in the eighth digit, we round off the decimal point to the fifth decimal place. Therefore, Table 3.6 shows parameters with fourth decimal place. The different combinations of these four parameters will result in a notably change in fitting error, predicted error, optimal fitting error, and final error. With a randomizer in GPR model, the result of four parameters is likely to change each time when GPR runs. In order to pick up the same parameters every time, the random state of GPR model is defined in 42. Also, the whole dataset is executed in the terminal environment.

As shown in the Table 3.6, β_p and $C_{\epsilon 4}$ almost reach to their upper bound. $C_{\epsilon 5}$ is around 0.2 which is modestly close to its lower bound. β_d is fluctuating which sometimes lays on the lower bound,and it shows an increasing of 1.01% in second run. Finally, it goes around 1%.

Table 3.6: GPR-surrogate and GA estimated parameters and corresponding performances

	GPR and GA		
Parameter	Final$_1$	Final$_2$	Final$_3$
β_p	0.7158	0.9683	0.9974
β_d	0.1038	1.1179	0.9845
$C_{\epsilon 4}$	0.9974	1	0.9718
$C_{\epsilon 5}$	0.1762	0.2639	0.2372
Fitting Error MAPE$_{srgt/sim}$	5%	6.26%	6.33%
Predicted Error MAPE$_{srgt/exp}$	30.87%	22.41%	14.13%
Optimal Fitting Error MAPE$_{srgt/sim}$	8.53%	15.04%	3.77%
Final Error MAPE$_{sim/exp}$	27.85%	22.41%	14.23%

PSO is operated by global and individual comparison. In our case, the initial particle



(a) The performance of $I_{3\%}$.

(b) The performance of $I_{15\%}$.

(c) The performance of $U_{3\%}$.

(d) The performance of $U_{15\%}$.

Figure 3.8: Comparison of physical experiments(red dots), GPR (blue line-dot curve), and simulation(purple dash-dot curve) in final_3 via GA.

population is set in 1000. The learning rate is [c_1,c_2]=[0.6, 0.5]. The stopping criteria of PSO is reach to maximum iteration 1000.

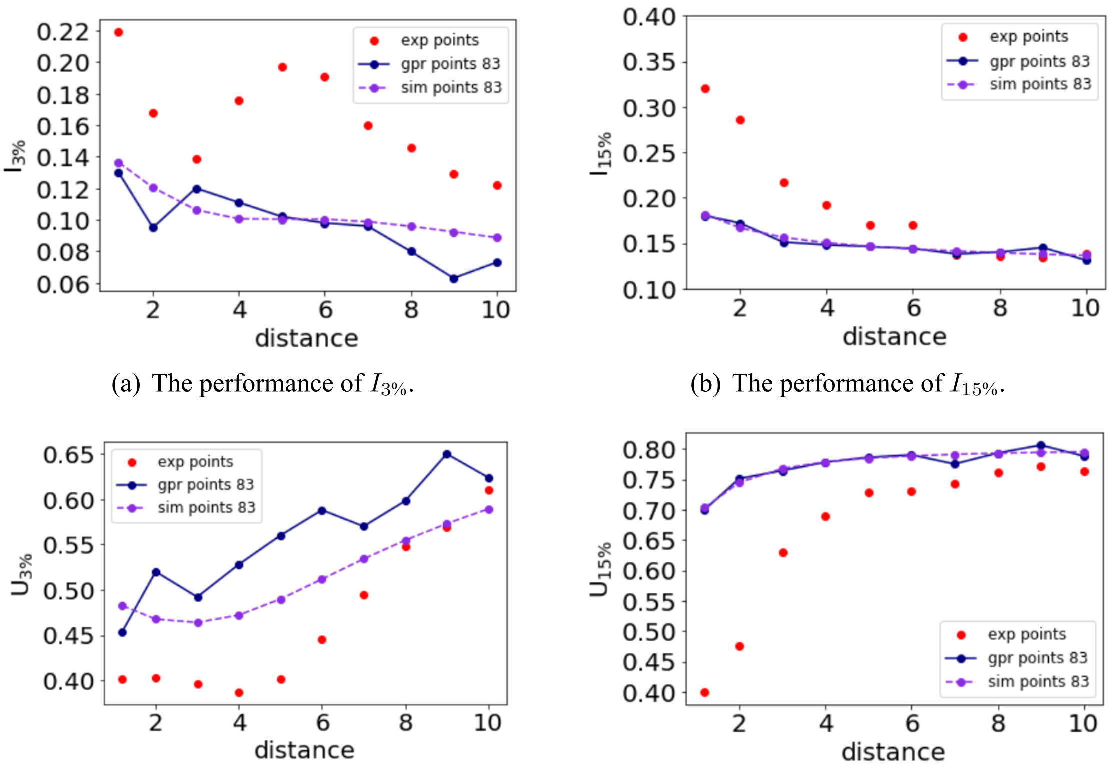
Table 3.7 presents information about the performance of GPR and Particle Swarm Optimization (PSO). Because final error MAPE$_{sim/exp}$ of final$_1$ is not within 23%, final$_1$ should go to the second round of simulator. It turns out to be final$_2$ with optimal fitting error MAPE$_{srgt/sim}$ (8.69%) and final error MAPE$_{sim/exp}$ (25.15%). There is a marginally increment in optimal fitting error MAPE$_{srgt/sim}$ up 0.03% compared with final$_1$, whereas final error MAPE$_{sim/exp}$ is in decline, dropping to 25.15% not below to 23%. There is a need to go on the third round. Final$_3$ turns out to be optimal fitting error MAPE$_{srgt/sim}$ (6.22%) and final error MAPE$_{sim/exp}$ (21.42%), as they all below to the standard of 10% and 23%, individually. It is the end of the estimation of GPR and PSO's parameters.

During these three times of experiment, it shows that the gap of fitting error MAPE$_{srgt/sim}$ and optimal fitting error MAPE$_{srgt/sim}$ are within 5%. While GPR is trained in historical data, it can gradually catch the trend of the simulator with estimation more and more accurate. Likewise, there is a narrow gap between predicted error MAPE$_{srgt/exp}$ and final error MAPE$_{sim/exp}$ because GPR can fit the simulation data precisely.

β_p, β_d, $C_{\epsilon 4}$, $C_{\epsilon 5}$ are prone to hit their upper bounds or lower bounds bu using PSO. After three times of estimation, we get β_p and $C_{\epsilon 5}$ laying in their lower bounds. The remaining ones are between 0.1 and 4. In contrast, there is no parameter hitting the upper bound or the lower bound in GA. Overall, neither the combination of parameters in GA nor the parameters in PSO are not the same.

Table 3.7: GPR-surrogate and PSO estimated parameters and corresponding performances

	GPR and PSO		
Parameter	Final$_1$	Final$_2$	Final$_3$
β_p	0.9254	0.1	0.1
β_d	0.1	3.843	2.25
$C_{\epsilon 4}$	1	0.4581	0.7
$C_{\epsilon 5}$	0.1	0.127	0.1
Fitting Error MAPE$_{srgt/sim}$	5%	5.03%	6.25%
Predicted Error MAPE$_{srgt/exp}$	30.65%	25.15%	20.16%
Optimal Fitting Error MAPE$_{srgt/sim}$	8.66%	8.69%	6.22%
Final Error MAPE$_{sim/exp}$	28.28%	25.15%	21.42%



(a) The performance of $I_{3\%}$.

(b) The performance of $I_{15\%}$.

(c) The performance of $U_{3\%}$.

(d) The performance of $U_{15\%}$.

Figure 3.9: Comparison of physical experiments(red dots), GPR (blue line-dot curve), and simulation(purple dash-dot curve) in final$_3$ via PSO.

Chapter 4

Conclusion

This paper emphasizes on data preprocessing, the building of surrogate model and metaheuristics instead of exploring the physical principle of marine turbine. As our surrogate model is built from historical simulation data which is not close to physical experiment, we apply some data preprocessing to increase accuracy of surrogate model's prediction. Then, we illustrate the use of the Gaussian surrogate model and metaheuristics to determine a set of unknown physical parameters based on historical information and define evaluative criteria to explain the model's accuracy. Our surrogate models are evaluated by MAPE, which compares the surrogate model's predictions with the physical experiment. By minimizing these surrogates' MAPE, the values of the unknown physical parameters are estimated via metaheuristics like GA and PSO. If the MAPE of optimal solution fitting error or final error are not within the range, we will add the optimal solution which is generated by each run of the simulation. Finally, we get final error $\text{MAPE}_{sim/exp}$ 14.23% via the third round of GA. The application of the surrogate methodology to the marine turbine problem provides the advantage of neglecting all the physical information about the system when building a model. This method also shows the potential of reducing the burden to solve marine turbine problems when computationally-expensive numerical simulations are required.

In our marine turbine's problem, we find twenty-six surrogate models and use the best of them to build the model, but we haven't tried Gaussian Mixture Model as a surrogate model. Although we used K-means clustering to retrieve the points which is close to physical experiments, we haven’t used Density Base Scan to select the best dataset. We suggest that the followers can go deep into these aspects in the future.

Bibliography

Anand, K., Ra, Y., Reitz, R. D., and Bunting, B. (2011). Surrogate model development for fuels for advanced combustion engines. *Energy & Fuels*, 25(4):1474–1484.

Barton, R. R. and Meckesheimer, M. (2006). Metamodel-based simulation optimization. *Handbooks in operations research and management science*, 13:535–574.

Braconnier, T., Ferrier, M., Jouhaud, J.-C., Montagnac, M., and Sagaut, P. (2011). Towards an adaptive pod/svd surrogate model for aeronautic design. *Computers & Fluids*, 40(1):195–209.

Coello, C. A. C., Pulido, G. T., and Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on evolutionary computation*, 8(3):256–279.

Elghali, S. B., Benbouzid, M., and Charpentier, J. F. (2007). Marine tidal current electric power generation technology: State of the art and current status. In *2007 IEEE International Electric Machines & Drives Conference*, volume 2, pages 1407–1412. IEEE.

Frigge, M., Hoaglin, D. C., and Iglewicz, B. (1989). Some implementations of the boxplot. *The American Statistician*, 43(1):50–54.

Haupt, R. L. and Ellen Haupt, S. (2004). Practical genetic algorithms.

Heermann, D. W. (1990). Computer-simulation methods. In *Computer Simulation Methods in Theoretical Physics*, pages 8–12. Springer.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

James, S. C., Johnson, E. L., Barco, J., and Roberts, J. D. (2017). Simulating current-energy converters: Snl-efdc model development, verification, and parameter estimation. *Renewable Energy*.

Kang, F., Xu, Q., and Li, J. (2016). Slope reliability analysis using surrogate models via new support vector machines with swarm intelligence. *Applied Mathematical Modelling*, 40(11-12):6105–6120.

Kennedy, J. (2010). Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766.

Moser, R. D., Kim, J., and Mansour, N. N. (1999). Direct numerical simulation of turbulent channel flow up to re τ= 590. *Physics of fluids*, 11(4):943–945.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

rey Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, volume 1, pages 82–87. Citeseer.

Shevlyakov, G., Andrea, K., Choudur, L., Smirnov, P., Ulanov, A., and Vassilieva, N. (2013). Robust versions of the tukey boxplot with their application to detection of outliers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6506–6510. IEEE.

Tseng, P. and Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423.

Van Laarhoven, P. J., Aarts, E. H., and Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125.

Wang, F. and Landau, D. (2001). Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86(10):2050.

Yamazaki, W., Rumpfkeil, M., and Mavriplis, D. (2010). Design optimization utilizing gradient/hessian enhanced surrogate model. In *28th AIAA Applied Aerodynamics Conference*, page 4363.