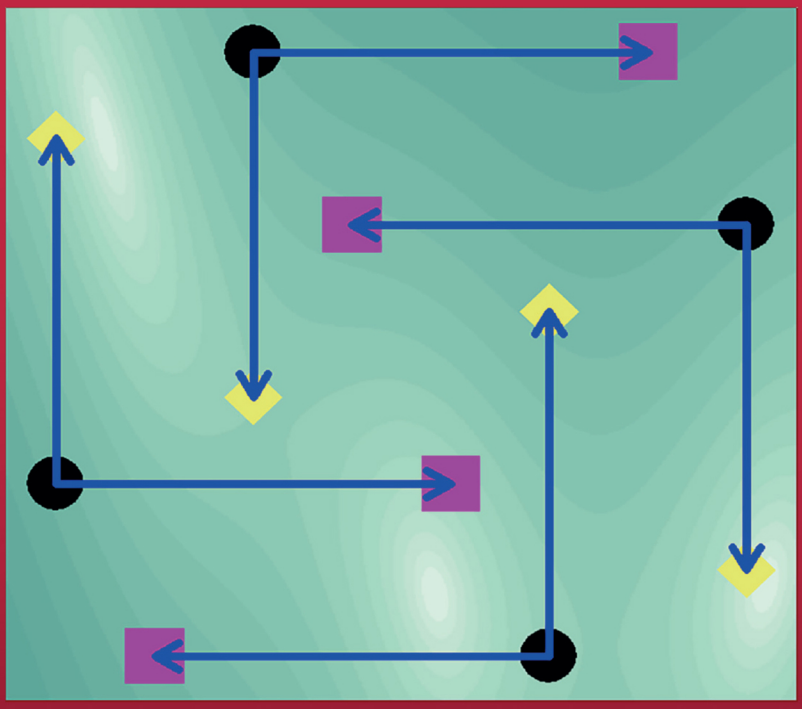
Texts in Statistical Science

Experimental Design for Data Science and Engineering



V. Roshan Joseph

A Chapman & Hall Book

CRC Press
Taylor & Francis Group

Experimental Design for Data Science and Engineering

Theory, experiments, computation, and data are considered as the four pillars of science and engineering. *Experimental Design for Data Science and Engineering* describes efficient statistical methods for making the experiments cheaper and computations faster for extracting valuable information from data, and helps identify discrepancies in the theory. The book also includes recent advances in experimental designs for dealing with large amounts of observational data.

Traditionally, the design and analysis of physical and computer experiments are treated differently, but this book attempts to create a unified framework using Gaussian process models. Although optimal designs are formulated using Gaussian process models, the focus is on obtaining practical experimental designs that are robust to model assumptions. A wide variety of topics are covered in the book – from designs for interpolating or integrating simple functions to designs that are useful for optimizing and calibrating complex computer models. It draws techniques that are spread across the fields of statistics, applied mathematics, operations research, uncertainty quantification, and information theory, and builds experimental design as a fundamental data analytic tool for engineering and scientific discoveries.

- Designs for both computer and physical experiments are discussed in a unified framework.
- Integrates several concepts from numerical analysis, Monte Carlo methods, sensitivity analysis, optimization, and machine learning with experimental design techniques in statistics.
- Methods are explained using many real experiments from the physical sciences and engineering.
- Experimental design techniques for analysis and compression of big data are discussed.
- All the numerical illustrations in the book are reproducible using R and Python codes provided on the author's GitHub site.

V. Roshan Joseph is A. Russell Chandler III Chair and Professor in the Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology, Atlanta. He is an author of more than 100 journal articles and has received many research awards. He is a Fellow of the American Statistical Association and the American Society for Quality, and a former editor of *Technometrics*.

CHAPMAN & HALL/CRC
Texts in Statistical Science Series

Joseph K. Blitzstein, *Harvard University, USA*
Julian J. Faraway, *University of Bath, UK*
Martin Tanner, *Northwestern University, USA*
Jim Zidek, *University of British Columbia, Canada*

Recently Published Titles

Applied Categorical and Count Data Analysis, Second Edition
Wan Tang, Hua He, and Xin M. Tu

Geographic Data Science with Python
Sergio Rey, Dani Arribas-Bel, and Levi John Wolf

Models for Multi-State Survival Data
Rates, Risks, and Pseudo-Values
Per Kragh Andersen and Henrik Ravn

Spatio–Temporal Methods in Environmental Epidemiology with R, Second Edition
Gavin Shaddick, James V. Zidek, and Alex Schmidt

A Course in the Large Sample Theory of Statistical Inference
W. Jackson Hall and David Oakes

Statistical Inference, Second Edition
George Casella and Roger Berger

Nonparametric Statistical Methods Using R, Second Edition
John Kloke and Joesph McKean

Generalized Linear Mixed Models
Modern Concepts, Methods and Applications, Second Edition
Walter W. Stroup, Marina Ptukhina, and Julie Garai

Analysis of Categorical Data with R, Second Edition
Christopher R. Bilder, Thomas M. Loughin

Applied Nonparametric Statistical Methods, Fifth Edition
Nigel C. Smeeton, Neil H. Spencer, and Peter Sprent

Linear Models with R, Third Edition
Julian J. Faraway

Statistics in Survey Sampling
Jae Kwang Kim

Time Series: A Data Analysis Approach Using R
Robert H. Shumway and David S. Stoffer

For more information about this series, please visit: https://www.routledge.com/Chapman--HallCRC-Texts-in-Statistical-Science/book-series/CHTEXSTASCI

Experimental Design for Data Science and Engineering

V. Roshan Joseph



CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

Designed Cover Image: Author's Original

First edition published 2026
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

© 2026 V. Roshan Joseph

Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

The Open Access version of this book, available at www.taylorfrancis.com, has been made available under a Creative Commons [Attribution-Non Commercial-No Derivatives (CC-BY-NC-ND)] 4.0 International license.

Any third party material in this book is not included in the OA Creative Commons license, unless indicated otherwise in a credit line to the material. Please direct any permissions enquiries to the original rightsholder.

For Product Safety Concerns and Information please contact our EU representative GPSR@taylorandfrancis.com. Taylor & Francis Verlag GmbH, Kaufingerstraße 24, 80331 München, Germany.

Trademark notice: Product or corporate names may be trademarks or registered trademarks and are used only for identification and explanation without intent to infringe.

ISBN: 978-1-041-11752-0 (hbk)
ISBN: 978-1-041-11758-2 (pbk)
ISBN: 978-1-003-66153-5 (ebk)

DOI: 10.1201/9781003661535

Typeset in CMR10 font
by KnowledgeWorks Global Ltd.

To my beloved family
Smitha, Reya, and Aneya



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

Preface **xi**

I Introduction **1**

1 Experiments **3**
1.1 Overview . . . 3
1.2 Applications . . . 4
1.3 Role of Uncertainty . . . 7
1.4 Outline of the Book . . . 9

2 Modeling Techniques **10**
2.1 Interpolation . . . 10
2.1.1 Polynomial Interpolation . . . 10
2.1.2 Radial Basis Functions . . . 13
2.1.3 Kriging . . . 16
2.1.4 Gaussian Process Modeling . . . 19
2.2 Regression . . . 23
2.2.1 Linear Regression . . . 23
2.2.2 Kernel Ridge Regression . . . 28
2.2.3 Gaussian Process Regression . . . 30

II Computer Experiments **37**

3 Model-based Designs **39**
3.1 Prediction-based Designs . . . 39
3.2 Maximum Entropy Designs . . . 43

4 Space-filling Designs **47**
4.1 Clustering-based Designs . . . 47
4.2 Minimax Designs . . . 49
4.3 Maximin Designs . . . 52
4.4 Latin Hypercube Designs . . . 56
4.5 MaxPro Designs . . . 60
4.5.1 Constrained Regions . . . 65
4.5.2 Qualitative Factors . . . 66
4.5.3 Branching Designs . . . 70
4.5.4 Nested Designs . . . 71
4.6 Minimum Energy Designs . . . 75
4.6.1 Bayesian Computation . . . 78
4.6.2 Candidate Set Generation . . . 80
4.6.3 Continuous Fidelity Parameter . . . 81

5 Representative Points **89**
5.1 Uniform Design . . . 89
5.1.1 Uniformity . . . 89
5.1.2 Integration . . . 91
5.2 Non-uniform Distributions . . . 95
5.2.1 Transformation Methods . . . 95
5.2.2 Support Points . . . 97
5.2.3 Uncertainty Propagation . . . 103
5.2.4 Population Quasi-Monte Carlo . . . 104

6 Screening Designs **111**
6.1 Sensitivity Analysis . . . 111
6.1.1 Variance-based Sensitivity Measures . . . 111
6.1.2 Derivative-based Sensitivity Measures . . . 116
6.2 Morris Screening Design . . . 118
6.3 MOFAT Designs . . . 120

7 Sequential Designs **127**
7.1 Emulation . . . 127
7.1.1 Prediction-based Sequential Design . . . 128
7.1.2 Entropy-based Sequential Design . . . 130
7.2 Optimization . . . 132
7.3 Inverse Designs . . . 135

III Physical Experiments **141**

8 Fractional Factorial Designs **143**
8.1 Two-level Designs . . . 143
8.2 Bayesian-inspired Designs . . . 149
8.3 Designs with More than Two Levels . . . 154
8.4 Mixture Designs . . . 158

9 Model Calibration **164**
9.1 Nonlinear Optimal Designs . . . 164
9.1.1 Analytical Models . . . 166
9.1.2 Computationally Expensive Models . . . 172
9.2 Robust Designs . . . 175
9.2.1 Parameter Uncertainties . . . 175
9.2.2 Model Uncertainties . . . 176

IV Data Science **183**

10 Data Subsampling **185**
10.1 Support Points-based Subsampling . . . 185
10.2 Data Splitting . . . 188
10.3 Data Twinning . . . 189
10.4 Supervised Compression . . . 191

11 Data Analysis **196**
11.1 Factor Selection and Ranking . . . 196

11.2 Twin-Gaussian Process 201

Bibliography **205**

Index **229**



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Preface

An experiment can be designed to collect data and obtain the maximum information from a system, or it may be that the data are already collected through an observational study and we want to summarize the data without losing any information. The data can be represented as points in the space of some variables, which can also be viewed as samples from a probability distribution. We will call the set of points at which we want to collect the data or the points that best summarize the data as *optimal samples*. This book is about optimal sampling, which is rooted in the discipline of experimental design in statistics.

Traditionally, the field of experimental design has focused on the planning and collection of data from physical systems. It is originated from agricultural experiments from the pioneering work of Ronald Fisher and expanded into science, engineering, and industry. There are many excellent books that deal with this subject in depth such as Wu and Hamada (2021). Recently the focus of this field has been shifted to computer experiments as many physical systems can be modeled and simulated in computers. Santner et al. (2018) is an excellent book that deals with design of computer experiments.

Since most computer models are deterministic in nature, the experimental designs turned out to be closely related to the points/knots used for interpolation and integration in *numerical analysis*, which has its own rich history. Fang and Wang (1993) is a nice book that explains the use of Quasi-Monte Carlo (QMC) techniques in numerical integration. This also leads to the field of *operations research*, where Monte Carlo techniques are extensively used for *simulation*. This makes a connection to *Bayesian statistics*, where simulating from probability distributions is a key technique for making inferences. However, when complex computer codes are involved, experimental design-based techniques can become useful for propagating uncertainties, which is a topic studied in the field of *uncertainty quantification*. When *optimization* is the objective, the samples are gathered sequentially so that we can move to the optimum location in the space with few evaluations of the objective function. This topic is intimately connected to sequential experimental designs in statistics and active learning techniques in *machine learning*.

Once the data are collected either through an experimental or observational study, classical methods in statistics focus on summarizing the data through sufficient statistics and parameterized models, whereas modern trends in machine learning and nonparametric regression methods is to directly model the full data. However, as the size of the data grows, modeling can be a computational bottleneck, in which compressing the data to a smaller size without losing information becomes useful. This topic has connections to *quantization* of probability distributions, which again has a long history stemming from communications and *information theory*. Although there are no experiments to perform, experimental design concepts can still be useful for selecting the most informative points from the data.

The aim of this book is to review some of the experimental design tools and techniques that are spread across the fields of statistics, numerical analysis, optimization, simulation, uncertainty quantification, and information theory. I hope that bringing several of these concepts into one place under the unifying theme of experimental design will stimulate interests from different communities and lead to research that will help advance the field faster. Although the coverage of topics is much broader than in a typical book on

experimental design, this book omits several techniques that are used in clinical trials, causal inference, survey sampling, and online experiments. The experimental design techniques discussed in this book are mainly tailored for applications in physical sciences, engineering, and data science.

This book is written in a somewhat unorthodox style in the sense that the presentation of topics does not follow their historical development. They are presented in an order of how they should have developed rather than how they got developed. For example, although physical experiments were developed much before computer experiments, in this book, physical experiments are presented after the computer experiments. The reason for doing this is that computer experiments are usually deterministic in nature and the data are error-free. On the other hand, the data from physical experiments are corrupted by noise. Thus, in principle, the data from computer experiments are cleaner and easier to handle. Therefore, the techniques for the design and analysis of computer experiments are discussed in Part II of the book, and then, the experimental design techniques for physical experiments are discussed in light of the developments made in computer experiments. This will provide a new perspective to the reader and, as I hope, will lead to new developments in the field.

I started to write this book as a research monograph, and therefore it overemphasizes some of my own research work. However, over time I have included several topics that now I feel this book can be genuinely used as a textbook or as a reference book for a second course on design of experiments. In fact, the material in this book grew out of a course on advanced experimental design that I have offered for graduate students at Georgia Tech.

A critical component of this book is the associated GitHub site which provides R codes for reproducing all the numerical illustrations of this book:

https://github.com/vroshanjoseph/rexpdes.

I have chosen to separate the codes from the book so that a reader who is not familiar with R programming will not be distracted by those codes. Although I have extensively cited R functions and packages throughout the book, the methods are explained without using any programming language. However, browsing the R codes on the GitHub site will help improve learning. For those readers who prefer to use Python, my student Youngseo Cho has translated several R scripts into Python and implemented Python wrappers for R-only packages which are provided on the GitHub site:

https://github.com/vroshanjoseph/pyexpdes.

I am thankful to many individuals and organizations that have inspired and supported me during the writing of this book. I am grateful for the support provided by my school through the endowment of A. Russell Chandler III and many grants from the US National Science Foundation, in particular DMS-2310637. I would like to thank Jeff Wu for his constant encouragement. A special thanks goes to my collaborators Shreyes Melkote, Surya Kalidindi, Rampi Ramprasad, William Brenneman, Bill Myers, Douglas Ray, Mark Losego, Patrick Knapp, and William Lewis, who have exposed me to challenging experimental design problems in science and engineering. I am thankful for the comments received from Hongquan Xu and Abhyuday Mandal, and to Bobby Gramacy for sharing his experience in publishing a book. This work would not have been possible without the contribution and help from my current and former students Simon Mak, Bill Huang, Shangkun Wang, Difan Song, Wei-Yang Yu, and Youngseo Cho.

V. Roshan Joseph
Georgia Institute of Technology
Atlanta, Georgia, USA

Part I

Introduction



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

1

Experiments

1.1 Overview

An important task in science and engineering is to develop a relationship between the inputs and outputs of a system. The "System" that we are interested in could be very general and may refer to a rocket engine, a machining process, or even a human brain. The inputs, for example, in a machining process, could be the speed of machining or a specific angle of the cutting tool, while the outputs could be the force on the tool or the temperature on the workpiece.

An approach to build the input-output relationship (also known as the response surface) is to collect data from the system and develop the relationship using statistical and machine learning methods. When data is expensive and resources are limited, we need to be careful about the data collection process so that we can obtain "useful" data with minimum cost and time. Performing carefully planned experiments on the system is one way to do this. Here an experiment refers to making purposeful changes to the inputs and observing the outputs as opposed to an observational study, where we simply observe both inputs and outputs without making any changes to the system. An experimental design is a plan for doing such an experiment, which prescribes where and how to collect the data from the system.

Another approach to build the input-output relationship is to use the physical laws governing the system (such as Newton's laws) and derive the relationship. One may think that with such an approach we do not need data or experiments, but that is not true! The physics-based models may contain unknown constants (such as friction coefficient), which can be specified only if we have data from the physical system. Moreover, the physics-based model may not be an easy-to-evaluate analytical function and may require sophisticated numerical methods, such as finite element methods for solving the underlying mathematical model, which could be a system of partial differential equations. The numerical implementation of the solution procedure on computers is usually known as a computer model or simulator. Since the computer model is computationally expensive, we again need to carefully plan the computer experiments. Thus, experimental designs are important in this context as well, but "how" to do the experiments becomes irrelevant and we only need to be concerned with "where" to do the experiments.

In some cases, a combination of the foregoing two approaches may be needed to fully develop the input-output relationship. For example, the investigator may not have full understanding of the system and may be able to develop a physics-based model that is only partially correct. This situation is very common, as most engineered or biological systems are very complex and difficult to model completely using physics. Therefore, we will also need to collect data from the physical system to correct for the bias due to missing physics and to calibrate the computer model. This will require both a physical experiment and a computer experiment.

DOI: 10.1201/9781003661535-1

This chapter has been made available under a CC-BY-NC-ND license 4.0.

1.2 Applications

Once the input-output relationship is developed, it can be used for many purposes, such as optimizing or controlling the system. Below are a few real examples.

Krishna et al. (2024) were interested in finding stable crystal structures. However, creating materials in a lab with different crystal structures and investigating their stability is tedious, expensive, and time consuming. Crystal structures that are found in nature are known to have low potential energy. Therefore, a more feasible approach would be to compute the potential energy of a crystal structure using Density Functional Theory (DFT) and find the optimal atomic configuration that minimizes the potential energy (Figure 1.1). Although this computational approach is cheaper, it is still not an easy problem because DFT computations are expensive and the atomic space to explore is huge. To add to that complexity, the potential energy surface can have many local minima (e.g., graphite and diamond are different configurations of carbon atoms). Experimental design techniques can be used to efficiently search the huge atomic space and find the global optimum with minimum number of DFT computations.

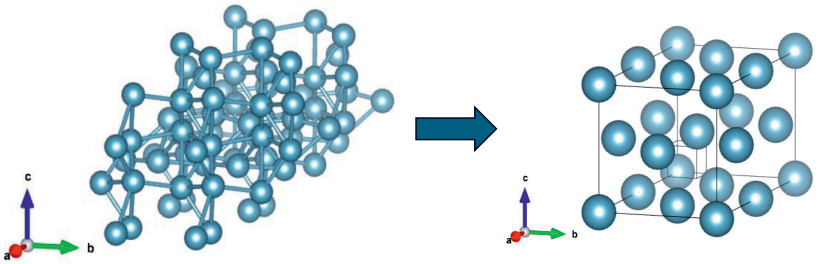


FIGURE 1.1: Optimization of crystal structures using Density Functional Theory. Figure courtesy: Rampi Ramprasad's lab.

Huang et al. (2021) described the problem of making hybrid membranes for chemical separations. Metal particles are infused into a polymer membrane to make hybrid membranes so that they do not easily dissolve during the chemical separation process. This is done using a vapor phase infiltration process. Ren et al. (2021) developed a reaction-diffusion model to predict the amount of metal particles trapped in the polymer membrane. However, this model has five unknown parameters (such as diffusion constants) that need to be specified before the model can be used for prediction. They can be estimated from the real data shown as a black solid line in Figure 1.2. This is a model calibration problem, but cannot be easily solved using the usual nonlinear regression techniques because here the computer model is expensive to evaluate. Therefore, experimental design techniques are necessary to efficiently solve this problem.

Gul et al. (2018) reported a problem faced by a machining software company. The right side of Figure 1.3 shows the tangential force acting on the tool over time in a solid end milling simulation. Some of the inputs that can be varied in the machining software console are shown on the left side of the figure. The simulator is deterministic; that is, it will give the same output of tangential force for the same input settings. However, in reality, the tool wears out and the material properties change from piece to piece. Therefore, the output force will not be exactly as predicted and will have some variability depending on the state

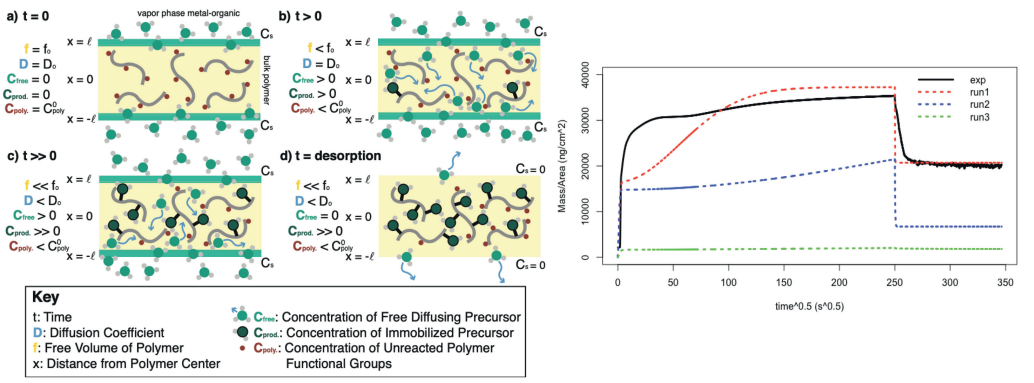


FIGURE 1.2: Calibration of reaction-diffusion model in a vapor-phase infiltration process. The real data is shown as a black solid line and three model evaluations as dashed lines. Reprinted with permission from Yi Ren, Emily K. McGuinness, Chaofan Huang, V. Roshan Joseph, Ryan P. Lively, and Mark D. Losego (2021). "Reaction–Diffusion Transport Model to Predict Precursor Uptake and Spatial Distribution in Vapor-Phase Infiltration Processes". In: Chemistry of Materials 33(13), 5210–5222. Copyright 2021 American Chemical Society.

of the tool and the material. The software company would like to add a confidence interval to the prediction to capture the uncertainty in the inputs. A simple way to construct this confidence interval is to use Monte Carlo (MC). That is, we can draw random samples from the distribution of the inputs and then run the simulator to produce a collection of the

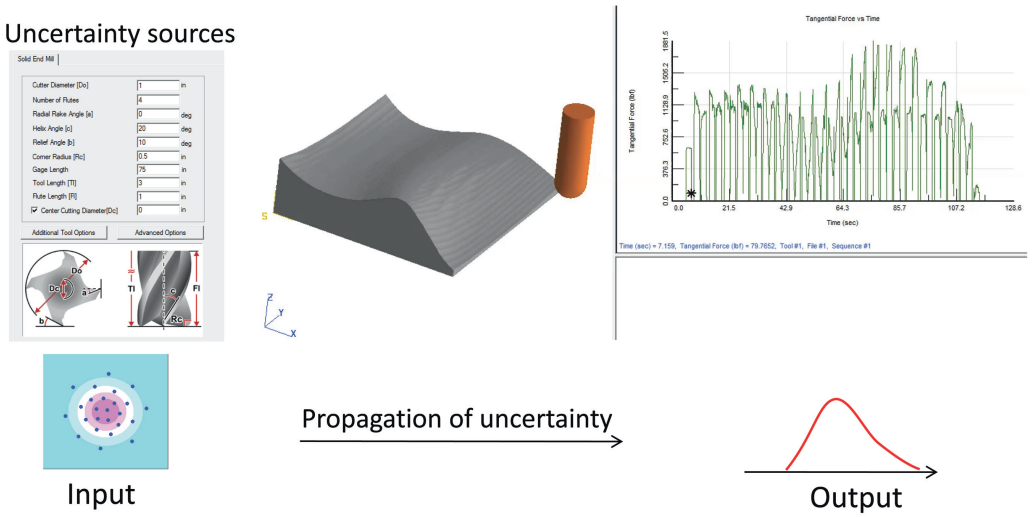


FIGURE 1.3: Uncertainty propagation in a solid end milling process. Figure courtesy: Shreyes Melkote's lab.

output curves from which we can obtain, say, the 95% confidence intervals. This confidence interval will be accurate if we use a large number of MC samples. However, this is inefficient because the simulation is not cheap to run. We can use experimental design techniques to optimally sample from the input distribution and propagate the uncertainty, thus saving the simulation time.

The objectives of the aforementioned examples are different: The first problem on crystal structures is about optimization, the second problem on membranes is about calibration, and the third problem on machining is about uncertainty propagation. There is no single optimal experimental design solution to all these problems. In fact, the optimal experimental design differs depending on the objective and the type of response surface. Figure 1.4 illustrates several examples using one-dimensional functions. For example, suppose that the objective is to approximate the response surface. If the response surface is rough, then a design that uniformly fills the space is good (top left panel of Figure 1.4); whereas, if the function is smooth, then a design that places more points near the boundary is good (top right panel). If the objective is to maximize a function, we should place more design points near its global maxima (bottom-left panel). On the other hand, if the objective is to propagate uncertainty, the design points should be placed in such a way as to mimic the distribution of the input (bottom right panel with function depicting the density of input).

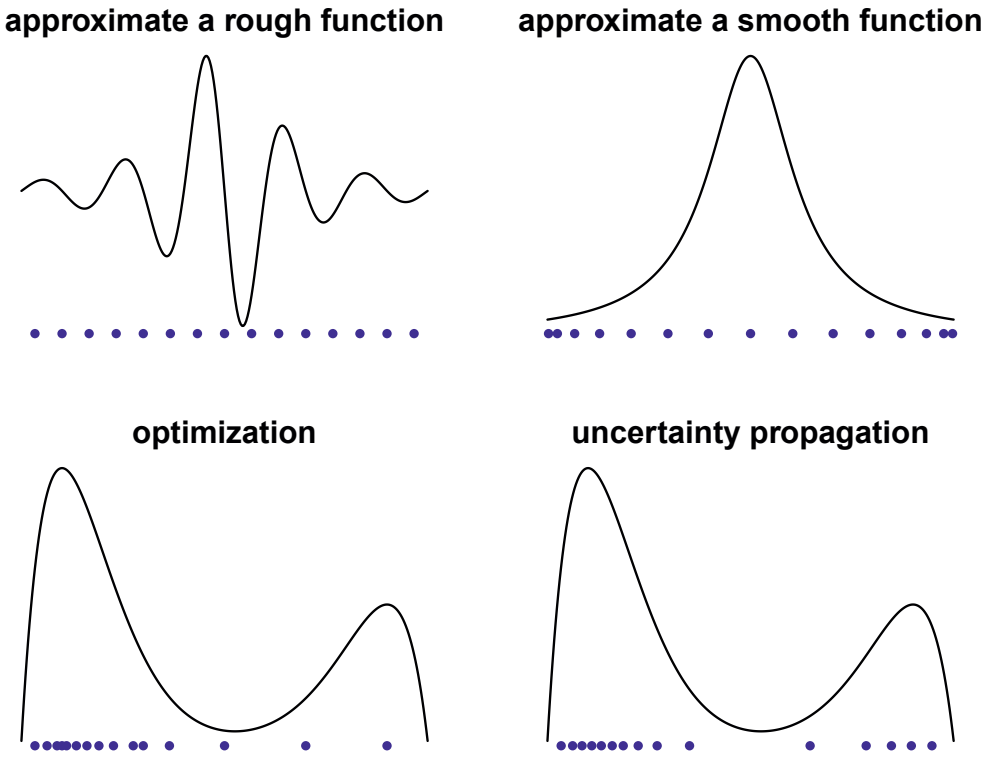


FIGURE 1.4: Optimal experimental design (solid circles) for different objectives.

An experimental design specifies where the data needs to be collected. Therefore, when trying to find the optimal experimental design, we do not have any data with us. In other words, we do not know anything about the response surface when designing the experiment.

But Figure 1.4 shows that the optimal experimental design depends on the underlying response surface (except for the last case). Thus, finding the optimal experimental design seems like an ill-defined problem. Fortunately, there is one way to get around this issue. We do not need to specify all the design points beforehand. We can start with an initial design, collect data, and then decide where to do the next experiment. This is called a sequential design approach (also known as active learning in machine learning literature). As far as the initial design is concerned, the uniformly spread points shown in the top left panel of Figure 1.4 seem like a good strategy. This is called a space-filling design and will be discussed in detail in a later chapter.

1.3 Role of Uncertainty

If we know everything about the underlying response surface, then we do not need to collect data or need any experiments. It is the uncertainty about the response surface that forces us to perform experiments and collect data. The data bring us new information and reduce our uncertainty about the response surface. Thus, it makes sense to perform experiments that maximize information or minimize uncertainty. As shown in Lindley (1956), the theory of optimal experimental design can be formulated using concepts from information theory (Shannon, 1948). However, we can develop most of the experimental design techniques discussed in this book without delving into information theory as long as we have a way to quantify uncertainties. We will quantify the uncertainties using some statistical modeling techniques.

To discuss about modeling techniques, we need to first introduce some notation. Let $\mathbf{x} = (x_1, \ldots, x_p)'$ be the p inputs (also known as *factors*) and y be the output (or *response*). There could be many outputs, but in most cases it is enough to develop the modeling techniques for multi-input/single-output system, which can be separately applied on each of the outputs. Assume the statistical model

$$y = h(\mathbf{x}) + \epsilon, \tag{1.1}$$

where $h(\mathbf{x})$ is the input-output relationship (response surface) and ϵ is the random error caused by changes in numerous other factors that might affect the output but not considered in the experiment. In a physical experiment, ϵ will also include the measurement errors caused by the instrument. If the experiment is performed using a deterministic computer model, we can ignore the term ϵ. It is common to assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is the variance of the random error. The normality assumption and the additive error assumption in (1.1) may not be valid, and some transformations may be needed to make it work in practice (Box and Cox, 1964).

Suppose that we performed an experiment with one factor at two values (also known as *levels*): $x = 0$ and $x = 1$ with three replicates. Data are shown as solid blue circles in Figure 1.5. Assume $h(\cdot)$ to be the function shown as a black solid line in the figure, but unknown to the experimenter. Linear regression models are commonly used in physical experiments (Wu and Hamada, 2021). Suppose that we fit the model $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ to the data, where $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$, for $i = 1, \ldots, n$ (here $n = 6$). The fitted line along with 95% confidence intervals are shown in the left panel of the figure. Gaussian process (GP) regression (Rasmussen and Williams, 2006) is rarely used in this context, but is also shown in the right panel of the same figure.

Although the predictions from both the linear and GP regressions are approximately the same, the prediction intervals are completely different. This is because of the way these two

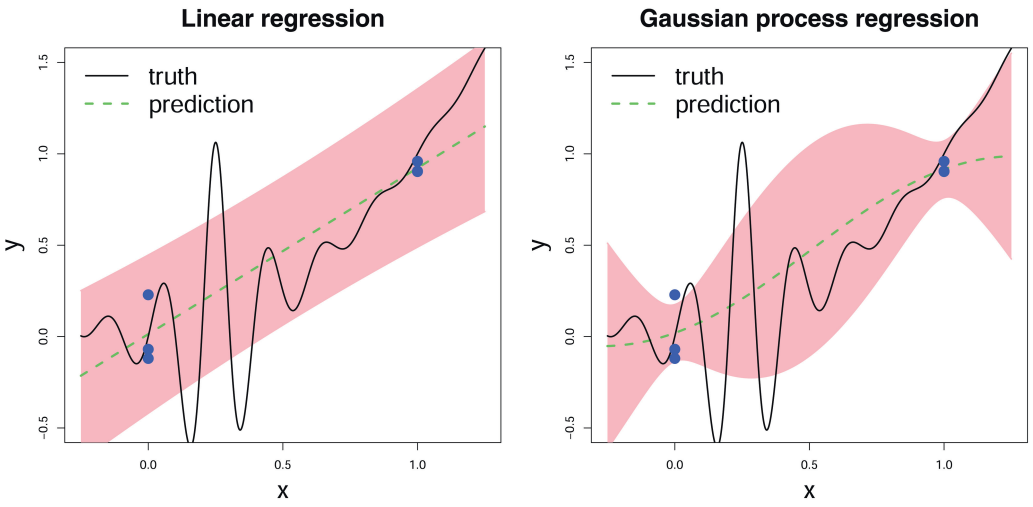


FIGURE 1.5: Comparison of linear and Gaussian process regression.

types of model quantify the prediction uncertainties, and it is important for us to understand their differences. The uncertainty of linear regression prediction includes the uncertainties in the estimated parameters ($\hat{\beta}_0$ and $\hat{\beta}_1$) and the random error (ϵ). The uncertainty is highest at the extremes ($x = 0$ and $x = 1$) and smallest at the middle ($x = 0.5$). On the other hand, GP regression is a nonparametric regression method. Therefore, its prediction uncertainty includes the uncertainties about the function $h(\cdot)$ and the random error. It has minimum uncertainty where we have observed the data and maximum uncertainty in the middle. This is completely opposite of linear regression!

Clearly, the uncertainty estimate from the GP regression makes more sense than that of the linear regression. The GP regression uncertainty estimate would suggest that we perform the next experiment in the middle, which will give us more information about the true function. On the other hand, the linear regression uncertainty estimate would suggest that we continue to perform experiments at extremes. This is because the linear regression uncertainty estimate is obtained by assuming that $y = \beta_0 + \beta_1 x$ is the true model. Thus, not being able to capture the uncertainties in the model is a major weakness of the linear regression approach.

Most of the experimental designs on physical experiments are developed by assuming linear regression models (Box et al., 1978; Montgomery, 2017; Wu and Hamada, 2021). Does this mean that they are not useful? Not really! When the number of factors (p) is high and when the number of experiments (n) is small, the GP regression will also give larger uncertainties at the boundaries coinciding with the results of the linear regression. Most physical experiments are of this kind (large p and small n); therefore, using a linear or GP regression framework would likely produce identical results. However, relying on the linear regression framework does impose some limitations in our quest for optimal designs.

Suppose that we have six continuous factors to experiment and a budget of about 30 experiments (also known as *runs*). One could consider a two-level fractional factorial design 2^{6-1}, which requires 32 runs. Alternatively, we could consider using a three-level fractional factorial design 3^{6-3} which requires 27 runs or a central composite design with 29 runs (12 axial points, 16 cube points, and 1 center point). The literature on experimental design is not clear on which of these designs should be used. This is because each of them assumes

a certain linear regression model and does not entertain the possibility of misspecifications of the model. The need to obtain designs that are robust to model misspecifications has long been recognized in the literature. However, the inability of linear regression models to fully capture the model uncertainties limits their usefulness. For example, one can obtain a 2^{6-1} minimum aberration design (Fries and Hunter, 1980), which is considered to be a model robust design (Cheng et al., 1999). However, it is robust only to a linear regression model that includes linear effects, and two-factor and higher-order interactions; it does not consider the possibility of nonlinear effects.

The foregoing discussion suggests that we would be better off developing the optimal design framework using GP regression than linear regression because of GP's ability to better quantify model uncertainties. This is the approach taken in this book.

1.4 Outline of the Book

Due to the reliance of GP models in many of the experimental design techniques discussed in the book, we will start with a gentle introduction to GP modeling in Chapter 2. However, before introducing GP, we start with the more familiar polynomial interpolation and then transition slowly to GP after a gentle introduction to radial basis functions and kriging. Extensions of these approaches to noisy data are also discussed, which become essential for the methods in Parts II and IV. Some important concepts in experimental design are also introduced during their discussion.

Part II of the book starts with GP-based experimental design methods. Chapter 3 is short because those methods are not very useful in practice because of their dependence on unknown hyperparameters of the GP model. However, Chapter 3 forms the basis for developing the more practical and useful experimental design methods in Chapters 4–7. Chapter 4 discusses some of the key concepts of the book on space-filling designs, which are motivated as the limiting cases of GP model-based designs, but also possess geometrical interpretations. They can be considered as model-robust designs, and thus useful for practical applications. Chapter 5 extends these ideas to non-uniform distributions, which lays the foundations for the data science methods discussed in Part IV. Chapter 6 discusses a very useful special case of space-filling design where the interest is in quickly screening factors. This chapter also introduces some useful concepts from global sensitivity analysis that form the basis for variable selection techniques discussed later in Chapter 11. Chapter 7 develops sequential design methods to achieve a specific objective, such as optimization, where the designs from Chapters 4 to 6 can be used as initial designs.

Chapter 8 in Part III on fractional factorial designs is kept intentionally short because of the availability of many excellent textbooks on this topic. However, this chapter can still be interesting to the readers because it tries to connect with many of the experimental design and modeling methods introduced in Parts I and II. Chapter 9 discusses the case where we have data from both computer and physical experiments.

Part IV shows the application of experimental design methods in subsampling (Chapter 10) and data analysis (Chapter 11). Experimental design is traditionally viewed in statistics as a technique for data collection and planning. This part is included in the book to create an appreciation of the broad applicability of design of experiments and reinforce it as a foundational topic of statistics.

2

Modeling Techniques

As mentioned in the Introduction, experiments will be performed to reduce our uncertainty about the response surface, and we need to use statistical modeling techniques to quantify the uncertainties. The main technique that we will use is Gaussian Process (GP) regression, but before explaining about it we will examine a few other closely related modeling concepts: polynomial models and radial basis functions from the numerical analysis literature and kriging from the geostatistics literature. As we will soon see, all of these modeling techniques will lead us to a very similar solution, but learning about these seemingly different ideas will give us new perspectives about the problem which will be useful to us in later chapters. To develop these ideas, first we will assume that the data are not corrupted by random error; that is, there is no ϵ in (1.1). This situation is common with deterministic computer models, such as the DFT in Figure 1.1 or the machining simulator in Figure 1.3. Because of the absence of random errors, we are looking for models that can interpolate the data. This is discussed in Section 2.1. Then, in Section 2.2, we will extend these interpolating models to deal with noisy data.

2.1 Interpolation

The interpolating models discussed in this section will be used extensively in Part II of the book on computer experiments.

2.1.1 Polynomial Interpolation

Consider a one-dimensional test function

$$h(x) = \frac{\sin(10\pi x)}{1 + 64(x - .25)^2} + x^2, \quad x \in [0, 1]. \tag{2.1}$$

We will assume that this function is unknown to the experimenter and has access to only some data. Suppose that the data are generated by doing $n = 10$ experiments, where the experimental design is an equi-spaced set of 10 points in $[0, 1]$. The test function (black dashed line) and the data $\{(x_i, y_i)\}_{i=1}^{10}$ (blue dots) are plotted on the left panel of Figure 2.1.

We can interpolate the n data points using an $(n-1)$th degree polynomial:

$$y = \beta_0 + \beta_1 x + \ldots + \beta_{n-1} x^{n-1}. \tag{2.2}$$

There are explicit solutions to one-dimensional polynomial interpolation such as the Lagrange interpolation formula. However, to connect the interpolation formula with the other methods discussed in this chapter, we will take a different approach.

DOI: 10.1201/9781003661535-2

This chapter has been made available under a CC-BY-NC-ND license 4.0.

First, rewrite the polynomial in (2.2) as

$$y = \beta_0 R_0(x) + \beta_1 R_1(x) + \ldots + \beta_{n-1} R_{n-1}(x), \tag{2.3}$$

where $R_i(x)$ is a polynomial of degree i with $R_0(x) \coloneqq 1$. For example, we can use the Chebyshev polynomials $R_1(x) = x$, $R_2(x) = 2x^2 - 1$, $R_3(x) = 4x^3 - 3x$, etc. for $x \in [-1, 1]$. The Chebyshev polynomials have an alternative expression: $R_i(x) = \cos\{i \cos^{-1}(x)\}$. It does not matter whether we use (2.2) or (2.3), but the latter will be more numerically stable. Then, to achieve interpolation, we must have

$$\sum_{j=0}^{n-1} \beta_j R_j(x_i) = y_i \quad \text{for} \quad i = 1, \ldots, n.$$

In matrix notation, this system of equations can be written as

$$\mathbf{R}\boldsymbol{\beta} = \mathbf{y},$$

where $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{n-1})'$, $\mathbf{y} = (y_1, \ldots, y_n)'$ and $\mathbf{R}$ is an $n \times n$ matrix with ijth element $R_j(x_i)$. Note that we use boldface letters to denote vectors and matrices, a convention that is followed throughout this book.

If $\mathbf{R}$ is a nonsingular matrix, then we can get a unique solution

$$\hat{\boldsymbol{\beta}} = \mathbf{R}^{-1}\mathbf{y}.$$

This gives us the polynomial interpolator $\hat{y}(x) = \mathbf{r}(x)'\hat{\boldsymbol{\beta}}$, where $\mathbf{r}(x) = (R_0(x), \ldots, R_{n-1}(x))'$. Thus,

$$\hat{y}(x) = \mathbf{r}(x)'\mathbf{R}^{-1}\mathbf{y}. \tag{2.4}$$

This interpolator is plotted in Figure 2.1 as a solid green line. We can see that it passes through the 10 points as expected out of an interpolator, but unfortunately the function does not seem to be a good approximation to the true function except for the middle region. The interpolator fluctuates wildly at the two ends, which is a known issue with high-degree polynomial interpolation called Runge's phenomenon.

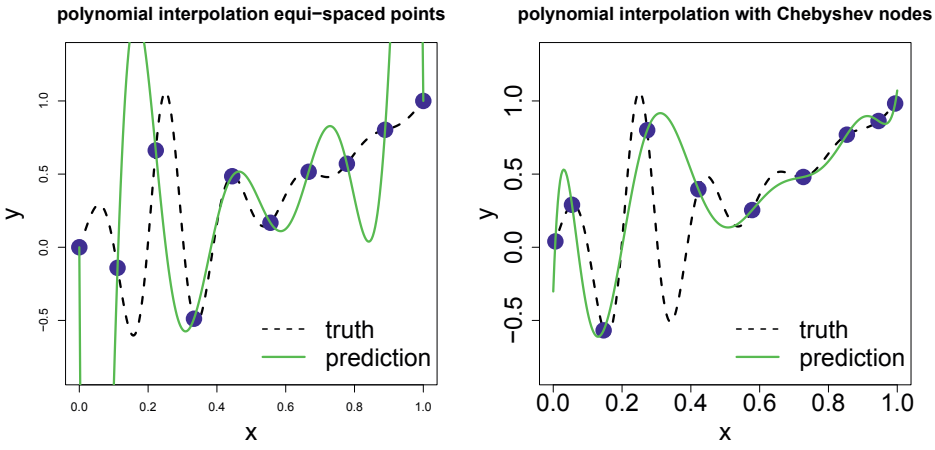


FIGURE 2.1: Polynomial interpolation with 10 equi-spaced (left) and 10 Chebyshev (right) design points.

We can overcome Runge's phenomenon and improve polynomial interpolation using a better experimental design. Let $h_{n-1}(x)$ be the $(n-1)$th degree polynomial that interpolates the data $\{(x_i, h(x_i))\}_{i=1}^n$ for $x_i \in [-1, 1]$, $i = 1, \ldots, n$. If $h(x)$ is differentiable at least n times, then its prediction error can be obtained as (Hildebrand, 1987, p. 63)

$$h(x) - h_{n-1}(x) = \frac{h^{(n)}(\xi)}{n!} \prod_{i=1}^{n} (x - x_i),$$

for some $\xi \in [-1, 1]$ where $h^{(n)}(x)$ is the nth derivative of $h(x)$. This gives an upper bound for the absolute error:

$$|h(x) - h_{n-1}(x)| \leq \frac{1}{n!} \max_{\xi} |h^{(n)}(\xi)| \left| \prod_{i=1}^{n} (x - x_i) \right|.$$

Chebyshev polynomial has the amazing property of minimizing the maximum absolute value of any polynomial of degree n after proper scaling. Therefore, the upper bound of the absolute error can be minimized by setting

$$\prod_{i=1}^{n} (x - x_i) = \frac{1}{2^{n-1}} \cos\{n \cos^{-1}(x)\},$$

where the scaling constant 2^{n-1} is used in the denominator to match the leading terms on both sides of the equation. This equation immediately suggests that x_i's should be the roots of the Chebyshev polynomial (also known as Chebyshev nodes). Thus, they can be obtained by solving

$$\cos\{n \cos^{-1}(x_i)\} = 0,$$

which gives the solution:

$$x_i = \cos\left\{\pi \frac{(i - .5)}{n}\right\} \quad \text{for } i = 1, \ldots, n. \tag{2.5}$$

The 10 experimental design points using Chebyshev nodes and the corresponding function evaluations are plotted in the right panel of Figure 2.1 (points are rescaled to $[0, 1]$ from $[-1, 1]$ using $x \leftarrow (x + 1)/2$). We can see that, compared to the equi-spaced design points, these design points are pushed more to the boundaries. The interpolator is also shown in the same figure as a green solid line, which provides a good approximation to the true function and a drastic improvement over the previous interpolator (left panel). This example clearly shows the importance of experimental design!

Although the use of Chebyshev nodes as the experimental design for polynomial interpolation in one-dimensional problems seems to work well, polynomial interpolation has an issue when there is more than one input variable. Consider the 2^2 full factorial design (Wu and Hamada, 2021) shown in the left panel of Figure 2.2. Note that the 2^2 design can be obtained by crossing the 2-point design on each variable: $(0, 1) \otimes (0, 1)$. We can obtain the interpolation by fitting a polynomial model of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2,$$

where the term $x_1 x_2$ represents the interaction between the two inputs. The terms in this polynomial can also be identified using a similar product representation: $(1, x_1) \otimes (1, x_2) = (1, x_1, x_2, x_1 x_2)$. Now consider a slight perturbation of two points shown in the right panel of the same figure. A polynomial to interpolate these four points is no longer obvious. In fact,

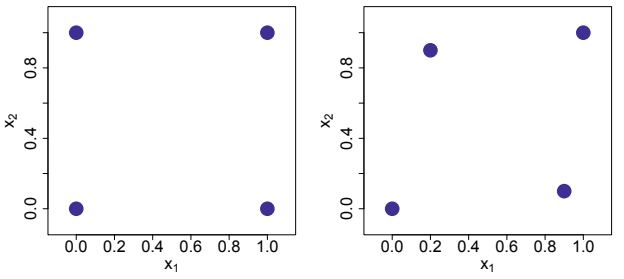


FIGURE 2.2: The left plot shows a 2^2 design and the right plot shows the same design by perturbing two of the points.

the interpolating polynomials may not be unique and may require specialized algorithms to identify them (Bates et al., 2003).

An approach to using polynomial models in higher-dimensional problems is to sample n points from each dimension using Chebyshev nodes and then form a full factorial design. But this would require n^p experiments, which is impractical when p is large. We can reduce the number of experiments using fractional factorial designs (Chapter 8) or sparse grids (Smolyak, 1963; Xiu, 2010; Plumlee, 2014), but such designs and the corresponding polynomials are not extensible for sequential designs (Chapter 7). In sequential experiments, we may want to add points in certain regions of the space without following any specialized structure for the design. This necessitates more flexible interpolation methods that can work on scattered data. Radial basis functions discussed in the next section is one such technique.

2.1.2 Radial Basis Functions

Given the design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and the corresponding function evaluations $y_i = h(\mathbf{x}_i)$, $i = 1, \ldots, n$, we are interested in obtaining an interpolator of the form

$$h_n(\mathbf{x}) = \sum_{i=1}^{n} \beta_i R_i(\mathbf{x}),$$

where $R_i(\mathbf{x})$ are some basis functions of $\mathbf{x}$, $i = 1, \ldots, n$. For one-dimensional polynomial interpolation, we took $R_1(x) = 1$, $R_2(x) = x$, ..., $R_n(x) = x^{n-1}$. However, the Mairhuber-Curtis theorem (Fasshauer, 2007, p. 3) tells us that we cannot fix the set of basis functions in advance for interpolating arbitrary scattered data in $\mathbb{R}^p$ for $p \geq 2$. This implies that the basis functions must depend on the data sites. Therefore, let

$$R_i(\mathbf{x}) = R\left(\mathbf{x} - \mathbf{x}_i\right),$$

where $R : \mathbb{R}^p \to \mathbb{R}$.

The interpolation condition gives rise to the following n equations

$$\sum_{j=1}^{n} \beta_j R(\mathbf{x}_i - \mathbf{x}_j) = y_i, \quad i = 1, \ldots, n.$$

In matrix notation, this system of equations can be written as $\mathbf{R}\boldsymbol{\beta} = \mathbf{y}$, where $\mathbf{R}$ is the $n \times n$ matrix with ijth element $R(\mathbf{x}_i - \mathbf{x}_j)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)'$, and $\mathbf{y} = (y_1, \ldots, y_n)'$. If $\mathbf{R}$ is

nonsingular, then we obtain the unique solution $\hat{\boldsymbol{\beta}} = \mathbf{R}^{-1}\mathbf{y}$. This gives us the radial basis function (RBF) interpolator

$$\hat{y}(\mathbf{x}) = \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{y}, \tag{2.6}$$

where $\mathbf{r}(\mathbf{x})$ is an $n \times 1$ vector with ith element $R(\mathbf{x} - \mathbf{x}_i)$. When $p = 1$, this is exactly the same as the polynomial interpolator in (2.4) except that we use radial basis functions instead of polynomial basis functions. However, unlike the polynomial interpolator, the RBF interpolator works for any p and does not require any structure for the design D.

Before proceeding further, we need to answer an important question. Is $\mathbf{R}$ invertible for any choice of $R(\cdot)$ and any design D? It turns out that the exact mathematical conditions on $R(\cdot)$ to make $\mathbf{R}$ invertible are still unknown. However, characterizations on $R(\cdot)$ are available to make $\mathbf{R}$ a positive definite matrix, and we know that a positive definite matrix can be inverted. For this, we need to understand the notion of a positive definite function (Fasshauer, 2007, Ch. 3)[1].

Definition 2.1 *A continuous function $R : \mathbb{R}^p \to \mathbb{R}$ is positive semi-definite on $\mathbb{R}^p$ if*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j R(\mathbf{x}_i - \mathbf{x}_j) \geq 0 \tag{2.7}$$

for any design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and for all $\mathbf{c} = (c_1, \ldots, c_n)'$. The function $R(\cdot)$ is called positive definite if (2.7) is 0 only for $\mathbf{c} = \mathbf{0}$.

Thus, if we choose $R(\cdot)$ to be a positive definite function, then $\mathbf{R}$ will be invertible provided that the data sites $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct. The literature on RBF further restricts $R(\cdot)$ to be a radial function (Fasshauer, 2007, Ch. 2).

Definition 2.2 *A function $R : \mathbb{R}^p \to \mathbb{R}$ is called radial provided there exists a function $\psi : \mathbb{R}^+ \to \mathbb{R}$ such that $R(\mathbf{x}) = \psi(\|\mathbf{x}\|)$, where $\|\cdot\|$ is usually the Euclidean norm.*

There are many choices for the radial basis functions, and we refer to the book by Fasshauer (2007) for details. An example is the Gaussian function

$$R(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}\|^2}{\theta^2}\right), \tag{2.8}$$

which we will use a lot in this book. This function has an unknown parameter θ (known as length-scale) which needs to be tuned based on the data to get good prediction.

Consider again the one-dimensional example in (2.1). Let D be $n = 10$ equi-spaced points in $[0, 1]$, i.e., $D = \{(i-1)/(n-1)\}_{i=1}^{n}$. Suppose that we use the Gaussian function in (2.8) as the radial basis function. Figure 2.3 shows the RBF interpolation for three different choices of θ. We can see that the RBF interpolation looks very good when $\theta = 0.1$, but not so good with $\theta = 1$ (similar to the left panel of Figure 2.1) and very poor with $\theta = 0.01$. In summary, RBF interpolation is very flexible and can give excellent results, but it comes with the additional complexity of properly tuning the length-scale parameter in the radial basis function.

One approach to tuning the length-scale parameter in the radial basis function is to use cross-validation. The leave-one-out cross-validation error is given by

$$cv_i = y_i - \hat{y}_{\sim i}(\mathbf{x}_i),$$

[1]To be consistent with the nomenclature used in the statistical literature, we use positive semi-definite and positive definite instead of positive definite and strictly positive definite, respectively.

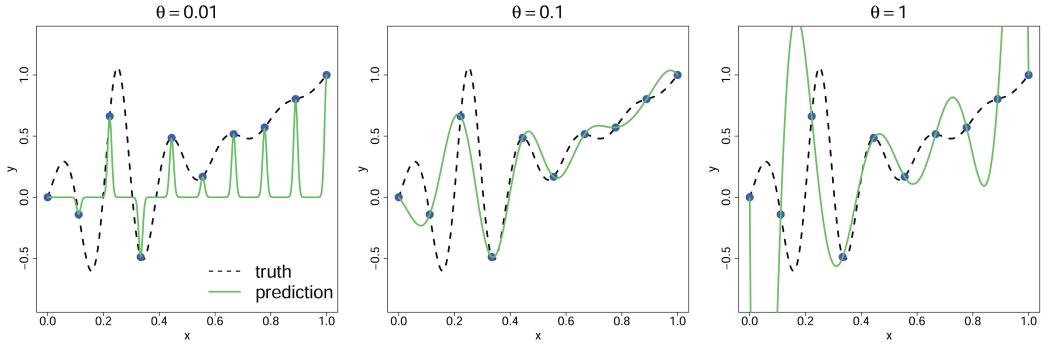


FIGURE 2.3: RBF interpolation with 10 equi-spaced points and three choices of θ in the Gaussian radial basis function.

where $\hat{y}_{\sim i}(\mathbf{x}_i)$ is the prediction at $\mathbf{x} = \mathbf{x}_i$ from the RBF interpolant that does not use the ith data point. Thus, $\hat{y}_{\sim i}(\mathbf{x}_i) = \mathbf{r}_{\sim i}(\mathbf{x}_i)'(\mathbf{R}_{\sim i})^{-1}\mathbf{y}_{\sim i}$, where $\mathbf{r}_{\sim i}$ denotes the vector $\mathbf{r}$ without its ith element (similarly $\mathbf{y}_{\sim i}$) and $\mathbf{R}_{\sim i}$ is the matrix $\mathbf{R}$ without its ith row and column. Taking the inverse of an $n \times n$ matrix requires $\mathcal{O}(n^3)$ computations. Therefore, computing cv_i for $i = 1, \ldots, n$ would require $\mathcal{O}(n^4)$ computations, which can be quite expensive even for moderately large n. Fortunately, there is an easy way to compute them in $\mathcal{O}(n^3)$ (Mitchell and Morris, 1992). The following results on partitioned matrices are useful (Petersen and Pedersen, 2006, p. 46), which we will use many times in the book.

Lemma 2.1 *Assuming the dimension of the blocks match*

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1} \\ -\mathbf{Q}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{Q}^{-1} \end{bmatrix}$$

$$\textit{and} \quad \begin{vmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{vmatrix} = |\mathbf{A}|\,|\mathbf{Q}|,$$

where $\mathbf{Q} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$.

Omitting the argument $\mathbf{x}_i$ from $\mathbf{r}_{\sim i}(\mathbf{x}_i)$ for notational simplicity, we obtain

$$\begin{aligned} \mathbf{R}^{-1} &= \begin{bmatrix} \mathbf{R}_{\sim i} & \mathbf{r}_{\sim i} \\ \mathbf{r}'_{\sim i} & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} (\mathbf{R}_{\sim i})^{-1} + (\mathbf{R}_{\sim i})^{-1}\mathbf{r}_{\sim i}\mathbf{r}'_{\sim i}(\mathbf{R}_{\sim i})^{-1}/Q & -(\mathbf{R}_{\sim i})^{-1}\mathbf{r}_{\sim i}/Q \\ -\mathbf{r}'_{\sim i}(\mathbf{R}_{\sim i})^{-1}/Q & 1/Q \end{bmatrix}, \end{aligned}$$

where $Q = 1 - \mathbf{r}'_{\sim i}(\mathbf{R}_{\sim i})^{-1}\mathbf{r}_{\sim i}$. Looking at the last row of the matrix, it is easy to see that

$$\mathbf{r}'_{\sim i}(\mathbf{R}_{\sim i})^{-1} = -\frac{(\mathbf{R}^{-1})_{i\sim i}}{(\mathbf{R}^{-1})_{ii}},$$

where $(\mathbf{R}^{-1})_{i\sim i}$ is the ith row of $\mathbf{R}^{-1}$ without its ith element. Therefore,

$$\mathbf{r}'_{\sim i}(\mathbf{R}_{\sim i})^{-1}\mathbf{y}_{\sim i} = -\frac{(\mathbf{R}^{-1})_{i\sim i}}{(\mathbf{R}^{-1})_{ii}}\mathbf{y}_{\sim i} = -\frac{(\mathbf{R}^{-1})_i\mathbf{y}}{(\mathbf{R}^{-1})_{ii}} + y_i.$$

Thus, we obtain

$$cv_i = \frac{(\mathbf{R}^{-1})_i\mathbf{y}}{(\mathbf{R}^{-1})_{ii}}. \tag{2.9}$$

This short-cut formula allows us to compute cv_i for all $i = 1, \ldots, n$ with only one inversion of the $\mathbf{R}$ matrix.

For the example in Figure 2.3, by minimizing the mean-squared cross-validation error

$$MSCV = \frac{1}{n}\sum_{i=1}^{n} cv_i^2, \tag{2.10}$$

we obtain $\hat{\theta} = 0.08$, which will give a result very close to the middle panel in Figure 2.3 that is very good.

A disadvantage of RBF interpolation is that it does not provide an uncertainty estimate for the prediction. Uncertainty estimation is important for us to decide where to do the next experiment. In the next section, we develop a probabilistic formulation of the problem which will provide identical predictions to those of RBF but with uncertainty estimates.

2.1.3 Kriging

Kriging is a multivariate interpolation technique named after Krige (1951) which was developed into the field of geostatistics by the pioneering work of Matheron (1963). See Cressie (1990) for its origins. Applications of kriging to spatial statistics problems are discussed in Cressie (1993) and its theoretical foundations in Stein (1999). The technique was brought to the field of computer experiments by Sacks et al. (1989).

The main idea of kriging is to view the deterministic function $h(\mathbf{x})$ as a realization of a stochastic process $Y(\mathbf{x})$. Let the stochastic process has a constant mean μ and covariance function $K(\mathbf{u}, \mathbf{v}) = cov\{Y(\mathbf{u}), Y(\mathbf{v})\}$. For a second-order stationary stochastic process, the covariance function can be written as

$$K(\mathbf{u}, \mathbf{v}) = \tau^2 R(\mathbf{u} - \mathbf{v}),$$

where τ^2 is the variance and $R(\cdot)$ the correlation function. What correlation functions can we use? Since $cor\{Y(\mathbf{u}), Y(\mathbf{u})\} = 1$, we must have $R(\mathbf{0}) = 1$. Moreover, since $cor\{Y(\mathbf{u}), Y(\mathbf{v})\} = cor\{Y(\mathbf{v}), Y(\mathbf{u})\}$, we must have $R(\mathbf{u} - \mathbf{v}) = R(\mathbf{v} - \mathbf{u})$. Thus, $R(\cdot)$ must be an even function. Finally, $var\{\mathbf{c'Y}\} \geq 0$ for any $\mathbf{c}$, i.e., we must have $\mathbf{c'Rc} \geq 0$. This implies that $R(\cdot)$ should be a positive semi-definite function. There are many functions that satisfy these conditions; Gaussian function in (2.8) is being one of them. However, there is no need to use functions that are radial (or isotropic). In fact, having different length-scale parameter for each input can be very advantageous. It can be shown that products of positive definite functions are also positive definite, which follows from a result in matrix algebra that Hadamard products of positive definite matrices are positive definite (Fasshauer, 2007, p. 30). Therefore, define the Gaussian product correlation function as

$$R(\boldsymbol{\Delta}) = \prod_{i=1}^{p} e^{-\Delta_i^2/\theta_i^2} = e^{-\sum_{i=1}^{p} \Delta_i^2/\theta_i^2}. \tag{2.11}$$

Some commonly used one-dimensional correlation functions are given in Table 2.1. Multidimensional correlation functions can be constructed from these one-dimensional correlation functions by taking products as in (2.11) or replacing Δ with a weighted Euclidean norm $\|\boldsymbol{\Delta}\|_{\boldsymbol{\theta}} = \sqrt{\sum_{i=1}^{p}(\Delta_i/\theta_i)^2}$. The choice of correlation function dictates the smoothness of the stochastic process. For example, the Gaussian correlation function leads to infinitely differentiable sample paths, whereas the sample paths with exponential correlation function are nowhere differentiable. See Rasmussen and Williams (2006, Ch. 4) and Fasshauer and McCourt (2015, Ch. 3) for details.

TABLE 2.1: Some commonly used correlation functions.

Name	$R(\Delta)$	Parameters
Gaussian	$e^{-\Delta^2/\theta^2}$	$\theta > 0$
Exponential	$e^{-\lvert\Delta\rvert/\theta}$	$\theta > 0$
Power exponential	$e^{-(\lvert\Delta\rvert/\theta)^\alpha}$	$\theta > 0$, $0 < \alpha \leq 2$
Matérn 3/2	$\left(1+\sqrt{3}\frac{\lvert\Delta\rvert}{\theta}\right)e^{-\sqrt{3}\lvert\Delta\rvert/\theta}$	$\theta > 0$
Matérn 5/2	$\left(1+\sqrt{5}\frac{\lvert\Delta\rvert}{\theta}+\frac{5}{3}\frac{\Delta^2}{\theta^2}\right)e^{-\sqrt{5}\lvert\Delta\rvert/\theta}$	$\theta > 0$
Rational quadratic	$\left(1+\frac{\Delta^2}{\theta^2}\right)^{-\alpha}$	$\theta > 0$, $\alpha > 0$

Now given the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we can form a linear predictor

$$\hat{y}(\mathbf{x}) = \mathbf{a}(\mathbf{x})'\mathbf{y}$$

and try to find the best linear predictor by minimizing the mean squared prediction error:

$$MSE(\mathbf{x}) = E\left\{\mathbf{a}(\mathbf{x})'\mathbf{Y} - Y(\mathbf{x})\right\}^2, \tag{2.12}$$

where the expectation is taken with respect to the stochastic process. Note that by taking a frequentist stance, we have replaced the observed data vector $\mathbf{y}$ by its random version $\mathbf{Y} = (Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n))'$. The linear predictor will be unbiased if

$$E\left\{\mathbf{a}(\mathbf{x})'\mathbf{Y}\right\} = \mu \text{ for all } \mathbf{x}.$$

Since $E(\mathbf{Y}) = \mu\mathbf{1}$, where $\mathbf{1}$ is a vector of n 1s, the unbiasedness condition reduces to $\mathbf{a}(\mathbf{x})'\mathbf{1} = 1$. Now the MSE can be computed as

$$\begin{aligned} MSE(\mathbf{x}) &= var\left\{\mathbf{a}(\mathbf{x})'\mathbf{Y} - Y(\mathbf{x})\right\} \\ &= \mathbf{a}(\mathbf{x})'var\{\mathbf{Y}\}\mathbf{a}(\mathbf{x}) - 2\mathbf{a}(\mathbf{x})'cov\{\mathbf{Y}, Y(\mathbf{x})\} + \tau^2 \\ &= \tau^2\mathbf{a}(\mathbf{x})'\mathbf{R}\mathbf{a}(\mathbf{x}) - 2\tau^2\mathbf{a}(\mathbf{x})'\mathbf{r}(\mathbf{x}) + \tau^2, \end{aligned}$$

where the matrix $\mathbf{R}$ and the vector $\mathbf{r}(\mathbf{x})$ are given by

$$\mathbf{R} = \begin{array}{c} \\ 1 \\ \vdots \\ i \\ \vdots \\ n \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 1 & \cdots & j & \cdots & n \end{array} \\ \begin{bmatrix} 1 & & \cdots & & \\ & & & & \\ \vdots & & & & \vdots \\ & & R(\mathbf{x}_i - \mathbf{x}_j) & & \\ \vdots & & & & 1 \end{bmatrix} \end{array} \quad \text{and} \quad \mathbf{r}(\mathbf{x}) = \begin{array}{c} \\ 1 \\ \vdots \\ i \\ \vdots \\ n \end{array}\!\!\begin{pmatrix} \\ \vdots \\ \\ R(\mathbf{x} - \mathbf{x}_i) \\ \\ \vdots \end{pmatrix}. \tag{2.13}$$

Thus, the optimization problem is to

$$\begin{aligned} &\min_{\mathbf{a}(\mathbf{x})} \ \mathbf{a}(\mathbf{x})'\mathbf{R}\mathbf{a}(\mathbf{x}) - 2\mathbf{a}(\mathbf{x})'\mathbf{r}(\mathbf{x}) + 1 \\ &\text{subject to } \mathbf{a}(\mathbf{x})'\mathbf{1} = 1. \end{aligned}$$

This can be solved using Lagrange multiplier method (the argument $\mathbf{x}$ is omitted for simplicity):

$$\min L = \mathbf{a}'\mathbf{R}\mathbf{a} - 2\mathbf{a}'\mathbf{r} + 1 + 2\lambda\{\mathbf{a}'\mathbf{1} - 1\}.$$

Differentiating L with respect to $\mathbf{a}$ and equating to 0, we obtain

$$\frac{\partial L}{\partial \mathbf{a}} = 2\mathbf{R}\mathbf{a} - 2\mathbf{r} + 2\lambda\mathbf{1} = 0,$$

which gives $\mathbf{a} = \mathbf{R}^{-1}(\mathbf{r} - \lambda\mathbf{1})$. Substituting this in the constraint and solving for λ, we obtain $\lambda = (\mathbf{r}'\mathbf{R}^{-1}\mathbf{1} - 1)/\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}$. Thus,

$$\hat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{r} - \frac{(\mathbf{r}'\mathbf{R}^{-1}\mathbf{1} - 1)}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\mathbf{R}^{-1}\mathbf{1}.$$

Thus, the best linear unbiased predictor (BLUP) is given by

$$\begin{aligned}\hat{y}(\mathbf{x}) &= \hat{\mathbf{a}}(\mathbf{x})'\mathbf{y} \\ &= \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{y} - \frac{(\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{1} - 1)}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\mathbf{1}'\mathbf{R}^{-1}\mathbf{y} \\ &= \hat{\mu} + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}), \end{aligned} \tag{2.14}$$

where

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}. \tag{2.15}$$

The estimate of μ in (2.15) can be readily seen as the generalized least squares (GLS) estimate when $E(\mathbf{y}) = \mu\mathbf{1}$ and $var(\mathbf{y}) = \tau^2\mathbf{R}$. Thus, the predictor in (2.14) can be viewed as (i) estimate μ from data using GLS, (ii) obtain the residuals $\mathbf{y} - \hat{\mu}\mathbf{1}$, and (iii) interpolate the residuals and add it back to $\hat{\mu}$.

The predictor (2.14) is known as *ordinary kriging*. Note that

$$\begin{aligned}\mathbf{I} = \mathbf{R}^{-1}\mathbf{R} &= \mathbf{R}^{-1}\big[\mathbf{r}(\mathbf{x}_1) \quad \cdots \quad \overset{i}{\mathbf{r}(\mathbf{x}_i)} \quad \cdots \quad \mathbf{r}(\mathbf{x}_n)\big] \\ &= \big[\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_1) \quad \cdots \quad \mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_i) \quad \cdots \quad \mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_n).\big]\end{aligned}$$

Thus, $\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_i) = \mathbf{I}_i$ for $i = 1, \ldots, n$, where $\mathbf{I}_i$ is the ith column of the identity matrix. Therefore,

$$\begin{aligned}\hat{y}(\mathbf{x}_i) &= \hat{\mu} + \mathbf{I}_i'(\mathbf{y} - \hat{\mu}\mathbf{1}) \\ &= \hat{\mu} + (y_i - \hat{\mu}) = y_i, \ \forall i.\end{aligned}$$

Therefore, the ordinary kriging predictor is an interpolator.

Substituting the ordinary kriging predictor in the expression for MSE and simplifying, we obtain

$$MSE(\mathbf{x}) = \tau^2\left\{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) + \frac{\{\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{1} - 1\}^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right\}, \tag{2.16}$$

which quantifies the uncertainty of the prediction. Since $\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_i) = \mathbf{I}_i$ and $R(\mathbf{0}) = 1$,

$$\begin{aligned}MSE(\mathbf{x}_i) &= \tau^2\left\{1 - \mathbf{r}(\mathbf{x})'\mathbf{I}_i + \frac{\{\mathbf{I}_i'\mathbf{1} - 1\}^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right\} \\ &= \tau^2\left\{1 - R(\mathbf{x}_i - \mathbf{x}_i) + \frac{\{1 - 1\}^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right\} = 0, \ \forall i.\end{aligned}$$

That is, there is no uncertainty at the observed locations. This makes sense because we observed the data at these locations with no error.

Consider again the one-dimensional example in (2.1) with $D = \{(i-1)/(n-1)\}_{i=1}^{n}$ and $n = 10$. We again need to specify the correlation parameter θ before a prediction can be made. We can again use cross-validation methods as in Section 2.1.2. We can also use maximum likelihood-based methods if we make an assumption about the distribution of the stochastic process. We will take this approach in the next section. So, for now, let us use $\hat{\theta} = 0.08$ as we obtained earlier by minimizing MSCV. The ordinary kriging predictor is shown in Figure 2.4 along with 95% prediction intervals $\pm 2\sqrt{MSE(x)}$ (the factor 2 is obtained by assuming normality, which will become clearer in the next section). The predictions are similar to those of the RBF interpolator, but here we also have their uncertainty estimates. As expected, the uncertainty reduces as the prediction point approaches the observed points and increases as the prediction point moves away from the observed points.

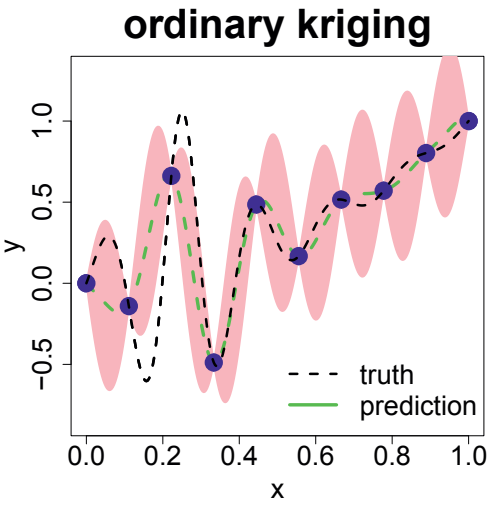


FIGURE 2.4: Ordinary kriging for the one-dimensional example. The 95% prediction interval is shown as a shaded band.

2.1.4 Gaussian Process Modeling

Gaussian process (GP) provides a Bayesian interpretation of the kriging methodology presented in the previous section. It was brought to the field of computer experiments by Currin et al. (1991), but it has been used previously in other settings (Wahba, 1978; O'Hagan, 1978). The book by Rasmussen and Williams (2006) was influential for its application in machine learning. The use of GPs for computer experiments is detailed in the books of Santner et al. (2018) and Gramacy (2020).

Definition 2.3 *A random function $Y(\mathbf{x})$ is a Gaussian process if for any choice of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and any $n \geq 1$, $\mathbf{Y} = (Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n))'$ has a multivariate normal distribution.*

Recall that a n-dimensional multivariate normal distribution is determined by its mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$. Thus, if $\mathbf{y} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then it has the density

$$f(\mathbf{y}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right\}.$$

GP can be viewed as a continuous extension of a multivariate normal distribution, which is determined by a mean function $\mu(\mathbf{x})$ and a covariance function or kernel $K(\mathbf{x}, \cdot)$, where $\mu(\mathbf{x}) = E\{Y(\mathbf{x})\}$ and $K(\mathbf{x}, \mathbf{u}) = cov\{Y(\mathbf{x}), Y(\mathbf{u})\}$.

We are interested in estimating the true input-output relationship $y = h(\mathbf{x})$ from the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. In a Bayesian approach, we would proceed by postulating a prior distribution for the function $h(\mathbf{x})$ and then find the posterior distribution of $h(\mathbf{x})$ given the data. The posterior distribution will give us an improved understanding of the function $h(\mathbf{x})$ compared to our prior understanding of it before seeing the data. We will use GP as a prior distribution of $h(\mathbf{x})$, that is,

$$h(\mathbf{x}) \sim GP(\mu(\mathbf{x}), K(\mathbf{x}, \cdot)). \tag{2.17}$$

For most of the developments in this book, we will assume that $\mu(\mathbf{x}) = \mu$ (a constant) and $K(\mathbf{x}, \mathbf{u}) = \tau^2 R(\mathbf{x} - \mathbf{u})$, where τ^2 is the variance and $R(\cdot)$ the correlation function. A few realizations of the GP for a one-dimensional input variable with $R(x) = e^{-x^2/\theta^2}$ and $\theta = 0.08$ are shown in the left panel of Figure 2.5. Suppose we evaluated the function and obtained the data, which are shown in the middle panel of the same figure as blue dots. The aim is to obtain the posterior distribution of the function given the data.

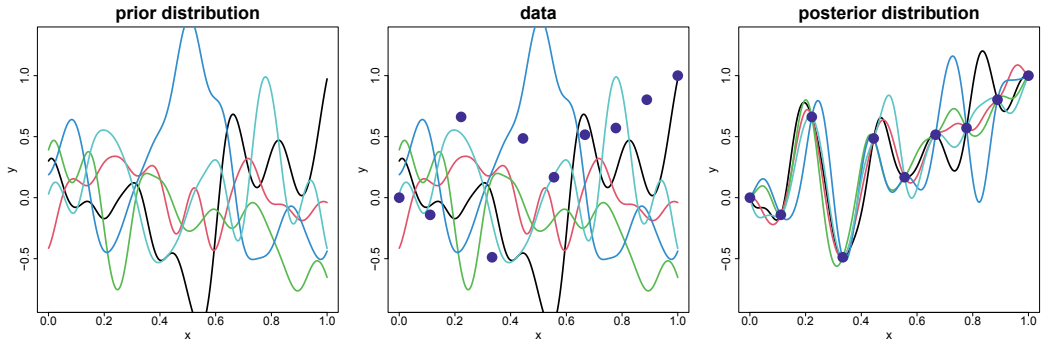


FIGURE 2.5: Five realizations from prior GP (left), data overlayed on the prior (middle), and five realizations from the posterior GP (right).

The following result from probability theory will be useful for deriving the posterior distribution.

Lemma 2.2 *Suppose* $\mathbf{y} = (\mathbf{y}_1', \mathbf{y}_2')'$, *where* $\mathbf{y}_1 \in \mathbb{R}^{n_1}$ *and* $\mathbf{y}_2 \in \mathbb{R}^{n_2}$. *If*

$$\begin{pmatrix}\mathbf{y}_1 \\ \mathbf{y}_2\end{pmatrix} \sim \mathcal{N}_{n_1+n_2}\left(\begin{pmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right),$$

then

$$\mathbf{y}_1|\mathbf{y}_2 \sim \mathcal{N}_{n_1}\left(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{y}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\right).$$

For the moment, assume that all the parameters of the model μ, τ^2, and the parameters in the correlation function ($\boldsymbol{\theta}$) are known. Then, for a fixed $\mathbf{x}$, (2.17) gives

$$\begin{pmatrix} h(\mathbf{x}) \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}_{1+n}\left(\begin{pmatrix} \mu \\ \mu\mathbf{1} \end{pmatrix}, \begin{bmatrix} \tau^2 & \tau^2\mathbf{r}(\mathbf{x})' \\ \tau^2\mathbf{r}(\mathbf{x}) & \tau^2\mathbf{R} \end{bmatrix}, \right)$$

where $\mathbf{r}(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^n$ and $\mathbf{R} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$ as in (2.13). Now using Lemma 2.2, we obtain the posterior distribution of $h(\mathbf{x})$ for a fixed $\mathbf{x}$:

$$h(\mathbf{x})|\mathbf{y} \sim \mathcal{N}\left(\mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}), \tau^2\{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}\right). \tag{2.18}$$

If we need a point estimate of $h(\mathbf{x})$, we can use the posterior mean $\hat{y}(\mathbf{x}) = \mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1})$, which is the same as the kriging predictor in (2.14) except that μ is assumed to be known here. The variance of the prediction $s^2(\mathbf{x}) = \tau^2\{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}$ can be used to quantify the uncertainty, which coincides with the $MSE(\mathbf{x})$ of ordinary kriging in (2.16) except for the last term. We will soon see that the extra term in (2.16) is the additional uncertainty in the prediction due to not knowing the value of μ. Thus, we can obtain the $(1-\alpha)$ credible interval as $\hat{y}(\mathbf{x}) \pm z_{1-\alpha/2}s(\mathbf{x})$, where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$ upper quantile of a standard normal distribution.

Eq. (2.18) gives the posterior distribution of $h(\mathbf{x})$ for a particular value of $\mathbf{x}$. We can similarly find the posterior distribution of $h(\mathbf{x})$ for a collection of $\mathbf{x}$ values, which will show that the posterior distribution for the whole function $h(\mathbf{x})$ is a GP:

$$h(\mathbf{x})|\mathbf{y} \sim GP\left(\mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}), \tau^2\{R(\mathbf{x} - \cdot) - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\cdot)\}\right). \tag{2.19}$$

Some realizations of this GP with $\theta = 0.08$ are also plotted in Figure 2.5 (right panel). We can see that all these realizations pass through the data points, and the overall uncertainty about $h(\mathbf{x})$ has been substantially reduced compared to the prior distribution.

The (hyper) parameters μ, τ^2, and $\boldsymbol{\theta}$ must be specified before GP can be used for prediction. The Bayesian formulation gives us a principled way to estimate them. One simple approach is to use empirical Bayes methods. The marginal distribution of the data is given by

$$\mathbf{y}|\mu, \tau^2, \boldsymbol{\theta} \sim \mathcal{N}_n(\mu\mathbf{1}, \tau^2\mathbf{R}).$$

Thus, the likelihood of the parameters is given by

$$L = \frac{1}{(2\pi\tau^2)^{n/2}|\mathbf{R}|^{1/2}} \exp\left\{-\frac{1}{2\tau^2}(\mathbf{y} - \mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1})\right\}.$$

The log-likelihood is

$$\log L = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\tau^2 - \frac{1}{2}\log|\mathbf{R}| - \frac{1}{2\tau^2}(\mathbf{y} - \mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}).$$

Maximizing $\log L$ with respect to μ and τ^2 gives

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}, \tag{2.20}$$

$$\hat{\tau}^2 = \frac{1}{n}(\mathbf{y} - \hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}). \tag{2.21}$$

The $\log L$ is a highly nonlinear function of the correlation parameters $\boldsymbol{\theta}$, and therefore, it is not easy to obtain an explicit expression for it. We can substitute for μ and τ^2 in $\log L$ and obtain the profile log-likelihood of $\boldsymbol{\theta}$ and can get an estimate of $\boldsymbol{\theta}$ by numerically maximizing it. Equivalently

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{n\log\hat{\tau}^2 + \log|\mathbf{R}|\right\}. \tag{2.22}$$

An alternative to empirical Bayes is hierarchical Bayes, where we can put priors on all the hyper-parameters and obtain their posterior distribution. However, performing a fully Bayesian inference can be computationally intensive because Markov chain Monte Carlo (MCMC)-based algorithms will be necessary to obtain the posterior distribution of $\boldsymbol{\theta}$. We will illustrate the hierarchical Bayes approach by placing a prior only on μ. For a more complete treatment, see Santner et al. (2018, Ch. 4).

Assume a non-informative prior for μ:

$$p(\mu) \propto 1.$$

Then,

$$\begin{aligned} p(\mu|\mathbf{y}) &\propto p(\mathbf{y}|\mu)p(\mu) \\ &\propto \exp\left\{-\frac{1}{2\tau^2}(\mathbf{y}-\mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1})\right\}. \end{aligned}$$

The following generalization of the Analysis of Variance (ANOVA) decomposition will be useful for our derivations.

Lemma 2.3 *Let* $\hat{\mu} = \mathbf{1}'\mathbf{R}^{-1}\mathbf{y}/\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}$. *Then,*

$$(\mathbf{y}-\mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1}) = (\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}) + (\mu-\hat{\mu})^2\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}.$$

Lemma 2.3 shows that

$$\begin{aligned} p(\mu|\mathbf{y}) &\propto \exp\left\{-\frac{1}{2\tau^2}[(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}) + (\mu-\hat{\mu})^2\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}]\right\} \\ &\propto \exp\left\{-\frac{1}{2\tau^2}(\mu-\hat{\mu})^2\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}\right\}. \end{aligned}$$

Thus,

$$\mu|\mathbf{y} \sim \mathcal{N}\left(\hat{\mu}, \frac{\tau^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right). \tag{2.23}$$

Now

$$p(h(\mathbf{x})|\mathbf{y}) = \int p(h(\mathbf{x})|\mathbf{y},\mu)\; p(\mu|\mathbf{y})d\mu.$$

Because both (2.18) and (2.23) are normals, this integration will give us another normal distribution:

$$h(\mathbf{x})|\mathbf{y} \sim \mathcal{N}\left(\hat{y}(\mathbf{x}), s^2(\mathbf{x})\right), \tag{2.24}$$

where we can obtain the expressions for the mean and variance as

$$\begin{aligned} \hat{y}(\mathbf{x}) &= E[E\{h(\mathbf{x})|\mathbf{y},\mu\}|\mathbf{y}] \\ &= E[\mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1})|\mathbf{y}] \\ &= \hat{\mu} + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}), \end{aligned} \tag{2.25}$$

$$\begin{aligned} s^2(\mathbf{x}) &= E[var\{h(\mathbf{x})|\mathbf{y},\mu\}|\mathbf{y}] + var[E\{h(\mathbf{x})|\mathbf{y},\mu\}|\mathbf{y}] \\ &= E[\tau^2\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}|\mathbf{y}] + var[\mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1})|\mathbf{y}] \\ &= \tau^2\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\} + \{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{1}\}^2 var(\mu|\mathbf{y}) \\ &= \tau^2\left\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) + \frac{\{\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{1}-1\}^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right\}. \end{aligned} \tag{2.26}$$

We can see that the expression for $s^2(\mathbf{x})$ is now identical to the $MSE(\mathbf{x})$ of ordinary kriging, and the last term that was missing in (2.18) captures the uncertainty of estimating μ.

The remaining parameters τ^2 and θ can be estimated again using empirical Bayes. The marginal likelihood is

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}|\mu)p(\mu)d\mu \\ &= \int \frac{1}{(2\pi\tau^2)^{n/2}|\mathbf{R}|^{1/2}} \exp\left\{-\frac{1}{2\tau^2}(\mathbf{y}-\mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1})\right\} d\mu \\ &= \frac{\exp\left\{-\frac{1}{2\tau^2}(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1})\right\}}{(2\pi\tau^2)^{n/2}|\mathbf{R}|^{1/2}} \int \exp\left\{-\frac{1}{2\tau^2}(\mu-\hat{\mu})^2\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}\right\} d\mu \\ &= \frac{\exp\left\{-\frac{1}{2\tau^2}(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1})\right\}}{(2\pi\tau^2)^{(n-1)/2}|\mathbf{R}|^{1/2}\sqrt{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}}, \end{aligned}$$

where we have again used Lemma 2.3 for simplification. Thus, τ^2 and $\boldsymbol{\theta}$ can be obtained as

$$\hat{\tau}^2 = \frac{1}{n-1}(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}) \tag{2.27}$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{n\log\hat{\tau}^2 + \log|\mathbf{R}| + \log(\mathbf{1}'\mathbf{R}^{-1}\mathbf{1})\right\}. \tag{2.28}$$

These are very similar to (2.21) and (2.22), and their differences are usually minor. Our R package `rkriging` (Huang and Joseph, 2024) uses the hierarchical Bayes formulas, which gives estimates $\hat{\mu} = 0.36$, $\hat{\tau}^2 = 0.215$, and $\hat{\theta} = 0.067$.

2.2 Regression

Now we will consider the case of data corrupted by random noise. Noise is common in physical experiments (Wu and Hamada, 2021) and stochastic simulations (Kleijnen, 2008). For such noisy data, we are looking for models that approximate rather than interpolate. These techniques will be used extensively in Parts III and IV of the book.

2.2.1 Linear Regression

Linear regression is one of the main data analysis tools in statistics and there are many excellent textbooks on this subject (Faraway, 2015, 2016). Therefore, we will only give a gentle introduction here so as to introduce some notation and concepts useful for developing experimental designs.

Consider again the same one-dimensional function as in (2.1), but now the data are generated by

$$y_i = h(x_i) + \epsilon_i,\ i = 1, \ldots, n, \tag{2.29}$$

where ϵ_i is the random error caused by changes in the other unobserved variables or due to measurement errors. Suppose that the data are generated by doing $n = 20$ experiments, where the experimental design is an equi-spaced set of 10 points in $[0, 1]$ with two replicates. The test function (black dashed line) and the data $\{(x_i, y_i)\}_{i=1}^{20}$ (blue dots) are plotted on Figure 2.6.

Suppose that we are interested in approximating $h(x)$ using a polynomial

$$y = \beta_0 + \beta_1 x + \ldots + \beta_m x^m. \tag{2.30}$$

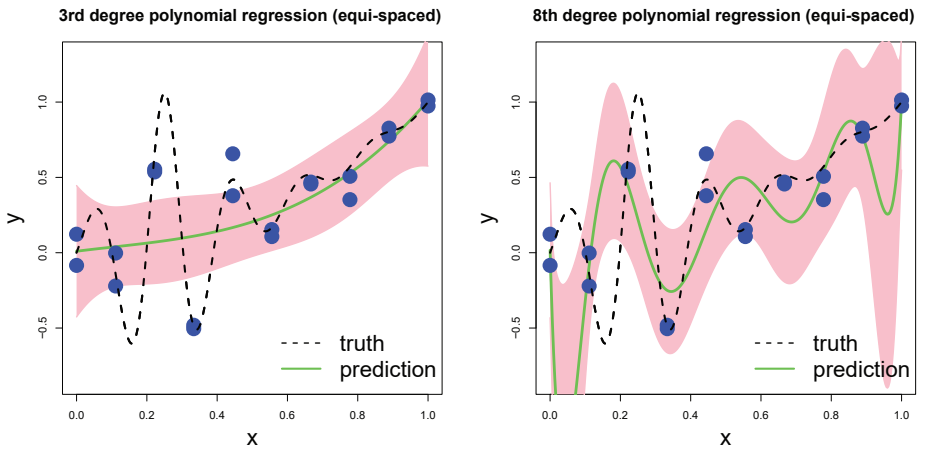


FIGURE 2.6: Polynomial regression with 10 equi-spaced design points and two replicates. Left plot shows the fit with a 3rd degree polynomial and the right shows an 8th degree polynomial. 95% prediction intervals are shown as shaded regions.

Linear regression approach takes this as the true model and assumes that the data is generated from

$$y_i = \beta_0 + \beta_1 x_i + \ldots + \beta_m x_i^m + \epsilon_i,\ i = 1, \ldots, n, \tag{2.31}$$

instead of (2.29). Thus, in this approach, we ignore the bias caused by the polynomial approximation to $h(x)$, which has important consequences in terms of uncertainty quantification. Note that although (2.31) captures nonlinear relationship, it is still called a linear regression model in statistics because it is linear in the parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_m)'$. These parameters can be chosen so that the estimated model is as close to the data as possible. Many loss functions can be used to measure this "closeness", the simplest of which is the squared error loss, which is the original method proposed by Legendre and Gauss. See Plackett (1972) for a fascinating history behind the method of least squares. Thus, the problem is to

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \{y_i - [\beta_0 + \ldots + \beta_m x_i^m]\}^2.$$

Let us write this in terms of the matrix notation:

$$\min_{\boldsymbol{\beta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}),$$

where $\mathbf{X} = \{x_i^{j-1}\}$ is the $n \times (m+1)$ model matrix. Differentiating with respect to $\boldsymbol{\beta}$ and equating to zero, we obtain

$$\mathbf{X}'\mathbf{X}\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}.$$

If $\mathbf{X}'\mathbf{X}$ is nonsingular, then we have a unique solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}. \tag{2.32}$$

Since we have only 10 unique points in x in our one-dimensional example, we must have $m \leq 9$ to make the columns of $\mathbf{X}$ linearly independent.

The predictors $\hat{y}(x) = \mathbf{x}'\hat{\boldsymbol{\beta}}$, where $\mathbf{x} = (1, x, \ldots, x^m)'$, with $m = 3$ and $m = 8$ are plotted as solid green lines in Figure 2.6. The third-degree polynomial fit causes a large bias at the

observed locations compared to the eighth-degree polynomial. However, the overall fit of the third-degree polynomial appears to be as good or even better than the eighth-degree polynomial.

If we further assume that $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$, $i = 1, \ldots, n$, then it is easy to show that

$$\hat{\boldsymbol{\beta}} \sim \mathcal{N}\left(\boldsymbol{\beta}, \sigma^2 (\mathbf{X}'\mathbf{X})^{-1}\right). \tag{2.33}$$

From this we can obtain the variance of the predictor as

$$var\{\hat{y}(\mathbf{x})\} = var\{\mathbf{x}'\hat{\boldsymbol{\beta}}\} = \sigma^2 \mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x},$$

which captures only the uncertainties in the estimation of $\boldsymbol{\beta}$. Thus, the 95% confidence intervals for the prediction of a new observation can be obtained as

$$\hat{y}(x) \pm 2\sigma\sqrt{1 + \mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}},$$

where an extra σ^2 is added to the variance to capture the uncertainties due to a new ϵ. They are plotted as a shaded region in Figure 2.6. We notice that the confidence intervals for the third-degree polynomial fit are narrower compared to those of the eighth-degree polynomial fit. However, there is more bias in the third-degree polynomial fit near the observed data points; therefore, these narrow confidence intervals do not indicate that they give a more accurate prediction. This discrepancy in the uncertainty quantification is due to the wrong assumption we made earlier in (2.31) that the third-degree polynomial is the true model. This is a major drawback of the linear regression models, which we can overcome using the nonparametric regression models discussed in the next two subsections.

Before proceeding further, let us investigate whether we can improve the linear regression fit using experimental design techniques. The estimate of $\boldsymbol{\beta}$ will be accurate if the variance of $\hat{\boldsymbol{\beta}}$ is "small". Since σ^2 is a constant, this amounts to minimizing the matrix $(\mathbf{X}'\mathbf{X})^{-1}$. To minimize it, we can define some meaningful scalar criteria (Wald, 1943; Elfving, 1952; Kiefer and Wolfowitz, 1960). A commonly used criterion is the determinant of $(\mathbf{X}'\mathbf{X})^{-1}$, which quantifies the volume of the confidence ellipsoid around $\hat{\boldsymbol{\beta}}$. Thus, to find the optimal design

$$\begin{aligned} \min_D |(\mathbf{X}'\mathbf{X})^{-1}| &= \min_D 1/|\mathbf{X}'\mathbf{X}| \\ &\equiv \max_D |\mathbf{X}'\mathbf{X}|. \end{aligned} \tag{2.34}$$

This is called a D-optimal design, where "D" stands for determinant. Another popular criterion is to minimize the sum of variances of $\hat{\beta}_i$'s, that is, the trace of the matrix $(\mathbf{X}'\mathbf{X})^{-1}$:

$$\min_D tr\left\{(\mathbf{X}'\mathbf{X})^{-1}\right\}. \tag{2.35}$$

Such a design is called an A-optimal design. The preceding two criteria focus on the accurate estimation of $\boldsymbol{\beta}$, which should indirectly help to improve the prediction. We can also define criteria that directly target prediction. A G-optimal design can be obtained by minimizing the maximum prediction variance over the entire domain $\mathcal{X}$:

$$\min_D \max_{x \in \mathcal{X}} \mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}, \tag{2.36}$$

whereas an I-optimal design can be obtained by averaging the prediction variance:

$$\min_D \int_{x \in \mathcal{X}} \mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}\, d\mathbf{x}. \tag{2.37}$$

More details of optimal designs can be found in the books Pukelsheim (1993); Silvey (1980); Atkinson et al. (2007), and López-Fidalgo (2023). A more practical treatment of the subject is given in Goos and Jones (2011).

Optimization of the foregoing design criteria is not easy because the objective function is nonlinear, multimodal, and expensive to compute. Therefore, general purpose optimization algorithms do not perform well in this problem. The most popular approach to solve them is to use some kind of exchange algorithm; that is, start with an initial design D and a set of candidates $\mathcal{C}$, and exchange a row in D with a row in $\mathcal{C}$ if there is an improvement in the objective function. Such algorithms converge to a local optimum, and therefore several initial designs are needed to find the globally optimal design. Here we will explain the Fedorov exchange algorithm (Fedorov, 1972) for finding D-optimal designs. See Cook and Nachtsheim (1980) and Nguyen and Miller (1992) for details of other exchange algorithms.

Let $\mathbf{x}_i$ be the ith row of $\mathbf{X}$, that is, $\mathbf{X} = (\mathbf{x}_1', \ldots, \mathbf{x}_n')'$. Then, $\mathbf{X}'\mathbf{X} = \sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i'$. Suppose that we exchanged the row $\mathbf{x}_i$ with a row $\mathbf{c}$ in the candidate set $\mathcal{C}$ and obtained a new model matrix $\mathbf{X}_{new}$, then the D-optimal objective function becomes $|\mathbf{X}_{new}'\mathbf{X}_{new}| = |\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}' - \mathbf{x}_i\mathbf{x}_i'|$. We need to decide which $\mathbf{x}_i$ to exchange with a $\mathbf{c} \in \mathcal{C}$. The following matrix results will be useful to simplify the computations, where the second is called the Sherman-Morrison matrix identity.

Lemma 2.4 *If* $\mathbf{A}$ *is an* $m \times m$ *matrix and* $\mathbf{a}$ *is an* $m \times 1$ *vector, then*

$$|\mathbf{A} + \mathbf{a}\mathbf{a}'| = |\mathbf{A}|(1 + \mathbf{a}'\mathbf{A}^{-1}\mathbf{a}) \quad \textit{and}$$

$$(\mathbf{A} + \mathbf{a}\mathbf{a}')^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{a}\mathbf{a}'\mathbf{A}^{-1}}{1 + \mathbf{a}'\mathbf{A}^{-1}\mathbf{a}}.$$

Applying the first matrix result two times

$$\begin{aligned}
|\mathbf{X}_{new}'\mathbf{X}_{new}| &= |\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}' - \mathbf{x}_i\mathbf{x}_i'| \\
&= |\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}'| \, \{1 - \mathbf{x}_i'(\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}')^{-1}\mathbf{x}_i\} \\
&= |\mathbf{X}'\mathbf{X}|(1 + \mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{c}) \, \{1 - \mathbf{x}_i'(\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}')^{-1}\mathbf{x}_i\}.
\end{aligned}$$

Now using the second matrix result and simplifying

$$\begin{aligned}
|\mathbf{X}_{new}'\mathbf{X}_{new}| &= |\mathbf{X}'\mathbf{X}|(1 + \mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{c}) \left\{1 - \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i + \frac{(\mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i)^2}{1 + \mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{c}}\right\} \\
&= |\mathbf{X}'\mathbf{X}|\{1 + \Delta(\mathbf{x}_i, \mathbf{c})\},
\end{aligned}$$

where

$$\Delta(\mathbf{x}_i, \mathbf{c}) = \mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{c} - \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i(1 + \mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{c}) + (\mathbf{c}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i)^2.$$

The optimal exchange can be obtained by maximizing $\Delta(\mathbf{x}_i, \mathbf{c})$, which is computationally much cheaper than using $|\mathbf{X}'\mathbf{X} + \mathbf{c}\mathbf{c}' - \mathbf{x}_i\mathbf{x}_i'|$. These exchanges can continue until there is no improvement. The algorithm is implemented in the R package `AlgDesign` (Wheeler and Braun, 2024).

Consider again the one-dimensional example. To obtain 10 distinct points, we can let $m = 9$. To obtain the candidate set, we first choose 301 equi-spaced points from $[0, 1]$, and then obtain $\mathbf{c}_j = (1, c_j, \ldots, c_j^9)$, $j = 1, \ldots, 301$. The D-optimal design is then found using the `optFederov` function in `AlgDesign`. They are plotted in Figure 2.7. We can see that the points are pushed toward the boundaries. The two polynomial fits are better than before, and in particular, the high fluctuations of the eighth-degree polynomial near the boundaries have disappeared.

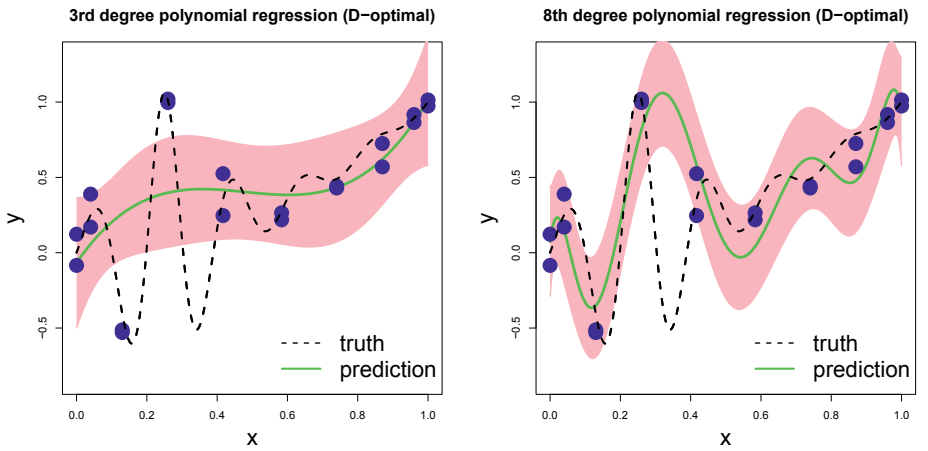


FIGURE 2.7: Polynomial regression with 10 D-optimal design points and two replicates. Left plot shows the fit with a 3rd degree polynomial and the right shows an 8th degree polynomial. 95% prediction intervals are shown as shaded regions.

A close examination of the design points in Figure 2.7 and the Chebyshev nodes in Figure 2.1 will reveal that they are very close. This is not an accident! Dette and Studden (1992) have shown that the empirical distribution of D-optimal designs converges to the arcsine density

$$f(x) = \frac{1}{\pi\sqrt{x(1-x)}}, \text{ for } x \in [0,1], \text{ or}$$
$$f(x) = \frac{1}{\pi\sqrt{1-x^2}}, \text{ for } x \in [-1,1]. \tag{2.38}$$

In Chapter 5, we will discuss experimental designs to represent a distribution, but for the moment we will use a simple approach. It is well known that $\{.5/n, 1.5/n, \ldots, (n-.5)/n\}$ is the best set of n points to approximate a uniform distribution in $[0,1]$. Then, the best set of points to approximate the arcsine density can be obtained using the inverse probability transform method (Section 5.2.1). For $x \in [-1,1]$, the distribution function can be obtained as

$$F(x) = \int_{-1}^{x} \frac{1}{\pi\sqrt{1-x^2}}\, dx = 1 - \frac{1}{\pi} cos^{-1}x,$$

which gives $F^{-1}(x) = \cos(\pi(1-x))$. Since $\{(i-.5)/n\}_{i=1}^{n}$ or $\{(n-i+.5)/n\}_{i=1}^{n}$ is the best point set of a uniform distribution, the representative points of the arcsine density can be obtained using the inverse probability transform as $\{F^{-1}([n-i+.5]/n)\}_{i=1}^{n}$, which is exactly the Chebyshev nodes given in (2.5). In other words, for large n, the D-optimal design points for fitting an $(n-1)$th degree polynomial can be well approximated using Chebyshev nodes. This makes a fascinating connection between the two fields of experimental design and approximation theory.

Now consider the case with more than one input variable. A linear regression model that includes only the main effects of the variables can be written as

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon.$$

We can also include the polynomial terms $x_j^2, x_j^3, \ldots$ for $j = 1, \ldots, p$, two-factor interaction terms $x_i x_j, x_i x_j^2, \ldots$, and higher-order interaction terms $x_i x_j x_k, \ldots$. Thus, the total number of effects that can be included in the linear regression model, P, can be very large. In fact, P can be much larger than the size of the data n. Let $\mathbf{X}$ be the $n \times P$ model matrix. Then, $rank(\mathbf{X}'\mathbf{X}) = n < P$, and therefore $\mathbf{X}'\mathbf{X}$ cannot be inverted to get the least squares estimate as in (2.32). We can overcome this issue using ridge regression (Hoerl and Kennard, 1970). Assume that the columns of $\mathbf{X}$ and $\mathbf{y}$ are standardized to have mean 0 and standard deviation 1. Now consider the regression model without intercept. Then, ridge regression solves the optimization problem:

$$\min_{\boldsymbol{\beta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}'\boldsymbol{\beta}, \tag{2.39}$$

where $\lambda > 0$ is a regularization parameter. This gives the solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_P)^{-1} \mathbf{X}'\mathbf{y}. \tag{2.40}$$

The regularization parameter λ can be chosen using cross-validation methods. The following result is useful for deriving leave-one-out cross-validation errors. Let $\mathbf{L} = \mathbf{X}(\mathbf{X}'\mathbf{X}+\lambda \mathbf{I}_P)^{-1}\mathbf{X}'$ be the hat matrix. Then the leave-one-out cross-validation error defined $y_i - \hat{y}_{\sim i}$ can be computed using the following short-cut formula (Wasserman, 2006, p.70).

Lemma 2.5 *For a linear predictor* $\hat{\mathbf{y}} = \mathbf{L}\mathbf{y}$*, the leave-one-out cross-validation error can be computed using*

$$cv_i = \frac{y_i - \hat{y}_i}{1 - l_i}, \; i = 1, \ldots, n,$$

where $l_i = \mathbf{L}_{ii}$ *is the hat value or leverage.*

Thus, the mean squared cross-validation error is given by $MSCV = \sum_{i=1}^n cv_i^2/n$. Golub et al. (1979) proposed an approximation to MSCV called generalized cross-validation (GCV) given by

$$GCV = \frac{1}{n} \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(1 - \bar{l})^2},$$

where $\bar{l} = \sum_{i=1}^n l_i/n = tr\{(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_P)^{-1}\mathbf{X}'\mathbf{X}\}/n$. GCV has some nice properties, such as the rotational invariance of the model matrix. Thus, λ can be estimated from data by minimizing either the MSCV or GCV.

However, ridge regression in (2.40) contains only a single regularization parameter λ, which indirectly assumes that main effects, two-factor interactions, etc. to be of the same importance. This is rarely true in practice. Therefore, we need to make some modifications to the ridge regression formulation to make it useful for the analysis of experiments. For this, we need to first introduce certain experimental design principles such as effect hierarchy and heredity (Wu and Hamada, 2021). We will do this later in Chapter 8 when we discuss the analysis of physical experiments.

2.2.2 Kernel Ridge Regression

Instead of using polynomial basis functions, we could use radial basis functions as in Section 2.1.2 to approximate the underlying function. This leads to the kernel ridge regression. The kernel $K(\cdot, \cdot)$ does not have to be radial, but should be symmetric and positive definite, that is, $K(\mathbf{u}, \mathbf{v}) = K(\mathbf{v}, \mathbf{u})$ and satisfy Definition 2.1. Although we will mainly use the shift-invariant kernels $K(\mathbf{u}, \mathbf{v}) = R(\mathbf{u}-\mathbf{v})$ in this book, we will describe the kernel ridge regression

for general type of kernels. Thus, given the noisy data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ we are interested in fitting the model

$$y_i = \sum_{j=1}^{n} c_j K(\mathbf{x}_i, \mathbf{x}_j) + \epsilon_i,$$

where $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ and y_i's are assumed to be centered. This leads to the least squares formulation:

$$\min_{\mathbf{c}} (\mathbf{y} - \mathbf{K}\mathbf{c})'(\mathbf{y} - \mathbf{K}\mathbf{c}),$$

where $\mathbf{K}$ is the $n \times n$ matrix with ijth element $K(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{c} = (c_1, \ldots, c_n)'$. However, as we have seen in Section 2.1.2, this problem has a unique solution $\hat{\mathbf{c}} = \mathbf{K}^{-1}\mathbf{y}$, provided the $\mathbf{x}_i$'s are distinct. This will make the predictor an interpolator, which is probably not desirable for noisy data because the predictor can have unnecessarily large fluctuations and can lead to poor approximation to the true underlying function. We can avoid this problem through regularization; that is, we will add a penalty term to the least squares objective function that controls the fluctuations of the predictor. Let $h_n(\mathbf{x}) = \mathbf{k}(\mathbf{x})'\mathbf{c}$ be the predictor, where $\mathbf{k}(\mathbf{x})$ is the $n \times 1$ vector with ith element $K(\mathbf{x}, \mathbf{x}_i)$. Thus, we wish to solve the problem

$$\min_{\mathbf{c}} (\mathbf{y} - \mathbf{K}\mathbf{c})'(\mathbf{y} - \mathbf{K}\mathbf{c}) + \lambda \|h_n\|^2,$$

where $\lambda > 0$ and $\|h_n\|$ is a norm of the function $h_n(\cdot)$ that controls its fluctuations.

To define a proper norm for the function $h_n(\cdot)$, we need to first define a class of functions in which $h_n(\cdot)$ is a member. Consider a class of functions formed by linear span of the kernel:

$$h(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, \mathbf{u}_i),$$

where m can be any positive integer and $\mathbf{u}_i$s can be any points in the domain $\mathcal{X}$. Define an inner product of any two elements of this function class as follows. Consider two functions $f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, \mathbf{u}_i)$ and $g(\cdot) = \sum_{i=1}^{n} \beta_i K(\cdot, \mathbf{v}_i)$. Then, define their inner product as

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j K(\mathbf{u}_i, \mathbf{v}_j).$$

A norm of the function induced by this inner product is $\|f\| = \sqrt{\langle f, f \rangle}$. Now, if we take a closure of the class of functions we defined earlier with respect to this norm, we get a Hilbert space of functions $\mathcal{H}$. This is called a reproducing kernel Hilbert space (RKHS) because of the reproducing property $\langle K(\cdot, \mathbf{u}), K(\cdot, \mathbf{v}) \rangle = K(\mathbf{u}, \mathbf{v})$. See Fasshauer and McCourt (2015, Ch.2) for details of RKHS.

Thus, the norm of the predictor $h_n(\cdot) = \sum_{i=1}^{n} c_i K(\cdot, \mathbf{x}_i)$ with respect to the RKHS of the kernel $K(\cdot, \cdot)$ is given by

$$\|h_n\|^2 = \langle h_n, h_n \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{c}'\mathbf{K}\mathbf{c}.$$

Hence, our problem is to

$$\min_{\mathbf{c}} (\mathbf{y} - \mathbf{K}\mathbf{c})'(\mathbf{y} - \mathbf{K}\mathbf{c}) + \lambda \mathbf{c}'\mathbf{K}\mathbf{c}. \tag{2.41}$$

In fact, it can be shown that this problem is equivalent to the infinite-dimensional optimization problem, which follows from the representer theorem (Wahba, 1990):

$$\min_{h_n \in \mathcal{H}} \sum_{i=1}^{n} \{y_i - h_n(\mathbf{x}_i)\}^2 + \lambda \|h_n\|^2.$$

Although this optimization appears difficult to solve, optimizing (2.41) is easy. Differentiating with respect to $\mathbf{c}$ and equating to 0, we obtain

$$-2\mathbf{K}(\mathbf{y} - \mathbf{K}\mathbf{c}) + 2\lambda\mathbf{K}\mathbf{c} = 0,$$

which gives

$$\hat{\mathbf{c}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}. \tag{2.42}$$

By comparing to the ridge regression solution in (2.40), it should now be clear why this method is called kernel ridge regression. To see that (2.40) can in fact be obtained as a special case, let $K(u, v) = \mathbf{r}(u)'\mathbf{r}(v)$, where $\mathbf{r}(u) = (u, u^2, \ldots, u^m)'$. Then $\mathbf{K} = \mathbf{R}\mathbf{R}'$, where $\mathbf{R}$ is an $n \times m$ matrix with ith row $\mathbf{r}(x_i)'$. Now (2.41) becomes

$$\min_{\mathbf{c}} (\mathbf{y} - \mathbf{R}\mathbf{R}'\mathbf{c})'(\mathbf{y} - \mathbf{R}\mathbf{R}'\mathbf{c}) + \lambda\mathbf{c}'\mathbf{R}\mathbf{R}'\mathbf{c}$$

which reduces to (2.39) by letting $\boldsymbol{\beta} = \mathbf{R}'\mathbf{c}$.

The kernel ridge regression predictor $\hat{y}(\mathbf{x}) = \mathbf{k}(\mathbf{x})'\hat{\mathbf{c}}$ can be used only after specifying the regularization parameter λ and the parameters $\boldsymbol{\theta}$ in the kernel function. We can again use the cross-validation methods as in Section 2.1.2 for this purpose. The leave-one-out cross-validation error can be obtained using Lemma 2.5

$$cv_i = \frac{y_i - \mathbf{H}_i\mathbf{y}}{1 - \mathbf{H}_{ii}},$$

where $\mathbf{H} = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}$. This formula cannot be used when $\lambda = 0$, but we can simplify this formula to accommodate this case. Using the matrix identity

$$\mathbf{I} - \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1} = \lambda(\mathbf{K} + \lambda\mathbf{I})^{-1}$$

we obtain

$$\begin{aligned} cv_i &= \frac{\left(\mathbf{y} - \{\mathbf{I} - \lambda(\mathbf{K} + \lambda\mathbf{I})^{-1}\}\mathbf{y}\right)_i}{\left(\mathbf{I} - \{\mathbf{I} - \lambda(\mathbf{K} + \lambda\mathbf{I})^{-1}\}\right)_{ii}} \\ &= \frac{\left((\mathbf{K} + \lambda\mathbf{I})^{-1}\right)_i \mathbf{y}}{\left((\mathbf{K} + \lambda\mathbf{I})^{-1}\right)_{ii}}, \end{aligned}$$

which coincides with the formula (2.9) that we derived earlier when $\lambda = 0$.

Consider again the data $\{(x_i, y_i)\}_{i=1}^{20}$ used in the previous section generated using the one-dimensional function in (2.1). Let us use the Gaussian kernel $K(u, v) = \exp\{-(u - v)^2/\theta^2\}$. Estimating λ and θ by minimizing MSCV or GCV yields the same results: $\hat{\lambda} = 0.038$ and $\hat{\theta} = 0.080$ for equi-spaced design points and $\hat{\lambda} = 0.036$ and $\hat{\theta} = 0.079$ for D-optimal (for the ninth-degree polynomial) design points. The predictors are shown in Figure 2.8, which are much better than the polynomial regressions shown in Figures 2.6 and 2.7.

2.2.3 Gaussian Process Regression

Because of their probabilistic formulation, extending kriging or Gaussian process to deal with noisy data is easy. Since kriging and Gaussian process formulations are similar, we will just focus on the Gaussian process in this section. Our statistical model is

$$y = h(\mathbf{x}) + \epsilon,\ h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot)),\ \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2).$$

We have the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Assume for the moment that the parameters μ, τ^2, σ^2, and the correlation parameters $\boldsymbol{\theta}$ are known. We will later use empirical Bayes methods for estimating them.

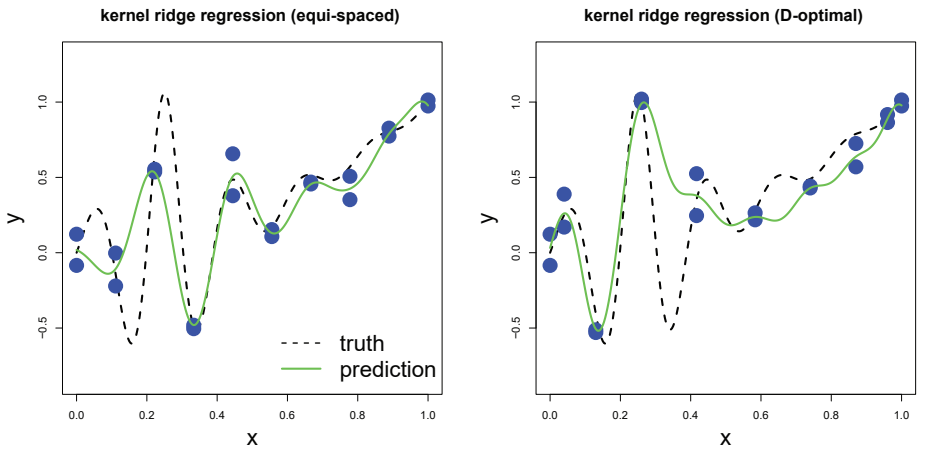


FIGURE 2.8: Kernel ridge regression using 10-point equi-spaced (left) and D-optimal (right) design points with two replicates.

The marginal distribution of $\mathbf{y} = (y_1, \ldots, y_n)'$ is

$$\mathbf{y} \sim \mathcal{N}(\mu\mathbf{1}, \tau^2\mathbf{R} + \sigma^2\mathbf{I})$$

and the joint distribution of $h(\mathbf{x})$ and $\mathbf{y}$ is

$$\begin{pmatrix} h(\mathbf{x}) \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}_{1+n}\left(\begin{pmatrix} \mu \\ \mu\mathbf{1} \end{pmatrix}, \begin{bmatrix} \tau^2 & \tau^2\mathbf{r}(\mathbf{x})' \\ \tau^2\mathbf{r}(\mathbf{x}) & \tau^2\mathbf{R} + \sigma^2\mathbf{I} \end{bmatrix}, \right)$$

where $\mathbf{r}(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^n$ and $\mathbf{R} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$ as in (2.13). These steps were easy to derive because ϵ is normally distributed. For non-normal data such as binomial or Poisson, the analysis becomes difficult and is not covered in this book. See Banerjee et al. (2003) and Diggle and Ribeiro Jr (2007) for Gaussian process regression with non-normal data.

Using Lemma 2.2, we obtain the posterior distribution of $h(\mathbf{x})$ for a fixed $\mathbf{x}$:

$$h(\mathbf{x})|\mathbf{y} \sim \mathcal{N}\left(\mu + \mathbf{r}(\mathbf{x})'(\mathbf{R} + \lambda\mathbf{I})^{-1}(\mathbf{y} - \mu\mathbf{1}), \tau^2\{1 - \mathbf{r}(\mathbf{x})'(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{r}(\mathbf{x})\}\right), \tag{2.43}$$

where $\lambda = \sigma^2/\tau^2$. We can see that the posterior mean

$$\hat{y}(\mathbf{x}) = \mu + \mathbf{r}(\mathbf{x})'(\mathbf{R} + \lambda\mathbf{I})^{-1}(\mathbf{y} - \mu\mathbf{1})$$

reduces to the kernel ridge regression predictor when $\mu = 0$. Although the predictions are the same, we now have a variance for the prediction which can be used for constructing confidence intervals. See Kanagawa et al. (2018) for a more comprehensive comparison between the two methods. The prediction variance for a new observation at $\mathbf{x}$ is given by

$$\begin{aligned} s^2(\mathbf{x}) &= var\{h(\mathbf{x}) + \epsilon|\mathbf{y}\} \\ &= var\{h(\mathbf{x})|\mathbf{y}\} + var(\epsilon) \ \text{ (by independence)} \\ &= \tau^2\{1 + \lambda - \mathbf{r}(\mathbf{x})'(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{r}(\mathbf{x})\}. \end{aligned}$$

Thus, the 95% prediction intervals are given by $\hat{y}(\mathbf{x}) \pm 2s(\mathbf{x})$.

The Gaussian process formulation also gives us an alternative to the cross-validation method to estimate the unknown parameters. The likelihood of the parameters is given by

$$L \propto \frac{1}{\tau^n |(\mathbf{R} + \lambda \mathbf{I})|^{1/2}} \exp\left\{ -\frac{1}{2\tau^2} (\mathbf{y} - \mu \mathbf{1})' (\mathbf{R} + \lambda \mathbf{I})^{-1} (\mathbf{y} - \mu \mathbf{1}) \right\}.$$

Maximizing L with respect to μ, τ^2, λ, and $\boldsymbol{\theta}$ gives

$$\hat{\mu} = \frac{\mathbf{1}'(\mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{y}}{\mathbf{1}'(\mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{1}}, \tag{2.44}$$

$$\hat{\tau}^2 = \frac{1}{n} (\mathbf{y} - \hat{\mu}\mathbf{1})' (\mathbf{R} + \lambda \mathbf{I})^{-1} (\mathbf{y} - \hat{\mu}\mathbf{1}), \tag{2.45}$$

$$(\hat{\lambda}, \hat{\boldsymbol{\theta}}) = \underset{\lambda, \boldsymbol{\theta}}{\operatorname{argmin}} \left\{ n \log \hat{\tau}^2 + \log |\mathbf{R} + \lambda \mathbf{I}| \right\}. \tag{2.46}$$

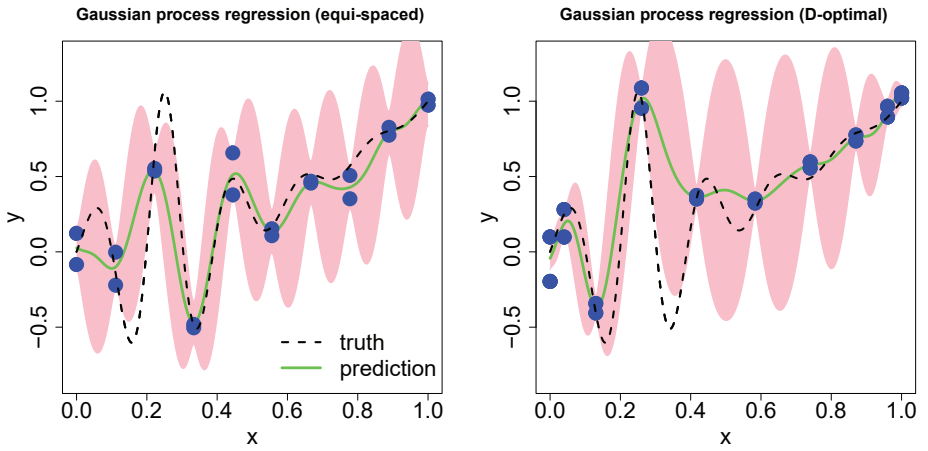


FIGURE 2.9: Gaussian process regression using 10-point equi-spaced (left) and D-optimal (right) design points with two replicates.

The main difference between Gaussian process regression and Gaussian process interpolation discussed in Section 2.1.4 is that the correlation matrix $\mathbf{R}$ is replaced with $\mathbf{R} + \lambda \mathbf{I}$. In fact, even for noiseless data, this modification using a small λ is useful for numerical stability (Ranjan et al., 2011; Gramacy and Lee, 2012; Peng and Wu, 2014). In the geostatistics literature, λ is called a *nugget*.

Consider again the 20 data points generated using the one-dimensional test function in (2.1). We fit the Gaussian process regression using the R package `rkriging` (Huang and Joseph, 2024) with a Gaussian correlation function $R(u - v) = \exp\{-(u - v)^2/\theta^2\}$. The predictions and 95% confidence intervals for the two design choices, equi-spaced points and D-optimal points for the ninth degree polynomial, are shown in Figure 2.9. Not surprisingly, the predictions are similar to those of kernel ridge regression, but the advantage of the Gaussian process regression is that we are able to provide uncertainty intervals for the predictions. Compared to the eighth degree polynomial given in the right panels of Figures 2.6 and 2.7, the predictions as well as the uncertainty intervals of the Gaussian process regression are much better. The uncertainty intervals of Gaussian process regression make more sense because they are narrower around the regions where we observed the data and wider in data-sparse regions.

Exercises

1. A function $\psi : [0, \infty) \to \mathbb{R}$ is called completely monotone if $(-1)^l \psi^{(l)}(x) \geq 0$, for $l = 1, 2, \ldots$, where $\psi^{(l)}(x)$ is the lth derivative of $\psi(x)$. Then, the following result holds (Fasshauer, 2007, p. 49).

   Theorem 2.1 *A function $\psi : [0, \infty) \to \mathbb{R}$ is completely monotone but not constant if and only if $\psi(\|\cdot\|^2)$ is positive definite and radial on $\mathbb{R}^p$ for any p.*

   Use this theorem to show that

   (a) the Gaussian correlation function: $R(\mathbf{x}) = \exp\{-\theta\|\mathbf{x}\|^2\}$, $\theta > 0$, and
   (b) the rational quadratic correlation function: $R(\mathbf{x}) = (1 + \theta\|\mathbf{x}\|^2)^{-\alpha}$, $\theta, \alpha > 0$

   are positive definite.

2. Show that
$$R(\mathbf{x}) = \frac{e^{-\theta_1 ||x||^2}}{1 + \theta_2 ||x||^2}, \mathbf{x} \in \mathbb{R}^p,$$
   is a positive definite correlation function for $\theta_1, \theta_2 > 0$.

3. Show that for any correlation function $R(\cdot)$ and any set of distinct points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^p$,
$$0 \leq \mathbf{r}(\mathbf{x})' \mathbf{R}^{-1} \mathbf{r}(\mathbf{x}) \leq 1, \ \forall \mathbf{x} \in \mathbb{R}^p,$$
   where $\mathbf{r}(\mathbf{x})' = (R(\mathbf{x} - \mathbf{x}_1), \cdots, R(\mathbf{x} - \mathbf{x}_n))$ and $\mathbf{R}$ is an $n \times n$ matrix with ijth element $R(\mathbf{x}_i - \mathbf{x}_j)$.

4. Suppose that $Y(\mathbf{x})$ is a stochastic process with mean
$$\mu(\mathbf{x}) = \beta_0 + \beta_1 f_1(\mathbf{x}) + \cdots + \beta_m f_m(\mathbf{x}) = \mathbf{f}(\mathbf{x})' \boldsymbol{\beta} \tag{2.47}$$
   and covariance function $cov\{Y(\mathbf{u}), Y(\mathbf{v})\} = \tau^2 R(\mathbf{u} - \mathbf{v})$. Assume that $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ are known functions and $f_0(\mathbf{x}) \equiv 1$. Let the data be $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. First, show that the linear predictor $\hat{y}(\mathbf{x}) = \mathbf{a}(\mathbf{x})' y$ will be unbiased if
$$\mathbf{F}' \mathbf{a}(\mathbf{x}) = \mathbf{f}(\mathbf{x}), \tag{2.48}$$
   where $\mathbf{F} = (\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_n))'$ is the $n \times (m + 1)$ model matrix corresponding to the linear model in (2.47). Then show that the best linear unbiased predictor (BLUP) that minimizes the mean squared prediction error in (2.12) is given by
$$\hat{y}(\mathbf{x}) = \mathbf{f}(\mathbf{x})' \hat{\boldsymbol{\beta}} + \mathbf{r}(\mathbf{x})' \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F} \hat{\boldsymbol{\beta}}), \tag{2.49}$$
   where
$$\hat{\boldsymbol{\beta}} = (\mathbf{F}' \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}' \mathbf{R}^{-1} \mathbf{y}$$
   is the generalized least squares estimate of $\boldsymbol{\beta}$. The predictor in (2.49) is called the *universal kriging* predictor.

5. Show that the mean squared prediction error of the universal kriging predictor in (2.49) is given by
$$MSE(\mathbf{x}) = \tau^2 \left\{1 - \mathbf{r}(\mathbf{x})' \mathbf{R}^{-1} \mathbf{r}(\mathbf{x}) + \boldsymbol{\delta}(\mathbf{x})' (\mathbf{F}' \mathbf{R}^{-1} \mathbf{F})^{-1} \boldsymbol{\delta}(\mathbf{x})\right\}, \tag{2.50}$$
   where $\boldsymbol{\delta}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{F}' \mathbf{R}^{-1} \mathbf{r}(\mathbf{x})$.

6. Consider the Gaussian process model

$$h(\mathbf{x})|\boldsymbol{\beta} \sim GP(\mathbf{f}(\mathbf{x})'\boldsymbol{\beta}, \tau^2 R(\cdot)),$$

where $\mathbf{f}(\mathbf{x})$ and $\boldsymbol{\beta}$ are as defined in (2.47). Given the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, find the posterior distribution of $h(\mathbf{x})$ under the following two cases:

   (a) Assume $\boldsymbol{\beta}$ to be known.

   (b) Assume a noninformative prior: $p(\boldsymbol{\beta}) \propto 1$.

7. Consider the ordinary kriging predictor in (2.14) or the Gaussian process predictor in (2.18). Show that its leave-one-out cross validation error is given by

$$cv_i = \frac{\mathbf{A}_i \mathbf{y}}{\mathbf{A}_{ii}},\ i = 1, \ldots, n,$$

where $\mathbf{A}_i$ and $\mathbf{A}_{ii}$ are the ith row and ith diagonal element of the matrix

$$\mathbf{A} = \mathbf{R}^{-1} - \frac{\mathbf{R}^{-1}\mathbf{1}\mathbf{1}'\mathbf{R}^{-1}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}.$$

8. Let the design be $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Consider the Gaussian process model

$$h(\mathbf{x}) = \mu + \frac{\nu}{\mathbf{r}(\mathbf{x})'\mathbf{c}} Z(\mathbf{x}), \quad Z(\mathbf{x}) \sim GP(0, R(\cdot)), \tag{2.51}$$

where $\mathbf{r}(\mathbf{x})$ is the $n \times 1$ correlation vector defined in (2.13), $\mathbf{c} \geq 0$, and $\|\mathbf{c}\| = 1$. Then show that the posterior distribution of $h(\mathbf{x})$ is given by

$$y(\mathbf{x})|\mathbf{y} \sim N\left(\hat{y}(\mathbf{x}), \frac{\nu^2}{\{\mathbf{r}(\mathbf{x})'\mathbf{c}\}^2}\{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}\right),$$

where

$$\hat{y}(\mathbf{x}) = \frac{\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1} diag(\mathbf{R}\mathbf{c})\mathbf{y}}{\mathbf{r}(\mathbf{x})'\mathbf{c}} \tag{2.52}$$

is called a rational kriging predictor (Joseph, 2025) because it is the ratio of two basis functions.

9. Consider again the rational GP model in (2.51).

   (a) Under a noninformative prior $p(\mu) \propto 1$, show that

$$\mu|\mathbf{y} \sim N\left(\hat{\mu}, \frac{\nu^2}{\mathbf{c}'\mathbf{R}\mathbf{c}}\right),$$

   where

$$\hat{\mu} = \frac{\mathbf{c}' diag(\mathbf{R}\mathbf{c})\mathbf{y}}{\mathbf{c}'\mathbf{R}\mathbf{c}}$$

   is the generalized least squares estimate of μ.

   (b) Show that $\hat{\mu}$ will always be in the range $[\min_i y_i, \max_i y_i]$ for any nonnegative and positive definite correlation function $R(\cdot)$.

   (c) Show that the posterior variance of μ can be minimized by taking $\mathbf{c}$ to be the Perron eigenvector of $\mathbf{R}$.

10. Consider the rational quadratic correlation function

$$R(\boldsymbol{\Delta}) = \frac{1}{1 + \frac{\|\boldsymbol{\Delta}\|^2}{\theta^2}}, \ \theta > 0.$$

Show that the rational kriging predictor in (2.52) with $\mathbf{c} = \mathbf{1}$ converges to the inverse distance weighting predictor (Shepard, 1968; Joseph and Kang, 2011)

$$\hat{y}_{IDW}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \|\mathbf{x} - \mathbf{x}_i\|^{-2} y_i}{\sum_{j=1}^{n} \|\mathbf{x} - \mathbf{x}_j\|^{-2}}.$$

as $\theta \to 0$.

11. Consider a Gaussian process model $h(\mathbf{x}) = \mu + Z(\mathbf{x})$, where $Z(\mathbf{x}) \sim GP(0, K(\cdot, \cdot))$. Let $R(\cdot)$ be any stationary correlation function and $\mathcal{X} = [0, 1]^p$. Show that if

$$K(\mathbf{u}, \mathbf{v}) = \tau^2 R(\mathbf{u} - \mathbf{v}) - \tau^2 \frac{\int_{\mathcal{X}} R(\mathbf{u} - \boldsymbol{\xi}) d\boldsymbol{\xi} \int_{\mathcal{X}} R(\boldsymbol{\xi} - \mathbf{v}) d\boldsymbol{\xi}}{\int_{\mathcal{X}} \int_{\mathcal{X}} R(\mathbf{u} - \mathbf{v}) d\mathbf{u} d\mathbf{v}},$$

then

$$\int_{\mathcal{X}} Z(\mathbf{x}) d\mathbf{x} = 0$$

with probability one. Then $Z(\mathbf{x})$ is called an orthogonal Gaussian process (Plumlee and Joseph, 2018).

12. Consider the 7-dimensional piston simulation function (Kenett and Zacks, 2021):

$$y = 2\pi \sqrt{\frac{M}{k + S^2 \frac{P_0 V_0}{T_0} \frac{T_a}{V^2}}},$$

$$V = \frac{S}{2k} \left(\sqrt{A^2 + 4k \frac{P_0 V_0}{T_0} T_a} - A \right), A = P_0 S + 19.62M - \frac{kV_0}{S}.$$

The ranges of the input variables are: piston weight, $M \in [30, 60]$; piston surface area, $S \in [.005, .02]$; initial gas volume, $V_0 \in [.002, .010]$; spring coefficient, $k \in [1000, 5000]$; atmospheric pressure, $P_0 \in [90000, 110000]$; ambient temperature, $T_a \in [290, 296]$; and filling gas temperature, $T_0 \in [340, 360]$. We will learn about optimal experimental designs in Part II, but for the moment let us use 70 uniform random samples $\mathbf{x}_i \overset{iid}{\sim} \mathcal{U}(0, 1)^7$, $i = 1, \ldots, 70$ as the experimental design. Rescale the samples using the ranges of the seven inputs and generate the output using the piston simulation function. Fit a Gaussian process model (ordinary kriging) using correlation functions

(a) Gaussian
(b) Matérn 3/2
(c) Matérn 5/2.

Generate a testing set $\{(\mathbf{X}_i, Y_i)\}_{i=1}^{N}$, where $N = 700$ and $\mathbf{X}_i \overset{iid}{\sim} \mathcal{U}(0, 1)^7$, $i = 1, \ldots, 700$. Visualize the performance of the GP model using a parity plot $\hat{y}(\mathbf{X}_i)$ versus Y_i under the three correlation functions. Compute the root mean squared error using

$$RMSE = \frac{1}{N} \sum_{i=1}^{N} \{Y_i - \hat{y}(\mathbf{X}_i)\}^2,$$

and identify the best correlation function.

13. Consider a simple example in which we hope to control a robot arm in a two-dimensional plane (An and Owen, 2001). The robot arm has four extendable segments of lengths L_1, L_2, L_3, L_4 and are at angles $\theta_1, \theta_2, \theta_3, \theta_4$ to the horizontal coordinate axis of the plane, respectively. The location of the end of the robot arm (u, v) is:

$$u = \sum_{i=1}^{4} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right), \quad v = \sum_{i=1}^{4} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right),$$

where $L_i \in [0, 1]$ and $\theta_i \in [0, 2\pi]$ for $i = 1, 2, 3, 4$. Take the response as the distance of the end of arm from the base: $y = \sqrt{u^2 + v^2}$. Repeat Exercise 12 using 80 samples as the experimental design and 800 samples as the testing set.

14. Consider the 6-dimensional OTL circuit function (Ben-Ari and Steinberg, 2007):

$$y = \frac{(V_{b1} + 0.74)\,\beta\,(R_{c2} + 9)}{\beta\,(R_{c2} + 9) + R_f} + \frac{11.35 R_f}{\beta\,(R_{c2} + 9) + R_f} + \frac{0.74 R_f \beta\,(R_{c2} + 9)}{(\beta\,(R_{c2} + 9) + R_f)\,R_{c1}},$$

$$V_{b1} = \frac{12 R_{b2}}{R_{b1} + R_{b2}}.$$

The ranges of the input variables are: $R_{b1} \in [50, 150]$, $R_{b2} \in [25, 70]$, $R_f \in [0.5, 3]$, $R_{c1} \in [1.2, 2.5]$, $R_{c2} \in [0.25, 1.2]$, and $\beta \in [50, 300]$. Repeat Exercise 12 using 60 samples as the experimental design and 600 samples as the testing set. In addition to RMSE, compute the interval score (Gneiting and Raftery, 2007) defined as

$$IS = \frac{1}{N} \sum_{i=1}^{N} \left[(u_i - l_i) + \frac{2}{\alpha} \{(l_i - t_i)_+ + (t_i - u_i)_+\} \right] \tag{2.53}$$

for assessing the accuracy of $(1 - \alpha)$ confidence intervals $[l_i, u_i]$, where $(x)_+ = x$ if $x > 0$ and 0 otherwise, and $\{t_i\}_{i=1}^{N}$ are the $N = 600$ function values at the testing locations. Which correlation function gives the smallest IS for $\alpha = 0.05$?

15. Consider the piston simulation function in Exercise 12. Fit a GP model using the R packages

    (a) `rkriging` (Huang and Joseph, 2024)

    (b) `mlegp` (Dancik, 2022)

    (c) `DiceKriging` (Roustant et al., 2021)

    using a Gaussian correlation function. Compare their performance in terms of RMSE and computational time.

Part II

Computer Experiments



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

3

Model-based Designs

As we know how to approximate the response surface for given data, we will turn our attention to how to optimally collect this data. As discussed in Chapter 1, we will use a two-stage strategy for experimental design: first an *initial design* and then a *sequential design*. In the initial design stage, we will not make any assumptions about the underlying input-output relationship, and we will try to collect the data to obtain a good approximation of the entire response surface. In the sequential design stage, we will continue the data collection process using the model estimated from the initial data in order to satisfy certain objectives of the experiment, such as optimization. We will focus on initial designs in this chapter and the next three chapters and discuss sequential designs in Chapter 7. The case of experimental designs for noisy data will be developed later in the chapter on physical experiments.

3.1 Prediction-based Designs

Suppose that we have two inputs (x_1 and x_2) and a budget to do seven experiments. Let $\mathcal{X}$ be the experimental region from which we need to choose these seven points. Unless otherwise mentioned, we assume that the experimental region can be scaled into a hypercube. So, here $\mathcal{X} = [0,1]^2$. If we do not use any experimental design ideas, then we could randomly sample seven points from $[0,1]^2$. Three such random designs are shown in Figure 3.1. We can see that the first design places the points mostly on the left side of $\mathcal{X}$, the second design places the points mostly on the middle side, and the third design places the points mostly on the right side. Clearly, they are not "good" designs to learn an input-output relationship in $\mathcal{X}$. It may so happen because they are the best designs for certain functions. For example, a function that varies more on the left side and almost constant on the right side can probably be well approximated using the first design. However, in the experimental design stage, we do not know anything about the function. Therefore, we should design the experiment in such a way that any possible function can be well approximated using the data. Therefore, we need to consider a class of functions and look for designs that will work well on average or minimize worst-case scenarios. For this purpose, we will use the stochastic framework provided by kriging or GP. The earlier developments using related ideas can be found in Sacks and Ylvisaker (1970) and Ylvisaker (1975). See Sacks et al. (1989) for a historical account. We will soon see that these model-based optimal designs cannot be used in practice because of their dependence on the unknown model parameters. However, the model-based approach will help us develop optimal design criteria that will guide us in developing more practical designs in the next chapter.

Consider a GP model for the response surface: $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$. Assume for the moment that μ, τ^2, and the correlation parameters $\boldsymbol{\theta}$ are known. Given the data

DOI: 10.1201/9781003661535-3

This chapter has been made available under a CC-BY-NC-ND license 4.0.

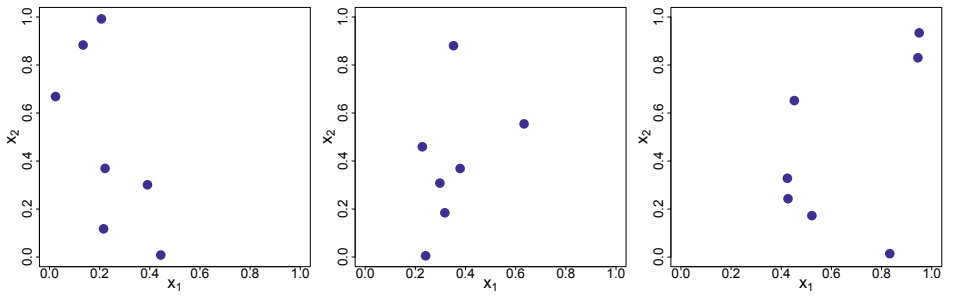


FIGURE 3.1: Three 7-run randomly sampled designs in $[0,1]^2$.

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the posterior distribution of $h(\mathbf{x})$ is given by

$$h(\mathbf{x})|\mathbf{y} \sim \mathcal{N}(\hat{y}(\mathbf{x}), s^2(\mathbf{x})),$$

where

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \mu + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}), \\ s^2(\mathbf{x}) &= \tau^2\{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}, \end{aligned}$$

where $\mathbf{r}(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^n$ and $\mathbf{R} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$ as in (2.13). Note that $\mathbf{r}(\mathbf{x})$ and $\mathbf{R}$ depend on the design $D = \{\mathbf{x}_i\}_{i=1}^n$ and the correlation parameters $\boldsymbol{\theta}$, but we do not explicitly show their dependence for simplicity of the notation.

Our aim is to find the design so that $\hat{y}(\mathbf{x})$ is as close as possible to the true response surface $h(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is the experimental region, which is usually the hypercube $[0,1]^p$ (after proper scaling). This closeness can be measured using the confidence interval: $\hat{y}(\mathbf{x}) \pm z_{1-\alpha/2}s(\mathbf{x})$, where $z_{1-\alpha/2}$ is the upper $(1-\alpha/2)$th quantile of the standard normal distribution. Thus, we can aim to minimize the width of the confidence interval $2z_{1-\alpha/2}s(\mathbf{x})$. Since this width varies with respect to $\mathbf{x}$, we can take the average:

$$\min_D \int_{\mathcal{X}} s(\mathbf{x})w(\mathbf{x})\, d\mathbf{x},$$

where $w(\mathbf{x})$ is a weight function that captures the importance of the response surface at $\mathbf{x}$. For the moment, we will assume that we are interested in the whole region $\mathcal{X}$ and therefore assign equal importance to all $\mathbf{x} \in \mathcal{X}$. Since τ does not affect the optimization, we will take it as 1. Thus, the optimal design criterion is to minimize the integrated root mean squared error

$$IRMSE(D;\boldsymbol{\theta}) = \int_{\mathcal{X}} \sqrt{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})}\, d\mathbf{x}. \tag{3.1}$$

Note that this criterion is independent of μ and τ^2, but is dependent on the correlation parameters $\boldsymbol{\theta}$. Although (3.1) is more interpretable (Joseph et al., 2019), it is mathematically more convenient to use the integrated mean squared error for finding the optimal designs (Sacks et al., 1989):

$$\begin{aligned} IMSE(D;\boldsymbol{\theta}) &= \int_{\mathcal{X}} \{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}\, d\mathbf{x}. \\ &= 1 - tr\left\{\mathbf{R}^{-1}\int_{\mathcal{X}} \mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x})'\, d\mathbf{x}\right\}, \end{aligned} \tag{3.2}$$

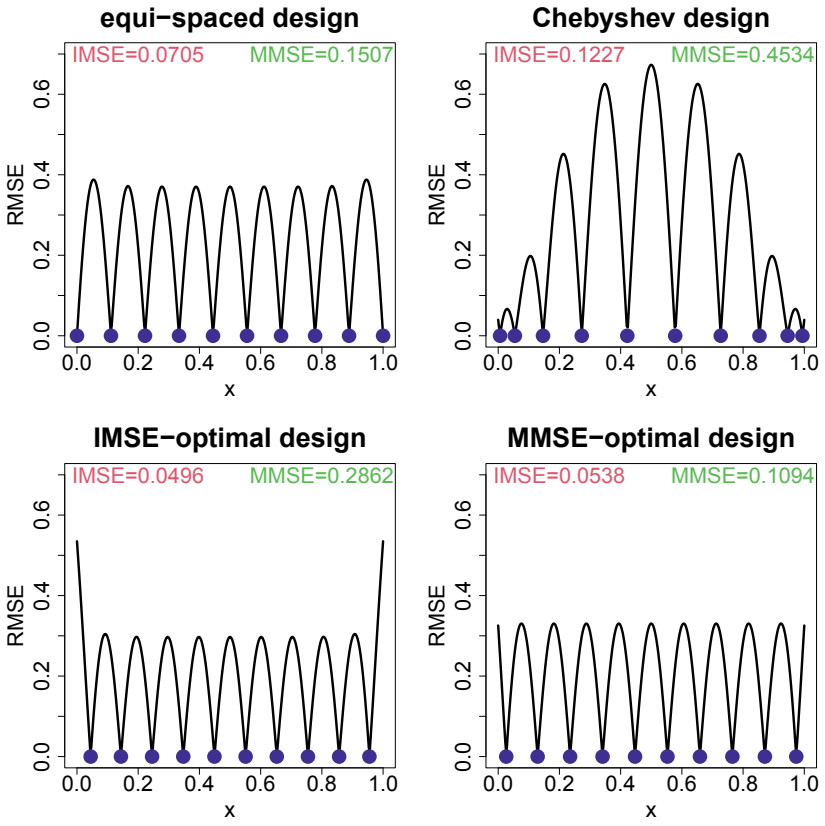


FIGURE 3.2: Root mean squared errors for four designs. The y-axis is proportional to the width of the confidence intervals if we were to fit a GP model with $R(\Delta) = \exp(-\Delta^2/\theta^2)$ and $\theta = 0.1$ using the data based on the design.

where $\mathcal{X} = [0,1]^p$. This criterion can be analytically integrated for certain choices of the correlation function, such as the Gaussian correlation (Leatherman et al., 2018), which can simplify optimization.

Consider a one-dimensional problem that is easy for us to visualize and understand the design criterion. The IMSE criterion can only be evaluated after specifying a correlation function and its parameters. Let $R(x) = \exp(-x^2/\theta^2)$ with $\theta = 0.1$. The top left panels of Figure 3.2 show the RMSE (same as the scaled-posterior standard deviation $s(\mathbf{x})/\tau$) for a 10-run equi-spaced design:

$$D_1 = \left\{ \frac{i-1}{n-1} \right\}_{i=1}^{n}$$

and a 10-run Chebyshev design

$$D_2 = \left\{ .5 + .5 \cos\left(\pi \frac{i-.5}{n} \right) \right\}_{i=1}^{n}$$

with $n = 10$. We can see that the RMSE is 0 at the design points and increases as the prediction point moves away from the design point. The RMSE is small for the Chebyshev design near the boundaries because more design points are allocated near the boundaries, but it increases the RMSE around the center. We can compare the two designs by computing the IMSE in (3.2). Using numerical integration, we obtain $IMSE(D_1; 0.1) = 0.0705$ and

$IMSE(D_2; 0.1) = 0.1227$, showing that the equi-spaced design is better. We can optimize the IMSE criterion to find the best possible design. This optimization of IMSE is difficult because the objective function contains $\mathbf{R}^{-1}$, requires numerical integration, and is multi-modal. We have used the derivative-free limited-memory BFGS optimization algorithm in the R function `optim` to find the optimal design by initializing the optimization using the equi-spaced design. The optimal design (D_3) is shown in the bottom left panel of the same figure, which gives $IMSE(D_3; 0.1) = 0.0496$. We can see that this design pulled the points to the center, which made the RMSE at the boundaries large, but overall it has a smaller IMSE.

Noting that

$$\int_{\mathcal{X}} \{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}\, d\mathbf{x} \leq \max_{\mathbf{x}\in\mathcal{X}} \{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\},$$

we can define another criterion, which is to minimize the maximum mean squared error (Sacks et al., 1989):

$$MMSE(D; \boldsymbol{\theta}) = \max_{\mathbf{x}\in\mathcal{X}} \{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}. \tag{3.3}$$

The IMSE and MMSE designs can be seen as the Gaussian process analogs of the I-optimal (2.37) and G-optimal designs (2.36) for linear regression models.

The optimal design (D_4) obtained by minimizing MMSE is shown on the bottom right panel of Figure 3.2. We can see that this design minimizes the largest uncertainty, which is obtained by slightly pushing the points toward the boundaries compared to the IMSE design. The $MMSE$ values of all four designs are also shown in Figure 3.2.

All of the foregoing computations were performed assuming a Gaussian correlation function with $\theta = 0.1$. In reality, we will not be able to guess the length-scale parameter before doing experiments. Therefore, it is important to find out if this parameter has an effect on optimal designs. The IMSE and MMSE values are plotted in the top panels of Figure 3.3. We can see that they change a lot with respect to θ. When θ is small or large, all designs perform similarly. This makes sense because small θ implies a rough surface, where all the 10-run designs perform poorly in approximating the response surface. Similarly, when θ is high, we have a smooth response surface and all designs do a good job of approximating it. It is in the middle region of θ, they behave differently. As in Sacks et al. (1989), we can define the relative efficiency to better discern the designs:

$$\begin{aligned} RE_I(D_i; \boldsymbol{\theta}) &= \frac{\min_i IMSE(D_i; \boldsymbol{\theta})}{IMSE(D_i; \boldsymbol{\theta})}, \\ RE_M(D_i; \boldsymbol{\theta}) &= \frac{\min_i MMSE(D_i; \boldsymbol{\theta})}{MMSE(D_i; \boldsymbol{\theta})}, \end{aligned}$$

for $i = 1, 2, 3, 4$.

The relative efficiencies of the four designs with respect to IMSE and MMSE are plotted in the bottom panels of Figure 3.3. We can see that different designs become optimal depending on the value of θ. For example, the Chebyshev design performs poorly with respect to IMSE when $\theta < 0.3$, but is the best when $\theta > 0.3$. This is bad news for our optimal designs! However, there is some hope. The equi-spaced design and the MMSE-optimal design perform reasonably well for all θ even if they are not the best. Thus, we need to find designs that are robust to θ, which can be formulated as

$$\max_{D} \min_{\boldsymbol{\theta}\in\Theta} \frac{IMSE(D^*(\boldsymbol{\theta}); \boldsymbol{\theta})}{IMSE(D; \boldsymbol{\theta})},$$

where

$$D^*(\boldsymbol{\theta}) = \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} IMSE(D; \boldsymbol{\theta})$$

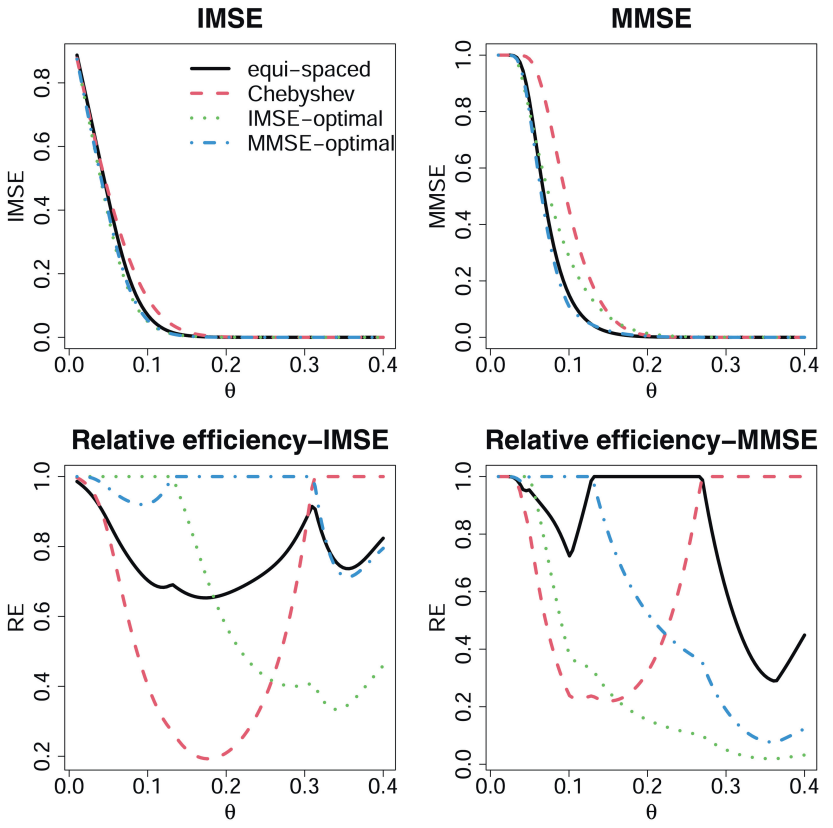


FIGURE 3.3: Relative efficiency of the four designs in Figure 3.2 with respect to IMSE and MMSE. The IMSE and MMSE optimal designs are found by fixing $\theta = 0.1$.

and $\Theta \subset [0, \infty)^p$. Robust MMSE-optimal designs can be defined similarly. This is an extremely difficult problem to solve, as we need to find IMSE- or MMSE-optimal designs for many values of $\boldsymbol{\theta}$ even if we discretize the p-dimensional space Θ.

As can be seen in Figure 3.3, all the designs perform poorly when $\theta \to 0$, that is, when the correlations go to 0. This is the case of near independence. For this case, $\mathbf{R} \to \mathbf{I}$, the identity matrix, and therefore, there is no need to invert the $\mathbf{R}$ matrix to compute the IMSE or MMSE criteria. This substantially simplifies the computational burden. Furthermore, the designs found in this limiting case seem to work reasonably well for large correlations as well. Thus, these designs, called space-filling designs, are much easier to construct and seem to be robust to misspecifications of the correlation structure. Therefore, we will use space-filling designs as initial designs in computer experiments. However, before delving into space-filling designs, we need to learn about one more important concept in model-based designs.

3.2 Maximum Entropy Designs

Maximum entropy designs were proposed by Shewry and Wynn (1987) for spatial sampling, which is introduced to the field of computer experiments by Currin et al. (1991). Entropy is

a measure of uncertainty (Shannon, 1948). Consider a discrete set $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ and denote the random vector $(Y_1, \ldots, Y_N)$ by $\mathbf{Y}_\mathcal{X}$, where $Y_i = h(\mathbf{X}_i)$ for some random function $h(\cdot)$. Then, the entropy of $\mathbf{Y}_\mathcal{X}$ is defined as

$$Ent(\mathbf{Y}_\mathcal{X}) = -\int f(\mathbf{y}) \log f(\mathbf{y})\, d\mathbf{y},$$

where $f(\mathbf{y})$ is the probability density function of $\mathbf{Y}_\mathcal{X}$. The objective is to choose a design D, which is a subset of $\mathcal{X}$. The design D gives the data $\mathbf{Y}_D$. We aim to choose a D such that it will minimize the entropy of the unobserved data: $\mathbf{Y}_{\mathcal{X}\setminus D}$ given the observed data $\mathbf{Y}_D$. The entropy has the following decomposition (Cover and Thomas, 1991):

$$Ent(\mathbf{Y}_\mathcal{X}) = Ent(\mathbf{Y}_D) + E\{Ent(\mathbf{Y}_{\mathcal{X}\setminus\mathcal{D}}|\mathbf{Y}_D)\}.$$

Since the left side is independent of D, the second term on the right side can be minimized by maximizing $Ent(\mathbf{Y}_D)$ with respect to D. This gives a maximum entropy design.

Assume that the input-output relationship is a realization of a GP: $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$, where μ,τ^2, and the correlation parameters $\boldsymbol{\theta}$ are known. Let $D = \{\mathbf{x}_i\}_{i=1}^n$ be the design and $\mathbf{y} = (y_1, \ldots, y_n)'$ the data. Then

$$\mathbf{y} \sim \mathcal{N}_n(\mu\mathbf{1}, \tau^2\mathbf{R}),$$

where $\mathbf{R} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$. Our aim is to choose D to maximize the entropy of $\mathbf{y}$. The entropy of $\mathbf{y}$ can be computed as:

$$\begin{aligned}
Ent(\mathbf{y}) &= -\int \left\{ -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\tau^2 - \frac{1}{2}\log|\mathbf{R}| - \frac{1}{2\tau^2}(\mathbf{y}-\mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1}) \right\} f(\mathbf{y}) d\mathbf{y} \\
&= \frac{n}{2}\log 2\pi + \frac{n}{2}\log\tau^2 + \frac{1}{2}\log|\mathbf{R}| + \frac{1}{2\tau^2} E\left\{(\mathbf{y}-\mu\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\mu\mathbf{1})\right\} \\
&= \frac{n}{2}\log 2\pi + \frac{n}{2}\log\tau^2 + \frac{1}{2}\log|\mathbf{R}| + \frac{1}{2\tau^2} tr\left\{\mathbf{R}^{-1} var(\mathbf{y})\right\} \\
&= \frac{n}{2}\log 2\pi + \frac{n}{2}\log\tau^2 + \frac{1}{2}\log|\mathbf{R}| + \frac{n}{2},
\end{aligned}$$

where we have used the result for the expectation of a quadratic form given by $E(\mathbf{y}'\mathbf{A}\mathbf{y}) = E(\mathbf{y})'\mathbf{A}E(\mathbf{y}) + tr\{\mathbf{A}var(\mathbf{y})\}$. Ignoring the constants and logarithm that would not affect the optimization, define the entropy of $\mathbf{y}$ as

$$Ent(\mathbf{y}) \coloneqq |\mathbf{R}|, \tag{3.4}$$

Thus, the maximum entropy design can be obtained as

$$D^* = \underset{D}{\operatorname{argmax}}\, |\mathbf{R}|. \tag{3.5}$$

A 10-run entropy design is obtained for the one-dimensional problem with $R(x) = \exp(-x^2/\theta^2)$ and $\theta = 0.1$. It is shown in the right panel of Figure 3.4 together with the RMSE profile. The optimal IMSE and MMSE designs in Figure 3.2 are also shown here for quick comparison. We can see that the entropy design points are pushed toward the boundaries compared to the IMSE and MMSE optimal designs. The maximum entropy design is close to the equi-spaced design in Figure 3.2, but has a smaller maximum RMSE.

Maximum entropy designs have the same practical limitations as the IMSE and MMSE designs mentioned in the previous section. First, they can be computed only after specifying the correlation parameters, which are unknown at the design stage. Second, although simpler

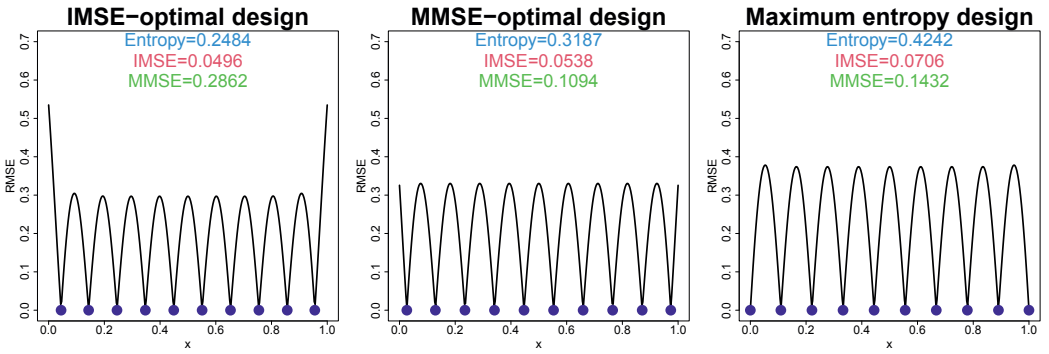


FIGURE 3.4: IMSE-optimal, MMSE-optimal, and Maximum entropy designs (blue dots) with $\theta = 0.1$.

than the IMSE/MMSE criteria, computing the determinant of $\mathbf{R}$ is still expensive because it requires $\mathcal{O}(n^3)$ operations, and therefore the optimization of entropy can be time consuming. Furthermore, the objective function can be multi-modal, making the optimization even harder. We have used local optimization algorithms with multiple starts or a good initial start to find the optimal design in Figure 3.4, but it may not be efficient in higher dimensions. Currin et al. (1991) used the DETMAX algorithm of Mitchell (1974). Advanced optimization algorithms for discrete problems can be found in Fampa and Lee (2022).

We will overcome the robustness and optimization issues of maximum entropy designs by finding them in low-correlation regimes (rough surfaces), which will lead to a type of space-filling designs discussed in the next chapter.

Exercises

1. Consider a Gaussian correlation function

$$R(\boldsymbol{\Delta}) = \exp\left\{-\sum_{l=1}^{p} \frac{\Delta_l^2}{\theta_l^2}\right\}.$$

Show that IMSE in (3.2) can be simplified to (Leatherman et al., 2018)

$$IMSE(D;\boldsymbol{\theta}) = 1 - tr\left\{\mathbf{R}^{-1}\mathbf{A}\right\}, \tag{3.6}$$

where

$$\mathbf{A}_{ij} = \sqrt{\mathbf{R}_{ij}} \prod_{l=1}^{p} \sqrt{\frac{\pi}{2}}\theta_l \left\{\Phi\left(\frac{2-(x_{il}+x_{jl})}{\theta_l}\right) - \Phi\left(\frac{-(x_{il}+x_{jl})}{\theta_l}\right)\right\}$$

and $\Phi(\cdot)$ is the standard normal distribution function.

2. Show that the IMSE in (3.6) can be written as (Binois et al., 2019)

$$IMSE(D;\boldsymbol{\theta}) = 1 - sum\left\{\mathbf{R}^{-1} \odot \mathbf{A}\right\},$$

where $sum\{\mathbf{A}\} = \sum_i \sum_j \mathbf{A}_{ij}$ and $\mathbf{A} \odot \mathbf{B}$ is the Hadamard (element-wise) product of $\mathbf{A}$ and $\mathbf{B}$. Discuss the computational savings of this expression compared to (3.6).

3. In robust parameter design experiments, the factors are classified as control factors $\mathbf{x}$ and noise factors $\mathbf{z}$. Cross arrays are widely used to design such experiments. See Wu and Hamada (2021, Ch. 11) for details. Assume a product correlation between $\mathbf{x}$ and $\mathbf{z}$, that is, $R(\mathbf{x}_i-\mathbf{x}_j,\mathbf{z}_i-\mathbf{z}_j)=R_x(\mathbf{x}_i-\mathbf{x}_j)R_z(\mathbf{z}_i-\mathbf{z}_j)$. Then, show that the IMSE-optimal cross array design D can be obtained by crossing the IMSE-optimal design D_x and IMSE-optimal design D_z (Joseph et al., 2019). This result is useful in practice because a large optimization can be divided into two smaller optimizations.

4. There are some factors whose nominal values can be controlled, but they can vary around their nominal values. Such factors are said to have internal noise. Examples include, part-to-part variability within their manufacturing tolerances and process parameter variability around its target. On the other hand, external noise factors considered in the previous exercise are completely uncontrollable including their nominal values. Unlike the external noise factors, we do not need a cross array design to incorporate internal noise factors. A factor with internal noise can be represented as $X=x+e$, where the nominal value x is controllable and the internal noise e is uncontrollable. Assume that $e\sim\mathcal{N}(0,\sigma^2)$. Then, under the Gaussian correlation function $R(\Delta)=\exp\{-\Delta^2/\theta^2\}$, the IMSE-optimal design $D=\{x_1,\ldots,x_n\}$ can be obtained by minimizing (Joseph et al., 2019)

$$\begin{aligned} IMSE(D;\boldsymbol{\theta}) &= \int_0^1\int_{-\infty}^{\infty}\{1-\mathbf{r}(x+e)'\mathbf{R}^{-1}\mathbf{r}(x+e)\}\frac{\phi^2(e)}{2\sigma\sqrt{\pi}}dedx \\ &= \frac{1}{2\sigma\sqrt{\pi}}\left[1-\int_0^1 tr\left\{\mathbf{R}^{-1}\mathbf{A}(x)\right\}dx\right], \end{aligned}$$

where

$$\mathbf{A}_{ij}(x)=\frac{\theta\sqrt{\mathbf{R}_{ij}}}{\sqrt{\theta^2+2\sigma^2}}\exp\left\{-\frac{2}{\theta^2+2\sigma^2}\left(x-\frac{x_i+x_j}{2}\right)^2\right\},$$

for $i,j=1,\ldots,n$. Show that this can be further simplified to

$$IMSE(D;\boldsymbol{\theta})=\frac{1}{2\sigma\sqrt{\pi}}\left[1-tr\left\{\mathbf{R}^{-1}\overline{\mathbf{A}}\right\}\right],$$

where

$$\overline{\mathbf{A}}_{ij}=\frac{\theta\sqrt{\pi\mathbf{R}_{ij}}}{\sqrt{2}}\left\{\Phi\left(\frac{2-x_i-x_j}{\sqrt{\theta^2+2\sigma^2}}\right)-\Phi\left(\frac{-(x_i+x_j)}{\sqrt{\theta^2+2\sigma^2}}\right)\right\}.$$

5. For $n=7$ and $p=2$, find the

   (a) IMSE-optimal design

   (b) MMSE-optimal design

   (c) maximum entropy design

   under an isotropic Gaussian correlation function $R(\boldsymbol{\Delta})=\exp\{-\|\boldsymbol{\Delta}\|^2/\theta^2\}$ with $\theta=1$. Investigate how the designs change when θ is reduced to $0.1, 0.01, 0.001, \ldots$.

4

Space-filling Designs

Although the model-based approach gives us a principled way to construct optimal designs, it has certain practical limitations. The model-based optimal designs depend on the unknown model parameters such as the correlations parameters. Unfortunately, they are not robust to the misspecification of those parameters. Moreover, the model-based criteria are computationally expensive and numerically unstable, making their optimization difficult and computationally expensive. Therefore, in this chapter, we will consider approximations of the model-based criteria and develop designs that aim to fill the experimental region. They have nice geometric interpretations and can be motivated by different types of modeling method, which indicates that they possess some sort of model robustness.

4.1 Clustering-based Designs

Let $D = \{\mathbf{x}_i\}_{i=1}^n$ be the design and $\mathcal{X} = [0,1]^p$ the experimental region. Consider the IRMSE criterion in (3.1):

$$IRMSE(D;\boldsymbol{\theta}) = \int_{\mathcal{X}} \sqrt{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})}\, d\mathbf{x},$$

where $\mathbf{r}(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^n$, $\mathbf{R} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$, and $R(\cdot)$ is the correlation function. Partition the experimental region into n Voronoi regions: $\mathcal{X} = V_1 \cup \ldots \cup V_n$, where

$$V_i = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\| \ \ \forall j \neq i\}. \tag{4.1}$$

Since $V_i \cap V_j = \emptyset$ for $i \neq j$,

$$IRMSE(D;\boldsymbol{\theta}) = \sum_{i=1}^n \int_{V_i} \sqrt{1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})}\, d\mathbf{x}.$$

Since the posterior variance decreases as we add more points $1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})$ will be smaller than the variance obtained using only one design point (Loeppky et al., 2010), which gives

$$1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) \leq 1 - R^2(\mathbf{x} - \mathbf{x}_i) \ \forall i. \tag{4.2}$$

Thus,

$$IRMSE(D;\boldsymbol{\theta}) \leq \sum_{i=1}^n \int_{V_i} \sqrt{1 - R^2(\mathbf{x} - \mathbf{x}_i)}\, d\mathbf{x}.$$

DOI: 10.1201/9781003661535-4

This chapter has been made available under a CC-BY-NC-ND license 4.0.

Under the *isotropic* Gaussian correlation function

$$R(\boldsymbol{\Delta}) = \exp\left(-\frac{\|\boldsymbol{\Delta}\|^2}{\theta^2}\right), \tag{4.3}$$

$$\begin{aligned} IRMSE(D;\theta) &\leq \sum_{i=1}^{n} \int_{V_i} \sqrt{1 - \exp\left(-2\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\theta^2}\right)}\, d\mathbf{x} \\ &\approx \sum_{i=1}^{n} \int_{V_i} \sqrt{1 - \left\{1 - 2\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\theta^2}\right\}}\, d\mathbf{x} \\ &= \frac{\sqrt{2}}{\theta} \sum_{i=1}^{n} \int_{V_i} \|\mathbf{x}-\mathbf{x}_i\|\, d\mathbf{x}, \end{aligned}$$

where the approximation will be accurate when $\max_i \max_{\mathbf{x}\in V_i} \|\mathbf{x}-\mathbf{x}_i\| << \theta$. By minimizing the upper bound, we obtain the optimal design:

$$\min_D \sum_{i=1}^{n} \int_{V_i} \|\mathbf{x}-\mathbf{x}_i\|\, d\mathbf{x}. \tag{4.4}$$

Note that the correlation parameter has been factored out in the upper bound and, therefore, we do not need it anymore to find the optimal design. The optimization in (4.4) is the quantization problem (Graf and Luschgy, 2007), which can be solved using clustering techniques. To do this, first generate a large uniform sample $\{\mathbf{X}_i\}_{i=1}^N$ from $\mathcal{X}$. Now using Monte Carlo (MC) approximation (see Section 5.1.2), the problem reduces to solving

$$\min_D \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{n} \delta_{ij} \|\mathbf{X}_j - \mathbf{x}_i\|, \tag{4.5}$$

where $\delta_{ij} = 1$ if $\mathbf{X}_j \in V_i$ and 0 otherwise.

Similarly, the IMSE criterion in (3.2) leads to the L_2-quantization problem

$$\min_D \sum_{i=1}^{n} \int_{V_i} \|\mathbf{x}-\mathbf{x}_i\|^2\, d\mathbf{x}. \tag{4.6}$$

and its MC approximation

$$\min_D \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{n} \delta_{ij} \|\mathbf{X}_j - \mathbf{x}_i\|^2. \tag{4.7}$$

The L_2-quantization problem is easier to solve mathematically and was proposed in communications theory by Lloyd (1957) and Max (1960). It is also related to the idea of optimum stratified sampling (Dalenius, 1950; Cox, 1957) and representative points of a distribution (Flury, 1990; Fang and Wang, 1993). Eq. (4.7) is the most popular criterion used in cluster analysis and can be solved using Lloyd's `kmeans` algorithm, which alternates between finding the Voronoi regions and their means. However, the problem of minimizing the sum of distances in (4.5) seems to have appeared much earlier in facility location analysis and computational geometry (Weber, 1929). An algorithm similar to k-means that uses iterative least squares for computing the centers can be used to solve (4.5), which is known as Weiszfeld's algorithm.

Suppose that we want to find $n = 7$ points in the experimental region $\mathcal{X} = [0,1]^2$. The clustering-based optimal design can be found using our R package `SFDesign` (Wang and Joseph, 2025c). The design using (4.4) is shown in Figure 4.1. We can see that the points

fill the space nicely. However, the points seem to be pulled too much toward the center, which may not be desirable in prediction problems. This is because most statistical and machine learning models are good for interpolation (predicting inside the convex hull of the 7 points), but extrapolation can be unreliable (predicting outside the convex hull). Lekivetz and Jones (2015) tried to overcome this issue by choosing a point from each of the Voronoi regions that is not at its centroid using a secondary criterion such as MaxPro (Joseph et al., 2015) to be discussed later.

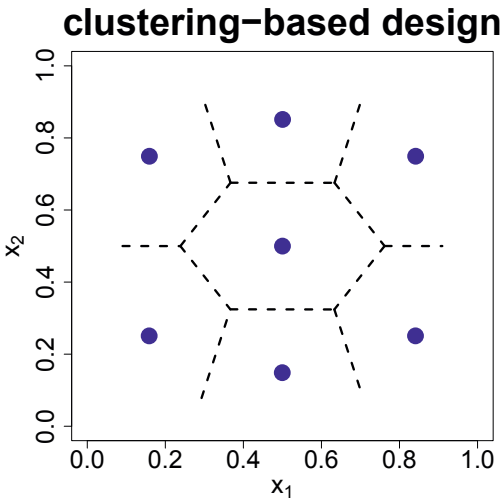


FIGURE 4.1: A 7-point clustering-based design. The Voronoi regions are also shown.

4.2 Minimax Designs

Consider the MMSE criterion in (3.3):

$$MMSE(D;\boldsymbol{\theta}) = \max_{\mathbf{x}\in\mathcal{X}}\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\}.$$

By partitioning the experimental region $\mathcal{X}$ using Voronoi regions as in (4.1), we obtain

$$\begin{aligned} MMSE(D;\boldsymbol{\theta}) &= \max_{i=1:n}\max_{\mathbf{x}\in V_i}\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\} \\ &\leq \max_{i=1:n}\max_{\mathbf{x}\in V_i}\{1-R^2(\mathbf{x}-\mathbf{x}_i)\} \\ &= 1-\min_{i=1:n}\min_{\mathbf{x}\in V_i} R^2(\mathbf{x}-\mathbf{x}_i), \end{aligned}$$

where we have used the same upper bound for the variance as in (4.2).

Suppose that we use the isotropic Gaussian correlation function in (4.3), then

$$\begin{aligned} MMSE(D;\theta) &\leq 1-\min_{i=1:n}\min_{\mathbf{x}\in V_i}\exp\left(-2\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\theta^2}\right) \\ &= 1-\exp\left(-2\max_{i=1:n}\max_{\mathbf{x}\in V_i}\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\theta^2}\right) \\ &= 1-\exp\left(-2\frac{\phi^2(D)}{\theta^2}\right), \end{aligned} \tag{4.8}$$

where

$$\phi(D) = \max_{i=1:n} \max_{\mathbf{x} \in V_i} \|\mathbf{x} - \mathbf{x}_i\|$$

is known as the *fill distance* (or covering radius) of the design D. A design that minimizes the fill distance is known as the minimax distance design or simply the minimax design (Johnson et al., 1990), that is,

$$\min_{D} \max_{i=1:n} \max_{\mathbf{x} \in V_i} \|\mathbf{x} - \mathbf{x}_i\|. \tag{4.9}$$

Such a design minimizes the upper bound of the MMSE in (4.8). Johnson et al. (1990) defines the minimax design in a slightly different way:

$$\min_{D} \max_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{x}_i \in D} \|\mathbf{x} - \mathbf{x}_i\|. \tag{4.10}$$

These two definitions are equivalent because

$$\begin{aligned} \max_{\mathbf{x} \in \mathcal{X}} \left(\min_{\mathbf{x}_i \in D} \|\mathbf{x} - \mathbf{x}_i\| \right) &= \max_{j=1:n} \max_{\mathbf{x} \in V_j} \left(\min_{\mathbf{x}_i \in D} \|\mathbf{x} - \mathbf{x}_i\| \right) \\ &= \max_{j=1:n} \left(\max_{\mathbf{x} \in V_j} \min_{\mathbf{x}_i \in D} \|\mathbf{x} - \mathbf{x}_i\| \right) \\ &= \max_{j=1:n} \max_{\mathbf{x} \in V_j} \|\mathbf{x} - \mathbf{x}_j\|. \end{aligned}$$

Thus, following (4.9), in order to find a minimax design, we need to first find the farthest point in each of the Voronoi regions. These are the points that are far away from the design points, and therefore, it is difficult to make accurate predictions at these points. The worst point is the farthest one. A minimax design tries to minimize the distance to the worst point. There could be more than one worst point in the experimental region. The least number of design points surrounding such a worst point is called the (minimax) index of the design. We want the index to be large because the prediction at the worst points is very difficult, but this difficulty reduces if we have more design points around it. Johnson et al. (1990) proved the following result.

Theorem 4.1 *Consider the correlation function $R(\mathbf{u}, \mathbf{v}) = \rho^k(d(\mathbf{u}, \mathbf{v}))$, where d is a distance metric, ρ is a decreasing function, and $k > 0$. Then, a minimax design of highest index is asymptotically MMSE-optimum as $k \to \infty$.*

We can see that the isotropic Gaussian correlation function we have used is a special case of the correlation function in this theorem. In particular, if we use the L_q distance metric $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_q$, where $\|\mathbf{x}\|_q = \{\sum_{i=1}^{p} |x_i|^q\}^{1/q}$, the minimax design becomes

$$\min_{D} \max_{i=1:n} \max_{\mathbf{x} \in V_i} \|\mathbf{x} - \mathbf{x}_i\|_q, \tag{4.11}$$

Unless otherwise mentioned, we will use Euclidean distance to find the minimax design.

To give an example, suppose that a petroleum company is planning to start some gas stations in a certain region. A minimax design would suggest placing the gas stations so that the distance of the farthest customer to his/her closest station is minimized. Thus, a minimax design will ensure that no customer is too far from one of company's gas stations. A minimax design of highest index would imply that the farthest customer is given the greatest possible choice in alternative gas stations.

It is easy to find minimax designs when $p = 1$. For example, for $n = 2$, the minimax design in $\mathcal{X} = [0, 1]$ is $\{.25, .75\}$. The worst points are $x = 0, .5$, and 1. All of them are

.25 away from the two design points, which is the minimum of the worst distances of any possible design. More generally, an n-point minimax design is given by

$$D = \left\{ \frac{i - .5}{n} \right\}_{i=1}^{n}. \tag{4.12}$$

In general, finding minimax designs for $p \geq 2$ is very difficult (but much easier than MMSE designs). Figure 4.2 shows the minimax design for $n = 7$ and $p = 2$. Compared to the clustering-based designs in Figure 4.1, the points are pushed toward the boundaries. Balls of radius $\phi(D)$ (the fill distance) are also plotted around each design point. We can see that these balls cover the entire region $\mathcal{X}$. In fact, a minimax design gives identical balls with a minimum radius to cover the entire experimental region. Therefore, the minimax design is also known as a covering design.

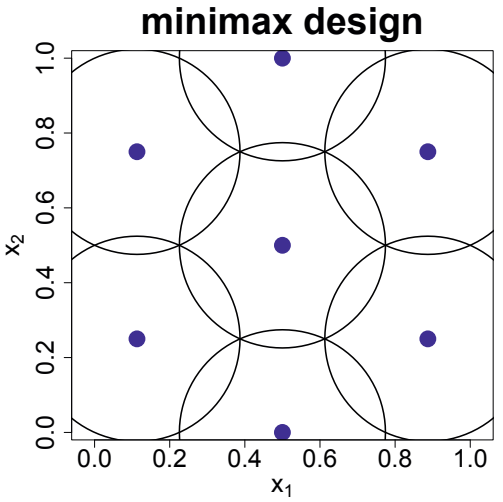


FIGURE 4.2: A 7-point minimax design.

Minimax designs overcome some of the practical limitations of model-based optimal designs. First, they do not need the specification of the length-scale parameter (θ). Theorem 4.1 indicates that they are close to optimal when $\theta \approx 0$ because θ is inversely proportional to k for Gaussian correlation function. The example in Figure 3.3 shows that designs generated using small values of θ are more likely to be robust compared to designs generated using large θ. Moreover, minimax designs can be thought of as optimal designs for modeling methods unrelated to GP such as a nearest-neighbor predictor (1-NN), because the predictive ability of a 1-NN predictor depends on how close the design point is to the prediction point. In fact, minimax designs can be motivated using purely geometric considerations (designs that minimize the gaps in the region) without using any modeling methods, which again points to its robustness.

Second, minimax designs are easier to generate than their counterpart MMSE-optimal designs because we do not need to compute $\mathbf{R}^{-1}$, which can be computationally expensive and numerically unstable for large n. Nevertheless, construction of minimax designs is still expensive for large n and p. Tan (2013) proposed a binary integer programming method to find minimax designs from a given set of candidate points. However, it becomes computationally intractable as the number of candidate points increases. Mak and Joseph (2018a)

proposed a clustering-based method to generate minimax designs. First, generate a large uniform sample $\{\mathbf{X}_j\}_{j=1}^N$ from $\mathcal{X}$. Then, for large q,

$$\begin{aligned}\left(\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{n}\delta_{ij}\|\mathbf{X}_j-\mathbf{x}_i\|^q\right)^{1/q} &\approx \max_{i=1:n}\max_{j=1:N}\delta_{ij}\|\mathbf{X}_j-\mathbf{x}_i\| \\ &= \max_{i=1:n}\max_{\mathbf{X}_j\in V_i}\|\mathbf{X}_j-\mathbf{x}_i\|,\end{aligned}$$

where $\delta_{ij}=1$ if $\mathbf{X}_j\in V_i$ and 0 otherwise. This shows that we can obtain the minimax design using Lloyd's k-means type algorithm by setting q to be a large value (but not too large to cause numerical instability). Since Lloyd's algorithm is likely to converge to a local optimum, they further improve the design using particle swarm optimization. Their implementation of the algorithm is available in the R package `minimaxdesign` (Mak, 2021). A more efficient method to construct minimax designs using a special class of lattice designs is proposed in He (2017a), but it works only for small p. See the R package `LatticeDesign`.

4.3 Maximin Designs

Consider the maximum entropy criterion in (3.5):

$$\max_D |\mathbf{R}|. \tag{4.13}$$

Let $\lambda_1,\ldots,\lambda_n$ be the eigenvalues of $\mathbf{R}$. Since the geometric mean is always smaller than the arithmetic mean,

$$\begin{aligned}|\mathbf{R}| &= \prod_{i=1}^{n}\lambda_i \le \left(\frac{1}{n}\sum_{i=1}^{n}\lambda_i\right)^n \\ &= \left(\frac{1}{n}tr\{\mathbf{R}\}\right)^n = 1 = |\mathbf{I}_n|.\end{aligned}$$

Thus, a maximum entropy criterion tries to make the correlation matrix as close as possible to an identity matrix; that is, it tries to minimize the off-diagonal elements of $\mathbf{R}$. Thus, a surrogate criterion to (4.13) is

$$\min_D \|\mathbf{R}-\mathbf{I}_n\|_\alpha, \tag{4.14}$$

where $\|\cdot\|_\alpha$ is the L_α-norm. An advantage of (4.14) over (4.13) is that the objective function of (4.14) can be computed in $\mathcal{O}(n^2)$ compared to $\mathcal{O}(n^3)$ of (4.13). This is a great computational benefit for optimization.

Consider the L_∞ norm. Since $\mathbf{R}$ is a positive matrix (for most commonly used correlation functions), (4.14) becomes

$$\min_D \max_{i\neq j}|\mathbf{R}_{ij}| = \min_D \max_{i\neq j}\mathbf{R}_{ij}. \tag{4.15}$$

Suppose that we use the isotropic Gaussian correlation function in (4.3). Then,

$$\begin{aligned}\max_{i\neq j}\mathbf{R}_{ij} &= \max_{i\neq j}\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\theta^2}\right) \\ &= \exp\left(-\min_{i\neq j}\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\theta^2}\right), \\ &= \exp\left(-4\frac{\psi^2(D)}{\theta^2}\right),\end{aligned}$$

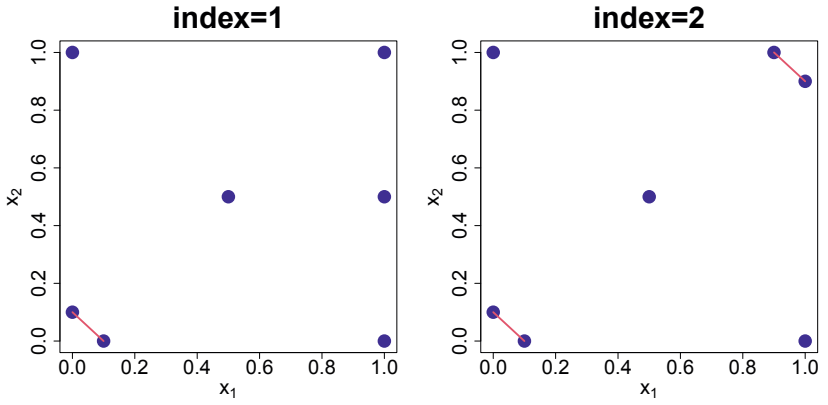


FIGURE 4.3: Two 7-point designs with the same separation distance but with different maximin indexes.

where

$$\psi(D) \coloneqq \frac{1}{2} \min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|$$

is called the *separation distance* (or packing radius) of the design D (Fasshauer, 2007, p. 136). A design that maximizes the separation distance is called a maximin distance design or simply a maximin design (Johnson et al., 1990), that is

$$D^* = \operatorname*{argmax}_D \min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|. \tag{4.16}$$

Thus, a maximin design tries to maximize the minimum pairwise distance among all the design points. Such a design minimizes the maximum off-diagonal correlation of $\mathbf{R}$, which approximately maximizes the entropy of the design. From a geometric point of view, by separating the points as much as possible, the hope is that a maximin design will fill the entire experimental region $\mathcal{X}$. Going back to the gas station example that we saw in the previous section, a maximin design ensures that no two gas stations are too close to each other. It minimizes the competition from each other by locating the stations as far apart as possible. Thus, a maximin design is more supplier-focused, whereas a minimax design is more customer-focused.

A maximin design need not be unique. Figure 4.3 shows two designs with the same separation distance, but clearly the first design is better because it has only a single pair of points with the minimum distance, while the second design has two pairs. Call the number of pairs of points at distance $2\psi(D)$ as the (maximin) index of the design. We want this index to be as small as possible because we do not want too many points to be close to each other. Thus, we can use the index of the design as a secondary criterion to refine the maximin designs. Johnson et al. (1990) proved the following result.

Theorem 4.2 *Consider the correlation function* $R(\mathbf{u}, \mathbf{v}) = \rho^k(d(\mathbf{u}, \mathbf{v}))$*, where* d *is a distance metric,* ρ *is a decreasing function, and* $k > 0$*. Then, a maximin design of lowest index is asymptotically a maximum entropy design as* $k \to \infty$*.*

Now consider the rational quadratic (RQ) correlation function (Rasmussen and Williams, 2006, p. 86) in Table 2.1, also known as inverse quadratic or Cauchy kernel (Fasshauer and

McCourt, 2015, p. 42):

$$R(\boldsymbol{\Delta}) = \frac{1}{\left(1 + \frac{\|\boldsymbol{\Delta}\|^2}{\theta^2}\right)}. \tag{4.17}$$

Then, the objective function in (4.14) becomes

$$\begin{aligned} \|\mathbf{R} - \mathbf{I}_n\|_\alpha^\alpha &= \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{\left(1 + \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\theta^2}\right)^\alpha} \\ &\leq \theta^{2\alpha} \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|^{2\alpha}}. \end{aligned}$$

The second term in the upper bound, the Total Reciprocal Distance (TRD), is interesting because it does not depend on θ and can be used as an alternative criterion to (4.16). It is easy to see that for large α,

$$\begin{aligned} \min_D TRD(D) &= \min_D \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|^{2\alpha}} \\ &\approx \min_D \frac{I}{\min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|^{2\alpha}} \\ &\equiv \max_D \frac{\min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|}{I^{1/(2\alpha)}}, \end{aligned} \tag{4.18}$$

where I is the index of the design. Thus, for α large enough, minimizing $TRD(D)$ not only maximizes the minimum distance but also minimizes the index of the design. In this sense, TRD seems to be a much better criterion to use than the maximin criterion in (4.16) (Morris and Mitchell, 1995). An advantage of (4.16) is that its objective function can be computed in $\mathcal{O}(n \log n)$ as opposed to $\mathcal{O}(n^2)$ for $TRD(D)$. The special case of $TRD(D)$ with $\alpha = 1$ is a criterion proposed by Audze and Eglais (1977), which has a physical analogy of minimizing the gravitational force among particles of equal mass placed inside a box.

As in the case of minimax designs, it is easy to find maximin design in one-dimensional problems. An n-point maximin design is given by

$$D = \left\{ \frac{i-1}{n-1} \right\}_{i=1}^{n}. \tag{4.19}$$

The problem becomes difficult for $p \geq 2$ except for some special choices of n such as $n = 2^p$. Optimizing $TRD(D)$ is easier because its gradient can be obtained as

$$\frac{\partial TRD(D)}{\partial \mathbf{x}_k} = 4\alpha \sum_{j \neq k} \frac{(\mathbf{x}_j - \mathbf{x}_k)}{\|\mathbf{x}_j - \mathbf{x}_k\|^{2\alpha+2}}, \quad k = 1, \ldots, n,$$

which can be used to perform a gradient-based optimization. Since $TRD(D)$ is multimodal, this optimization is likely to converge to a local optimum. Therefore, multiple random starts are needed to find the global optimum. To find the maximin design in (4.16), we can use a derivative-free global optimization, which can be slow for large n and p. This algorithm is implemented in the R package `SFDesign` (Wang and Joseph, 2025c).

The maximin design for $n = 7$ and $p = 2$ obtained using `SFDesign` is shown in Figure 4.4, which has a separation distance of $\psi(D^*) = 0.26795$. Circles are drawn around each point with radius $\psi(D^*)$. A maximin design maximizes the radius of identical non-overlapping

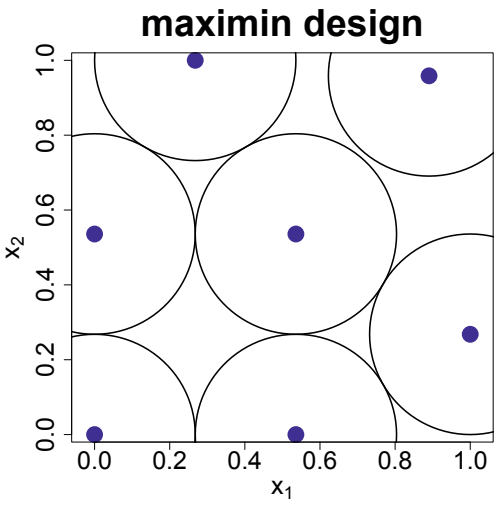


FIGURE 4.4: A 7-point maximin design.

balls that can be placed around each design point, which makes this problem related to the well-known sphere packing problem in mathematics (Conway and Sloane, 2013). However, most of the theoretical results on sphere packing are about arranging spheres in $\mathbb{R}^p$ as opposed to a hypercube in our case. The boundary constraints make the maximin designs different from the optimal sphere-packing configurations, but there has been some effort to use the results of the sphere-packing to solve the maximin design problem (He, 2017b). These are difficult mathematical problems, although the solutions for some specific cases are trivial. As Borodachov et al. (2019, p.11) put it "The proof of best-packing configurations in $\mathbb{R}^3$ was an open problem for nearly four centuries, but anyone who has ever stacked oranges already knew the answer". See the website http://www.packomania.com/ for the best sphere-packing results on squares and cubes, and the references therein.

An advantage of maximin designs over cluster-based and minimax designs is that it can be easily adapted for sequential construction. For example, we can start with a point $\mathbf{x}_1$, and then sequentially add $\mathbf{x}_2, \ldots, \mathbf{x}_n$ by

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}} \min_{i=1:k} \|\mathbf{x} - \mathbf{x}_i\|, \quad \text{for} \quad k = 1, \ldots, n-1. \tag{4.20}$$

This method was originally proposed by Kennard and Stone (1969) for physical experiments. Several variants and improvements to this algorithm are proposed in Shang and Apley (2021). Note that these are sequential construction methods for initial designs, whereas the sequential design methods to be discussed in Chapter 7 are done after acquiring all the data, including the responses.

The left panel of Figure 4.5 shows six points sequentially added to the center point using the maximin criterion. This can be done using the `maximin.augment` function in `SFDesign` (Wang and Joseph, 2025c). The separation distance of the resulting design is $\psi(D) = 0.25$, which is smaller than that of the optimal maximin design in Figure 4.4. Six more points are added to this design sequentially and we obtain the design in the right plot of Figure 4.5, which looks like a good design (but not the optimal 13-point maximin design).

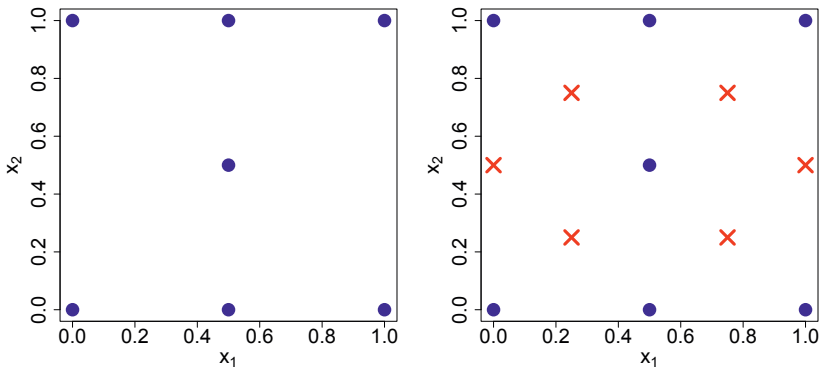


FIGURE 4.5: The left plot shows a sequentially constructed maximin design starting with the center point. The right plot shows this design augmented with another six points (red crosses) using maximin criterion.

4.4 Latin Hypercube Designs

We might be dealing with hundreds of factors, but only a few might be actually important to the output. The well-known Pareto principle says that 80% of the variability in the output can be explained using 20% of the factors in the system. In the experimental design literature, this is known as the effect sparsity principle (Wu and Hamada, 2021, p. 169). How does this commonly observed phenomenon affect the experimental design?

Consider a 4-run design with two factors. The maximin and minimax designs are shown in Figure 4.6. The cluster-based design in this case is the same as the minimax design. Now suppose x_2 is not important, that is, it has a negligible effect on the output. Then, this would mean that we have performed two experiments at $x_1 = 0$ and $x_1 = 1$ for the design on the left ($x_1 = 0.25$ and $x_1 = 0.75$ for the design on the right). Since there is no error in the output, we will get the same values for the output. This is known before doing the experiment, and therefore we have simply wasted our time performing these two experiments unnecessarily. The same situation will happen if x_1 is unimportant and only x_2 is important. How can we protect ourselves from such undesirable situations, which are quite likely to happen in practice?

If we had known x_2 to be irrelevant, then we should have performed the four experiments using an optimal design for x_1 and vice versa. Such a design is shown in the left panel of Figure 4.7, where we have used a minimax design for each input: $\{(i - .5)/4\}_{i=1}^{4}$. We can see that *projection* of the four design points on each axis gives four distinct values for the two factors. Therefore, this design will perform well even if one of the factors turned out to be unimportant after the experiment.

A design with good one-dimensional projections is easy to construct. Divide each axis into n equally spaced intervals. Then, the desired projection can be achieved by requiring that each row and column contain only one point, as shown in Figure 4.7. In higher dimensions, this can be obtained by letting

$$\left\{ \frac{\pi_j(i) - .5}{n} \right\}_{i=1}^{n}, \tag{4.21}$$

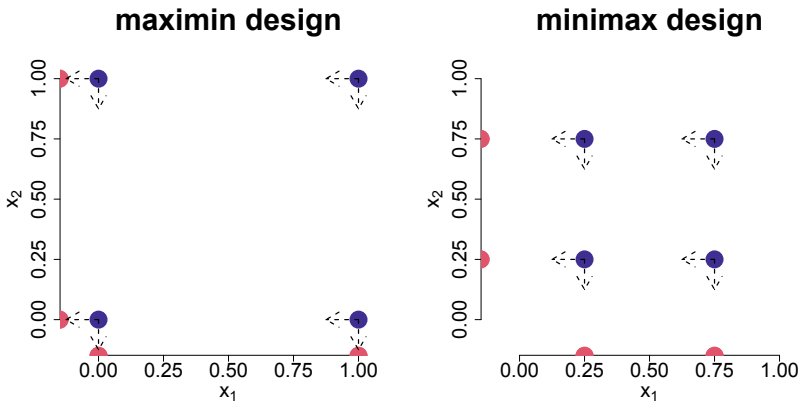


FIGURE 4.6: Maximin and minimax designs in four runs with their projections onto each axis.

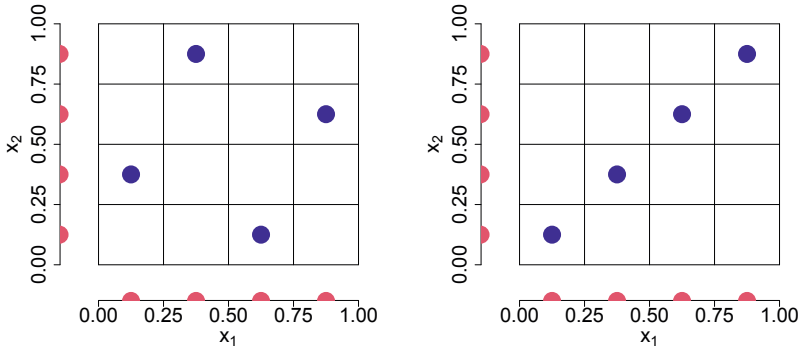


FIGURE 4.7: Latin hypercube designs in four runs with their projections shown on each axis.

for the jth column, where $\{\pi_j(1), \ldots, \pi_j(n)\}$ is a random permutation of $\{1, \ldots, n\}$. Such a design is called a Latin hypercube design (LHD) (McKay et al., 1979). It is also possible to randomly perturb the points within each cell, which can be achieved by using $\{(\pi_j(i) - u_{ij})/n\}_{i=1}^n$, where $u_{ij} \overset{iid}{\sim} \mathcal{U}(0,1)$, $i = 1, \ldots, n$ and $j = 1, \ldots, p$. From a design perspective, random perturbations are not needed and we will use the center point of each cell, that is, $u_{ij} = 0.5$ as in (4.21).

Although an LHD gives good one-dimensional projections, the random permutations used in their construction may not always be good. One such random LHD is shown on the right panel of Figure 4.7. Clearly, this LHD is not a good design because the two factors are perfectly correlated and leave large gaps in the experimental region. Tang (1993) integrated the Latin hypercube idea with orthogonal arrays (OA) to obtain a better LHD. For example, using an OA of strength two can ensure a better spread of points in two-dimensional projections. Owen (1994) tried to improve LHDs by reducing the correlations among the columns. Park (1994) proposed to search for LHDs that minimize IMSE or maximize entropy as in Chapter 3. However, this requires the knowledge of the correlation

parameters, making the optimal LHDs non-robust to their mis-specifications. Morris and Mitchell (1995) proposed to optimize the LHDs using the maximin criterion, which ensures good space-fillingness as well as good one-dimensional projections. Such a design is called a maximin LHD or in short MmLHD. In particular, they proposed to use the TRD criterion in (4.18):

$$\min_{D\in\mathcal{L}} \sum_{i=1}^{n} \sum_{j\neq i} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|^{2\alpha}},$$

where $\mathcal{L}$ is the class of LHDs. Integrating maximin with LHD turned out to be the right combination because it is simple and effective, which made MmLHDs the most popular designs for computer experiments. The design shown on the left panel of Figure 4.7 is actually the MmLHD.

Permuting the rows of a design does not alter the design. Therefore, by fixing the first column, we can obtain $(n!)^{p-1}$ random LHDs. Searching through this large number of possible designs and finding the best LHD according to the maximin criterion is not an easy problem. Morris and Mitchell (1995) used a simulated annealing algorithm for efficient global optimization. In simulated annealing, at each iteration, a new design is proposed given the current design. If the new design improves the objective function, then it is accepted with a certain probability, and the iteration continues. To create the new design, Morris and Mitchell (1995) proposed to randomly choose a column and interchange two elements of that column, which maintains the LHD structure. Jin et al. (2005) further improved the algorithm by efficient updating of distance computations within each iteration. This algorithm is implemented in the R package `SLHD` (Ba, 2015). Optimization using a genetic algorithm is provided in the R package `LHD` (Wang et al., 2022). Simulated annealing followed by a deterministic optimization step is implemented in the R package `SFDesign` (Wang and Joseph, 2025c). The construction of MmLHDs using number-theoretic methods is proposed in Wang et al. (2018). Such methods are fast, but not as flexible as algorithmic methods with respect to the choice of n and p. An integer programming approach for L_1-MmLHD is given in Vázquez and Xu (2024), which can be used to generate small designs. An MmLHD with $n = 7$ and $p = 2$ generated using the `SFDesign` package is shown in Figure 4.8. We

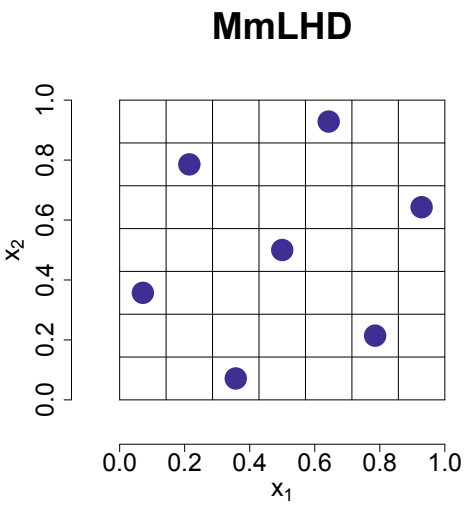


FIGURE 4.8: A 7-point MmLHD.

can see that the points nicely fill the space as the maximin design in Figure 4.4 but with better one-dimensional projections.

Several modifications of LHDs have been proposed in the literature. Ye (1998) proposed orthogonal LHDs that give LHDs with zero correlations among all columns. Joseph and Hung (2008) integrated orthogonality with maximin to produce orthogonal-maximin LHD (OMLHD). It tends to perform well if our aim is to fit a GP model which has a regression model with linear main effects as its mean or a blind kriging model (Joseph et al., 2008). OMLHDs can be generated using the `SA2008` function in the R package `LHD` (Wang et al., 2022).

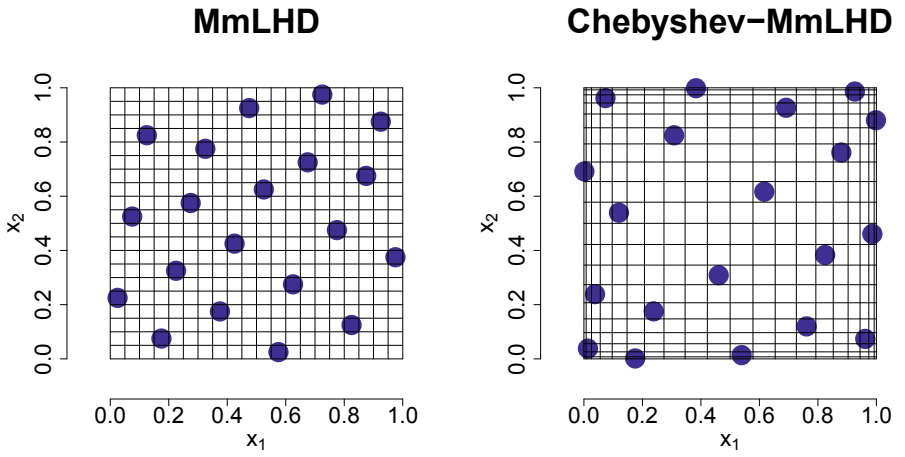


FIGURE 4.9: Left plot shows a 20-run MmLHD and the right plot the same MmLHD with Chebyshev transformation.

All of the foregoing LHDs use the centered values in (4.21), which gives a minimax design for the one-dimensional projections. Dette and Pepelyshev (2010) proposed to use the Chebyshev design for the one-dimensional projections. To integrate this idea with MmLHDs, we can first find an MmLHD and then transform the values using (2.5), that is, $D_{ij} \leftarrow \{1 + \cos(\pi D_{ij})\}/2$, where D is the MmLHD. Figure 4.9 shows a 20-rum MmLHD and MmLHD with Chebyshev transformation. We can see that the Chebyshev transformation pushes points toward the boundaries. This approach has a nice connection to the TRD criterion in (4.18). Consider the one-dimensional case, i.e., $p = 1$. It is easy to see that

$$\lim_{\alpha \to 0} \left\{ \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{|x_i - x_j|^{2\alpha}} \right\}^{1/\alpha} = -2 \sum_{i=1}^{n} \sum_{j \neq i} \log |x_i - x_j|.$$

The right side is known as the logarithmic potential energy. A design that minimizes the logarithmic potential energy criterion has the arc-sine distribution

$$f(x) = \frac{1}{\pi x \sqrt{1 - x}}$$

as $n \to \infty$ (Dette and Studden, 1992). The n-point Chebyshev design gives the best quantile approximation to this density. As we saw earlier in Figure 3.3, the Chebyshev design performs well for approximating smooth functions, but can perform poorly with rough functions. In this sense, the Chebyshev transformation does not produce a design that is robust to the mis-specification of the correlation parameters.

4.5 MaxPro Designs

Maximin designs focus on filling the p-dimensional space, and we saw that it can lead to undesirable projections on the lower dimensional subspaces. We tried to overcome the issue by constraining the class of maximin designs to LHDs, which ensures good one-dimensional projections. However, the projections of MmLHDs to subspaces with dimensions $2, \ldots, p-1$ can still be poor. Draguljić et al. (2012) attempted to improve the projection properties of MmLHD by computing the TRD criterion in (4.18) over all possible subspaces. There are a total of $p + \binom{p}{2} + \ldots + \binom{p}{p-1} = 2^p - 2$ subspaces, which can be huge for large p and can make the TRD criterion computationally intractable to optimize.

Another way to formulate this problem is to introduce weights for each variable and modify the criterion to

$$TRD_w(D) = \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{\{\sum_{k=1}^{p} w_k (x_{ik} - x_{jk})^2\}^{\alpha}},$$

where $w_i \geq 0$, $i = 1, \ldots, p$, and $\sum_{i=1}^{p} w_i = 1$. For example, if we know that x_1 and x_2 are going to be important, then we can take $w_1 = w_2 = 1/2$ and set all the other w_i's to be 0. However, the problem is that at the design stage we do not know which variables are going to be important. One way to address this problem is to use a Bayesian approach. We can assume a prior for $\mathbf{w} = (w_1, \ldots, w_{p-1})'$ and then average $TRD_w(D)$ over all possible values of $\mathbf{w}$.

Suppose that we do not have any prior information on the importance of variables. Then, it makes sense to assign a uniform prior for $\mathbf{w}$:

$$f(\mathbf{w}) = (p-1)!, \quad \text{for } \mathbf{w} \in \mathcal{W} = \{\mathbf{w} : \mathbf{w} \geq 0, \sum_{i=1}^{p-1} w_i \leq 1\}. \tag{4.22}$$

Then,

$$\int_{\mathcal{W}} TRD_w(D) f(\mathbf{w})\, d\mathbf{w} = \sum_{i=1}^{n} \sum_{j \neq i} \int_{\mathcal{W}} \frac{1}{\{\sum_{k=1}^{p} w_k (x_{ik} - x_{jk})^2\}^{\alpha}} f(\mathbf{w})\, d\mathbf{w}.$$

High-dimensional integration can also be computationally expensive. However, note that when $\alpha = p = 2$, we can get an explicit solution:

$$\int_0^1 \frac{1}{\{w_1 (x_{ik} - x_{jk})^2 + (1 - w_1)(x_{ik} - x_{jk})^2\}^2}\, dw_1 = \frac{1}{(x_{i1} - x_{j1})^2 (x_{i2} - x_{j2})^2}.$$

Now the following result can be proved using mathematical induction (Joseph et al., 2015).

Theorem 4.3 *If $\alpha = p$, then under the prior distribution in (4.22),*

$$\int_{\mathcal{W}} \frac{1}{\{\sum_{k=1}^{p} w_k (x_{ik} - x_{jk})^2\}^{\alpha}} f(\mathbf{w})\, d\mathbf{w} = \frac{1}{\prod_{k=1}^{p} (x_{ik} - x_{jk})^2}.$$

Thus, the design criterion becomes

$$\min_{D} \sum_{i=1}^{n} \sum_{j \neq i} \frac{1}{\prod_{k=1}^{p} (x_{ik} - x_{jk})^2}. \tag{4.23}$$

Since this criterion maximizes the projection capacity of the design, Joseph et al. (2015) called it a maximum projection (MaxPro) design. This criterion is not more expensive to compute than the TRD criterion in (4.18) because it just replaces the sum in the denominator with a product. The product term ensures that no two coordinates can be equal; thus, the Latin hypercube requirement is automatically enforced in the criterion.

Another justification for the MaxPro criterion can be given as follows. Suppose we use the *anisotropic* Gaussian correlation function in (2.11) with the following noninformative prior: $f(\theta_k) \propto 1/\theta_k^3$, $k = 1, \ldots, p$, then,

$$\begin{aligned} E\{\|\mathbf{R} - \mathbf{I}_n\|_1\} &\propto \sum_{i=1}^{n}\sum_{j\neq i}\int \exp\left\{-\sum_{k=1}^{p}\frac{(x_{ik}-x_{jk})^2}{\theta_k^2}\right\}\prod_{k=1}^{p}\frac{1}{\theta_k^3}d\boldsymbol{\theta} \\ &= \sum_{i=1}^{n}\sum_{j\neq i}\prod_{k=1}^{p}\int \exp\left\{-\frac{(x_{ik}-x_{jk})^2}{\theta_k^2}\right\}\frac{1}{\theta_k^3}d\theta_k \\ &\propto \sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{\prod_{k=1}^{p}(x_{ik}-x_{jk})^2}. \end{aligned} \tag{4.24}$$

We can see that the isotropic Gaussian correlation function leads to maximin criterion, whereas the anisotropic Gaussian correlation function with noninformative prior on the length-scales leads to MaxPro criterion. The product term in the denominator is not a legitimate distance metric but can be seen as the limiting case of the Minkowski distance as follows (Joseph et al., 2019):

$$\lim_{\alpha\to 0}\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_\alpha}{p^{1/\alpha}} = \left\{\prod_{k=1}^{p}|x_{ik}-x_{jk}|\right\}^{1/p}. \tag{4.25}$$

In other words, if we use the Minkowski distance metric with a small α to find maximin design using (4.16), it will be close to a MaxPro design.

Yet another justification for MaxPro designs can be obtained using a product form of rational quadratic kernels:

$$R(\boldsymbol{\Delta}) = \prod_{k=1}^{p}\left(1+\frac{\Delta_k^2}{\theta_k^2}\right)^{-1}. \tag{4.26}$$

Then,

$$\begin{aligned} \|\mathbf{R} - \mathbf{I}_n\|_1 &= \sum_{i=1}^{n}\sum_{j\neq i}\prod_{k=1}^{p}\left(1+\frac{(x_{ik}-x_{jk})^2}{\theta_k^2}\right)^{-1} \\ &= \prod_{k=1}^{p}\theta_k^{2p}\sum_{i=1}^{n}\sum_{j\neq i}\prod_{k=1}^{p}\frac{1}{\theta_k^2+(x_{ik}-x_{jk})^2} \\ &\leq \prod_{k=1}^{p}\theta_k^{2p}\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{\prod_{k=1}^{p}(x_{ik}-x_{jk})^2}, \end{aligned}$$

which is minimized by a MaxPro design for all values of $\boldsymbol{\theta}$.

Joseph et al. (2015) proposed a gradient-based optimization to obtain the MaxPro design, where the gradient of the objective function in (4.23) is given by

$$\frac{\partial}{\partial x_{rs}}\left\{\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{\prod_{k=1}^{p}(x_{ik}-x_{jk})^2}\right\} = 4\sum_{i\neq r}\frac{1}{\prod_{k=1}^{p}(x_{ik}-x_{rk})^2}\frac{1}{(x_{is}-x_{rs})}.$$

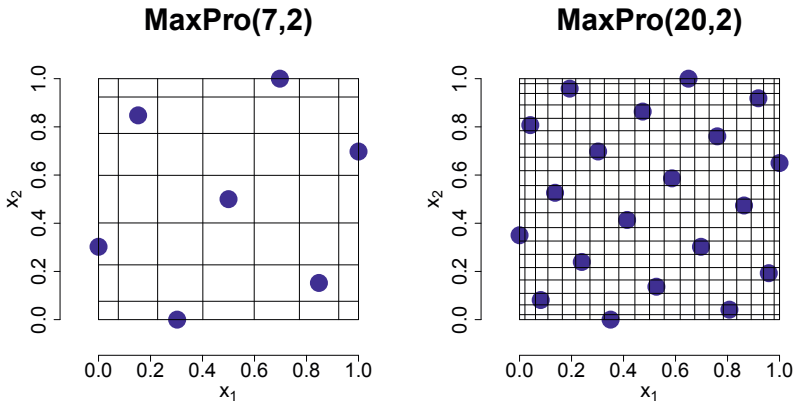


FIGURE 4.10: MaxPro designs with $n = 7$ (left) and $n = 20$ (right) for $p = 2$.

However, the objective function contains too many local optima and therefore many random starts would be needed to find the global optimum. Instead of making several random starts, they proposed using a simulated annealing algorithm similar to the one used in Morris and Mitchell (1995) to first find a MaxPro LHD and then perform the gradient-based optimization. This is implemented in the R package `MaxPro` (Ba and Joseph, 2018). The implementation in the R package `SFDesign` (Wang and Joseph, 2025c) uses the same strategy but an additional deterministic optimization is carried out after the simulated annealing step, which seems to improve the results. The MaxPro designs for $n = 7$ and $n = 20$ for $p = 2$ are shown in Figure 4.10. We can see that these designs fill the space nicely and have a Latin hypercube structure. However, the points on each axis are not equally spaced as in the LHD. The points are pushed toward the boundaries, but not to the level that we saw with the Chebyshev transformation in Figure 4.9.

Designs in higher dimensions are difficult to visualize. So, a two-dimensional projection of points of a MaxPro design with $n = 50$ in five dimensions is shown as a scatter plot matrix in Figure 4.11. We can see that the points are nicely spread out in the two-dimensional projections. The minimum of the minimum distances among the points in projections with dimensions $1, 2, 3, 4$, and in full-dimensional space are plotted in Figure 4.12 for the MaxPro and MmLHD designs. We can see that in the full-dimensional space, MmLHD has a higher minimum distance, which is expected because MmLHD is obtained by maximizing the minimum distance in the full-dimensional space. However, when the points are projected onto dimensions 2, 3, or 4, MaxPro has a larger minimum distance. This is nice because we do not know how many dimensions are going to be important before the experiment. The minimum distance for MaxPro in their one-dimensional projections is smaller than that of MmLHD because the MaxPro design has more points closer to the boundaries. In general, a MaxPro design is expected to perform better than an MmLHD in projections of dimensions $2, \ldots, p-1$, and therefore it provides better protection against uncertainty in the varying importance of variables.

Zhang et al. (2021) found that small maximin designs (small $n^{1/p}$) are not good for estimating correlation parameters. This is an artifact of poor projections of maximin designs. Figure 4.13 shows the histogram of the pairwise distance among the points of a 64-run maximin design in six dimensions. Since $64 < 2^6$, all the points are pushed to the corners of the hypercube $[0, 1]^6$ and, therefore, there are only a few distinct pairwise distances. The figure also shows the pairwise distances from a MaxPro design with 64 runs. We can

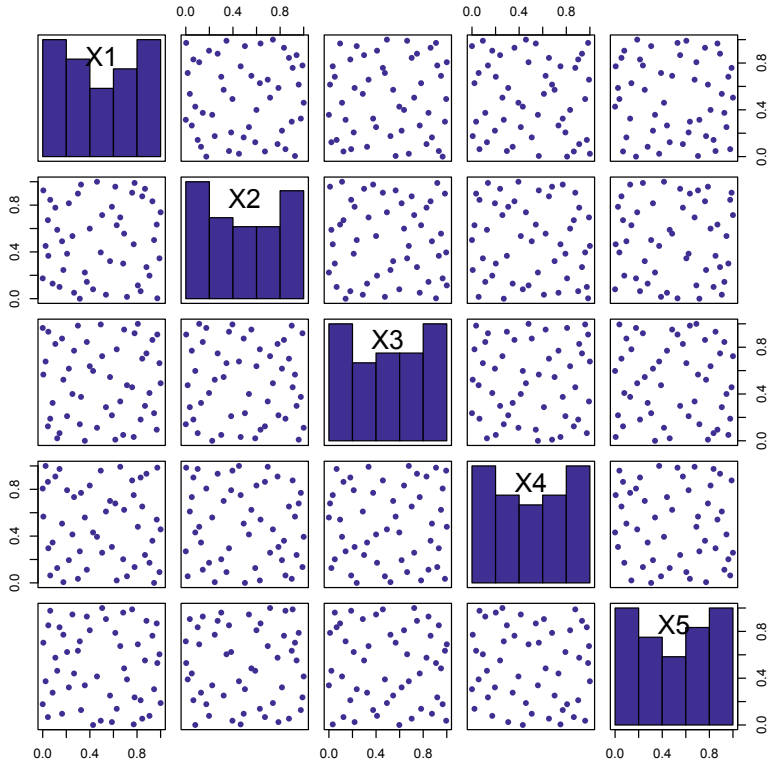


FIGURE 4.11: Scatter plot matrix of MaxPro design points with $n = 50$ and $p = 5$.

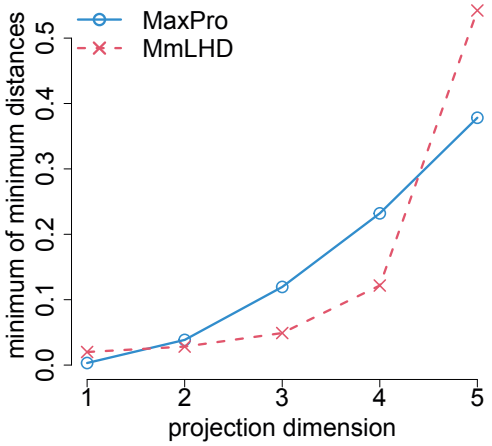


FIGURE 4.12: The minimum of minimum distances among points projected onto a subspace.

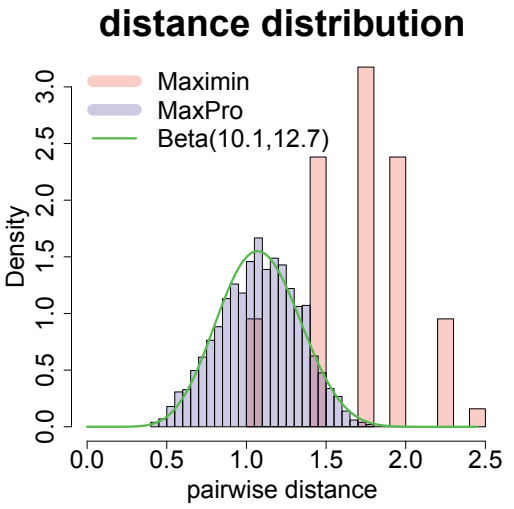


FIGURE 4.13: The histograms show the distribution of the pairwise distances among points of a Maximin (64,6) and MaxPro (64,6). The Beta density is scaled in $[0, \sqrt{6}]$.

see that the distances are almost continuous, which can be well approximated by using a scaled Beta distribution with shape parameters 10.1 and 12.7. Zhang et al. (2021) empirically found that designs whose pairwise distances with approximately a Beta distribution perform well in estimating the correlation parameters. Therefore, they have proposed optimization algorithms to generate designs whose pairwise distance distributions mimic a Beta distribution with shape parameters chosen based on n and p. Although MaxPro designs are developed aiming at prediction performance, surprisingly, MaxPro designs somehow inherited the Beta distribution property for the pairwise distances and therefore, we can expect them to perform well in the estimation of correlation parameters as well.

A big advantage of MaxPro designs over MmLHDs is that the MaxPro criterion allows for a sequential construction of the designs. We can start with $\mathbf{x}_1 \in \mathcal{X}$, and then add $\mathbf{x}_2, \ldots, \mathbf{x}_n$ one-by-one as follows (Joseph, 2016):

$$\mathbf{x}_{k+1} = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^{k} \frac{1}{\prod_{j=1}^{p} (x_j - x_{ij})^2}, \; k = 1, \ldots, n-1. \tag{4.27}$$

This approach is implemented in the R function `MaxProAugment` in the `MaxPro` package (Ba and Joseph, 2018). The greedy optimization will not produce a design better than the design obtained using the simulated annealing/gradient-based optimization. However, its flexibility allows us to solve many different kinds of problems.

As an application of the sequential MaxPro construction in (4.27), consider the problem of generating a validation set to test the goodness of a fitted model. For illustration, consider the two-dimensional Branin function taken from the Virtual Library of Simulation Experiments[1] (Surjanovic and Bingham, 2013):

$$f(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10, \; x_1 \in [-5, 10], x_2 \in [0, 15]. \tag{4.28}$$

The image of the function after rescaling the domain to $[0, 1]^2$ is shown in the left panel of

[1] https://www.sfu.ca/ ssurjano/

Figure 4.14. A 10-run MaxPro design is shown on the plot as blue dots. Now we generate 10 additional points for validation using (4.27). They are shown in the same plot as red triangles. We fitted an ordinary kriging predictor using the R package `rkriging` (Huang and Joseph, 2024)(Huang and Joseph, 2024) and then computed the root mean squared prediction error (RMSE) on the validation set. For comparison, the RMSEs are computed for 100 randomly generated 10-point validation sets. They are shown as a boxplot in the right panel of Figure 4.14. We can see that the RMSE from the sequential MaxPro validation set is larger than the upper whisker of the boxplot. This is because if the validation set is generated randomly then it is likely to have points near the design points which are easy to predict, and therefore gives smaller RMSEs. On the other hand, the sequential MaxPro validation points are placed away from the design points (in all subspaces) and thus give us a validation set that is difficult to predict.

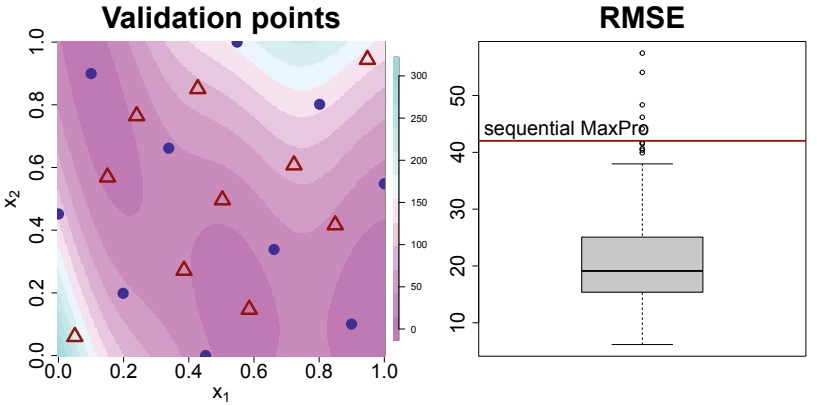


FIGURE 4.14: The left panel shows the MaxPro design (blue dots) and the validation points generated using sequential MaxPro (red triangles) overlayed on the image of Branin function. The right panel shows the boxplot of RMSEs of 100 random validation sets.

4.5.1 Constrained Regions

So far we have been looking at experimental regions that are hypercubes. However, sometimes certain factor combinations cannot be used. For example, if we know that the combination of high pressure and high temperature can lead to system failure, then there is no need to experiment with such combinations. Suppose that the feasible region of inputs is defined by a set of constraints:

$$\mathcal{X} = \left\{ \mathbf{x} \in [0,1]^p : g_k(\mathbf{x}) \leq 0 \ \forall k \right\}, \tag{4.29}$$

where $\{g_k(\mathbf{x}) \leq 0\}_{k=1}^K$ are arbitrary K inequality constraints. These could be black-box functions and expensive to evaluate. Here is a toy example taken from Huang et al. (2021):

$$\begin{aligned} &g_1(x) = x_1 - \sqrt{50(x_2 - 0.52)^2 + 2} + 1 \leq 0, \\ &g_2(x) = \sqrt{120((x_2 - 0.48)^2 + 1)} - 0.75 - x_1 \leq 0, \\ &g_3(x) = 0.65^2 - x_1^2 - x_2^2 \leq 0, \\ &\text{where } 0 \leq x_1, x_2 \leq 1. \end{aligned} \tag{4.30}$$

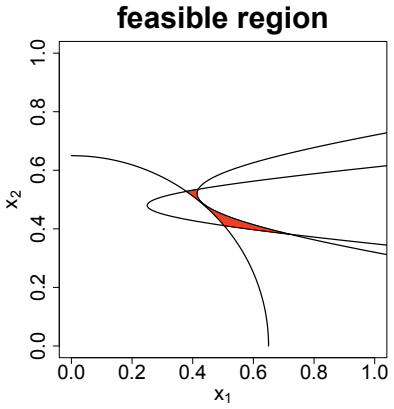


FIGURE 4.15: Feasible region of the constraints in (4.30) is shown as a shaded region.

The feasible region is shown in Figure 4.15 as a shaded region. Suppose that we have a budget to perform 20 experiments.

Here is a general approach for generating experimental designs from a constrained region: (i) first generate a large uniform sample from a hypercube that encloses the feasible region, (ii) reject the samples that do not satisfy the constraints, and (iii) choose the experimental design points from the remaining samples (candidate set). To obtain 20 design points for the experimental region shown in (4.30), we first generated 10,000 random Latin hypercube samples. This is shown in the left panel of Figure 4.16. Rejecting the points that do not satisfy (4.30) gives 538 points. This forms the candidate set, which is shown in the middle panel of the figure. Now we can use the sequential MaxPro method in (4.27) to obtain the MaxPro design. This is shown in the right panel of Figure 4.16. A better method to generate the candidate set, which in turn improves the quality of the MaxPro design, will be discussed later in Section 4.6.2.

4.5.2 Qualitative Factors

So far we have been looking at inputs that are continuous. However, in reality, some inputs can be qualitative in nature. For example, the machining simulator described in Section 1.2

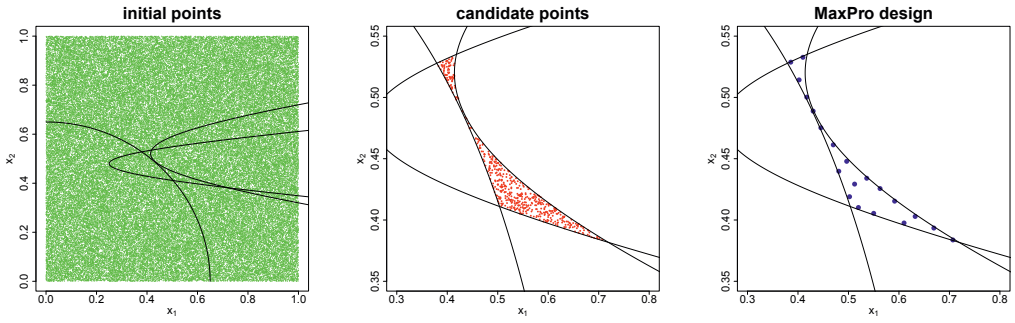


FIGURE 4.16: 10,000 Latin hypercube samples (left), feasible points (middle), and MaxPro design (right).

can specify the workpiece materials such as various Titanium alloys (Ti-6AL-4V or Ti-6Al-6V-2Sn) or tool path optimization types (none, in-cut, air-cut, or both). Such factors are called nominal qualitative factors.

In general, input in a system can be classified as quantitative or qualitative. The quantitative inputs can be continuous or discrete-numeric. The discrete-numeric factors are those that take only discrete values such as $1, 2, 3, \ldots$. For example, one can specify the number of flutes of the milling tool in the machining simulator, which is a discrete-numeric factor. Qualitative inputs can be nominal or ordinal. Ordinal factors are those that have some order in their values. For example, an ordinal factor can take the following values: Excellent, Very Good, Good, Fair, and Poor. In the machining example, one such factor would be the condition of the tool.

How can we develop space-filling designs that can accommodate qualitative factors? An approach taken in the literature is to divide an LHD into several slices and assign each slice to a level of the qualitative factor. Qian (2012) developed Sliced Latin hypercube design (SLHD) which ensures that each slice is also an LHD. As an example, consider two continuous factors x_1 and x_2, and a nominal qualitative factor x_3 with three levels. Suppose that we have a budget for nine experiments. The left panel of Figure 4.17 shows a random LHD, where the three levels of the qualitative factors are shown as black circles ("1"), red triangles ("2"), and green pluses ("3"). We can see that this is not a good design because $x_1 \in [0, 1/3]$ when the level of qualitative factor is 1, $x_1 \in [1/3, 2/3]$ when the qualitative factor level is 2, and $x_1 \in [2/3, 1]$ when the qualitative factor level is 3. Thus, x_1 does not explore the whole range within the level of the qualitative factor. On the other hand, the sliced LHD shown in the right panel of Figure 4.17 is a good design because both x_1 and x_2 explore the whole range within each level of the qualitative factor. As shown in the figure, we have a 9×9 LHD if we ignore the qualitative factor (dashed lines), in addition to 3×3 LHDs within each level of the qualitative factor (solid lines).

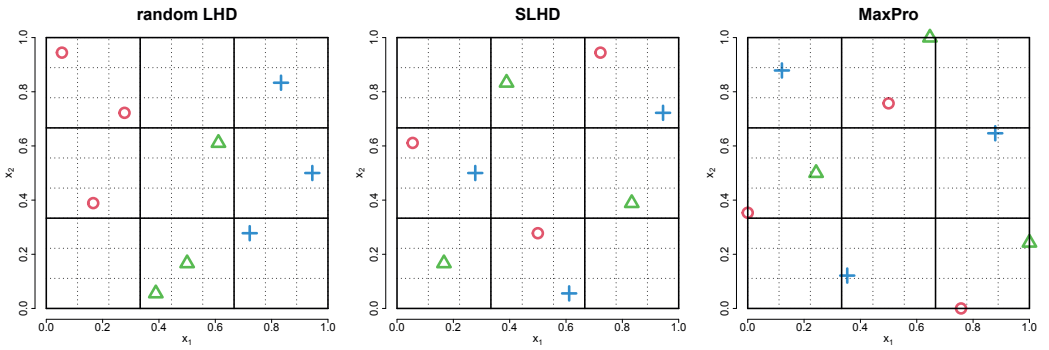


FIGURE 4.17: Designs with two quantitative factors (x_1, x_2) and one qualitative factor with three levels (x_3): random LHD (left), Sliced LHD (middle), and MaxPro design (right).

If there are p qualitative factors with number of levels $L_1, \ldots, L_p$, then we can first create a "new" qualitative factor with $t = \prod_{i=1}^{p} L_i$ levels and then construct an SLHD with t slices. The efficient construction of SLHDs is given in Yang et al. (2013) and Yin et al. (2014). Ba et al. (2015) proposed maximin SLHD that ensures good maximin properties for the entire design as well as the slices. The SLHD shown on the right panel of Figure 4.17 is in fact a maximin SLHD generated using the R package `SLHD` (Ba, 2015). However, SLHDs

have the drawback that the designs can become extremely large as the number of qualitative factors and their levels increase. Deng et al. (2015) proposed marginally coupled designs (MCDs) that can accommodate a larger number of nominal factors. The MCD structure combines an orthogonal array (OA) (see, e.g., Wu and Hamada (2021)) for nominal factors with an LHD for continuous factors in such a way that the LHD can be partitioned by the levels of any single nominal factor into multiple smaller LHDs. In other words, the LHD for continuous factors combined with any single nominal factor column in an MCD forms an SLHD. Because the MCD uses an OA instead of a full factorial combination for the nominal factors, it can achieve a more economical run-size than the SLHD. Nevertheless, the MCD structure is restrictive: it requires not only the existence of an OA of a given size, but also the existence of a special LHD which can form an SLHD when paired with each single column of the OA. Joseph et al. (2020) proposed MaxPro designs to accommodate qualitative factors that are flexible in their run sizes and thus overcome the limitations of SLHD and MCD. Moreover, their approach allows for the inclusion of discrete-numeric and ordinal factors as well.

Before we can discuss MaxPro designs for qualitative factors, we need to understand how to fit a GP model with qualitative factors. First, we convert ordinal factors into discrete numeric factors through *scoring method* (see, e.g., Wu and Hamada (2021, Ch.14)). That is, if the ordinal factor has levels "poor", "fair" and "good", then depending on the nature of the classification, the experimenter can choose some discrete numeric levels such as $\{1, 2, 3\}$ or $\{1, 4, 5\}$ to represent the three ordinal levels. Thus, we will only need to consider continuous, nominal, and discrete numeric factors assuming that the ordinal factors are already converted to discrete numeric factors through scoring.

Qian et al. (2008) proposed several correlation construction schemes along with a general framework to build GP models with quantitative and nominal-type qualitative factors. Here we assume the following exchangeable correlation structure for the nominal factor, which was also used by Joseph and Delaney (2007):

$$R(\mathbf{w}_i - \mathbf{w}_j; \alpha, \beta, \gamma) = \exp\left\{ -\sum_{l=1}^{p_1} \alpha_l |x_{il} - x_{jl}| - \sum_{k=1}^{p_2} \beta_k |u_{ik} - u_{jk}| - \sum_{h=1}^{p_3} \gamma_h I(v_{ih} \neq v_{jh}) \right\}, \quad (4.31)$$

where $\mathbf{w}_i = (\mathbf{x}_i, \mathbf{u}_i, \mathbf{v}_i)$, $\mathbf{x}$ represents the p_1 continuous factors, $\mathbf{u}$ represents the p_2 discrete numeric factors (including ordinal factors), $\mathbf{v}$ represents the p_3 nominal factors, and $I(v_{ih} \neq v_{jh})$ is the indicator function that takes values 1 if $v_{ih} \neq v_{jh}$ and 0 otherwise. Assume that the discrete numeric factors are also scaled to $[0, 1]$ so that the correlation parameters of the three types of factors are on the same scale. Unlike the Gaussian correlation function in (4.24), an exponential correlation function is used here for continuous factors. We will see that using an informative prior on the correlation parameters of the exponential correlation function, we can derive the same MaxPro criterion as with the Gaussian correlation function. This modification using an informative prior is crucial for the discrete numeric factor because they cannot have n levels in the design.

Assume the following priors for the correlation parameters:

$$\begin{aligned} \alpha_l &\overset{iid}{\sim} Gamma(2, \bar{\alpha}_l), l = 1, \ldots, p_1, \\ \beta_k &\overset{iid}{\sim} Gamma(2, \bar{\beta}_k), k = 1, \ldots, p_2, \\ \gamma_h &\overset{iid}{\sim} Gamma(2, \bar{\gamma}_h), h = 1, \ldots, p_3. \end{aligned}$$

A Gamma prior is chosen here so that we can get an explicit expression for the objective function. Furthermore, the shape parameter in the gamma distribution is chosen to be 2 so

that the results agree with the MaxPro criterion for the continuous factors in (4.23). Thus

$$
\begin{aligned}
&E\{\|\mathbf{R}-\mathbf{I}_n\|_1\}\\
=&\int\!\!\int\!\!\int \sum_{i=1}^{n}\sum_{j\neq i} R(\mathbf{w}_i-\mathbf{w}_j;\alpha,\beta,\gamma)\prod_{l=1}^{p_1}\bar{\alpha}_l^2\alpha_l e^{-\bar{\alpha}_l\alpha_l}\prod_{k=1}^{p_2}\bar{\beta}_k^2\beta_k e^{-\bar{\beta}_k\beta_k}\prod_{h=1}^{p_3}\bar{\gamma}_h^2\gamma_h e^{-\bar{\gamma}_h\gamma_h}d\alpha d\beta d\gamma\\
=&\sum_{i=1}^{n}\sum_{j\neq i}\prod_{l=1}^{p_1}\int\bar{\alpha}_l^2\alpha_l d\alpha_l e^{-\{|x_{il}-x_{jl}|+\bar{\alpha}_l\}\alpha_l}\prod_{k=1}^{p_2}\int\bar{\beta}_k^2\beta_k e^{-\{|u_{ik}-u_{jk}|+\bar{\beta}_k\}\beta_k}d\beta_k\\
&\prod_{h=1}^{p_3}\int\bar{\gamma}_h^2\gamma_h e^{-\{I(v_{ih}\neq v_{jh})+\bar{\gamma}_h\}\gamma_h}d\gamma_h\\
=&\sum_{i=1}^{n}\sum_{j\neq i}\prod_{l=1}^{p_1}\frac{\bar{\alpha}_l^2}{\{|x_{il}-x_{jl}|+\bar{\alpha}_l\}^2}\prod_{k=1}^{p_2}\frac{\bar{\beta}_k^2}{\{|u_{ik}-u_{jk}|+\bar{\beta}_k\}^2}\prod_{h=1}^{p_3}\frac{\bar{\gamma}_h^2}{\{I(v_{ih}\neq v_{jh})+\bar{\gamma}_h\}^2}.
\end{aligned}
$$

This gives a new design criterion which is to minimize

$$
\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{\prod_{l=1}^{p_1}\{|x_{il}-x_{jl}|+\bar{\alpha}_l\}^2\prod_{k=1}^{p_2}\{|u_{ik}-u_{jk}|+\bar{\beta}_k\}^2\prod_{h=1}^{p_3}\{I(v_{ih}\neq v_{jh})+\bar{\gamma}_h\}^2}. \tag{4.32}
$$

Intuitively, smaller values for $\bar{\alpha}_l$, $\bar{\beta}_k$, and $\bar{\gamma}_h$ should be used in (4.32) in order to increase its sensitivity to design changes. For continuous factors, we can simply set $\bar{\alpha}_l = 0$ ($l = 1,\ldots,p_1$) because this would force each continuous factor to have n distinct levels. For nominal and discrete numeric factors, however, the total available number of levels is less than n for each factor. Therefore, we must have $\bar{\beta}_k > 0$ ($k = 1,\ldots,p_2$) and $\bar{\gamma}_h > 0$ ($h = 1,\ldots,p_3$) to prevent the denominator in (4.32) to become zero when $u_{ik} = u_{jk}$ or $v_{ih} = v_{jh}$ for some (i,j) values. Let L_h denote the number of distinct levels of the hth nominal factor, $h = 1,\ldots,p_3$. Because $P\{I(v_{ih}\neq v_{jh}) = 0\} = 1/L_h$, we have $E\{I(v_{ih}\neq v_{jh})\} = 1 - 1/L_h$ for $h = 1,\ldots,p_3$. This suggests choosing $\bar{\gamma}_h = 1/L_h$, which is the smallest possible value of $\bar{\gamma}_h$ to make $E\{I(v_{ih}\neq v_{jh}) + \bar{\gamma}_h\}$ independent of L_h. Similarly, for discrete numeric factors, we can also set $\bar{\beta}_k = 1/m_k$, where m_k is the number of levels of the kth discrete numeric factor. It can be seen that as m_k increases, the corresponding discrete numeric factor behaves more like a continuous factor and the corresponding $\bar{\beta}_k \to 0$ as desired.

With the foregoing choices of the hyperparameter values, the optimal design criterion becomes

$$
\min_{D}\sum_{i=1}^{n-1}\sum_{j\neq i}\frac{1}{\prod_{l=1}^{p_1}(x_{il}-x_{jl})^2\prod_{k=1}^{p_2}\left\{|u_{ik}-u_{jk}|+\frac{1}{m_k}\right\}^2\prod_{h=1}^{p_3}\left\{I(v_{ih}\neq v_{jh})+\frac{1}{L_h}\right\}^2}. \tag{4.33}
$$

This criterion contains the MaxPro criterion in (4.23) as a special case where there are only continuous factors ($p_1 \geq 1, p_2 = p_3 = 0$). As a result, it possesses all the desirable properties of a MaxPro design and maximizes the space-filling properties not only in the full design space but also in all possible projections to sub-dimensional spaces. For nominal factors, this criterion also shares similar ideas as the J_2-optimality criterion proposed by Xu (2002) in the construction of mixed-level OAs and near OAs by maximizing the dissimilarity of the rows, which is defined as $\min_D \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\{\sum_{h=1}^{p_3} I(v_{ih} = v_{jh})\}^2$.

The MaxPro design in (4.33) can be generated using the `MaxProQQ` function in the R package `MaxPro` (Ba and Joseph, 2018). The MaxPro design in nine runs with two continuous factors and one qualitative nominal factor with three levels is shown in the right panel of Figure 4.17. We can see that the MaxPro design has the SLHD property although no such restriction was placed during optimization. On the other hand, MaxPro design is very flexible: it can be generated for any number of runs and any number of quantitative and

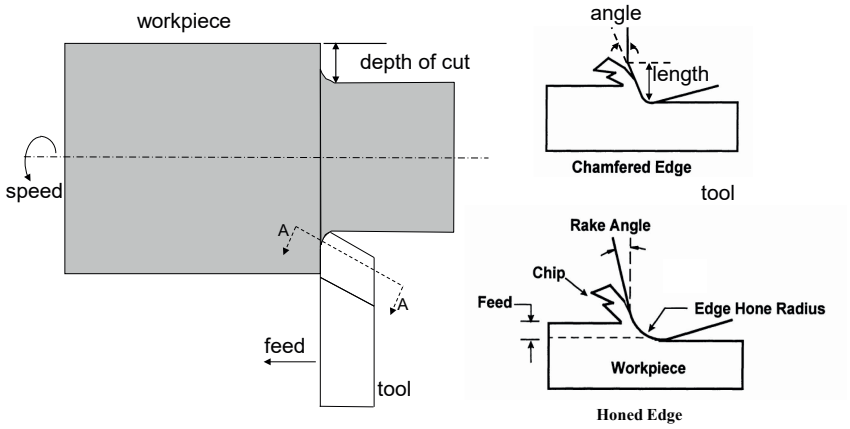


FIGURE 4.18: (left) Schematic of a turning process, where A-A is a perpendicular section through the tool. (right) Illustration of hone and chamfer tool edges. (Figure courtesy: Shreyes Melkote).

qualitative factors. See Chitre et al. (2025) for a recent application of `MaxProQQ` in liquid formulation design.

4.5.3 Branching Designs

Hung et al. (2009) reported a computer experiment aimed at optimizing a turning process for hardened bearing steel with a cBN cutting tool (see Figure 4.18). This process is commonly referred to as hard turning. Since the material being machined is very hard (hardness in excess of 60 Rockwell C), the cutting tool is subjected to large forces, stresses, and temperatures during the operation. In practice, the cutting edge of the tool is shaped such that it can withstand extreme conditions. Two commonly employed cutting edge shapes, hone and chamfer, are shown on the right side of the same figure. Note that they represent the idealized view of the instantaneous cutting action in the cross section A-A shown on the left side. The chamfer tool design can be changed using two factors: chamfer length and chamfer angle, whereas the hone design is fixed. In other words, the two factors length and angle are *nested* within the chamfer edge and there are no factors nested within hone edge. Following Taguchi (1987) and Phadke (1995, p.168), we will call the tool edge as a *branching* factor. Thus, when the branching factor takes the level chamfer, there are two additional factors present in the experiment; but when the branching factor takes the level hone, there are no factors. There are a few other factors that are common to both of the tool edges such as the cutting edge radius, tool nose radius, and rake angle. The machining parameters such as cutting speed, feed, and depth of cut are also factors that do not depend on the type of tool edges. To distinguish them from the branching and nested factors, they are called as *shared* factors. All factors involved in this experiment and their allowable ranges are shown in Table 4.1. The experiments were performed using commercially available finite element software AdvantEdge.

Since the nested factors can be different under each level of the branching factor, projections within each branching factor level become very important. Hung et al. (2009) developed branching Latin hypercube designs that consider distances in the space of shared

TABLE 4.1: Factors and their ranges in the hard turning experiment.

Type of Factor	Notation	Factor	Ranges
Branching factor	z_1	Cutting edge shape	Hone or chamfer
Nested factors	$v_1 \vert z_1$=chamfer	Angle (degree)	[-20, -17]
	$v_2 \vert z_1$=chamfer	Length (μm)	[-140, -115]
	$v_1 \vert z_1$=hone	None	None
	$v_2 \vert z_1$=hone	None	None
Shared factors	x_1	Cutting edge radius (μm)	[-25, -5]
	x_2	Rake angle (degree)	[-15, -5]
	x_3	Tool nose radius (mm)	[-1.6, -0.4]
	x_4	Cutting speed (m/min)	[-240, -120]
	x_5	Feed (mm/rev)	[-0.15, -0.05]
	x_6	Depth of cut (mm)	[-0.25, -0.1]

factors and in the space of nested and shared factors within each level of the branching factor. Since branching factor is usually a qualitative factor, we can instead use a MaxProQQ design as discussed in the previous section. However, modifications to the design are needed to accommodate the nested factors.

We will explain the approach using the hard turning experiment as an example. We have a budget for 30 runs. First, we can create a MaxPro (30,9) design using `MaxProQQ` with the cutting edge shape as a nominal factor and the other eight factors as continuous factors. Now we can pick two columns and remove the entries corresponding to honed edge tool and rescale and optimize the entries corresponding to chamfer edge tool. We can compute the MaxPro criterion for the shared factor design and the nested-shared designs within the two levels of the branching factor, and optimize them with respect to the choice of the two columns. The quality of the final design also depends on the initial MaxPro (30,9) design. Therefore, the whole process can be repeated several times and we can choose the design that is best according to the MaxPro criterion. A scatterplot matrix of the optimal design thus obtained is shown in Figure 4.19. The real design and data can be obtained from the book's GitHub site.

4.5.4 Nested Designs

Sometimes, computer codes can be developed to solve mathematical equations describing the physical system at various fidelity levels such as in finite element analysis. For example, Figure 4.20 shows a problem of heat conduction in a metal plate for two mesh arrangements subject to mixed boundary conditions: a constant temperature of 100 ^oC at the bottom edge, heat convection to an external temperature of 0 ^oC at the top and left edges, and thermal insulation at the right edge (Cameron et al., 1986). The left image shows the temperature field on a coarse mesh, while the right image uses a finer mesh. The numerical solution will be more accurate with a finer mesh, but the computational time will be longer. Therefore, for a given computational budget, it makes sense to perform many coarse mesh simulations (low-fidelity) and a few fine mesh simulations (high-fidelity).

Suppose that there are K fidelity levels with "1" being the lowest fidelity and "K" the highest fidelity. Let $D_k = \{\mathbf{x}_i^{(k)}\}_{i=1}^{n_k}$ be the experimental design at level k with n_k runs and $\{y_k = h_k(\mathbf{x}_i^{(k)})\}_{i=1}^{n_k}$ the corresponding output data, where $n_1 \geq n_2 \geq \ldots \geq n_K$. Kennedy

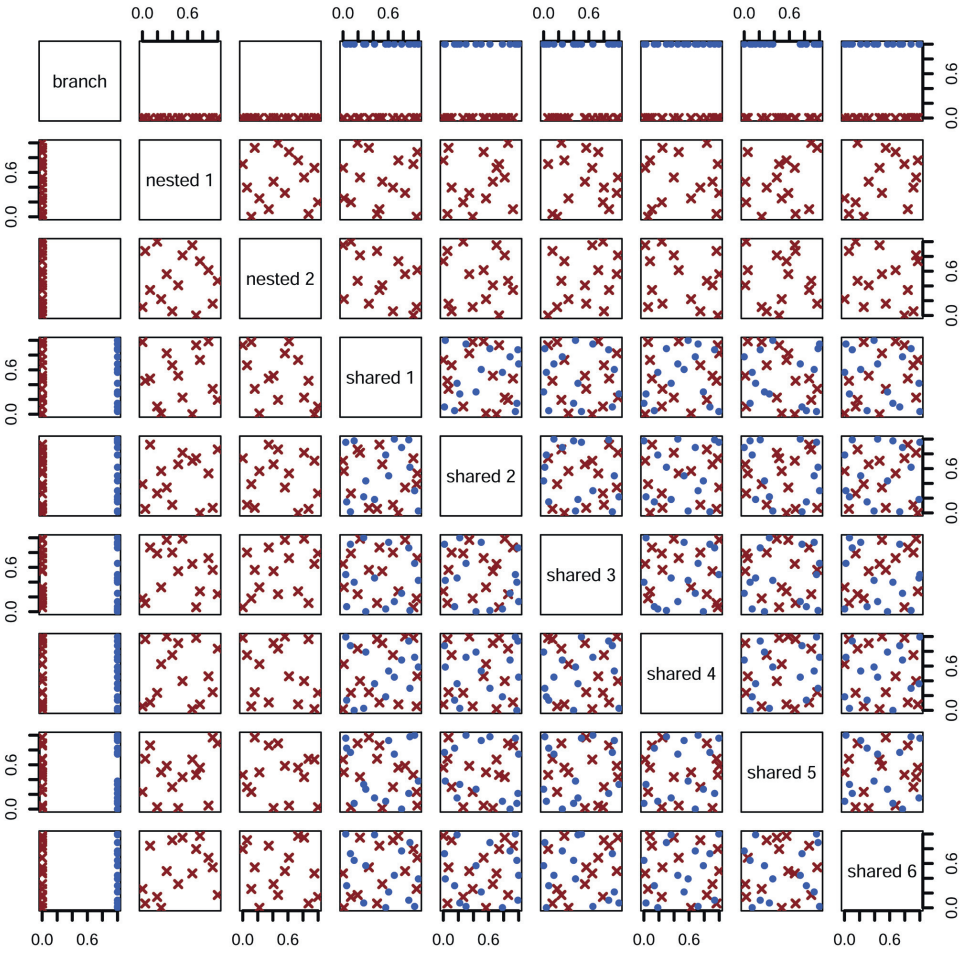


FIGURE 4.19: Scatterplot matrix of the 30-run branching design for the hard turning experiment.

and O'Hagan (2000) proposed the following model to integrate the data:

$$h_k(\mathbf{x}) = h_{k-1}(\mathbf{x}) + \delta_k(\mathbf{x}),\ k = 2, \ldots, K, \tag{4.34}$$

where $\delta_k(\mathbf{x})$ denotes the discrepancy between the outputs at the $(k-1)$th and kth fidelity levels. The following GP assumption is made:

$$\delta_k(\mathbf{x}) \sim GP(0, \tau_k^2 R_k(\cdot)),\ k = 2, \ldots, K \tag{4.35}$$

$$y_1(\mathbf{x}) \sim GP(\mu, \tau_1^2 R_1(\cdot)). \tag{4.36}$$

How should we choose $D_1, \ldots, D_K$ to perform multi-fidelity simulations so that we can fit the foregoing model and obtain a good approximation of the response surface at the highest fidelity level? Kennedy and O'Hagan (2000) recommend nested designs $D_1 \supseteq D_2 \supseteq \ldots \supseteq D_k$ because such designs will help estimate $\delta_k(\cdot)$'s without relying too much on modeling

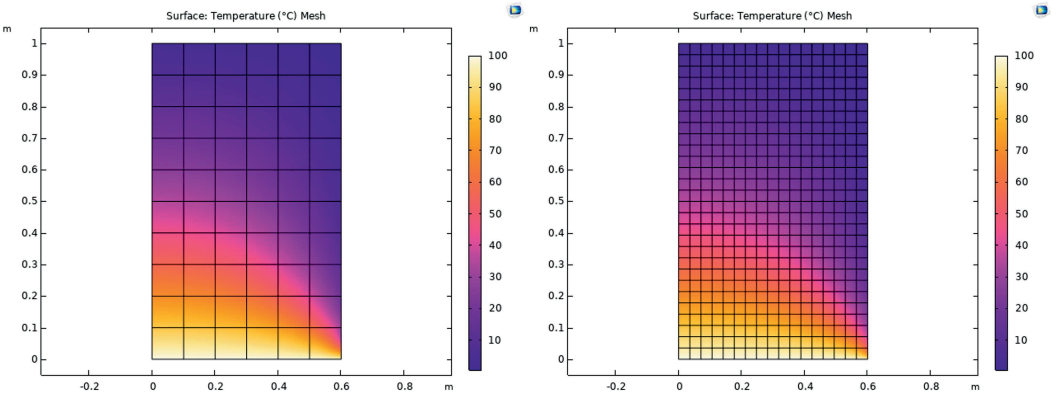


FIGURE 4.20: Heat transfer through a metal plate with coarse mesh (left) and fine mesh (right). Figure courtesy: Maruthi Annamaraju (Surya Kalidindi's lab).

assumptions. Qian et al. (2006) describe an application with two fidelity levels in which D_1 is chosen using an orthogonal array-based LHD and D_2 selected as a subset of D_1 using the minimax criterion. Qian (2009), Qian et al. (2009), and Qian et al. (2009) provide several algebraic methods for the construction of nested space-filling deigns.

To understand the appropriateness of nested space-filling designs, we will first look at the model-based designs. For notational simplicity, consider the case of $K = 2$; that is, we only have two levels of fidelity: "1" (low) and "2" (high). Let $\mathbf{y}_1$ and $\mathbf{y}_2$ be the data obtained by performing the experiments using designs D_1 and D_2. Since there are two designs involved here, we need to introduce a slightly more general notation to specify the correlation matrices. Let $A = \{\mathbf{a}_i\}_{i=1}^m$ and $B = \{\mathbf{b}_i\}_{i=1}^n$, where $\mathbf{a}_i, \mathbf{b}_i \in \mathcal{X} \subset \mathbb{R}^p$. Then, let $\mathbf{R}(A, B)$ be the $m \times n$ matrix with ijth element $R(\mathbf{a}_i - \mathbf{b}_j)$. Thus, so far, we have been denoting $\mathbf{R}(D, D)$ as simply $\mathbf{R}$. It is easy to see that

$$\begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{pmatrix} \sim \mathcal{N}_{n_1+n_2}\left(\mu\begin{pmatrix}\mathbf{1}_{n_1}\\ \mathbf{1}_{n_2}\end{pmatrix}, \begin{bmatrix}\tau_1^2\mathbf{R}_1(D_1,D_1) & \tau_1^2\mathbf{R}_1(D_1,D_2)\\ \tau_1^2\mathbf{R}_1(D_2,D_1) & \tau_1^2\mathbf{R}_1(D_2,D_2)+\tau_2^2\mathbf{R}_2(D_2,D_2)\end{bmatrix}\right). \quad (4.37)$$

Now consider the maximum entropy design:

$$\max_{D_1,D_2}\begin{vmatrix}\tau_1^2\mathbf{R}_1(D_1,D_1) & \tau_1^2\mathbf{R}_1(D_1,D_2)\\ \tau_1^2\mathbf{R}_1(D_2,D_1) & \tau_1^2\mathbf{R}_1(D_2,D_2)+\tau_2^2\mathbf{R}_2(D_2,D_2)\end{vmatrix}.$$

Using the results on partitioned matrices (Lemma 2.1) this is equivalent to

$$\begin{aligned}&\max_{D_1,D_2}|\tau_1^2\mathbf{R}_1(D_1,D_1)|\times\\ &\quad|\tau_1^2\mathbf{R}_1(D_2,D_2)+\tau_2^2\mathbf{R}_2(D_2,D_2)-\tau_1^2\mathbf{R}_1(D_2,D_1)\mathbf{R}_1^{-1}(D_1,D_1)\mathbf{R}_1(D_1,D_2)|.\end{aligned} \quad (4.38)$$

Note that if $D_2 \subseteq D_1$, then $\mathbf{R}_1(D_2,D_2) - \mathbf{R}_1(D_2,D_1)\mathbf{R}_1^{-1}(D_1,D_1)\mathbf{R}_1(D_1,D_2) = 0$. Thus, if D_2 is nested within D_1, then (4.38) simplifies to

$$\max_{D_1,D_2\subseteq D_1}|\mathbf{R}_1(D_1,D_1)|\times|\mathbf{R}_2(D_2,D_2)|. \quad (4.39)$$

As an approximation to (4.39), we could consider the two smaller sequential optimizations:

$$\begin{aligned}D_1^* &= \operatorname*{argmax}_{D_1}|\mathbf{R}_1(D_1,D_1)|,\\ D_2^* &= \operatorname*{argmax}_{D_2\subseteq D_1^*}|\mathbf{R}_2(D_2,D_2)|.\end{aligned} \quad (4.40)$$

This suggests that first we can find a maximum entropy design for the low-fidelity level and then find a design for the high-fidelity level by solving (4.40). However, these designs cannot be obtained in practice because we do not know the correlation parameters in $R_1(\cdot)$ and $R_2(\cdot)$. Therefore, as before, we will look for appropriate space-filling designs that can approximate the foregoing solutions.

As discussed in Section 4.5, MaxPro designs will be good choices for D_1^* and D_2^* because they tend to maximize the entropy. The only thing that we need to develop is how to find a MaxPro design that is nested within another design. One way to do this is to construct D_2^* by removing points from D_1^* one at a time. Since

$$\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{\prod_l\{|x_{il}-x_{jl}|+\beta_l\}^2}=\sum_{i=1,i\neq k}^{n}\sum_{j\neq i}\frac{1}{\prod_l\{|x_{il}-x_{jl}|+\beta_l\}^2}+\sum_{j\neq k}\frac{1}{\prod_l\{|x_{kl}-x_{jl}|+\beta_l\}^2}$$

we can identify the best point to remove from D_1 as

$$\mathbf{x}_k^*=\underset{\mathbf{x}_k\in D_1}{\operatorname{argmax}}\sum_{j\neq k}\frac{1}{\prod_{l=1}^{p}\{|x_{kl}-x_{jl}|+\beta_l\}^2}. \tag{4.41}$$

Note that we have added a constant $\beta_l > 0$ in the denominator. This is because now the choice of values in D_2 is not continuous. It can only take the discrete values in D_1. Therefore, as in (4.33), we can let $\beta_l = 1/n_1$ for all $l = 1, \ldots, p$. If we were to use a Gaussian correlation instead of the exponential correlation in (4.31), then the criterion would become:

$$\mathbf{x}_k^*=\underset{\mathbf{x}_k\in D_1}{\operatorname{argmax}}\sum_{j\neq k}\frac{1}{\prod_{l=1}^{p}\{(x_{kl}-x_{jl})^2+\beta_l\}}, \tag{4.42}$$

which is implemented in the `maxpro.remove` function of the R package `SFDesign` (Wang and Joseph, 2025c). Here we need to choose $\beta_l = 1/n_1^2$ for $l = 1, \ldots, p$.

The foregoing algorithmic construction method for nested design is much more general and flexible compared to the algebraic construction methods mentioned earlier. Moreover, it is easy to extend the method to any number of fidelity levels. Consider a simple example with two fidelity levels and $p = 2$. Suppose that we want $n_1 = 100$ runs from the low-fidelity level and $n_2 = 10$ from the high-fidelity level. Figure 4.21 shows the 100 points for D_1 (blue dots) obtained using MaxPro and the 10 points for D_2 (red triangles) selected from D_1 using the `maxpro.remove` function.

Nested designs that take into account the cost and accuracy of the codes at each fidelity level, known as stacking designs, are developed in Sung et al. (2024). Heo and Sung (2025) recently proposed a nonlinear model to link the data at two adjacent fidelity levels:

$$h_k(\mathbf{x}) = W_k(\mathbf{x}, h_{k-1}(\mathbf{x})),$$

which generalizes the additive model in (4.34). The nonlinear function W_k is modeled as another Gaussian process, similar to the deep Gaussian process (Damianou and Lawrence, 2013; Sauer et al., 2023). Sequential designs (Chapter 7) using this model based on the nested design structure are discussed in Heo and Sung (2025). In situations where fidelity levels cannot be ordered from lowest to highest, Ji et al. (2024) proposed a graphical Gaussian process model that embeds the scientific dependencies of the multi-fidelity codes through a directed acyclic graph (DAG). Furthermore, they proposed a nested breadth-first-traversal design for efficient experimentation with such codes.

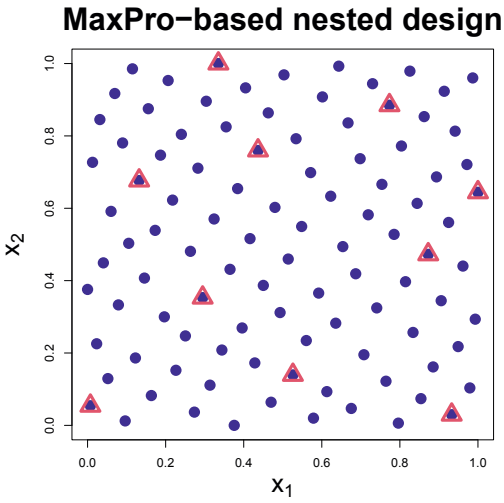


FIGURE 4.21: Nested design for multi-fidelity simulations with two levels of fidelity. D_1 is shown using blue dots and D_2 as red triangles.

4.6 Minimum Energy Designs

Figure 4.22 shows the population density in various zip code locations of the United States without Alaska and Hawaii (Rozzi, 2022). The points are plotted in pink with their size proportional to the log-population density. Suppose that we are interested in placing 200 sensors to measure pollution level across the United States. Once we have the measurements from the sensors, we can use kriging or GP to predict the pollution at any place in United States. How should we place the sensors so that the predictions are accurate? A 200-point maximin design generated using the R package `maximin` (Sun and Gramacy, 2024) is shown in the left panel of Figure 4.22, which does not make much sense because we know that pollution will be higher in densely populated regions. So, ideally, we should place more sensors in the higher-density regions and less sensors in the lower-density regions. Thus, we need some sort of weighted space-filling design (Bowman and Woods, 2013) where the weights are proportional to the population density. The minimum energy designs proposed by Joseph et al. (2015) is one such option.

Minimum energy design (MED) has a physical analogy. Consider n charged particles. Without loss of generality, take them as positively charged. If they are placed inside a box, they will repel each other and occupy positions in the box that will minimize the total potential energy. Let $q(\mathbf{x})$ be the charge of the particle when it is at position $\mathbf{x} \in \mathcal{X} = [0,1]^p$. Then, the total potential energy of n particles is given by

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{q(\mathbf{x}_i)q(\mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|}.$$

The MED can be obtained by minimizing E with respect to the design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. If $q(\mathbf{x}) = 1$ for all $\mathbf{x} \in [0,1]^p$, then the criterion reduces to the TRD criterion in (4.18) with

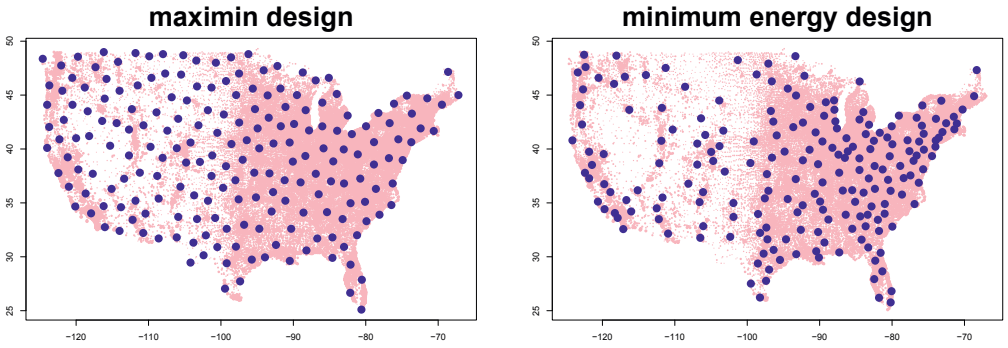


FIGURE 4.22: Maximin and minimum energy designs are shown as blue dots. The zip code locations across United States from the 2020 census are shown as pink dots, where the size of the points are taken to be proportional to the population density.

$\alpha = 1/2$. Therefore, consider a generalized version of the energy criterion:

$$\min_D \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{q(\mathbf{x}_i)q(\mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|_\alpha} \right)^k \right\}^{1/k},$$

where $\|\mathbf{x}\|_\alpha = \{\sum_{i=1}^p |x_i|^\alpha\}^{1/\alpha}$. As $k \to \infty$, this criterion becomes

$$\min_D \max_{i \neq j} \frac{q(\mathbf{x}_i)q(\mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|_\alpha} \equiv \max_D \min_{i \neq j} \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_\alpha}{q(\mathbf{x}_i)q(\mathbf{x}_j)}, \tag{4.43}$$

which can be viewed as a weighted version of the maximin criterion in (4.16). Joseph et al. (2015) and Joseph et al. (2019) proved the following asymptotic result.

Theorem 4.4 *Suppose that the charge function $q(\cdot)$ is Lipschitz continuous, i.e., $|q(\mathbf{u}) - q(\mathbf{v})| \leq Ld(\mathbf{u}, \mathbf{v})$, for $\mathbf{u}, \mathbf{v} \in \mathcal{X} = [0,1]^p$ and a constant $L > 0$. Let $D^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*\}$ be an n-point minimum energy design using (4.43) with the smallest index and $\mathscr{B}$ be the Borel σ-algebra of $\mathcal{X}$. Define the following probability measures on $(\mathcal{X}, \mathscr{B})$:*

$$\mathcal{P}_n(A) = \frac{\text{card}\{\mathbf{x}_i^* : 1 \leq i \leq n, \mathbf{x}_i^* \in A\}}{n}, \text{ for any } A \in \mathscr{B}. \tag{4.44}$$

Then, there exists a probability measure $\mathcal{P}$ such that $\mathcal{P}_n$ converges to $\mathcal{P}$ weakly for all fixed $\alpha \in (0, \infty)$ as $n \to \infty$. Moreover, $\mathcal{P}$ has a density proportional to $1/q^{2p}(\mathbf{x})$ over $\mathcal{X}$.

Interestingly, MED is known by the name minimal Riesz energy points in the sphere-packing literature. Results similar to Theorem 4.4 for the case of $\alpha = 2$ can be found in Borodachov et al. (2008a,b).

Theorem 4.4 is very useful because if we have a target density in mind, say $f(\mathbf{x})$, then we can take the charge function to be $q(\mathbf{x}) = f(\mathbf{x})^{-1/(2p)}$ so that the MED will approximately follow the desired distribution. With this choice of charge function, MED becomes

$$\max_D \min_{i \neq j} f^{1/(2p)}(\mathbf{x}_i) f^{1/(2p)}(\mathbf{x}_j) \|\mathbf{x}_i - \mathbf{x}_j\|_\alpha. \tag{4.45}$$

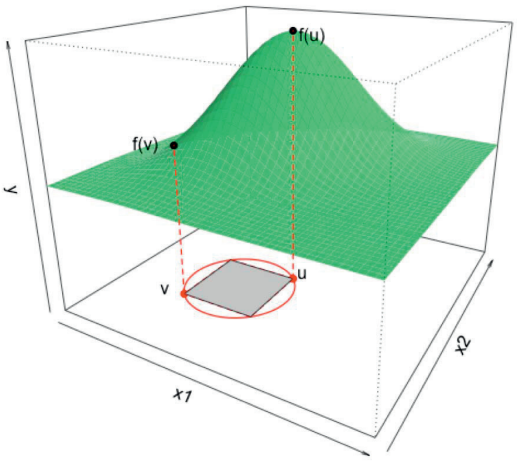


FIGURE 4.23: Probability-balancing interpretation of minimum energy designs.

MED has a nice probability interpretation. For $\alpha = 2$, the criterion in (4.45) is equivalent to

$$\max_D \min_{i \neq j} \sqrt{f(\mathbf{x}_i) f(\mathbf{x}_j)} V_S(\mathbf{x}_i, \mathbf{x}_j),$$

where $V_S(\mathbf{x}_i, \mathbf{x}_j) = \pi^{p/2}/\Gamma(p/2+1)\{d(\mathbf{x}_i, \mathbf{x}_j)/2\}^p$ is the volume of the sphere with center at $(\mathbf{x}_i + \mathbf{x}_j)/2$ and passing through the two points $\mathbf{x}_i$ and $\mathbf{x}_j$. See Figure 4.23 for an illustration. The term $\sqrt{f(\mathbf{x}_i) f(\mathbf{x}_j)}$ is the geometric mean of the density values at $\mathbf{x}_i$ and $\mathbf{x}_j$. Thus, $P_{ij}(D) = \sqrt{f(\mathbf{x}_i) f(\mathbf{x}_j)} V_S(\mathbf{x}_i, \mathbf{x}_j)$ is approximately the probability of $\mathbf{x}$ falling in the sphere. Let $i^* = arg\min_{j \neq i} P_{ij}(D)$. Then, the MED criterion can be written as

$$\max_D \min_{i=1:n} P_{ii^*}(D).$$

Now maximizing the minimum probability will tend to make the probabilities $P_{ii^*}(D)$ equal for all $i = 1, \ldots, n$. Thus, roughly speaking, an MED tries to balance the probabilities among adjacent points of the design. As shown in (4.25), for $\alpha = 0$, the criterion in (4.45) is equivalent to

$$\max_D \min_{i \neq j} \sqrt{f(\mathbf{x}_i) f(\mathbf{x}_j)} V_R(\mathbf{x}_i, \mathbf{x}_j),$$

where $V_R(\mathbf{x}_i, \mathbf{x}_j) = \prod_{l=1}^p |x_{il} - x_{jl}|$ is the volume of the hyper-rectangle, which has $\mathbf{x}_i$ and $\mathbf{x}_j$ at the two opposite corners and sides parallel to the axes (see Figure 4.23). Thus, the same probability-balancing interpretation as in the Euclidean case holds for this criterion as well, which can be obtained by replacing the hypersphere volume element with the hyperrectangle volume element. For $f(\mathbf{x}) = 1$, the two cases $\alpha = 2$ and $\alpha = 0$ correspond to the maximin and MaxPro designs, respectively.

Joseph et al. (2015) proposed the following sequential construction of the MED:

$$\mathbf{x}_{k+1} = \operatorname*{argmax}_{\mathbf{x} \in \mathcal{X}} \min_{i=1:k} f(\mathbf{x}) f(\mathbf{x}_i) \|\mathbf{x} - \mathbf{x}_i\|_\alpha^{2p}, \text{ for } k = 2, \ldots, n, \tag{4.46}$$

where

$$\mathbf{x}_1 = \operatorname*{argmax}_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}).$$

Consider again the problem of placing sensors across United States for measuring the pollution level based on the population density $f(\mathbf{x})$. The sequential algorithm in (4.46) is implemented in the R package `mined`. The 200 points selected using MED with $\alpha = 1$ are shown in the right panel of Figure 4.22. We can now see that more sensors are assigned to zip codes where more people live, which might help in building a more accurate prediction model for the pollution level across United States.

4.6.1 Bayesian Computation

Suppose the data $\mathbf{y}$ is generated from a sampling model $f(\mathbf{y}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the unknown set of parameters. In classical statistics, an estimate of $\boldsymbol{\theta}$ is obtained by maximizing the likelihood, i.e., $\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}} f(\mathbf{y}|\boldsymbol{\theta})$, and then the uncertainty of $\hat{\boldsymbol{\theta}}$ is assessed through repeated sampling of $\mathbf{y}$'s from the sampling model. In Bayesian statistics, a prior distribution $f(\boldsymbol{\theta})$ is postulated representing the uncertainty of $\boldsymbol{\theta}$ before seeing the data, and then the updated distribution known as the posterior distribution is obtained through Bayes' theorem (assuming $\boldsymbol{\theta}$ to be continuous):

$$f(\boldsymbol{\theta}|\mathbf{y}) = \frac{f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})}{\int f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})d\boldsymbol{\theta}}.$$

The denominator (normalizing constant) can be a high-dimensional integral which is not easy to calculate. Therefore, inference is usually done by sampling several values of $\boldsymbol{\theta}$ from $f(\boldsymbol{\theta}|\mathbf{y})$ using Markov Chain Monte Carlo (MCMC) methods that do not require knowledge of the normalizing constant.

MCMC methods usually require a large number of samples (thousands or even millions) to get a good approximation of the posterior. This can be time-consuming if the unnormalized posterior $h(\boldsymbol{\theta}) = f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})$ is expensive to evaluate. In such cases, we can think about using an experimental design to evaluate the unnormalized posterior and approximate it using a GP model (Joseph, 2012, 2013).

How to obtain an experimental design that is suitable for approximating the posterior? Clearly, we should have more points placed in the high-density regions of the posterior and few or no points in the low-density regions of the posterior. This looks like an ideal application of minimum energy designs with the charge function taken to be $q(\mathbf{x}) = f(\boldsymbol{\theta}|\mathbf{y})^{-1/(2p)}$. Thus,

$$\begin{aligned} D^* &= \underset{D}{\text{argmax}} \min_{i \neq j} f(\boldsymbol{\theta}_i|\mathbf{y}) f(\boldsymbol{\theta}_j|\mathbf{y}) \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|_\alpha^{2p} \\ &= \underset{D}{\text{argmax}} \min_{i \neq j} h(\boldsymbol{\theta}_i) h(\boldsymbol{\theta}_j) \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|_\alpha^{2p} \\ &= \underset{D}{\text{argmax}} \min_{i \neq j} \left\{ \log h(\boldsymbol{\theta}_i) + \log h(\boldsymbol{\theta}_j) + 2p \log \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|_\alpha \right\}. \end{aligned} \tag{4.47}$$

Notice that the normalizing constant has been factored out in the optimization. So we do not need the normalizing constant for finding the MED, which is a good thing! We also took the logarithm in the last step because the unnormalized posteriors can become very small and can lead to numerical inaccuracies.

There is an issue with the foregoing formulation. Optimization in (4.47) would require numerous evaluations of the unnormalized posterior, which would defeat the purpose of using an experimental design to approximate the expensive posterior. Joseph et al. (2019) proposed an efficient algorithm to generate an MED with few evaluations of the unnormalized posterior. The main idea is to use an annealed version of the unnormalized posterior density:

$$h^\gamma(\boldsymbol{\theta}), \quad \gamma \in [0, 1], \tag{4.48}$$

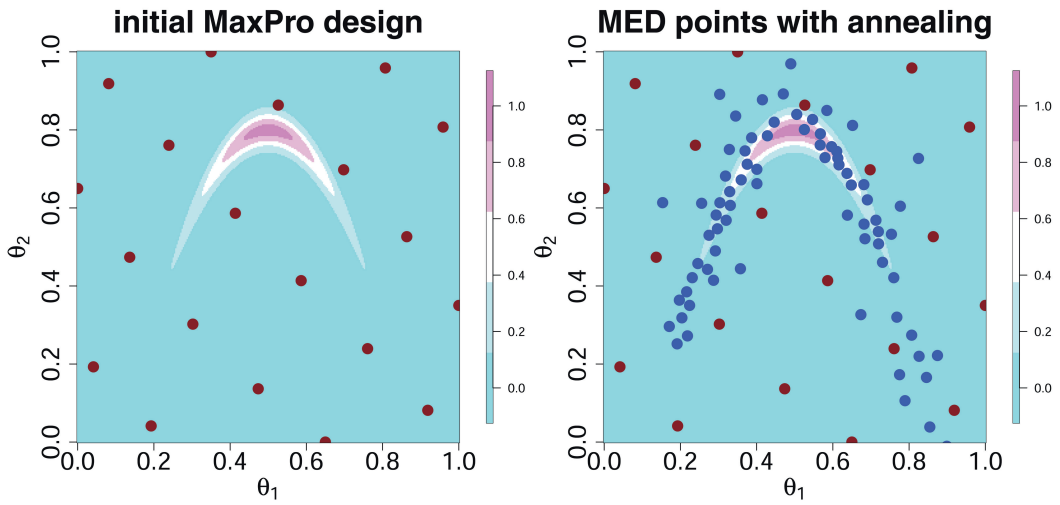


FIGURE 4.24: The left panel shows the initial 20-run MaxPro design points (red) overlayed on the banana-shaped posterior. The right panel shows the 80 MED points (blue) sequentially added in four steps with annealing.

where $\gamma = 1$ gives the target distribution. When $\gamma = 0$, we get a uniform distribution in the support of the posterior. Assume that, after proper scaling, the support of the posterior is $\mathcal{X} = [0,1]^p$. Then, we can start with a design in $[0,1]^p$ and then slowly increase γ to 1, while sequentially learning the unnormalized posterior.

For example, consider the banana-shaped density function given by Haario et al. (1999)

$$f(\boldsymbol{\theta}) \propto \exp\left\{-\frac{1}{2}\frac{\theta_1^2}{100} - \frac{1}{2}(\theta_2 + .03\theta_1^2 - 3)^2\right\}, \ \boldsymbol{\theta} \in \mathbb{R}^2. \tag{4.49}$$

This is not an expensive posterior and is used here only for illustrative purposes. Suppose that we know that the high-density region of the posterior lies in $[-40, 40] \times [-25, 10]$. We can re-scale this rectangle to a unit square $[0,1]^2$. Let us start with a 20-run MaxPro design on this square. The 20 points are shown over the image of the posterior in the left panel of Figure 4.24. We can see that the MaxPro design has completely missed the high-density region of the posterior. But now we add 20 MED points generated using the sequential algorithm given in (4.46) with $\gamma = 1/4$, where $h(\cdot)$ is replaced by a local approximation $\hat{h}(\cdot)$. This process continues with $\gamma = 2/4, 3/4$, and finally $\gamma = 1$. The algorithm is implemented in the R package `mined` (Wang and Joseph, 2022). The 100 MED points thus generated are shown in the right panel of Figure 4.24. We can now see that the points are placed in the high density region of the posterior, which is clearly a good experimental design to approximate the posterior.

Consider the Design of Experiments-based Interpolation technique (DoIt) proposed in Joseph (2013) for approximating posteriors. Let $D = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n\}$ be the design and $h_i = h(\boldsymbol{\nu}_i)$ for $i = 1, \ldots, n$. Then, the square-root of unnormalized density is approximated using a basis function expansion:

$$\sqrt{h(\boldsymbol{\theta})} \approx \sum_{i=1}^{m} c_i g(\boldsymbol{\theta}; \boldsymbol{\nu}_i, \boldsymbol{\Sigma}), \tag{4.50}$$

where the square-root is taken to ensure that the predicted values of the density will be non-negative,

$$g(\boldsymbol{\theta};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \exp\left\{-\frac{1}{2}\sum_{i=1}^{p}\frac{(\theta_i-\mu_i)^2}{\sigma_i^2}\right\},$$

and $\boldsymbol{\Sigma} = diag\{\sigma_1^2,\ldots,\sigma_p^2\}$. The σ_i^2's can be estimated using cross-validation methods as described in Section 2.1.2. Integrating from $-\infty$ to ∞ with respect to each θ_i, we obtain the normalizing constant

$$\begin{aligned}\int \hat{h}(\boldsymbol{\theta})\mathrm{d}\theta &= \int \hat{\mathbf{c}}'\mathbf{g}(\boldsymbol{\theta})\mathbf{g}(\boldsymbol{\theta})'\hat{\mathbf{c}}\ \mathrm{d}\theta \\ &= \pi^{d/2}|\boldsymbol{\Sigma}|^{1/2}\hat{\mathbf{c}}'\mathbf{G}_2\hat{\mathbf{c}}, \end{aligned} \tag{4.51}$$

where $\mathbf{G}_2$ is an $n\times n$ matrix with ijth element $g(\boldsymbol{\nu}_i;\boldsymbol{\nu}_j,2\boldsymbol{\Sigma})$. Thus, the approximation to the posterior density becomes

$$\hat{f}(\boldsymbol{\theta}|\mathbf{y}) = \frac{\{\hat{\mathbf{c}}'\mathbf{g}(\boldsymbol{\theta})\}^2}{\pi^{d/2}|\boldsymbol{\Sigma}|^{1/2}\hat{\mathbf{c}}'\mathbf{G}_2\hat{\mathbf{c}}}. \tag{4.52}$$

From this the marginal densities can be obtained as

$$\hat{p}(\theta_k|\mathbf{y}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}\phi(\theta_k;(\boldsymbol{\nu}_{ik}+\boldsymbol{\nu}_{jk})/2,\boldsymbol{\Sigma}_{kk}/2)}{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}}, \tag{4.53}$$

where $\phi(\theta;\mu,\sigma^2)$ is the normal density with mean μ and variance σ^2, and $d_{ij} = \hat{c}_i\hat{c}_j g(\boldsymbol{\nu}_i;\boldsymbol{\nu}_j,2\boldsymbol{\Sigma})$.

Since DoIt is applied on $\sqrt{h(\boldsymbol{\theta})}$, a 100-point MED is generated as in Figure 4.24 with $\sqrt{h(\boldsymbol{\theta})}$ as the target density. The DoIt marginal densities of θ_1 and θ_2 of the banana-shaped density in (4.49) are shown in Figure 4.25. The true posterior in this case can be obtained using numerical integration, which is shown in the same figure as the green lines. For comparison, the densities of 10,000 MCMC samples generated using the adaptive Metropolis algorithm of Vihola (2012) are also shown in the same figure as red lines. We can see that the DoIt gives a much better approximation to the true posterior than the MCMC. In this case DoIt used only 100 evaluations compared to 10,000 evaluations used by MCMC, which will be advantageous when we deal with computationally expensive posteriors.

4.6.2 Candidate Set Generation

Recall that in Section 4.5.1, a candidate set was first generated in the feasible region before finding the experimental design. When the feasible region is small in $\mathcal{X} = [0,1]^p$, the rejection sampling used there can become inefficient. In Figure 4.16, out of 10,000 uniform samples in $[0,1]^2$, only 538 samples were found to be feasible. The design generated from such a small candidate set may not be very good. Moreover, if the constraints are expensive to evaluate, then we waste a lot of resources by rejecting a large number of samples. We can make this process more efficient using minimum energy designs (Huang et al., 2021).

The key idea is to convert the inequality constraints $\{g_k(\mathbf{x})\le 0\}_{k=1}^K$ into a probability distribution so that we can use MED for sampling. Golchi and Loeppky (2015) proposed the following probabilistic relaxation using the probit function,

$$f_\tau(\mathbf{x}) \propto \Phi(-\tau g(\mathbf{x}))\ , \tag{4.54}$$

where Φ is the standard normal cumulative distribution function and $\tau\ge 0$ is the parameter that controls the rigidity of the constraint. We can see that the function ρ_τ assigns a value

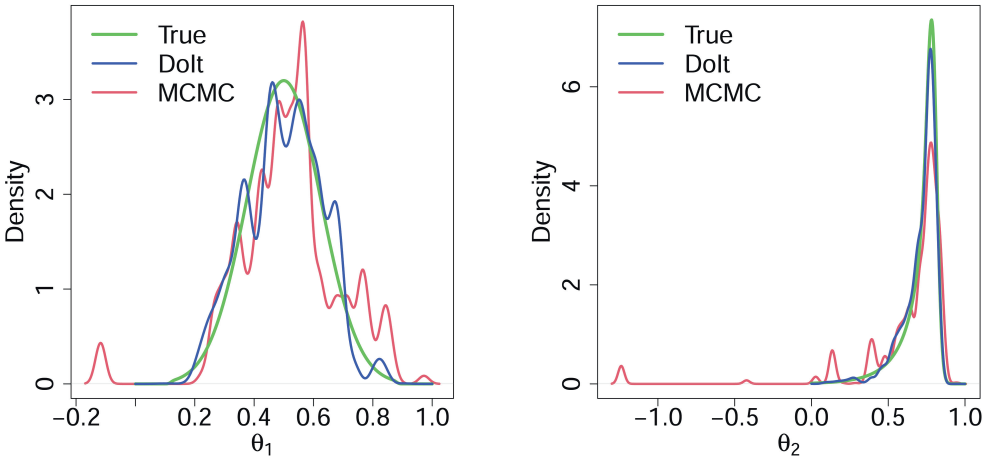


FIGURE 4.25: The marginal posterior densities of the banana-shaped posterior using numerical integration (green), DoIt using the 100-point MED (blue), and Adaptive Metropolis algorithm with 10,000 samples (red).

close to 1 for $\mathbf{x}$ that meets the constraint and a value close to 0 otherwise. Moreover, in the limit,

$$\lim_{\tau\to\infty} f_\tau(\mathbf{x}) \propto \lim_{\tau\to\infty} \Phi(-\tau g(\mathbf{x})) = I(g(\mathbf{x}) \le 0) \; . \tag{4.55}$$

This can be generalized to multiple inequality constraints $\{g_k(x) \le 0\}_{k=1}^K$ by

$$f_\tau(\mathbf{x}) \propto \prod_{k=1}^K \Phi(-\tau g_k(\mathbf{x})) \; . \tag{4.56}$$

Now choose an increasing sequence of the rigidity parameter $0 = \tau_0 < \tau_1 < \cdots < \tau_T$ and solve the MED problem sequentially:

$$\mathbf{x}_{m+1}^t = \underset{\mathbf{x}}{\text{argmax}} \min_{i=1:m} \left\{ \sum_{k=1}^K \log \Phi(-\tau_t g_k(\mathbf{x})) + \sum_{k=1}^K \log \Phi(-\tau_t g_k(\mathbf{x}_i)) + 2p \log \|\mathbf{x}_i - \mathbf{x}\|_\alpha \right\}, \tag{4.57}$$

for $t = 1, \ldots, T$. The full details of the algorithm can be found in Huang et al. (2021).

Consider again the constraints in (4.30). The left panel of Figure 4.26 shows 2,155 points generated using the foregoing algorithm with $T = 8$ aimed at constructing a 53-run MaxPro design. The right panel shows the 915 feasible candidate points. Thus, $915/2155 \times 100 = 42.5\%$ of the points are feasible, which is very high compared to the earlier $538/10000 \times 100 = 5.4\%$ of feasible points generated using 10,000 Latin hypercube samples. This will be extremely beneficial when the constraints are expensive to evaluate.

4.6.3 Continuous Fidelity Parameter

In many simulation codes, the fidelity of the results can be controlled using a continuous parameter, such as the mesh density in finite element analysis. As shown in Figure 4.20, a finer mesh can give more accurate results than that of a coarser mesh, but with higher computational cost.

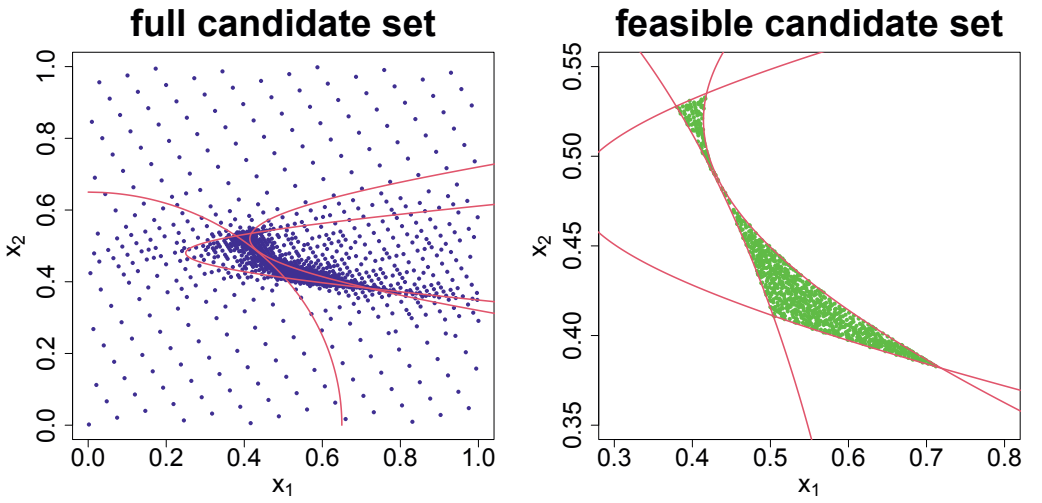


FIGURE 4.26: Left panel shows 2,155 candidate points obtained by applying the constrained MED algorithm. Right panel shows the 915 feasible candidate points.

Let the mesh density tuning parameter be t, where t could be the side length of a square or cuboidal mesh element. Tuo et al. (2014) proposed the following model:

$$y(\mathbf{x}, t) = y(\mathbf{x}, 0) + \delta(\mathbf{x}, t) = \phi(\mathbf{x}) + \delta(\mathbf{x}, t), \tag{4.58}$$

where $\phi(\mathbf{x})$ is the true response and $\delta(\mathbf{x}, t)$ denotes the bias in the simulation at input $\mathbf{x}$ under mesh density t. The true response function $\phi(\mathbf{x})$ is unattainable in finite element analysis (FEA) because it is impossible to run the simulation with $t = 0$. The beauty of Tuo et al.'s approach is that by modeling data from different fidelity levels, we can extrapolate and predict the response at $t = 0$. Similar to the model used in Kennedy and O'Hagan (2000), the two terms $\phi(\mathbf{x})$ and $\delta(\mathbf{x}, t)$ can be modeled by two independent Gaussian processes, where $\phi(\mathbf{x})$ has a stationary covariance function:

$$\text{cov}\{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)\} = \sigma_0^2 R_0(\mathbf{x}_1 - \mathbf{x}_2). \tag{4.59}$$

However, $\delta(\mathbf{x}, t)$ cannot be modeled by a stationary Gaussian process because $\delta(\mathbf{x}, t) \to 0$ as $t \to 0$ regardless of $\mathbf{x}$, breaking the necessary condition for a stationary process. Therefore, Tuo et al. (2014) proposed to use a nonstationary covariance function:

$$\text{cov}\{\delta(\mathbf{x_1}, t_1), \delta(\mathbf{x_2}, t_2)\} = \sigma_1^2 R_1(\mathbf{x}_1 - \mathbf{x}_2) K(t_1, t_2). \tag{4.60}$$

Specifically, they proposed using a Brownian motion kernel:

$$K(t_1, t_2) = \min(t_1, t_2)^l,$$

where l is typically set to 4 motivated by the error analysis of a finite element simulation. Note that $\text{Var}\{\delta(\mathbf{x}, t)\} = \sigma^2 t^l \to 0$ as $t \to 0$ and therefore the bias disappears from the model. Another choice for covariance is proposed in Gul et al. (2018). Let $Z(t)$ be a stationary GP with a covariance function $\tau^2 R_t(t - \cdot)$. Then, the process $Z(t) - Z(0)$ will be 0 when $t = 0$ and has the covariance function given by

$$\begin{aligned} K(t_1, t_2) &= cov\{Z(t_1) - Z(0), Z(t_2) - Z(0)\} \\ &= \tau^2\{R_t(t_1 - t_2) - R_t(t_1) - R_t(t_2) + 1\}. \end{aligned} \tag{4.61}$$

See Ji et al. (2024) for a recent application of this nonstationary covariance function in a multi-fidelity nuclear physics problem.

How to optimally choose a design $D = \{(\mathbf{x}_i, t_i)\}_{i=1}^n$? Yuchi et al. (2023) proposed an approach using minimum energy design (MED) which we describe below. Their main idea is to design a multi-fidelity simulation with the same computational cost as a single fidelity simulation. Assume that the computational time (T) of a single simulation is proportional to the number of mesh elements (M):

$$T = \beta M, \tag{4.62}$$

where β is a proportionality constant that depends on the meshing characteristics such as geometry and adaptation. The number of mesh elements (M) is related to the mesh density (t) by

$$t \propto \frac{1}{M^{1/k}}, \tag{4.63}$$

where k is the dimension of the mesh (usually 2 or 3). Therefore, we may choose $\{M_i\}_{i=1}^n$, which will give us $\{t_i\}_{i=1}^n$.

Suppose FEA simulations within a range of $\tilde{M}$ and $\bar{M}$ converge and produce reasonable results. If we were to perform the entire set of simulations at a single fidelity level, we can choose the upper bound $\bar{M}$ as the number of mesh elements. Suppose that we have a budget to perform $\bar{n}$ simulations at $\bar{M}$. Thus, the total time budget is $\bar{n} \times \beta\bar{M}$. We would like to divide this computational budget for simulations into n distinct levels $M_1, \ldots, M_n$. Thus, $\sum_{i=1}^n \beta M_i = \bar{n}\beta\bar{M}$. Therefore, the levels must satisfy the constraint

$$\sum_{i=1}^{n} M_i = \bar{n}\bar{M}. \tag{4.64}$$

Since we need more simulations at a small M compared to a large M, Yuchi et al. (2023) proposed to use the following density for choosing the optimal levels of M:

$$f(M) \propto \frac{1}{M}, \text{ for } M \in [\tilde{M}, \bar{M}].$$

The minimum energy design with this target density can be obtained by solving

$$\max_{M_1,\ldots,M_n} \min_{i\neq j} \frac{1}{\sqrt{M_i M_j}} |M_i - M_j|. \tag{4.65}$$

We can use the sequential construction method in (4.46) to find the optimal levels of M. Yuchi et al. (2023) proposed an inverse probability transform method to solve the same problem, which is much easier in this particular case. See Section 5.2.1 for more details. It is easy to see that the density of the target distribution is

$$f(M) = \frac{1/M}{\log(\bar{M}/\tilde{M})}, \text{ for } M \in [\tilde{M}, \bar{M}]$$

and the cumulative distribution function is

$$F(M) = \frac{\log(M/\tilde{M})}{\log(\bar{M}/\tilde{M})}, \text{ for } M \in [\tilde{M}, \bar{M}].$$

Thus, $u = F(M)$ has a uniform distribution in $[0, 1]$. Therefore, under this transformation, (4.65) becomes

$$\max_{u_1,\ldots,u_n} \min_{i\neq j} |u_i - u_j|.$$

We already know the solution to this one-dimensional maximin design problem, which is given in (4.19). Thus, the solution to (4.65) is given by

$$\begin{aligned} M_i &= F^{-1}(u_i) \\ &= \tilde{M}\left(\bar{M}/\tilde{M}\right)^{u_i}, \quad i = 1, \ldots, n, \end{aligned} \tag{4.66}$$

where $u_i = (i-1)/(n-1)$. Substituting (4.66) in (4.64), we obtain

$$\sum_{i=1}^{n}\left(\bar{M}/\tilde{M}\right)^{u_i} = \bar{n}\left(\bar{M}/\tilde{M}\right). \tag{4.67}$$

Let $\lambda = \bar{M}/\tilde{M}$. Then, (4.67) becomes

$$\sum_{i=1}^{n}\lambda^{\frac{i-1}{n-1}} = \bar{n}\lambda,$$

which gives

$$\frac{\lambda^{n/(n-1)}-1}{\lambda^{1/(n-1)}-1} = \bar{n}\lambda.$$

Solving for n and choosing the nearest integer solution, we obtain

$$n = 1 + \left[\frac{\log\lambda}{\log\nu}\right], \tag{4.68}$$

where $\nu = (\bar{n}\lambda - 1)/\{\lambda(\bar{n}-1)\}$ and $[x]$ denotes the nearest integer of x. Once n is obtained, $\{M_1, \ldots, M_n\}$ can be obtained from (4.66). Since the proportionality constant in (4.63) will be absorbed into σ_1^2, we can let the optimal level of t's to be $\left\{1/M_i^{1/k}\right\}_{i=1}^{n}$ or

$$\{t_i\}_{i=1}^{n} = \left\{\left(\frac{M_1}{M_n}\right)^{1/k}, \left(\frac{M_1}{M_{n-1}}\right)^{1/k}, \ldots, 1\right\}. \tag{4.69}$$

Consider a toy example from Sung et al. (2024), in which the output depends on two inputs (x_1, x_2) and a tuning parameter M:

$$y = \phi(\mathbf{x}) + \frac{16}{M}e^{-1.4x_1}\cos(3.5\pi x_2),$$

where

$$\phi(\mathbf{x}) = \left(1 - e^{-.5/x_2}\right)\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}. \tag{4.70}$$

The true input-output relationship is $y = \phi(\mathbf{x})$, but it cannot be computed because the cost is proportional to M. As in Sung et al. (2024), let $M \in [2, 16]$. If we were to perform a single-fidelity simulation at $M = 16$, we would have a total budget of $\bar{n} = 50$ simulations. We will be able to perform more simulations with the same computational cost spreading M in $[2, 16]$. From (4.68), we obtain $n = 118$. Using (4.66) and (4.69), we obtain n levels for $t = 1/M$. Now, to obtain the design, we can start with a MaxPro design in 118 runs in three factors. The levels in the third column corresponding to t can be replaced with their optimal levels. Treating it as a discrete-numeric factor, we can use the `MaxProQQ` function (Ba and Joseph, 2018) to obtain the final design. A scatter plot of this design is shown in Figure 4.27. We can see a clear nonuniform pattern for t.

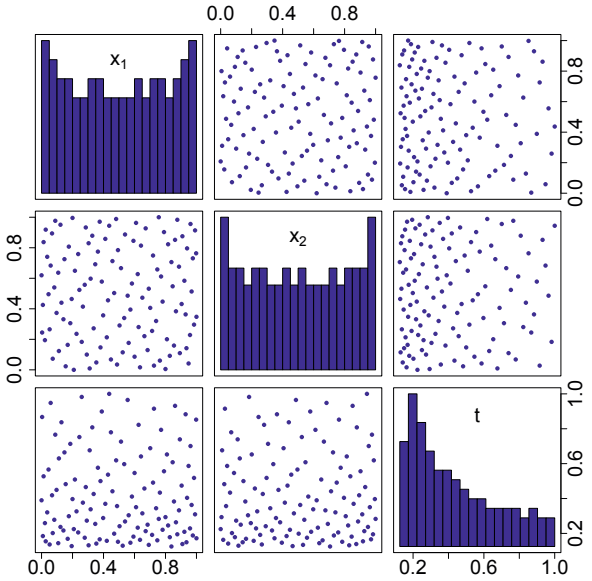


FIGURE 4.27: Scatter plot matrix of the design for the multi-fidelity simulation example.

Let $\phi(\mathbf{x}) \sim GP(\mu, \sigma_0^2 R_0(\mathbf{x} - \cdot))$ and $\delta(\mathbf{x}, t) \sim GP(0, \sigma_1^2 R_1(\mathbf{x} - \cdot)K(t, \cdot))$ in (4.58). Since the sum of two GPs is a GP, under (4.58), $y(\mathbf{x}, t)$ also follows a GP:

$$y(\mathbf{x}, t) \sim GP(\mu, \sigma_0^2 R_0(\mathbf{x} - \cdot) + \sigma_1^2 R_1(\mathbf{x} - \cdot)K(t, \cdot)).$$

Thus, all hyperparameters can be estimated as described in Section 2.1.4. We used the kernel in (4.61) with $R_t(\cdot)$ as a Gaussian correlation function. The fitted model can be used to predict the output at $t = 0$ (or $M = \infty$). We have generated 100 testing points using `MaxProAugment` as described in Section 4.5. In this toy example, the true outputs can be computed using (4.70). The RMSE for the 100 test points is obtained as 0.054. For comparison, we also fitted a single-fidelity model using a MaxPro design of size $\bar{n} = 50$ at $M = 16$. This gives an RMSE of 0.420, which is much higher than what we obtained using multifidelity modeling.

Exercises

1. Suppose that $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$, where the isotropic correlation function $R(\boldsymbol{\Delta}) = G(\|\boldsymbol{\Delta}\|)$ is nonnegative and non-increasing in $\|\boldsymbol{\Delta}\|$. Show that (Pronzato and Müller, 2012)

$$MMSE(D) \leq 1 - \frac{G^2(\phi(D))}{1 + (n-1)G(2\psi(D))},$$

   where $\phi(D)$ is the fill distance and $\psi(D)$ the separation distance of the design D.

2. Generate a 20-run design in two dimensions using the clustering-based design in (4.4) using the `SFDesign` package. Find 20 cluster centers using the R packages `Kmedians`

(Godichon-Baggioni and Surendran, 2023) and `Gmedian` (Cardot, 2022) based on a large Monte Carlo sample. Compare the three designs in terms of separation distance and fill distance.

3. Consider an $LHD(n, p)$ in $[0, 1]^p$. Show that the average rectangular and squared-Euclidean distances between the n points are always constant given by (Joseph and Hung, 2008; van Dam et al., 2009)

$$\frac{p(n+1)}{3n} \text{ and } \frac{p(n+1)}{6n},$$

respectively.

4. Generate the following 20-run designs in two dimensions, and report their separation and fill distances.

   (a) Clustering-based design

   (b) Maximin design

   (c) Minimax design

   (d) Maximin LHD

   (e) MaxPro design

5. In order to optimize a solid end milling process (Joseph, 2016), a 60-run MaxPro design was generated for the six variables: hardness of the workpiece material (x_1), rake angle (x_2), helix angle (x_3), relief angle (x_4), corner radius (x_5), and flute length (x_6) of the cutting tool. The simulations were carried out using the commercially available Production Module software (Third Wave Systems, Minneapolis, MN). The response (y) is the average tangential cutting force on the tool (averaged over time). The dataset is given in the file `milling.txt`.

   (a) Fit an ordinary kriging model with Gaussian correlation function using the R package `rkriging`. Report the estimates of correlation parameters.

   (b) Find the optimum settings of the six variables in $[0, 1]^6$ by minimizing the surrogate model for the average tangential force obtained in (a).

6. Generate a 25-run MaxPro design in the experimental region defined by the following constraints:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 &\geq x_2 \\ (x_1 - .5)^2 + (x_2 - .5)^2 &\leq 0.25 \\ x_1, x_2 &\in [0, 1]^2. \end{aligned}$$

   Plot the experimental region and show the design points on it.

7. Suppose there are two continuous factors (x_1 and x_2), one discrete-numeric factor (x_3) taking values $\{1, 2, 3\}$, and a nominal factor (x_4) with four levels. Generate a 24-run MaxProQQ design. Plot the design by showing the points in a x_1 vs x_2 plot with different colors for the levels of x_3 and different plotting symbols for the levels of x_4.

8. Suppose that we need to perform a computer experiment using three variables: material type (x_1), speed (x_2), and depth of cut (x_3). There are three materials to test: steel, copper, and aluminum; speed can be varied between 1,000 and 2,000; and depth of cut from 0.1 to 0.3. Construct a good space-filling design in 9 runs. Provide the design matrix in both coded and actual values.

9. Consider the hard turning experiment discussed in Section 4.5.3. The design and data from Hung et al. (2009) are given in the file `turning.csv`. Compare this design with the design obtained in Section 4.5.3 in terms MaxPro criteria. Which design is better?

10. Consider two alternative approaches to designing the hard turning experiment in Section 4.5.3.

    Approach 1: Generate a MaxPro (15,8) design for the chamfer edge tool and a MaxPro (15,6) design for the honed edge tool. Assemble them with a column for the cutting edge shape to obtain the final design.

    Approach 2: Generate a MaxPro (30,6) design for the shared factors and a MaxPro (15,2) for the nested factors under the chamfer edge. Assemble them with a column for the cutting edge shape to obtain the final design.

    Which approach produced a better design? Compare them with the design obtained in Section 4.5.3 in terms of MaxPro criteria.

11. Suppose that we want to create a nested design with $n_1 = 100$ runs for the low-fidelity level and $n_2 = 20$ for the high-fidelity level. Construct nested designs using the following three strategies.

    (a) Use the method described in Section 4.5.4 using the `SFDesign` package.

    (b) Use the `MaxProAugment` function in the `MaxPro` package starting with a MaxPro design for the high-fidelity level.

    (c) Use the `NestedX` function in the R package `RNAmf` (Heo and Sung, 2021)

    Compare the three designs using visualizations and MaxPro criterion.

12. Three models with varying fidelity are available to simulate a system. Model 1 is the cheapest of the three but has the lowest accuracy. Model 3 is the most expensive code, but has the highest accuracy. Model 2's cost and accuracy are between those of the other two models. Two variables (x_1 and x_2) need to be studied. Create a nested space-filling design D_1 with $n_1 = 100$ runs, D_2 with $n_2 = 50$ runs, and D_3 with $n_3 = 10$ runs using the following two approaches.

    Approach 1 (forward): Generate D_3 in n_3 runs using a MaxPro design. Also generate a candidate set of points of size $100n_1$. Now use `MaxProAugment` function to augment $n_2 - n_3$ runs to D_3 using the candidate set to obtain D_2. Now augment $n_1 - n_2$ runs to D_2 using the candidate set to obtain D_1. Plot the three designs on one plot by differentiating them using different symbols and colors for the points.

    Approach 2 (backward): Generate D_1 in n_1 runs using a MaxPro design. Now use `maxpro.remove` function to select n_2 runs from D_1 to obtain D_2. Now use `maxpro.remove` function again to select n_3 runs from D_2 to obtain D_3. As before, plot the three designs on one plot.

    Compare the nested designs produced by the two approaches using separation distance, fill distance, and MaxPro criterion. Which approach produced a better nested space-filling design?

13. Dasgupta et al. (2011) reported a case study on predicting the density of nanowires (y) with respect to the thickness of polymer films (x) in a solution-based growth process. Eight experiments were conducted with two replicates (except for one run). The data are in the file `nanowire.csv`. The density of nanowires is assumed to follow a Poisson distribution with mean:

$$\mu(x) = \theta_1 \exp(-\theta_2 x^2) + \theta_3\{1 - \exp(-\theta_2 x^2)\}\Phi(-x/\theta_4),$$

where $\Phi(\cdot)$ is the standard normal distribution function. Assume the following prior distribution for the parameters:

$$\log\theta_1, \log\theta_3, \log\theta_4 \overset{iid}{\sim} \mathcal{N}(0, 10^2) \text{ and } \theta_2 \sim \mathcal{U}(0, 1).$$

Generate a minimum energy design in 100 runs. Compare the marginal densities of the four parameters using the following three methods: (i) empirical density of the minimum energy design points, (ii) DoIt approximation obtained using the minimum energy design, and (iii) empirical density from 10,000 MCMC samples.

14. Suppose that we want to generate a 101-run MaxPro design in the five-dimensional space:

$$\mathcal{X} = \left\{ 0 \le x_i \le 1, i = 1, ..., 5, \text{ and } \sum_{i=1}^{5} i x_i \le 5 \right\}.$$

Use the constrained minimum energy design to generate a candidate set in $T = 8$ steps. What percentage of the candidate points are feasible? Use the candidate set to obtain the desired 101-run MaxPro design using the `MaxProAugment` function.

15. Yuchi et al. (2023) report a multi-fidelity experiment using finite element analysis to study the deflection of a cantilever beam. A continuous static half-sine pressure field is applied vertically downward onto the top surface of the beam.

$$\text{pressure} = 2000 x_1 \sin(\pi z / 200 x_2),$$

where the input variable $x_1 \in [0, 1]$ is a weighting parameter, $x_2 \in [0, 1]$ controls the length of the beam $200x_2$, x_3 denotes the width and height of the beam with square cross section, and $z \in [0, 200x_2]$ is the distance from the fixed surface. A reasonable range for the number of mesh cells is $\tilde{M} = 1,000$ and $\bar{M} = 8,000$. Let the time budget be equivalent to the total computational time of running eight simulations with $\bar{M}$. Three experimental designs were investigated. First, a Single-Mesh design with eight points in the input space $[0, 1]^3$ obtained using `MaxPro`. The Double-Mesh design uses a two-layer nested design with half of the time budget allocated to each layer. The first layer consists of 32 simulations at $\tilde{M}$ and the second layer contains four simulations at $\bar{M}$. Finally, for the Multi-Mesh method described in Section 4.6.3, $n = 18$ simulations with different values of $M \in [\tilde{M}, \bar{M}]$ are obtained. The designs and data can be downloaded from the book's GitHub site. Generate the multi-mesh design using `MaxProQQ` as explained in Section 4.6.3 and compare it with the design in Yuchi et al. (2023).

5

Representative Points

Figure 3.1 in Chapter 3 shows seven points randomly sampled from a uniform distribution in $[0,1]^2$. We argued that they are not good as experimental designs because of their poor space-filling properties. However, is it possible to sample points so that they are more uniformly distributed in the space? In this chapter, we investigate this question in detail, leading to the concept of uniform designs. Later we will generalize this concept to obtain representative points from any distribution. These representative point sets are not directly aimed at approximating functions, but they have certain optimality properties when the objective is integration or uncertainty propagation.

5.1 Uniform Design

This section will focus on methods for optimal sampling from a uniform distribution in $[0,1]^p$. See the book Fang et al. (2005) for more details on uniform design.

5.1.1 Uniformity

First, consider the case of a one-dimensional variable. Suppose that we randomly sample seven values in $\mathcal{U}[0,1]$ and obtain (after sorting): $D = \{0.202, 0.266, 0.372, 0.573, 0.898, 0.908, 0.945\}$. We can assign a probability mass of $1/n$ for each of them:

$$Pr(x = x_i) = \frac{1}{n}$$

and obtain its cumulative distribution function as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I\{x_i \leq x\}, \tag{5.1}$$

where $I(\cdot)$ is an indicator function, that is, $I(A) = 1$ if A occurs and 0 otherwise. $F_n(x)$ is called an empirical distribution function of the design D. It is plotted in the left panel of Figure 5.1. The distribution function of a uniform random variable is given by $F(X) = x$ for $x \in [0,1]$. This is plotted as a green dashed line in the same figure. We can compare $F_n(x)$ to $F(x)$ to measure how close is D to a uniformly distributed variable. One measure is

$$\sup_{x\in[0,1]} |F_n(x) - F(x)|.$$

This is called a star discrepancy in numerical mathematics, which is the same as the Kolmogorov-Smirnov goodness-of-fit test statistic in statistics. It can be shown that

DOI: 10.1201/9781003661535-5

This chapter has been made available under a CC-BY-NC-ND license 4.0.

(Fang and Wang, 1993, p.16)

$$\left\{\frac{i-.5}{n}\right\}_{i=1}^{n}$$

minimizes the star discrepancy. This is exactly the same as the minimax design in (4.12), making a strong connection to the space-filling designs. The empirical distribution under the optimal design (i.e., uniform design), is shown in the right panel of Figure 5.1.

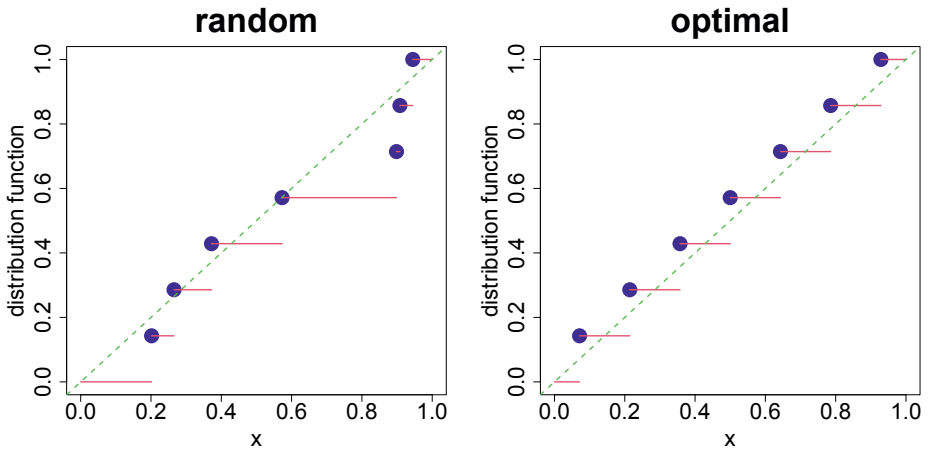


FIGURE 5.1: Empirical distribution function of seven randomly sampled points (left) and optimally sampled points (right).

Now consider the higher-dimensional cases. As before, we can define the empirical distribution of a design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in [0,1]^p$, as

$$F_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} I(\mathbf{x}_i \le \mathbf{x}).$$

Here $\sum_{i=1}^{n} I(\mathbf{x}_i \le \mathbf{x})$ is simply the number of points falling in the p-dimensional hypercube $[\mathbf{0}, \mathbf{x}] = [0, x_1] \times \ldots \times [0, x_p]$. The star discrepancy becomes

$$\mathcal{SD}(D) = \sup_{\mathbf{x}\in[0,1]^p} |F_n(\mathbf{x}) - F(\mathbf{x})|,$$

where $F(\mathbf{x}) = x_1 x_2 \cdots x_p$.

Unfortunately, the minimizer of the star discrepancy for $p \ge 2$ is not clear at all. In fact, it is not even easy to compute when $p \ge 2$. An easier measure is the star L_2-discrepancy given by

$$\mathcal{SL}_2(D) = \int_{[0,1]^p} \{F_n(\mathbf{x}) - F(\mathbf{x})\}^2 d\mathbf{x},$$

which coincides with the Cramér-von Mises statistic for testing the goodness-of-fit of a uniform distribution. Warnock (1972) gave an explicit expression for computing the star L_2-discrepancy:

$$\mathcal{SL}_2 = \frac{1}{3^p} - \frac{2^{1-p}}{n}\sum_{i=1}^{n}\prod_{l=1}^{p}(1-x_{il}^2) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{l=1}^{p}\{1-\max(x_{il}, x_{jl})\}. \tag{5.2}$$

Although uniform designs do a good job in approximating a uniform distribution in $[0,1]^p$, their optimality regarding approximating a given function is far from clear. However, they do possess strong optimality properties if the objective is to integrate a function, which is discussed in the next section.

5.1.2 Integration

Suppose our objective is to integrate a function:

$$I = \int_{[0,1]^p} h(\mathbf{x}) d\mathbf{x}. \tag{5.3}$$

For one-dimensional integration, one can use the common numerical integration techniques such as trapezoidal or Simpson's rule, which essentially approximate the integral as

$$\hat{I} = \sum_{i=1}^{n} w_i h(x_i), \tag{5.4}$$

where $\{x_1, \ldots, x_n\}$ are the integration points and w_i's are some weights. It is known that the absolute integration error $|\hat{I} - I|$ is $\mathcal{O}(1/n^2)$ for trapezoidal rule and $\mathcal{O}(1/n^3)$ for Simpson's rule. However, this rate of convergence is severely reduced in higher dimensions. To see this, suppose that we construct the m-point integration rule in p-dimensions using tensor products. Then, there will be a total of $n = m^p$ function evaluations. This means that the rate of convergence with respect to the number of function evaluations will be $\mathcal{O}(1/n^{2/p})$ for the trapezoidal rule and $\mathcal{O}(1/n^{3/p})$ for Simpson's rule. Therefore, these methods are not good for high-dimensional integration.

Another approach to evaluate (5.3) is to recognize that I can be written as the expectation of a uniform random variable:

$$I = E\{h(X)\},$$

where $X \sim \mathcal{U}[0,1]^p$. A simple estimate of the mean is the sample average:

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}_i),$$

where $\mathbf{x}_i \overset{iid}{\sim} \mathcal{U}[0,1]^p$, $i = 1, \ldots, n$. By the strong law of large numbers $\hat{I} \to I$ almost surely as $n \to \infty$. In fact, we can compute the variance of $\hat{I}$ as

$$var\{\hat{I}\} = \frac{1}{n^2} \sum_{i=1}^{n} var\{h(\mathbf{x}_i)\} = \frac{\tau^2}{n},$$

where $\tau^2 = var\{h(\mathbf{X})\}$. Thus, a 95% confidence interval for I can be constructed as $\hat{I} \pm 2\tau/\sqrt{n}$. In other words, the integration error reduces to zero at the rate $\mathcal{O}_P(1/\sqrt{n})$. Although this is a poor convergence rate for a one-dimensional integration compared to the trapezoidal or Simpson's rule, it is much better for high-dimensional integration when p is more than 4 or 6. See the book Robert et al. (2010) for details on Monte Carlo methods.

O'Hagan (1987) criticizes the use of Monte Carlo methods for integration, but it is only the "uniformity" of points and not the "randomness" that matters. This is captured in the Koksma-Hlawka inequality for the integration error (Niederreiter, 1992)

$$|\hat{I} - I| \leq \mathcal{SD}(D) V(h),$$

where $V(h)$ is a measure of variation of h. Thus, by minimizing the discrepancy measure of the point set with respect to a uniform distribution, we can minimize the upper bound of the integration error for any h. The foregoing development relies on the theory of reproducing kernel Hilbert space. As discussed in Sections 2.2.2 and 2.2.3, since the theory of

reproducing kernel Hilbert space is intimately connected with Gaussian processes, we will use a statistician's approach and describe the details using Gaussian processes (Briol et al., 2019).

Assume that

$$h(\mathbf{x}) \sim GP(\mu, \tau^2 K(\mathbf{x}, \cdot)),$$

where $cov\{h(\mathbf{u}), h(\mathbf{v})\} = \tau^2 K(\mathbf{u}, \mathbf{v})$. Then,

$$\begin{aligned} E\{\hat{I} - I\} &= E\left\{\frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i) - \int_{[0,1]^p} h(\mathbf{x}) d\mathbf{x}\right\} \\ &= \frac{1}{n}\sum_{i=1}^{n} \mu - \int_{[0,1]^p} \mu \, d\mathbf{x} \\ &= \mu - \mu = 0. \end{aligned}$$

Therefore, the variance becomes

$$\begin{aligned} var\{\hat{I} - I\} &= var\left\{\frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i) - \int_{[0,1]^p} h(\mathbf{x}) d\mathbf{x}\right\} \\ &= E\left\{\frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i) - \int_{[0,1]^p} h(\mathbf{x}) d\mathbf{x}\right\}^2 \\ &= E\left\{\frac{1}{n}\sum_{i=1}^{n} \int_{[0,1]^p} [h(\mathbf{x}) - h(\mathbf{x}_i)] d\mathbf{x}\right\}^2 \\ &= \frac{1}{n^2} E\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} \int_{[0,1]^p} [h(\mathbf{x}) - h(\mathbf{x}_i)] d\mathbf{x} \int_{[0,1]^p} [h(\mathbf{x}) - h(\mathbf{x}_j)] d\mathbf{x}\right\} \\ &= \frac{1}{n^2} E\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} \int_{[0,1]^p}\int_{[0,1]^p} [h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_j)] d\mathbf{u} d\mathbf{v}\right\} \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \int_{[0,1]^p}\int_{[0,1]^p} E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_j)]\right\} d\mathbf{u} d\mathbf{v} \end{aligned}$$

Note that

$$\begin{aligned} E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_j)]\right\} &= cov\{h(\mathbf{u}) - h(\mathbf{x}_i), h(\mathbf{v}) - h(\mathbf{x}_j)\} \\ &= \tau^2 K(\mathbf{u}, \mathbf{v}) - \tau^2 K(\mathbf{u}, \mathbf{x}_j) - \tau^2 K(\mathbf{v}, \mathbf{x}_i) + \tau^2 K(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

Therefore,

$$\begin{aligned} &\int_{[0,1]^p}\int_{[0,1]^p} E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_j)]\right\} d\mathbf{u} d\mathbf{v} = \\ &\tau^2 \int_{[0,1]^p}\int_{[0,1]^p} K(\mathbf{u}, \mathbf{v}) d\mathbf{u} d\mathbf{v} - \tau^2 \int_{[0,1]^p} K(\mathbf{x}, \mathbf{x}_i) d\mathbf{x} - \tau^2 \int_{[0,1]^p} K(\mathbf{x}, \mathbf{x}_j) d\mathbf{x} + \tau^2 K(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

Thus,

$$var\{\hat{I} - I\} = \tau^2 \mathcal{K}(D),$$

where

$$\mathcal{K}(D) = \int_{[0,1]^p}\int_{[0,1]^p} K(\mathbf{u}, \mathbf{v}) d\mathbf{u} d\mathbf{v} - \frac{2}{n}\sum_{i=1}^{n} \int_{[0,1]^p} K(\mathbf{x}, \mathbf{x}_i) d\mathbf{x} + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j) \quad (5.5)$$

is called a kernel discrepancy (Hickernell, 1998a, 1999).

Hickernell also derived various discrepancy measures by choosing different types of kernels. For example, the star L_2 discrepancy in (5.2) can be obtained by using the kernel

$$K(\mathbf{u},\mathbf{v}) = \prod_{l=1}^{p}\{1-\max(u_l,v_l)\},$$

which is known as the Brownian sheet kernel. Hickernell (1998a,b) also noticed that both $\mathcal{SD}(D)$ and $\mathcal{SL}_2(D)$ do not produce point sets with good projections. Therefore, he proposed the following two discrepancy measures

$$\begin{aligned}\mathcal{CL}_2(D) &= \left(\frac{13}{12}\right)^p - \frac{2}{n}\sum_{i=1}^{n}\prod_{l=1}^{p}\left[1+\frac{1}{2}|x_{il}-.5| - \frac{1}{2}(x_{il}-.5)^2\right] \\ &+ \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{l=1}^{p}\left[1+\frac{1}{2}|x_{il}-.5| + \frac{1}{2}|x_{jl}-.5| - \frac{1}{2}|x_{il}-x_{jl}|\right] \qquad (5.6)\end{aligned}$$

and

$$\mathcal{W}(D) = -\left(\frac{4}{3}\right)^p + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{l=1}^{p}\left[\frac{3}{2} - |x_{il}-x_{jl}| + (x_{il}-x_{jl})^2\right], \qquad (5.7)$$

which are called the centered L_2 discrepancy and wrap-around discrepancy, respectively. They can be obtained using the kernels

$$K(\mathbf{u},\mathbf{v}) = \prod_{l=1}^{p}\left[1+\frac{1}{2}|u_l-.5| + \frac{1}{2}|v_l-.5| - \frac{1}{2}|u_l-v_l|\right]$$

and

$$K(\mathbf{u},\mathbf{v}) = \prod_{l=1}^{p}\left[\frac{3}{2} - |u_l-v_l| + (u_l-v_l)^2\right],$$

respectively.

Minimizing the discrepancy measure is not easy. Therefore, most of the research in the area of Quasi-Monte Carlo (QMC) has focused on generating point sets with a low discrepancy. They use number-theoretic methods for their construction (Fang and Wang, 1993), and thus are quick to generate. These point sets can achieve an almost $\mathcal{O}(1/n^{1-\epsilon})$ convergence rate for some $\epsilon > 0$, which is much better than the Monte Carlo convergence rate of $\mathcal{O}(1/\sqrt{n})$. See the books Leobacher and Pillichshammer (2014) and Owen (2013) for more details of QMC. The middle panel of Figure 5.2 shows a 20-point Sobol' sequence in $[0,1]^2$ generated using the R package `spacefillr` (Morgan-Wall, 2025). This package implements a type of Sobol' sequence that has better two-dimensional projections (Joe and Kuo, 2008). Clearly, it is more uniformly scattered in $[0,1]^2$ than a 20-point Monte Carlo sample shown on the left panel of the same figure.

Although low-discrepancy sequences are fast to generate and have many applications, they are not suitable for computer experiments where each evaluation of the computer code is costly. For an expensive h, it is perfectly fine to spend some time optimizing the discrepancy measure so that we have an optimal set for integration. The discrepancy measure can be minimized by using some exchange algorithms similar to those used for maximin or MaxPro Latin hypercube designs. See, for example, Jin et al. (2005). A 20-point uniform design in $[0,1]^2$ obtained by the `SFDesign` package (Wang and Joseph, 2025c), which minimizes the wrap-around discrepancy is shown in the right panel of Figure 5.2. Visually, this design has

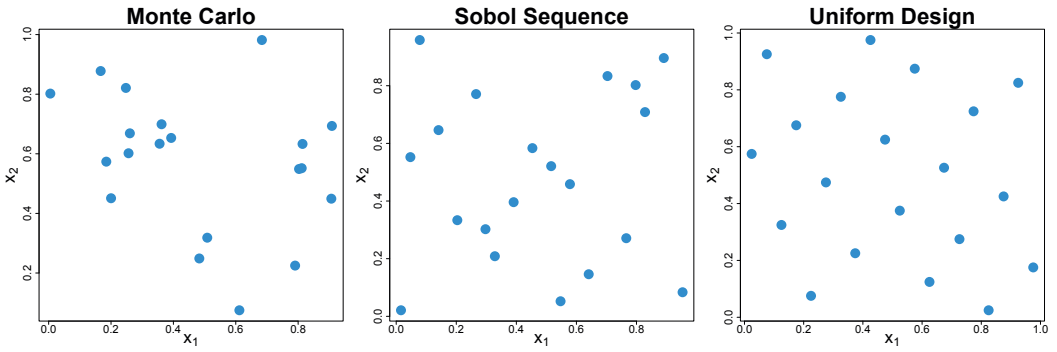


FIGURE 5.2: A 20-point Monte Carlo sample (left), Sobol' sequence (middle), and uniform design (right) in $[0,1]^2$.

much better space-fillingness and uniformity compared to the Sobol' sequence. Quantitatively, the uniform design has $\mathcal{W}(D) = 0.0412$, which is much smaller than $\mathcal{W}(D) = 0.0591$ of the Sobol' sequence and $\mathcal{W}(D) = 0.1810$ of the Monte Carlo sample.

As an example, consider the problem of evaluating a two-dimensional integral:

$$I = \int_0^1 \int_0^1 e^{-(x_1-.5)^2-(x_2-.5)^2}\, dx_1\, dx_2.$$

This can be analytically integrated to obtain $I = \pi\{\Phi(.5\sqrt{2}) - \Phi(-.5\sqrt{2})\}^2 = 0.8511$, where $\Phi(\cdot)$ is the standard normal distribution function. Now consider estimating the integral using sample average, where the sample is generated using three methods: Monte Carlo, Sobol' sequence, and uniform design. We have varied n from 10 to 100 at an interval of 10, and generated samples 30 times for each n. Thus, for each n, we have 30 different values of the integral for each of the three methods. Their medians along with 5% and 95% quantiles (as shaded regions) are plotted in the right panel of Figure 5.3. The left panel of the figure shows their corresponding $\mathcal{W}(D)$ values. We can see that the wrap-around discrepancy values are the largest for MC and the smallest for uniform design. Although the MC samples give approximately the true value on average, they have a high variability. The uniform design

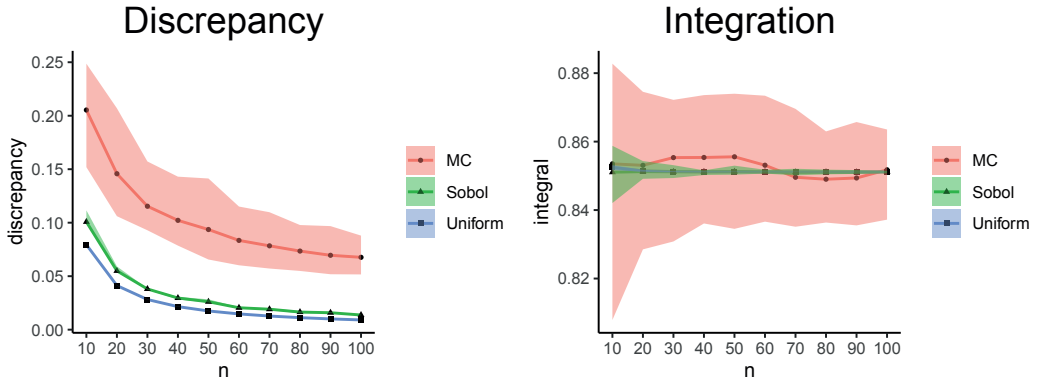


FIGURE 5.3: Wrap-around discrepancy values (left) and the two-dimensional integral values (right) using three different sampling methods: Monte Carlo, Sobol' sequence, and Uniform design.

gives the smallest variability around the true value. This is the advantage of having a better experimental design (smaller discrepancy value), which provides better protection against worst-case scenarios.

5.2 Non-uniform Distributions

In this section, we will consider the problem of optimally sampling from non-uniform distributions. The results of this section also have important applications in data science, which we will explore later in Part IV of the book.

5.2.1 Transformation Methods

Suppose our aim is to obtain a representative sample of a distribution $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^p$. Once we have a sample $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we can use it to approximate an integral

$$I = \int h(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x}$$

with the sample average

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}_i).$$

The aim is to find an optimal design D so that the integration error $|\hat{I} - I|$ is small for *any* integrand $h(\cdot)$.

Since we already know how to solve the foregoing problem for the case of a uniform distribution, is it possible to somehow make use of that result for the case of general distributions? To answer this, first consider the case of a one-dimensional problem. The following result from probability is useful.

Lemma 5.1 *If $U \sim \mathcal{U}(0,1)$, then $X = F^{-1}(U)$ has a distribution function $F(\cdot)$.*

This result is easy to show because

$$Pr\{X \leq x\} = Pr\{F^{-1}(U) \leq x\} = Pr\{U \leq F(x)\} = F(x).$$

Thus, if we have a random sample $u_i \overset{iid}{\sim} \mathcal{U}(0,1)$, for $i = 1, \ldots, n$, then $\{F^{-1}(u_i)\}_{i=1}^n$ is a random sample from F. As shown in Fang and Wang (1993), $\{F^{-1}(u_i)\}_{i=1}^n$ minimizes the star discrepancy $\sup |F(x) - F_n(x)|$ if $\{u_i\}_{i=1}^n$ minimizes the star discrepancy with respect to a uniform distribution. Thus, we can obtain an estimate of I using

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} h(F^{-1}(u_i)),$$

where $\{u_i\}_{i=1}^n$ is an MC sample, a QMC sample, or a uniform design. This result is obvious if we use a change-of-variable technique to evaluate the integral with $F(x) = u$:

$$I = \int h(x) f(x) \, dx = \int_0^1 h(F^{-1}(u)) \, du.$$

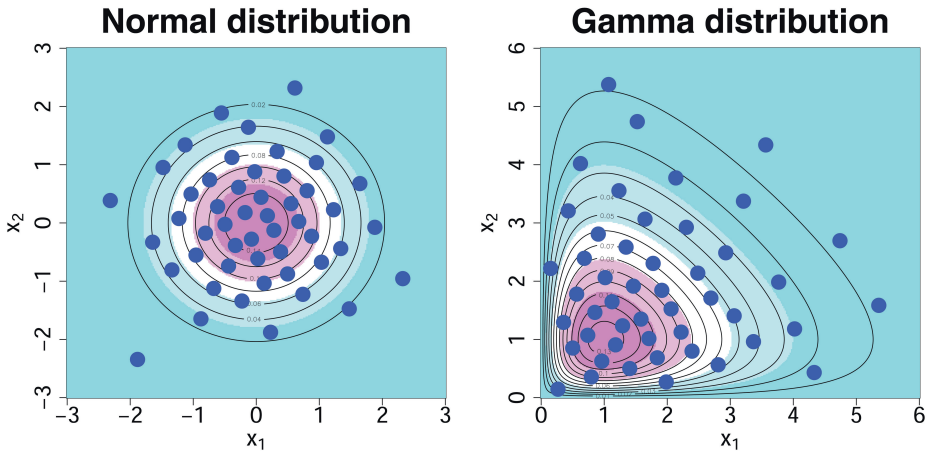


FIGURE 5.4: 50 representative points of a bivariate normal distribution (left) and Gamma distribution (right) obtained using inverse probability transform of a uniform design. The image and contours of the distributions are also shown.

Unfortunately, in general, the foregoing result cannot be used for multidimensional integration except for some special cases. An important special case where this transformation technique works is when the random variables are independent:

$$F(\mathbf{x}) = \prod_{i=1}^{p} F_i(x_i),$$

where $F_i(x_i)$ is the distribution function of the ith variable. Now we can do the inverse probability transform for each variable and obtain the integral as

$$\begin{aligned} I &= \int h(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \\ &= \int h(\mathbf{x}) \prod_{i=1}^{p} f_i(x_i)\, dx_1 \dots dx_p \\ &= \int_{[0,1]^p} h(F_1^{-1}(u_1), \dots, F_p^{-1}(u_p))\, du_1 \dots du_p. \end{aligned}$$

The left panel of Figure 5.4 shows 50 points generated from an independent bivariate normal distribution using the inverse probability transform of a uniform design. The right panel of the same figure shows 50 points when the two variables follow an independent Gamma distribution with shape and rate parameters 2 and 1, respectively. Both sets of points seem to be representative of the corresponding distributions. Li et al. (2020) explain the conditions under which an inverse probability transformation will lead to a good representative set of the distribution.

When variables are dependent, the following Rosenblatt transformation (Rosenblatt, 1952) can sometimes be used:

$$\begin{aligned} u_1 &= F_1(x_1) \\ u_2 &= F_2(x_2|x_1) \\ &\vdots \\ u_p &= F_p(x_p|x_1, \dots, x_{p-1}), \end{aligned}$$

where $F_i(x_i|x_1,\ldots,x_{i-1})$ is the conditional distribution function of x_i given $x_1,\ldots,x_{i-1}$ and $u_i \overset{iid}{\sim} \mathcal{U}(0,1)$. To show the validity of this transformation, first note that the Jacobian of transformation is given by

$$\begin{aligned}|J| &= \left|\frac{\partial(u_1,\ldots,u_p)}{\partial(x_1,\ldots,x_p)}\right| \\ &= f_1(x_1)f_2(x_2|x_1)\cdots f_p(x_p|x_1,\ldots,x_{p-1}) \\ &= f(x_1,x_2,\ldots,x_p),\end{aligned}$$

which follows from the fact that the Jacobian matrix is an upper triangular matrix and the determinant is a product of its diagonals. Since $f(x_1,\ldots,x_p) = f_u(u_1,\ldots,u_p)|J|$, the probability density of $u_1,\ldots,u_p$ has to be 1. As an illustrative example, consider the problem of generating a design from the following bivariate normal distribution

$$\begin{pmatrix}x_1\\x_2\end{pmatrix} \sim \mathcal{N}_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right). \tag{5.8}$$

Using Lemma 2.2, we obtain

$$\begin{aligned}x_1 &\sim \mathcal{N}(0,1)\\ x_2|x_1 &\sim \mathcal{N}(\rho x_1, 1-\rho^2).\end{aligned}$$

Thus, we can generate the points using

$$\begin{aligned}x_{1i} &= \Phi^{-1}(u_{1i})\\ x_{2i} &= \rho x_{1i} + \sqrt{1-\rho^2}\Phi^{-1}(u_{2i}),\end{aligned}$$

for $i = 1,\ldots,n$, where $\{(u_{1i},u_{2i})\}_{i=1}^n$ is a uniform design and $\Phi(\cdot)$ is the standard normal distribution function. Figure 5.5 shows 50 points thus generated with $\rho = 0.5$. We can see that the points have a good representation of the distribution shown as an image.

Fang and Wang (1993) shows how Rosenblatt transformation can be used for generating QMC or uniform design in a ball or a simplex. However, this transformation would not work in regions with more complex constraints, such as those discussed in Section 4.5.1. Moreover, the foregoing transformation methods have limited applicability in other statistical problems. For example, in Bayesian statistics, one can only find the unnormalized posterior density. That is, we usually do not know the probability density or its distribution function. The variables are also likely to be dependent with unknown conditional distributions. Therefore, in general, we cannot use the inverse probability transformation or the Rosenblatt transformation for sampling. An even more difficult case is the usual data analysis problems, where we have only the data with no knowledge about the distribution from which the data are generated. Thus, we need a more general approach to sampling from non-uniform distributions, which is discussed in the next section.

5.2.2 Support Points

To find the representative points of general type of distributions, we can proceed as in Section 5.1 by defining a goodness-of-fit statistic and reverse engineering the technique to find optimal samples. However, as mentioned earlier, commonly used goodness-of-fit measures such as Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling statistics are not easy to compute in high dimensions, which makes their optimization difficult. To this end, Mak and Joseph (2018c) proposed using a goodness-of-fit measure based on an energy distance metric, which is easy to compute and optimize.

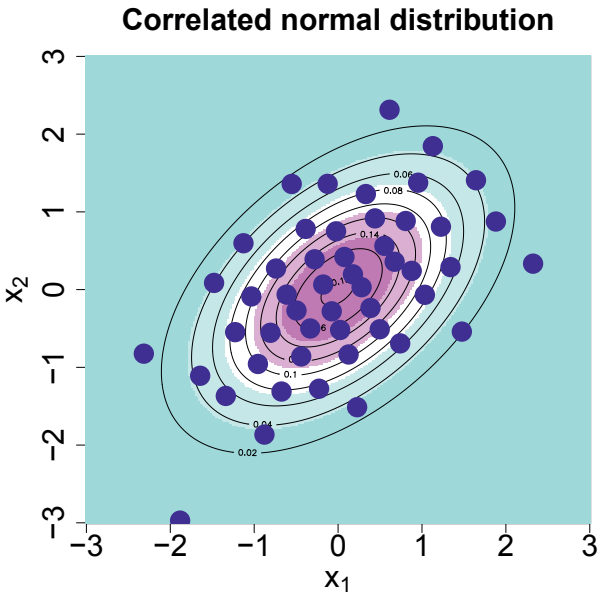


FIGURE 5.5: Representative points of a correlated bivariate normal distribution obtained using Rosenblatt transformation of a uniform design overlayed on the image of the distribution.

Székely et al. (2004); Székely and Rizzo (2013) proposed the following energy distance to measure the "distance" between two distributions F and G:

Definition 5.1 *Let F and G be two distribution functions on $\varnothing \neq \mathcal{X} \subseteq \mathbb{R}^p$ with finite means, and let $\mathbf{X}, \mathbf{X}' \overset{iid}{\sim} G$ and $\mathbf{Y}, \mathbf{Y}' \overset{iid}{\sim} F$. The energy distance between F and G is defined as:*

$$\mathcal{E}(F, G) \equiv 2\mathbb{E}\|\mathbf{X} - \mathbf{Y}\| - \mathbb{E}\|\mathbf{X} - \mathbf{X}'\| - \mathbb{E}\|\mathbf{Y} - \mathbf{Y}'\|, \tag{5.9}$$

where $\|\cdot\|$ is the Euclidean distance. An important property of the energy distance is that $\mathcal{E}(F, G)$ is non-negative, and attains a minimum value of 0 if and only if F and G are the same distribution. See the book Székely and Rizzo (2023) for more details on energy distance.

When $G = F_n$, the empirical distribution function for $\{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$, this energy distance becomes:

$$\mathcal{E}(F, F_n) = \frac{2}{n}\sum_{i=1}^{n}\mathbb{E}\|\mathbf{x}_i - \mathbf{Y}\|_2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i - \mathbf{x}_j\|_2 - \mathbb{E}\|\mathbf{Y} - \mathbf{Y}'\|_2. \tag{5.10}$$

This can be used as a goodness-of-fit statistic to test if a sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ belongs to the distribution F or not. Instead of using it as a goodness-of-fit statistic, Mak and Joseph (2018c) proposed to find the best sample possible to fit into a given distribution F. They called the sample support points of the distribution, which can be formally defined as follows.

Definition 5.2 *For a given distribution F on $\mathcal{X} \subseteq \mathbb{R}^p$ with finite mean, the* support points *for F are defined as:*

$$\underset{D}{\operatorname{argmin}}\ \mathcal{E}(F, F_n) = \underset{D}{\operatorname{argmin}} \left\{ \frac{2}{n} \sum_{i=1}^{n} \mathbb{E}\|\mathbf{x}_i - \mathbf{X}\| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\| \right\}, \tag{5.11}$$

where $\mathbf{X} \sim F$, and F_n is the empirical distribution function of the point set $D = \{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$.

Furthermore, Mak and Joseph (2018c) showed the following important result.

Theorem 5.1 *Let $\mathbf{X} \sim F$ and $\mathbf{X}_n \sim F_n$, where F_n is the empirical distribution function of the support points in* (5.11)*. Then $\mathbf{X}_n \xrightarrow{d} \mathbf{X}$, where the notation $\xrightarrow{d}$ stands for convergence in distribution.*

This theorem justifies the use of support points as a representative sample of F. Although the convergence in distribution occurs only as $n \to \infty$, support points give the best representative samples for any given n according to the energy distance measure.

From a design perspective, the formulation in 5.11 has the following interpretation. By *minimizing* the first term $\sum_{i=1}^{n} \mathbb{E}\|\mathbf{x}_i - \mathbf{Y}\|$, one minimizes the expected Euclidean distance of each point to F, thereby ensuring that all design points are close to the desired distribution. Likewise, by *maximizing* the second term $\sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|$, one maximizes the sum of the pairwise distance between the design points, thereby ensuring that the points are pushed apart in the experimental region $\mathcal{X}$. For uniform distribution $F = \mathcal{U}[0,1]^p$, these two objectives are quite similar to the goals of the minimax and maximin designs that we saw in Chapter 4.

Mak and Joseph (2018c) also proved the following Koksma-Hlawka type inequality for the absolute integration error

$$|\hat{I} - I| \leq V(h)\sqrt{\mathcal{E}(F, F_n)},$$

which shows that the upper bound for the error can be minimized by using support points. Furthermore, under certain regularity conditions, they showed that

$$|\hat{I} - I| = \mathcal{O}\left(\frac{1}{\sqrt{n}(\log n)^{\epsilon}}\right)$$

for some $\epsilon > 0$. This shows that support points have a better convergence rate than a Monte Carlo sample. Although the theoretical improvement shown for the convergence rate is small, Mak and Joseph (2018c) found the convergence rate to be almost $\mathcal{O}(1/n)$ in practice.

To connect with the Gaussian process framework (Mak and Joseph, 2025), assume that $h(\mathbf{x})$ follows a Brownian motion with

$$E\{h(\mathbf{u}) - h(\mathbf{v})\} = 0 \text{ and } var\{h(\mathbf{u}) - h(\mathbf{v})\} = \tau^2||\mathbf{u} - \mathbf{v}||. \tag{5.12}$$

Brownian motion is a special case of a nonstationary Gaussian process, which is also known by the name of intrinsic kriging (Matheron, 1963). It is popularized in the field of computer experiments by Zhang and Apley (2014). Now

$$\begin{aligned} E\{\hat{I} - I\} &= E\left\{ \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}_i) - \int h(\mathbf{x}) dF(\mathbf{x}) \right\} \\ &= \frac{1}{n} \sum_{i=1}^{n} \int E\left\{h(\mathbf{x}_i) - h(\mathbf{x})\right\}\ dF(\mathbf{x}) = 0. \end{aligned}$$

Therefore, following the same steps as in Section 5.1.2, the variance becomes

$$\begin{aligned} var\{\hat{I} - I\} &= var\left\{\frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i) - \int h(\mathbf{x})dF(\mathbf{x})\right\} \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int\int E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_j)]\right\} dF(\mathbf{u})dF(\mathbf{v}) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int\int E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{v}) - h(\mathbf{x}_i)]\right\} dF(\mathbf{u})dF(\mathbf{v}) \\ &\quad - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int\int E\left\{[h(\mathbf{u}) - h(\mathbf{x}_i)][h(\mathbf{x}_j) - h(\mathbf{x}_i)]\right\} dF(\mathbf{u})dF(\mathbf{v}). \end{aligned}$$

Note that for any $\mathbf{x}_i$,

$$cov\{h(\mathbf{u}) - h(\mathbf{x}_i), h(\mathbf{v}) - h(\mathbf{x}_i)\} = \frac{\tau^2}{2}\left\{\|\mathbf{u} - \mathbf{x}_i\| + \|\mathbf{v} - \mathbf{x}_i\| - \|\mathbf{u} - \mathbf{v}\|\right\}.$$

Therefore,

$$\begin{aligned} var\{\hat{I} - I\} &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int\int\left[\frac{\tau^2}{2}\{\|\mathbf{u} - \mathbf{x}_i\| + \|\mathbf{v} - \mathbf{x}_i\| - \|\mathbf{u} - \mathbf{v}\|\}\right] dF(\mathbf{u})dF(\mathbf{v}) \\ &\quad - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int\int\left[\frac{\tau^2}{2}\{\|\mathbf{u} - \mathbf{x}_i\| + \|\mathbf{x}_j - \mathbf{x}_i\| - \|\mathbf{u} - \mathbf{x}_j\|\}\right] dF(\mathbf{u})dF(\mathbf{v}) \\ &= \frac{\tau^2}{n}\sum_{i=1}^{n}\int\|\mathbf{u} - \mathbf{x}_i\|dF(\mathbf{u}) - \frac{\tau^2}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i - \mathbf{x}_j\| - \frac{\tau^2}{2}\iint\|\mathbf{u} - \mathbf{v}\|dF(\mathbf{u})dF(\mathbf{v}) \\ &= \frac{\tau^2}{2}\mathcal{E}(F, F_n). \end{aligned}$$

Thus, support points minimize the variance of the integration error if the integrand follows a Brownian motion (Mak and Joseph, 2025).

A major benefit of using energy distance is that the optimization of support points can be performed efficiently. We start by approximating the expectation in (5.11) by a Monte Carlo average. Assume that we have a large sample from F: $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$. Then the problem is to minimize

$$\frac{2}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N}\|\mathbf{x}_i - \mathbf{X}_j\| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i - \mathbf{x}_j\| \tag{5.13}$$

The key observation here is that this objective function is the difference of two convex functions in $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. This structure is quite appealing, because the formulation can then be optimized using difference-of-convex (d.c.) programming methods. Since global optimization algorithms for d.c. programs are slow, Mak and Joseph (2018c) used a local optimization algorithm called the convex-concave procedure (Yuille and Rangarajan, 2003), which is a specific case of majorization-minimization technique that is widely used in statistics (Lange, 2016). If $(\mathbf{x}_1^0, \ldots, \mathbf{x}_n^0)$ is the current iterate, then the next iterate can be obtained using

$$\mathbf{x}_i = \left(\sum_{j=1}^{N}\|\mathbf{x}_i^0 - \mathbf{X}_j\|^{-1}\right)^{-1}\left(\frac{N}{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\mathbf{x}_i^0 - \mathbf{x}_j^0}{\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|} + \sum_{j=1}^{N}\frac{\mathbf{X}_j}{\|\mathbf{x}_i^0 - \mathbf{X}_j\|}\right) \tag{5.14}$$

for $i = 1, \cdots, n$. This can be continued until convergence. The computational complexity of this algorithm is $\mathcal{O}(Nnp)$, which can be reduced to $\mathcal{O}(n^2 p)$ using a stochastic version of the majorization-minimization algorithm. The algorithm is implemented in the R package `support` (Mak, 2018). A faster $\mathcal{O}(pN \log N)$ algorithm is discussed in Section 10.3.

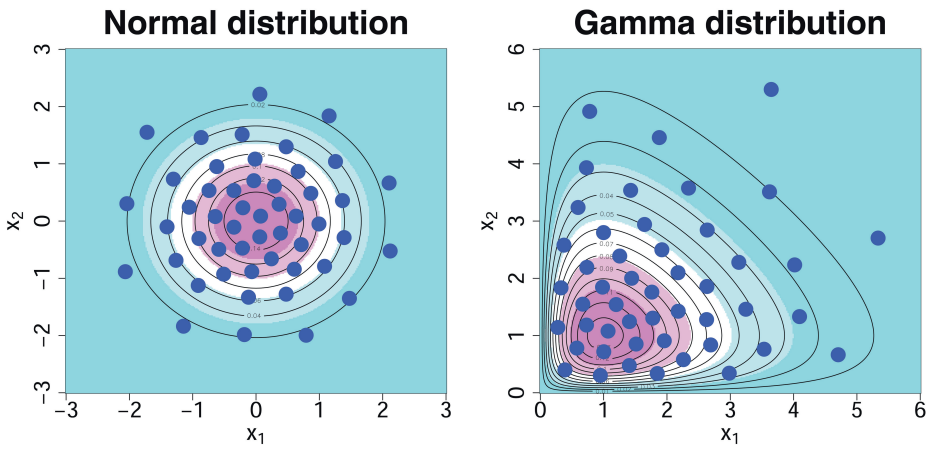


FIGURE 5.6: 50 support points of a bivariate normal distribution (left) and Gamma distribution (right). The image and contours of the distributions are also shown.

The 50 support points generated from a bivariate normal and Gamma distribution are shown in Figure 5.6. Clearly, they look much nicer than those in Figure 5.4 generated using inverse probability transform of a uniform design. The transformation distorts the space-fillingness in the transformed space, whereas support points are optimized in the original space, giving a much better representation to the target distribution.

Consider the correlated bivariate normal distribution discussed in the previous section. We generate a Monte Carlo sample of 10,000 points and then used `support` package to obtain 100 support points. They are shown in the left panel of Figure 5.7. Clearly, this point set looks much better than the Rosenblatt-transformed point set shown in Figure 5.5. Now consider the more complex banana-shaped posterior density example in (4.49), where we know the density only up to a proportionality constant. Here we used the adaptive Metropolis algorithm implemented in the R package `adaptMCMC` (Scheidegger, 2024) to obtain 100,000 samples, which are then used to find 100 support points. The support points are plotted on the right panel of Figure 5.7, which again gives a good representation of the underlying distribution.

If only a few variables in $h(\mathbf{x})$ are active, then as discussed in Sections 4.4 and 4.5, support points will also have projection issues, which can lead to poor space-filling properties in the active dimensions. Mak and Joseph (2018b) extended the support points' framework by using ideas similar to those of MaxPro designs. The main idea is to use a product form of the rational quadratic kernel given by

$$K(\mathbf{u}, \mathbf{v}) = \prod_{l=1}^{p} \frac{1}{(u_l - v_l)^2 + \lambda},$$

where $\lambda > 0$. The derivation similar to the kernel discrepancy in (5.5) leads to the following definition.

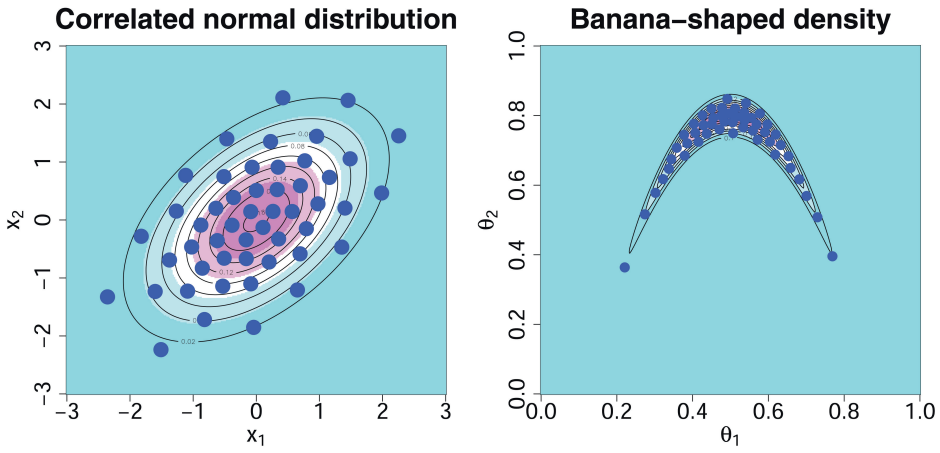


FIGURE 5.7: 100 support points of the correlated bivariate normal distribution in (5.8) (left) and banana-shaped density in (4.49) (right). The image and contours of the distributions are also shown.

Definition 5.3 *For a given distribution F on $\mathcal{X} \subseteq \mathbb{R}^p$ with finite mean, the* projected support points $D = \{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$ *for F is defined as:*

$$\underset{D}{\operatorname{argmin}} \left\{ \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \prod_{l=1}^{p} \frac{1}{(x_{il} - x_{jl})^2 + \lambda} - \frac{2}{n} \sum_{i=1}^{n} \mathbb{E} \left[\prod_{l=1}^{p} \frac{1}{(x_{il} - X_l)^2 + \lambda} \right] \right\}, \tag{5.15}$$

where $\mathbf{X} \sim F$ and $\lambda > 0$.

It is interesting to look at the case of $F = \mathcal{U}[0,1]^p$. Then the objective function in (5.15) becomes (Mak and Joseph, 2025)

$$\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{\prod_{l=1}^{p} [(x_{il} - x_{jl})^2 + \lambda]} - \frac{2}{n\lambda^{p/2}} \sum_{i=1}^{n} \prod_{l=1}^{p} \left[\tan^{-1}\left(\frac{x_{il}}{\sqrt{\lambda}} \right) + \tan^{-1}\left(\frac{1 - x_{il}}{\sqrt{\lambda}} \right) \right]. \tag{5.16}$$

For small λ, the first term is the same as the objective function for MaxPro, whereas the second term pulls the points near the boundaries to interior of the experimental region to make the MaxPro design uniform.

Kernel herding (Chen et al., 2012) is a closely related approach to support points. It is a sequential one-point-at-a-time algorithm given by

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}}{\operatorname{argmax}} \left\{ E[K(\mathbf{x}, \mathbf{X})] - \frac{1}{n+1} \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i) \right\}, \; k = 1, \ldots, n-1,$$

where $K(\mathbf{u}, \mathbf{v})$ is a symmetric positive definite kernel such as the Gaussian kernel. Support points have several advantages over kernel herding. First, support points are obtained using difference-of-convex programming techniques, which gives a much better representative set than the one-point-at-a-time greedy algorithm. Second, support points use distance-based kernels, making them similar to maximin and minimax designs, which have nice geometrical interpretations. Moreover, the Euclidean kernel used in support points does not have any length-scale parameters as in the Gaussian kernel, which makes the optimal point set robust

to the choice of such parameters. Another concept, maximum mean discrepancy (MMD), defined based on the reproducing kernel Hilbert space, has become a popular tool in the machine learning literature (Gretton et al., 2012). MMD is closely related to the kernel discrepancy in (5.5); see Oates (2021) for a recent review. MMD also uses symmetric positive definite kernels that contain unknown length-scale parameters and is therefore susceptible to their misspecifications. This is an advantage of support points, which can be considered as a more robust representative set of the distribution.

A different approach to generate representative points is to use the cluster-based designs discussed in Section 4.1. They are known as MSE-rep points (Fang and Wang, 1993) or principal points (Flury, 1990). Unlike the support points, their empirical distribution need not converge to the target distribution (see (5.19)), and therefore the points need to be associated with some weights to correct for the distortion in the distribution. Fang et al. (2025) is a recent book that explains this approach.

5.2.3 Uncertainty Propagation

Consider the machining problem mentioned in Chapter 1 and illustrated in Figure 1.3. In this problem, there are several input variables with known distribution. The aim is to propagate this input uncertainty through the expensive machining simulator and obtain the distribution of the output (force on the tool).

There are several ways to tackle this uncertainty propagation problem. First, we could generate a large Monte Carlo sample $\mathbf{x}_i \overset{iid}{\sim} F$, $i = 1, \ldots, n$, and run the simulator n times, giving the output $\{y_1, \ldots, y_n\}$, where $y_i = h(\mathbf{x}_i)$. For this set of output, we can obtain the empirical distribution of $h(\mathbf{x})$ or its kernel density estimator. This can be expensive and, therefore, this approach can be used only when the simulator is cheap to run. We can reduce the cost by using a transformed low-discrepancy sequence instead of the Monte Carlo sample. We can further reduce this cost by using support points, because it can obtain the same accuracy for the output distribution with fewer samples compared to a low-discrepancy sequence. However, if the simulator is too expensive to run, then we should first approximate it using a surrogate model such as a Gaussian process model. We can use the projected support points or the minimum energy design discussed in Section 4.6 as the experimental design because the input distributions are likely to be nonuniform. Once the surrogate modeling is done, we can use a large set of Monte Carlo samples to find the output distribution.

Consider, for example, the function that models the flow rate (y) of water through a borehole (Morris et al., 1993):

$$y = \frac{2\pi x_3(x_4 - x_6)}{\log\left(\frac{x_2}{x_1}\right)\left[1 + \frac{2x_7x_3}{\log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right]},$$

where the uncertainty distributions of the eight input variables are specified in Table 5.1. For illustrative purposes, assume that the borehole model is expensive to evaluate, but not too expensive that it would require surrogate modeling.

Because the energy distance uses the Euclidean distance, we should not try to generate support points directly on the space defined by the variables in Table 5.1. We should scale the variables so that the sum of squared differences in the Euclidean distance makes sense. Note that if $X, X' \overset{iid}{\sim} F$,

$$E\{(X - X')^2\} = 2var\{X\}.$$

Therefore, we should scale the variables so that their variances are equal. This can be done using the usual standardization technique. Let μ_i and σ_i be the mean and standard

TABLE 5.1: Input variables and corresponding uncertainty distributions for the borehole model. Here $\mathcal{LN}$ denotes the log-normal distribution.

Input variable	Notation	Uncertainty distribution
Borehole radius	x_1	$\mathcal{N}(0.1, 0.01618^2)$
Radius of influence	x_2	$\mathcal{LN}(7.71, 1.0056^2)$
Upper transmissivity	x_3	$\mathcal{U}[63070, 115600]$
Upper head	x_4	$\mathcal{U}[990, 1110]$
Lower transmissivity	x_5	$\mathcal{U}[63.1, 116]$
Lower head	x_6	$\mathcal{U}[700, 820]$
Borehole length	x_7	$\mathcal{U}[1120, 1680]$
Borehole conductivity	x_8	$\mathcal{U}[9855, 12045]$

deviation of x_i. Then, let

$$z_i = \frac{x_i - \mu_i}{\sigma_i},\ i = 1, \ldots, p.$$

The support points can be generated in this standardized space. Once we obtain the support points, we can get the support points in the original space using $x_i = \mu_i + \sigma_i z_i$.

For the borehole model, we first generate 100,000 MC samples. Since the borehole model is an easy-to-evaluate function, we can evaluate it on these 100,000 samples and obtain the output distribution using the kernel density estimate. This is shown as a black dotted line in Figure 5.8. Now consider uncertainty propagation using support points. First, we standardize the 100,000 MC samples and obtain $n = 100$ support points $\{\mathbf{z}_j\}_{j=1}^{100}$ using the R package `support` (Mak, 2018). They are transformed back to their original scale using $x_{ji} = \mu_i + \sigma_i z_{ji}$. The borehole function is evaluated at these 100 points and the kernel density estimates of the y_i's are plotted in the same figure as a solid green line. We can see that it gives a good approximation to the "true" density obtained using the 100,000 MC samples. For comparison, the density of 100 MC samples is also shown in the same figure as a red-dashed line. We can see that this density estimate from the support points is closer to the true density.

5.2.4 Population Quasi-Monte Carlo

Consider the Bayesian computational problem, where we are interested in obtaining a sample $D = \{\mathbf{x}\}_{i=1}^n$ from the posterior distribution

$$f(\mathbf{x}) = \frac{h(\mathbf{x})}{Z},$$

where $h(\mathbf{x})$ is the unnormalized posterior density (a product of likelihood and prior) and $Z = \int h(\mathbf{x}) d\mathbf{x}$ is the normalizing constant. Since the computation of Z involves a high-dimensional integral, sampling is commonly done using Markov chain Monte Carlo (MCMC) methods that do not require knowledge of Z. Since the MCMC samples are correlated, a large number of samples are needed to obtain convergence to the target distribution. Suppose that we have such a large sample $\{\mathbf{X}_i\}_{i=1}^N$. To reduce storage requirements and subsequent computations, it is common to reduce the number of samples using a procedure called *thinning*. This is usually done by selecting every kth sample (say $k = 10$) in the MCMC chain. Clearly a much better thinned sample can be obtained using support points (Mak and Joseph, 2018b). However, this approach requires N samples from the unnormalized posterior, which can sometimes be costly when $h(\mathbf{x})$ is expensive to evaluate.

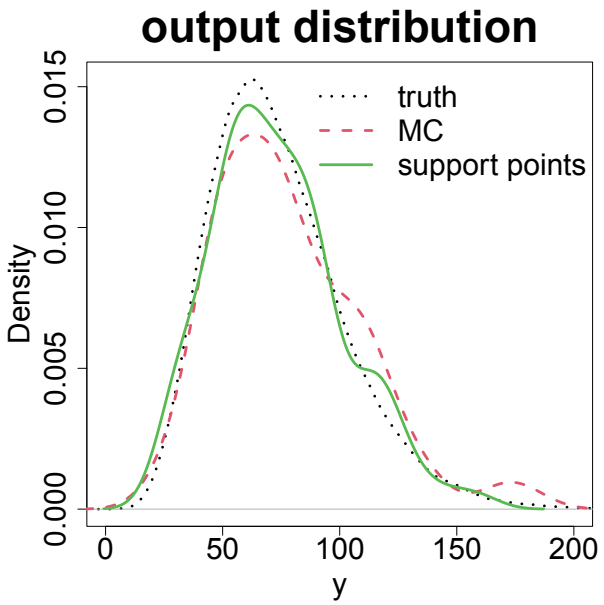


FIGURE 5.8: The output distribution of the borehole function using 100 MC samples (red-dashed) and 100 support points (green-solid). The true output distribution is shown in black-dotted line.

Suppose that the evaluation of $h(\mathbf{x})$ is expensive and, therefore, we do not have a large MCMC sample. In Section 4.6.1, we used minimum energy designs (MEDs) to solve this problem. However, unlike support points, MEDs do not guarantee the best possible sample of size n. Thus, a much better approximation to the target density can be obtained by using support points instead of MEDs. To find the support points, we have to solve the following optimization problem:

$$\min_{D} \left\{ \frac{2}{n} \sum_{i=1}^{n} \int \|\mathbf{x}_i - \mathbf{X}\| \frac{h(\mathbf{X})}{Z} d\mathbf{X} - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\| \right\}.$$

Unfortunately, the normalizing constant Z does not factor out of the objective function as in the case of MED and therefore this optimization cannot be easily done.

Suppose that we are able to obtain a large sample from a proposal density $q(\mathbf{X})$. Then, the foregoing optimization problem can be written as

$$\min_{D} \left\{ \frac{2}{n} \sum_{i=1}^{n} \int \|\mathbf{x}_i - \mathbf{X}\| \frac{h(\mathbf{X})}{Z q(\mathbf{X})} q(\mathbf{X}) d\mathbf{X} - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\| \right\},$$

where

$$Z = \int h(\mathbf{X}) d\mathbf{X} = \int \frac{h(\mathbf{X})}{q(\mathbf{X})} q(\mathbf{X}) d\mathbf{X}.$$

Let $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim q(\mathbf{X})$. Then, we can solve the following problem

$$\min_{D} \left\{ \frac{2}{n} \sum_{i=1}^{n} \sum_{j=1}^{N} \|\mathbf{x}_i - \mathbf{X}_j\| w_j - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\| \right\}, \tag{5.17}$$

Algorithm 1 Population Monte Carlo.

Target Distribution: $f = h/Z$ where Z is an unknown normalizing constant.
Initialization: set parameters for the initial proposals $\{q_k^{(1)} = \mathcal{N}(\cdot|\mu_k^{(1)}, \Sigma)\}_{k=1}^K$.
For $t = 1, \ldots, T$ do
• **Sampling:** draw J samples from each proposal $k = 1, \cdots, K$

$$x_{k,j}^{(t)} \sim q_k^{(t)}, \quad j = 1, \cdots, J, \quad k = 1, \cdots, K,$$

so that a total of KJ samples are simulated.
• **Weighting:** compute deterministic mixture weights

$$w_{k,j}^{(t)} = \frac{h(x_{k,j}^{(t)})}{K^{-1} \sum_{i=1}^K q_i^{(t)}(x_{k,j}^{(t)}|\mu_i^{(t)}, \Sigma)}, \quad j = 1, \cdots, J, \quad k = 1, \cdots, K,$$

and normalize them by

$$\bar{w}_{k,j}^{(t)} = \frac{w_{k,j}^{(t)}}{\sum_{i=1}^K \sum_{l=1}^J w_{i,l}^{(t)}}.$$

• **Adaptation:** perform resampling to update the proposal centers

$$\mu_k^{(t+1)} \overset{\text{i.i.d.}}{\sim} \sum_{k=1}^K \sum_{j=1}^J \bar{w}_{k,j}^{(t)} \delta_{x_{k,j}^{(t)}}, \quad k = 1, \cdots, K.$$

Return: $\{(x_{k,j}^{(t)}, w_{k,j}^{(t)})\}_{t=1}^T{}_{k=1}^K{}_{j=1}^J$ where $w_{k,j}^{(t)}$ are the unnormalized weights.

where w_j's are the importance weights given by

$$w_j = \frac{h(\mathbf{X}_j)/q(\mathbf{X}_j)}{\sum_{j=1}^N h(\mathbf{X}_j)/q(\mathbf{X}_j)}.$$

Huang et al. (2022) calls the optimal sample from (5.17) as *importance support points* (ISP).

Although (5.17) can be efficiently solved, finding a good importance sampling proposal $q(\mathbf{X})$ in high dimensions is not an easy task. Therefore, Huang et al. (2022) proposed to integrate this idea within the Population Monte Carlo (PMC) framework (Cappé et al., 2004). PMC iterates over three steps. It starts with K samples. We can place a normal distribution centered at each of the samples, which becomes K proposals: $q_1(\mathbf{x}), \ldots, q_K(\mathbf{x})$. In the first step `sampling`, we draw J samples from each of the proposals, giving a total of KJ samples. In the second step `weighting`, importance weights are computed for each of the samples. In the last step `adaptation`, K new samples are obtained from the KJ samples by resampling with the importance weights. This procedure continued for T steps, which is summarized in Algorithm 1. From these weighted samples we can estimate an integral $I = \int h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$ using

$$\hat{I}^{\text{PMC}} = \frac{1}{TKJ} \sum_{t=1}^T \sum_{k=1}^K \sum_{j=1}^J w_{k,j}^{(t)} \, h(x_{k,j}^{(t)})/\hat{Z},$$

where $\hat{Z} = \sum_{t=1}^T \sum_{k=1}^K \sum_{j=1}^J w_{k,j}^{(t)}/TKJ$.

Huang et al. (2022)'s main idea is to use importance support points resampling in (5.17) for the adaptation step. The K support points will provide a much better representation of the KJ samples and therefore much better centers for the new proposals. They named the modified procedure Population Quasi-Monte Carlo (PQMC). Huang et al. (2022) have

also proposed several improvements to the sampling and weighting steps, but we will not describe the details here.

As an example, consider the following mixture normal distribution:

$$f(x) = \frac{1}{5}\sum_{i=1}^{5} \mathcal{N}(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i)$$

where $\mu_1 = (0.250, 0.250)'$, $\mu_2 = (0.500, 0.900)'$, $\mu_3 = (0.825, 0.700)'$, $\mu_4 = (0.275, 0.675)'$, $\mu_5 = (0.850, 0.150)'$, $\mathbf{\Sigma}_1 = 40^{-2}[2, 0.6; 0.6, 1]$, $\mathbf{\Sigma}_2 = 40^{-2}[2, -0.4; -0.4, 2]$, $\mathbf{\Sigma}_3 = 40^{-2}[2, 0.8; 0.8, 2]$, $\mathbf{\Sigma}_4 = 40^{-2}[3, 0; 0, 0.5]$, and $\mathbf{\Sigma}_5 = 40^{-2}[2, -0.1; -0.1, 2]$. Let $K = 20, J = 10$, and $T = 4$. This setting makes a total of 800 density evaluations. Figure 5.9 shows the samples and the final set of resampled centers for PMC (with multinomial resampling) and PQMC. We can see that the samples and proposal centers of PQMC are nicely placed in the high-density region compared to those of the PMC. Thus, PQMC gives much higher quality weighted samples than PMC.

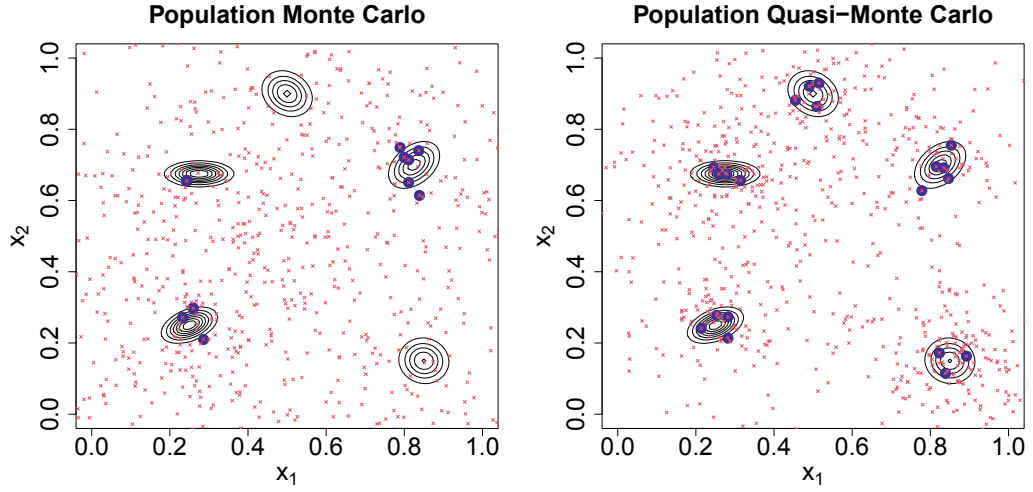


FIGURE 5.9: The 800 samples (red crosses) and the final set of proposal centers (blue solid circles) from PMC (left) and PQMC (right) for the mixture normal example.

Exercises

1. Show that the point set $\{(i-.5)/n\}_{i=1}^{n}$ minimizes the star discrepancy of a $\mathcal{U}(0,1)$ (Fang and Wang, 1993, p.16).

2. Consider the L_q-quantization problem:

$$\min_{D} \sum_{i=1}^{n} \int_{V_i} \|\mathbf{x} - \mathbf{x}_i\|^q \, f(\mathbf{x}) d\mathbf{x}, \tag{5.18}$$

where $\mathbf{x} \in \mathbb{R}^p$ and $V_1, \ldots, V_n$ are the Voronoi regions of $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The optimal quantizer has an asymptotic distribution given by (Zador, 1982)

$$f_D(\mathbf{x}) \propto \{f(\mathbf{x})\}^{\frac{p}{p+q}}. \tag{5.19}$$

To test this result, generate 10,000 Monte Carlo samples from $\mathcal{N}(0,1)$ and find $n = 100$ cluster centers using `kmeans`. Compare the empirical density of the points to the theoretical density in (5.19).

3. Assume that $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$, where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$. In Section 5.1.2, the aim was to obtain a point estimate of the integral

$$I = \int_{\mathcal{X}} h(\mathbf{x}) dF(\mathbf{x})$$

given some function evaluations $\{y_i = h(\mathbf{x}_i)\}_{i=1}^n$ made at some design points $D = \{\mathbf{x}_i\}_{i=1}^n$, where $F(\mathbf{x})$ is the distribution function of $\mathbf{x}$. Suppose that we are also interested in obtaining the posterior distribution of the integral. Show that

$$I|\mathbf{y} \sim \mathcal{N}\left(\mu + \mathbf{z}'\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}), \tau^2\{z_0 - \mathbf{z}'\mathbf{R}^{-1}\mathbf{z}\}\right), \tag{5.20}$$

where $z_0 = \int_{\mathcal{X}} \int_{\mathcal{X}} R(\mathbf{u} - \mathbf{v}) dF(\mathbf{u}) dF(\mathbf{v})$, $\mathbf{z} = \int_{\mathcal{X}} \mathbf{r}(\mathbf{x}) dF(\mathbf{x})$, and $\mathbf{r}(\mathbf{x})$ and $\mathbf{R}$ are as defined earlier in (2.13). The Bayesian quadrature problem (Diaconis, 1988; O'Hagan, 1991) is to find the design D to minimize the posterior variance of I. See the book by Hennig et al. (2022) for details.

4. Generate the following designs in 25 runs and two dimensions:

   (a) MmLHD
   (b) MaxPro LHD
   (c) Uniform design.

   Compare them using the centered L_2 and wrap-around discrepancy measures. The discrepancy measures of a design can be computed using the R package `DiceDesign` (Franco et al., 2023).

5. Generate 25 representative points of a $\mathcal{N}_2(\mathbf{0}, \mathbf{I})$ using the following methods:

   (a) Inverse probability transform of Monte Carlo samples
   (b) Inverse probability transform of Sobol' points
   (c) Support points.

   Compute the energy distance of the three sets of points and compare them using box plots from 30 simulations.

6. Suppose that we want to generate a uniform design in the region:

$$\mathcal{X} = \{\mathbf{x} : 0 \le x_1 \le x_2 \le \cdots \le x_p \le 1\}.$$

Fang and Wang (1993) show that this can be done by sampling $\mathbf{u} \in \mathcal{U}(0,1)^p$ and then obtaining $\mathbf{x}$ as follows:

$$\begin{aligned} x_p &= u_p^{1/p} \\ x_{p-1} &= u_p^{1/p} u_{p-1}^{1/(p-1)} \\ &\vdots \\ x_1 &= u_p^{1/p} u_{p-1}^{1/(p-1)} \cdots u_2^{1/2} u_1. \end{aligned}$$

Generate a 50-run uniform design in $[0,1]^2$ using `SFDesign` and then obtain a uniform design in $\mathcal{X}$ using this transformation method.

7. Suppose that we want to generate a uniform design in the region:

$$\mathcal{X} = \{\mathbf{x} : x_1 + x_2 + \cdots + x_p \leq 1, x_i \geq 0 \forall i\}.$$

Fang and Wang (1993) shows that this can be done by sampling $\mathbf{u} \in \mathcal{U}(0,1)^p$ and then obtaining $\mathbf{x}$ as follows:

$$\begin{aligned} x_1 &= u_1^{1/p}\{1 - u_2^{1/(p-1)}\} \\ x_i &= u_1^{1/p} \prod_{j=2}^{i} u_j^{1/(p-j+1)}\{1 - u_{i+1}^{1/(p-i)}\},\ i = 2, \ldots, p-1 \\ x_p &= u_1^{1/p} \prod_{j=2}^{i} u_j^{1/(p-j+1)}. \end{aligned}$$

Generate a 100-run uniform design in $\mathcal{X}$ using this transformation method.

8. Suppose that the experimental region is a hypersphere in a five-dimensional space. Obtain a 100-run uniform design inside the hypersphere. To do this, first generate 100,000 uniform random points inside a sphere using the R package `uniformly` (Laurent, 2024) and then use support points. Show the scatterplot matrix of the 100-run design.

9. Joseph (2016) reported a case study on the robust parameter design optimization of a solid end milling process using simulation. Six variables from the process are selected for optimization; one related to the workpiece material: hardness (x_1) and five related to the cutting tool: rake angle (x_2), helix angle (x_3), relief angle (x_4), corner radius (x_5), and flute length (x_6). Workpiece material hardness is a noise factor to the process, whereas the five tool parameters can be controlled. The objective of the experiment is to find the robust setting of the tool parameters that minimizes the average tangential cutting force (y) irrespective of the material hardness. The following ranges are chosen for simulation: $x_1 \in [95, 125]$, $x_2 \in [0, 0.05]$, $x_3 \in [14, 26]$, $x_4 \in [7, 13]$, $x_5 \in [0.45, 0.50]$, and $x_6 \in [0.7, 1.3]$. The simulation is done using the Production Module software of Third Wave Systems. The data are given in the file `milling.csv`. Assume $x_1 \sim \mathcal{N}(110, 15^2)$. Do the following.

   (a) Fit a Gaussian process model with a Gaussian correlation function to the data to predict the average tangential cutting force.

   (b) Obtain 100 representative points $\{x_{1i}\}_{i=1}^{100}$ for the hardness distribution.

   (c) Find the robust setting of the five factors $x_2 \sim x_6$ to minimize

$$\widehat{E}\{\hat{y}(\mathbf{x})\} = \frac{1}{100}\sum_{i=1}^{100} \hat{y}(x_{1i}, \mathbf{x}_{\sim 1}).$$

   (d) Plot the density of the average tangential cutting force at the robust setting induced by the uncertainty of the material hardness.

10. Consider a Gaussian process model $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$. So far we have been estimating the unknown parameters μ, τ^2, and the correlation parameters $\boldsymbol{\theta}$ using empirical Bayes methods. Suppose we want to do a fully Bayesian inference. Assume:

$$f(\mu, \tau^2) \propto \frac{1}{\tau^2} \text{ and } \theta_i \overset{iid}{\sim} \mathcal{LN}(0, 1).$$

Given the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we can integrate out μ and τ^2 from the unnormalized posterior to obtain:

$$f(\boldsymbol{\theta}|\mathbf{y}) \propto |\mathbf{R}|^{-1/2}(\mathbf{1}'\mathbf{R}^{-1}\mathbf{1})^{-1/2}\left\{(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1})\right\}^{-(n-1)/2} f(\boldsymbol{\theta}),$$

where $\hat{\mu} = \mathbf{1}'\mathbf{R}^{-1}\mathbf{y}/\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}$ and $f(\boldsymbol{\theta}) \propto \prod_{i=1}^p \exp\{-(\log\theta_i)^2/2\}/\theta_i$. This is an expensive density to sample because we need to compute $|\mathbf{R}|$ and $\mathbf{R}^{-1}$ for each evaluation of the unnormalized posterior. Therefore, we can use PQMC for efficient sampling instead of MCMC. Consider the solid end milling simulation discussed in the previous exercise. Use PQMC to sample the six-dimensional space of the correlation parameters with $K = 40$, $J = 5$, and $T = 5$. The posterior density of $h(\mathbf{x})$ is given by (Santner et al., 2018):

$$\frac{h(\mathbf{x}) - \hat{y}(\mathbf{x})}{s(\mathbf{x})}|\boldsymbol{\theta}, \mathbf{y} \sim \mathcal{T}_{n-1}, \tag{5.21}$$

where

$$\begin{aligned}
\hat{y}(\mathbf{x}) &= \hat{\mu} + \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}),\\
s^2(\mathbf{x}) &= \hat{\tau}^2\left(1 - \mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) + \frac{\{1-\mathbf{r}(\mathbf{x})'\mathbf{R}^{-1}\mathbf{1}\}^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}\right),\\
\hat{\tau}^2 &= \frac{1}{n-1}(\mathbf{y}-\hat{\mu}\mathbf{1})'\mathbf{R}^{-1}(\mathbf{y}-\hat{\mu}\mathbf{1}),
\end{aligned}$$

and $\mathcal{T}_{n-1}$ denotes a t-distribution with $n-1$ degrees of freedom. Obtain the posterior density of the prediction at the robust setting found in the previous exercise with $x_1 = 110$. Compare this to the posterior density obtained using empirical Bayes.

6

Screening Designs

Sparsity is commonly observed in most systems, that is, even if the system has hundreds of factors, only a few may be important for the output. In the previous two chapters, we learned about designs that can fill the space of all factors irrespective of their effects. However, if we had known the important factors a priori, we could have used our resources judiciously to fill the space of only the important factors. We learned about MaxPro designs, which ensure good projections into lower-dimensional spaces by taking an average of all possibilities. See, for example, the 50-run MaxPro design in five factors shown in Figure 4.11. It gives reasonably good projections in all two-dimensional spaces, but it does make compromises. If we had known that only x_4 and x_5 will be important, we could have done much better. In this chapter, we will learn experimental design techniques to quickly identify the important factors, which are called screening designs.

6.1 Sensitivity Analysis

Before developing screening designs, we first need to understand how to quantify the importance of a factor. If the function is linear, that is, $h(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$, then it is easy to quantify the importance of each factor by using $|\beta_i|$ assuming that the variability in the x_i's is the same. However, this task becomes non-trivial when the function is nonlinear. We will now discuss two methods for quantifying the importance of factors: one variance-based and the other derivative-based.

6.1.1 Variance-based Sensitivity Measures

Sobol' (1990, 1993) proposed variance-based sensitivity indices to measure the importance of a factor. To obtain the indices, we first decompose the function $y = h(\mathbf{x})$, $\mathbf{x} \in [0,1]^p$, using functional ANOVA:

$$h(\mathbf{x}) = h_0 + \sum_{i=1}^{p} h_i(x_i) + \sum_{i<j} h_{ij}(x_i, x_j) + \ldots + h_{12\ldots p}(x_1, \ldots, x_p), \tag{6.1}$$

where, for uniqueness, we must have

$$\int h_I(\mathbf{x}_I) dx_k = 0 \quad \text{for any } k \in I. \tag{6.2}$$

DOI: 10.1201/9781003661535-6

This chapter has been made available under a CC-BY-NC-ND license 4.0.

Using (6.2), it is easy to see that the overall mean is

$$h_0 = \int h(\mathbf{x}) d\mathbf{x},$$

the ith main effect is

$$h_i(x_i) = \int h(\mathbf{x}) d\mathbf{x}_{\sim i} - h_0,$$

the two-factor interaction between x_i and x_j is

$$h_{ij}(x_i, x_j) = \int h(\mathbf{x}) d\mathbf{x}_{\sim i,\sim j} - h_i(x_i) - h_j(x_j) - h_0,$$

and so on, where the notation $\mathbf{x}_{\sim i}$ indicates all variables except x_i. Since

$$\int h_{I_1}(\mathbf{x}_{I_1}) h_{I_2}(\mathbf{x}_{I_2}) d\mathbf{x} = 0 \text{ if } I_i \neq I_2,$$

all the terms in (6.1) are orthogonal. Therefore,

$$\int h^2(\mathbf{x}) d\mathbf{x} = h_0^2 + \sum_{i=1}^{p} \int h_i^2(x_i) dx_i + \ldots + \int h_{12\ldots p}^2(x_1, \ldots, x_p) d\mathbf{x}.$$

Let

$$V = \int h^2(\mathbf{x}) d\mathbf{x} - h_0^2 \text{ and } V_I = \int h_I^2(\mathbf{x}_I) d\mathbf{x}_I.$$

Then,

$$V = \sum_{i=1}^{p} V_i + \sum_{i<j} V_{ij} + \ldots + V_{12\ldots p},$$

which gives a decomposition of the total variance into sum of variances due to main effects, two-factor interactions, etc., just like the usual ANOVA. Therefore, $S_I = V_I/V$ gives the fraction of variation in $h(\cdot)$ explained by $\mathbf{x}_I$. This is similar to the R^2 measure used in linear regression to assess the overall fit, but here we have a set of indices to quantify the importance of each effect or a group of effects. These are called Sobol' sensitivity indices. In particular,

$$S_i = \frac{V_i}{V},\ i = 1, \ldots, p, \tag{6.3}$$

is called the first-order Sobol' sensitivity index.

A large value of S_i indicates that x_i has a large effect on the output. However, we cannot ignore x_i if S_i is small. This is because although x_i's main effect is small, it may have non-negligible interaction effects with other factors. Thus, to quantify the importance of x_i, we should add all the variance terms involving x_i and get the total variance (Homma and Saltelli, 1996):

$$V_i^{tot} = V_i + \sum_{j \neq i} V_{ij} + \ldots.$$

It is easy to see that this is the same as

$$V_i^{tot} = V - V_{\sim i}.$$

Thus, we can define Sobol' total index as

$$T_i = \frac{V_i^{tot}}{V} = 1 - \frac{V_{\sim i}}{V}. \tag{6.4}$$

We can declare a factor to be unimportant if T_i is small.

Computation of Sobol' indices is not trivial, as it involves high-dimensional integrals. Monte Carlo methods are commonly used for high-dimensional integration. First sample m random vectors from a p-dimensional uniform distribution:

$$\mathbf{a}_i \overset{iid}{\sim} \mathcal{U}[0,1]^p,\ i = 1, \ldots, m,$$

which is the same as drawing mp samples $\{a_{ij}\}$, $i = 1, \ldots, m$, $j = 1, \ldots, p$ from a univariate uniform distribution. We will collect all samples into a matrix $\mathbf{A} = (\mathbf{a}_1', \ldots, \mathbf{a}_m')'$, which can be viewed as a random $m \times p$ design matrix.

Monte Carlo estimate of the overall mean can be obtained as

$$\hat{h}_0 = \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{a}_i).$$

By the strong law of large numbers, this estimate will converge to the true value h_0 at the rate of $\mathcal{O}(1/\sqrt{m})$. The Quasi-Monte Carlo methods that give a convergence rate of almost $\mathcal{O}(1/m)$ discussed in the previous chapter can also be used here. The Monte Carlo estimate of the main effect can be obtained as

$$\hat{h}_i(x_i) = \frac{1}{m} \sum_{j=1}^{m} h(a_{j1}, \ldots, x_i, \ldots, a_{jp}) - \hat{h}_0,$$

and similarly higher-order interaction effects can be computed.

The overall variance $V = \int h^2(\mathbf{x}) d\mathbf{x} - h_0^2 = \int \{h(\mathbf{x}) - h_0\}^2 d\mathbf{x}$ can be estimated as

$$\hat{V} = \frac{1}{m-1} \sum_{i=1}^{m} \{h(\mathbf{a}_i) - \hat{h}_0\}^2. \tag{6.5}$$

Similarly, an unbiased estimate of $V_1 = \int h_1^2(x_1) dx_1$ can be obtained as

$$\begin{aligned} \hat{V}_1 &= \frac{1}{m-1} \sum_{i=1}^{m} \hat{h}_1^2(a_{i1}) \\ &= \frac{1}{m-1} \sum_{i=1}^{m} \left(\frac{1}{m} \sum_{j=1}^{m} h(a_{i1}, a_{j2}, \ldots, a_{jp}) - \hat{h}_0 \right)^2, \end{aligned}$$

and similarly $\hat{V}_2, \ldots, \hat{V}_p$. Each of them requires m^2 evaluations of $h(\cdot)$, which can be expensive for large m. Therefore, more efficient estimation methods are needed to compute the variances of the main effects and higher-order interaction effects. Sobol' (1993) has developed some novel methods to do this, which are also called pick and freeze methods (Da Veiga et al., 2021). Before we develop these methods, we will first rewrite the indices using the expectation and variance operators.

The main effect can be written as

$$\begin{aligned} h_i(x_i) &= \int h(\mathbf{x}) d\mathbf{x}_{\sim i} - h_0 \\ &= E\{h(\mathbf{x}) | x_i\} - E\{h(\mathbf{x})\}. \end{aligned}$$

Then,

$$\begin{aligned} V_i &= \int h_i^2(x_i) dx_i \\ &= var\{h_i(x_i)\} \\ &= var\{E[h(\mathbf{x})|x_i]\}. \end{aligned}$$

The key idea to simplify the computation is to switch the expectation and variance operators using the conditional variance formula:

$$var\{h(\mathbf{x})\} = var\{E[h(\mathbf{x})|x_i]\} + E\{var[h(\mathbf{x})|x_i]\}.$$

Thus,

$$V_i = V - E\{var[h(\mathbf{x})|x_i]\}.$$

Suppose that we have two evaluations $h(\mathbf{a})$ and $h(\mathbf{b})$, where $\mathbf{a}, \mathbf{b} \overset{iid}{\sim} \mathcal{U}[0,1]^p$, then an unbiased estimate of the variance is

$$\begin{aligned}\widehat{var}\{h(\mathbf{x})\} &= \left\{h(\mathbf{a}) - \frac{h(\mathbf{a}) + h(\mathbf{b})}{2}\right\}^2 + \left\{h(\mathbf{b}) - \frac{h(\mathbf{a}) + h(\mathbf{b})}{2}\right\}^2 \\ &= \frac{1}{2}\{h(\mathbf{a}) - h(\mathbf{b})\}^2.\end{aligned}$$

If we need to estimate the conditional variance by fixing the value of x_i, then modify the $\mathbf{b}$ vector by replacing its ith value by a_i. Let us denote this vector by $\mathbf{b}^{(i)}$. Then

$$\widehat{var}\{h(\mathbf{x})|x_i = a_i\} = \frac{1}{2}\{h(\mathbf{a}) - h(\mathbf{b}^{(i)})\}^2.$$

A better estimate of $E\{var[h(\mathbf{x})|x_i]\}$ can be obtained by averaging the variance estimate over several realizations. Let $\mathbf{b}_i \overset{iid}{\sim} \mathcal{U}[0,1]^p$ for $i = 1, \ldots, m$. This can be grouped into a matrix $\mathbf{B} = (\mathbf{b}_1', \ldots, \mathbf{b}_m')'$. Recall that we already have the evaluations $h(\mathbf{A}) = (h(\mathbf{a}_1), \ldots, h(\mathbf{a}_m))'$, which was used to calculate the overall variance in (6.5). We will now do a new set of evaluations $h(\mathbf{B}^{(\mathbf{A}_i)}) = (h(\mathbf{b}_1^{(i)}), \ldots, h(\mathbf{b}_m^{(i)}))'$, where $\mathbf{b}_j^{(i)}$ is the vector $\mathbf{b}_j$ with its ith value replaced by the ith value of $\mathbf{a}_j$ and therefore

$$\mathbf{B}^{(\mathbf{A}_i)} = \begin{bmatrix} & & & i & & \\ b_{11} & \cdots & a_{1i} & \cdots & b_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{m1} & \cdots & a_{mi} & \cdots & b_{mp} \end{bmatrix}, \; i = 1 \ldots, p,$$

that is, $\mathbf{B}^{(\mathbf{A}_i)}$ is the $\mathbf{B}$ matrix with its ith column replaced by the ith column of $\mathbf{A}$.

Then, the variance for the ith main effect can be estimated using

$$\hat{V}_i = \hat{V} - \frac{1}{2m}\sum_{j=1}^{m}\{h(\mathbf{a}_j) - h(\mathbf{b}_j^{(i)})\}^2 = \hat{V} - \frac{1}{2m}\|h(\mathbf{A}) - h(\mathbf{B}^{(\mathbf{A}_i)})\|^2. \tag{6.6}$$

This method was proposed by Jansen (1999) and developed further by Saltelli et al. (2010). Its computation requires only $2m$ function evaluations. Thus, the first-order Sobol' indices of all the factors can be computed with $m(p+1)$ function evaluations, which is quite reasonable.

Similarly, to compute the Sobol' total index, first write the V_i^{tot} as

$$\begin{aligned}V_i^{tot} &= V - V_{\sim i} \\ &= var\{h(\mathbf{x})\} - var\{E[h(\mathbf{x})|\mathbf{x}_{\sim i}]\} \\ &= E\{var[h(\mathbf{x})|\mathbf{x}_{\sim i}]\}.\end{aligned}$$

To compute the conditional variance, we need to fix all values of the vector $\mathbf{a}_j$ except its ith value, which is replaced by the ith value of $\mathbf{b}_j$. Denote this vector by $\mathbf{a}_j^{(i)}$. We will now

do a new set of evaluations $h(\mathbf{A}^{(\mathbf{B}_i)}) = (h(\mathbf{a}_1^{(i)}), \ldots, h(\mathbf{a}_m^{(i)}))'$, where $\mathbf{A}^{(\mathbf{B}_i)}$ is the $\mathbf{A}$ matrix with its ith column replaced by the ith column of $\mathbf{B}$. Thus, we obtain the estimate (Jansen, 1999)

$$\hat{V}_i^{tot} = \frac{1}{2m}\sum_{j=1}^{m}\{h(\mathbf{a}_j) - h(\mathbf{a}_j^{(i)})\}^2 = \frac{1}{2m}\|h(\mathbf{A}) - h(\mathbf{A}^{(\mathbf{B}_i)})\|^2. \tag{6.7}$$

Thus, the total Sobol' indices of all the p factors can also be computed with $m(p+1)$ function evaluations. There are many other efficient methods for computing the first-order and total sensitivity indices. See books Saltelli (2008) and Da Veiga et al. (2021) for details.

Consider, for example, the function that models the flow of water through a borehole (Morris et al., 1993):

$$y = \frac{2\pi x_3(x_4 - x_6)}{\log\left(\frac{x_2}{x_1}\right)\left[1 + \frac{2x_7x_3}{\log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right]},$$

where the inputs are: radius of the borehole (x_1) in $[.05, .15]$ m, radius of influence (x_2) in $[100, 50000]$ m, transmissivity of upper aquifer (x_3) in $[63070, 115600]$ m^2/yr, potentiometric head of upper aquifer (x_4) in $[990, 1110]$ m, transmissivity of lower aquifer (x_5) in $[63.1, 116]$ m^2/yr, potentiometric head of lower aquifer (x_6) in $[700, 820]$ m, length of borehole (x_7) in $[1120, 1680]$ m, and hydraulic conductivity of borehole (x_8) in $[9855, 12045]$ m/yr. The response (y) is the water flow rate with units m^3/yr. The first-order and total Sobol' indices can be obtained using the `soboljansen` function in the R package `sensitivity` (Iooss et al., 2024). We used 10,000 Monte Carlo samples for $\mathbf{A}$ and $\mathbf{B}$. The estimated total Sobol' indices are shown in the left panel of Figure 6.1 as a bar plot. Each bar is divided into two parts with S in light blue color (which quantifies main effects) and $T - S$ in pink color (which quantifies interaction effects). We can see that the radius of the borehole x_1 is the most important factor. The factors $\{x_4, x_6, x_7, x_8\}$ have non-negligible effects, whereas $\{x_2, x_3, x_5\}$ are unimportant.

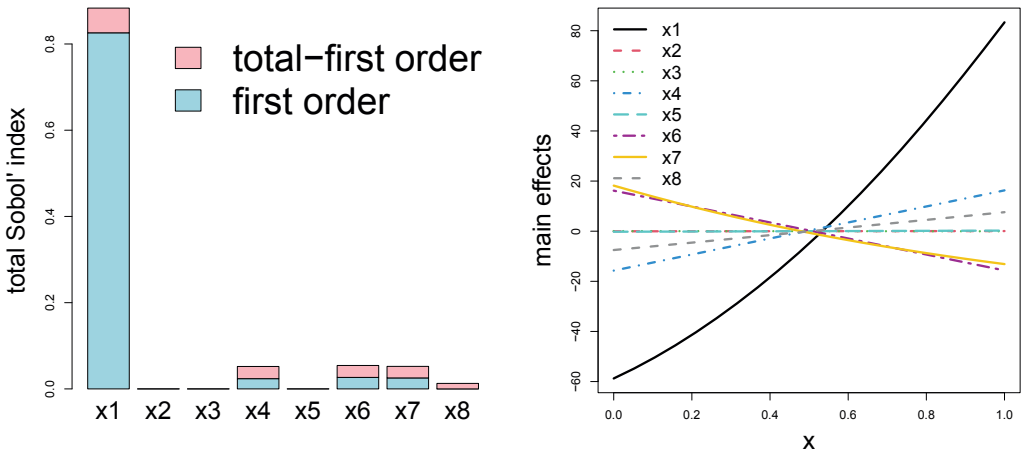


FIGURE 6.1: (left) Total Sobol' indices of the borehole function. The first-order indices are shown in light blue and the difference between total and first-order in pink. (right) Main-effects plot.

The Monte Carlo-based estimation of the Sobol' indices can become very expensive for computationally intensive models. An approach to reduce the computational burden is to first evaluate the computer model using a space-filling design and then approximate it using a Gaussian process model. In fact, analytical expressions for the main effects and first-order Sobol' indices can be easily derived assuming a product Gaussian correlation function (Oakley and O'Hagan, 2004; Chen et al., 2005). However, the formulas get a bit cumbersome for higher-order sensitivity indices. It is much easier to use the Monte Carlo methods to compute the sensitivity indices once we fit the Gaussian process model. For the borehole function, we used a MaxPro design in $n = 80$ runs and fitted a Gaussian process model with Gaussian correlation function using `rkriging` (Huang and Joseph, 2024). The estimated main effects are plotted in the right panel of Figure 6.1. The plot visually agrees with the first-order indices that we computed earlier shown in the left panel of the same figure. However, this approach has several limitations: (i) space-filling designs may not be efficient or optimal for estimating the Sobol' indices; (ii) for high-dimensional problems, we may need very large space-filling designs; and (iii) the Gaussian process model may not be accurate for high-dimensional problems. Due to these limitations, later in this chapter we will develop screening designs that directly target the estimation of the sensitivity indices.

6.1.2 Derivative-based Sensitivity Measures

The local sensitivity of the function $h(\mathbf{x})$ with respect to x_i at $x_i = x_i^0$ can be measured using the derivative $\partial h(\mathbf{x})/\partial x_i$ evaluated at $x_i = x_i^0$. Using this, a global measure can be defined as (Sobol' and Kucherenko, 2009)

$$\nu_i = \int_{[0,1]^p} \left(\frac{\partial h(\mathbf{x})}{\partial x_i} \right)^2 d\mathbf{x}. \tag{6.8}$$

Sobol' and Kucherenko (2009) showed the following relationship with total Sobol' indices:

$$T_i \leq \frac{\nu_i}{\pi^2 V}. \tag{6.9}$$

If $\nu_i = 0$, then $T_i = 0$. Thus, unimportant factors will have $\nu_i \approx 0$, and therefore can be used to filter them. The derivatives of complex models can be obtained using finite-difference methods. This can be repeated over several points in the unit hypercube. Let $\mathbf{a}_j \overset{iid}{\sim} \mathcal{U}(0,1)^p$ for $j = 1, \ldots, m$. Then, ν_i can be estimated by using

$$\hat{\nu}_i = \frac{1}{m} \sum_{j=1}^{m} \left(\widehat{\frac{\partial h(\mathbf{x})}{\partial x_i}} \right)^2_{\mathbf{x}=\mathbf{a}_j}.$$

Assuming at least one $\hat{\nu}_i > 0$, we can use the following normalized version to quantify the importance of a factor:

$$\bar{\nu}_i = \frac{\hat{\nu}_i}{\sum_{j=1}^{p} \hat{\nu}_j}.$$

Consider again the borehole function. Suppose that we use $m = 4$ samples. The estimation can be done using the `delsa` function in the R package `sensitivity` (Iooss et al., 2024). Figure 6.2 shows the estimated sensitivities for 100 repetitions as a boxplot. We can see that it correctly identifies $\{x_2, x_3, x_5\}$ as unimportant factors. Since we used only four samples, the computation is also much faster than that of total Sobol' indices.

Although for the borehole function, the derivative-based sensitivity method correctly identifies x_1 as the most important factor, unfortunately, in general a large ν_i does not

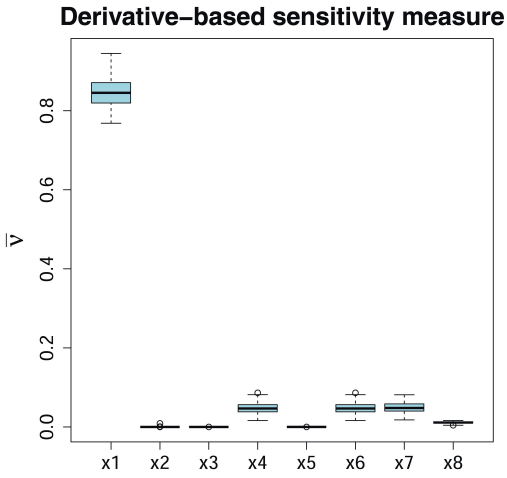


FIGURE 6.2: Derivative-based sensitivity indices of the borehole function, where the estimation is repeated 100 times using four samples in $[0, 1]^8$.

imply that the factor i is important. To understand this, consider a simple additive function in four factors:

$$h(\mathbf{x}) = \sum_{k=1}^{4} \frac{1}{k} \sin\left(\{k^2 + 1\}\pi x_k\right).$$

It is plotted in the left panel of Figure 6.3. The total Sobol' and derivative-based sensitivity measures for the four factors are plotted in the right panel of the same figure. We can see that the total Sobol' indices identify x_1 as the most important factor and x_4 as the

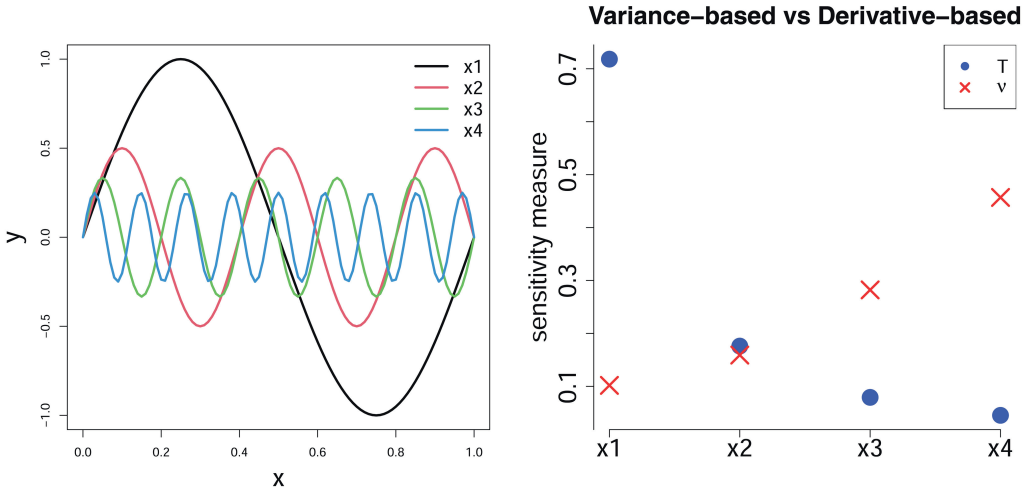


FIGURE 6.3: (left) Plot of four-dimensional additive sine function. (right) Total Sobol' indices and ν_i's for the four factors.

least important, whereas the derivative-based measure gives completely opposite order for the importance of the factors. The reason should be clear from the plot of the function: the response is smooth with respect to x_1 but has a large effect, while it is quite wiggly with respect to x_4 but the overall variability is small. This example shows that although derivative-based sensitivity measures can be used for screening the factors, they cannot be used to rank order the factors according to their importance.

6.2 Morris Screening Design

The total Sobol' indices for the borehole function shown in Figure 6.1 were calculated using $10000 \times (8 + 1) = 90,000$ function evaluations. This was fine because borehole function is inexpensive to evaluate. However, this cannot be done for computationally expensive models. Can we develop an experimental design method that will help us identify the important factors with a few evaluations of the function? Morris (1991) proposed such a method using one-factor-at-a-time (OFAT) designs. In general, OFAT designs are viewed negatively in the statistical literature due to its estimation inefficiency and its inability to estimate interaction effects. However, Morris' idea was to use a collection of OFAT designs, which overcomes some of these limitations.

Morris' method for screening can be explained as follows. Define elementary effects to estimate the local sensitivity of factors by changing one-factor-at-a-time. Let $\mathbf{x}_i(\Delta) = (x_1, \ldots, x_i \pm \Delta, \ldots, x_p)'$, where the $+$ or $-$ sign for Δ is chosen so that $x_i + \Delta \in [0, 1]$ or $x_i - \Delta \in [0, 1]$. Then, the p elementary effects are given by

$$d_i(\mathbf{x}) = \frac{h(\mathbf{x}_i(\Delta)) - h(\mathbf{x})}{\Delta}, \; i = 1, \ldots, p.$$

As $\Delta \to 0$, $d_i(\mathbf{x})$ converges to the derivative of $h(\mathbf{x})$, which shows a close connection between the derivative-based sensitivity measures and Morris screening method. In fact, the derivative-based measures developed in Sobol' and Kucherenko (2009) are motivated by the Morris method. However, since Morris' method uses a large value of Δ, in practice, its results are a compromise between the variance-based and derivative-based methods.

Let m $\mathbf{x}$'s be randomly chosen from a regular grid of L levels: $\{0, 1/(L-1), \ldots, 1\}^p$ to compute the elementary effects, where Δ is taken as a multiple of $1/(L-1)$. An example with $m = L = 4$, $p = 2$, and $\Delta = 1/(L-1)$ is shown in the left panel of Figure 6.4. This illustration uses "strict" OFAT designs according to the classification by Daniel (1973), where a run is based on the previous run. One can also use the "standard" OFAT design as shown in the right panel of the same figure, where all the changes are made from a standard or base run (shown as black circles in the figure). The "strict" version is more commonly used in practice (Saltelli, 2008; Da Veiga et al., 2021).

Let $d_{i(j)}$ be the elementary effect of x_i at $\mathbf{x} = \mathbf{x}_j$, $j = 1, \ldots, m$. Then, we can compute the following summary measures to understand the global effects of factors:

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} d_{i(j)} \text{ and } \sigma_i = \sqrt{\frac{1}{m-1} \sum_{j=1}^{m} (d_{i(j)} - \mu_i)^2}.$$

Based on these two measures, we can classify the factor effects as (i) negligible (low $|\mu_i|$ and low σ_i), (ii) linear and additive (high $|\mu_i|$ and low σ_i), and (iii) nonlinear and/or interactions

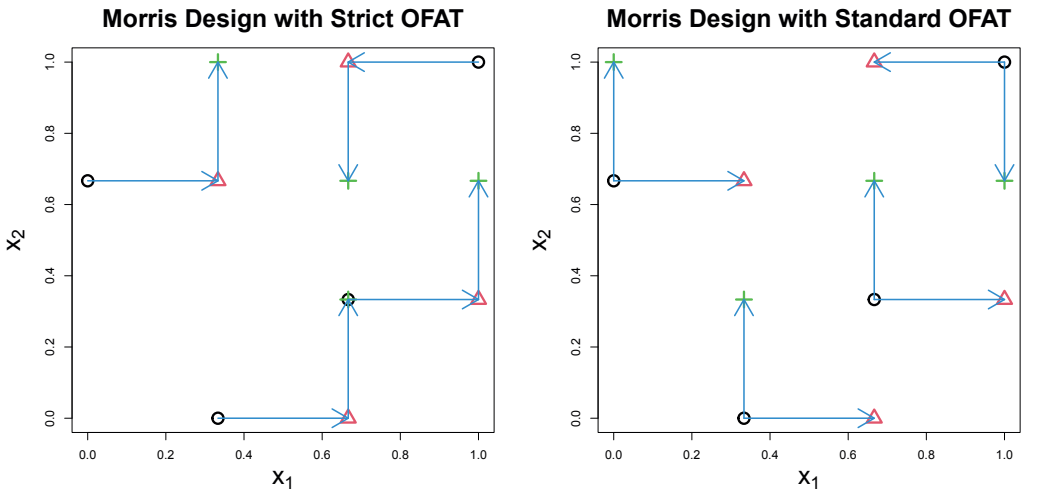


FIGURE 6.4: Morris design with strict OFAT (left) and standard OFAT (right). The OFAT structure is shown using arrows.

(high σ_i). Campolongo et al. (2007) proposed another measure by taking the absolute values of $d_{i(j)}$'s:

$$\mu_i^* = \frac{1}{m}\sum_{j=1}^{m} |d_{i(j)}|, \tag{6.10}$$

which is related to the derivative-based sensitivity measure $\int |\partial h(\mathbf{x})/\partial x_i|\, d\mathbf{x}$.

Consider again the borehole function. Suppose that we choose $L = m = 4$ and $\Delta = 3/(L-1)$. The total number of function evaluations will be $m(p+1) = 36$. We can use the `morris` function in the R package `sensitivity` (Iooss et al., 2024) to generate the design and compute the elementary effects. A plot of μ^* against σ is shown in the left panel of Figure 6.5. We can see that it correctly identifies x_1 as the most important effect followed by $\{x_7, x_4, x_6, x_8\}$. It only uses 36 evaluations of the function compared to the 90,000 evaluations used for computing the total Sobol' indices using Monte Carlo methods, which is very impressive. Since there is a lot of randomness in the Morris screening design, we have repeated this exercise 100 times and the results are plotted as a boxplot in the right panel of Figure 6.5. We can see that the method can successfully screen out the unimportant factors $\{x_2, x_3, x_5\}$.

Although not as random as Monte Carlo-based Sobol' designs, there is still randomness in Morris' screening method: m base points for the OFAT's are randomly chosen from the experimental region, and the order in which the variables are changed within an OFAT is also randomized. This randomness can produce designs with poor space-filling properties, which may not be good for follow-up experiments. Campolongo et al. (2007) proposed to improve the space-fillingness by maximizing the distance between OFATs. The distance between two OFATs is defined as the sum of the pairwise distances between the points in the two OFATs. Sun et al. (2013) investigated other possible distance metrics. However, as we discussed in Chapter 4, the use of maximizing the minimum or the sum of distances can push the points to the boundaries and can lead to poor projections of the points. Campolongo et al. (2011) proposed to use quasi-random sequences such as Sobol' points (see the next chapter) as the base runs along with standard OFAT designs (they called it radial designs) instead

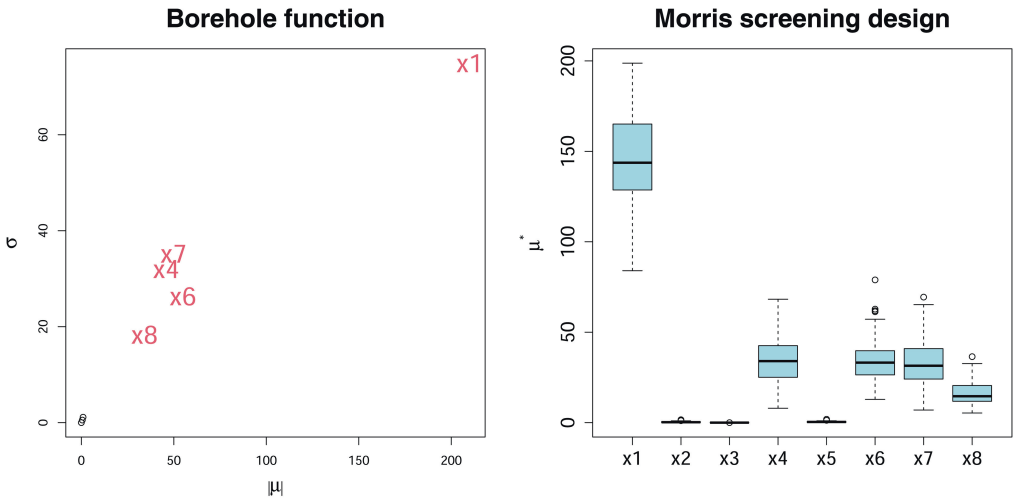


FIGURE 6.5: (left) Results from a Morris screening design on the borehole function. (right) Boxplots of μ^* from 100 repetitions.

of strict OFAT designs. Pujol (2009) tried to improve the projections using simplex-based designs instead of OFAT designs, but they require linear model fitting to estimate the effects which can introduce bias. Another attempt to improve Morris screening design known as the maximum OFAT design (MOFAT design) is discussed in detail in the next section.

6.3 MOFAT Designs

Xiao et al. (2023) noticed that the Monte Carlo-based Sobol' design for estimating total Sobol' indices can be viewed as a collection of standard OFATs. As discussed in Section 6.1.1, to estimate the total Sobol' indices, we need to first generate two random matrices $\mathbf{A} = (a_{ij})_{m\times p}$ and $\mathbf{B} = (b_{ij})_{m\times p}$, where $a_{ij}, b_{ij} \overset{iid}{\sim} \mathcal{U}(0,1)$. Then, the design is given by

$$\mathbf{D} = \begin{pmatrix} \mathbf{A} \\ \mathbf{A}^{(1)} \\ \vdots \\ \mathbf{A}^{(p)} \end{pmatrix}, \tag{6.11}$$

where $\mathbf{A}^{(i)}$ is the matrix $\mathbf{A}$ with its ith column replaced by the ith column of $\mathbf{B}$, which is a simplified notation for $\mathbf{A}^{(\mathbf{B}_i)}$ that we used earlier. If we take the first rows of $\mathbf{A}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(p)}$ and put them together as a design, we get the first OFAT design. Similarly, by taking the second rows of these matrices, we get the second OFAT, and so on. For example, let $m = 4$ and $p = 2$. Figure 6.6 shows two realizations of a 12-run design with $\mathbf{A}$, $\mathbf{A}^{(1)}$, and $\mathbf{A}^{(2)}$ represented by black circles, red triangles, and green pluses, respectively, where the OFAT structure is shown using arrows.

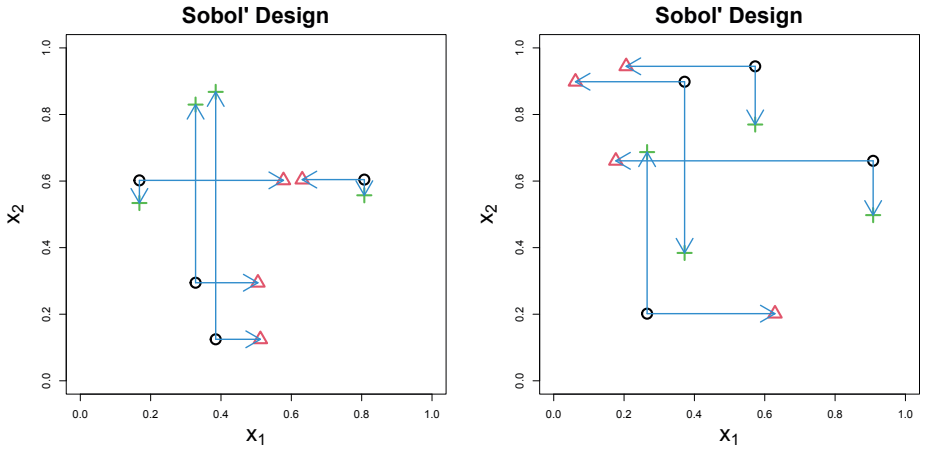


FIGURE 6.6: Two realizations of Monte Carlo-based Sobol' design. The OFAT structure is shown using arrows.

Although Sobol' design can be viewed as a collection of OFATs, they have several differences from a Morris screening design: (i) Morris method allows for any type of OFAT design (strict or standard), but Sobol' design has to be of standard type; (ii) Morris design will always be a collection of m OFATs, but a Sobol' design reduces to m OFATs only if $\mathbf{A}_j \neq \mathbf{A}_j^{(i)}$ for all i and j, which always happens with Monte Carlo samples (but not with quasi-random samples); (iii) Morris design has mp random values, whereas Sobol' design has $2mp$ random values because it needs two random matrices of size $m \times p$; and (iv) by construction Morris design always has constant Δ's, whereas Sobol' design has varying Δ's.

It is clear from Figure 6.6 that the Sobol' design on the right panel is better than the one shown on the left panel. Thus, we can try to improve the Sobol' design using some optimization techniques. Here we explain the approach of Xiao et al. (2023).

As can be seen in (6.7) and (6.10), both $\widehat{V}_i^{tot}$ and μ_i^* are sample averages to approximate some expectations. Sobol' (2001) uses Monte Carlo samples to compute the average. Therefore, the ith columns of $\mathbf{A}$ and $\mathbf{A}^{(i)}$ should be two independent uniform samples from $[0, 1]$. The best uniform sample of size m that minimizes the discrepancy from a uniform distribution is given by $\{.5/m, 1.5/m, \ldots, (m-.5)/m\}$ (Fang and Wang, 1993). Thus, we can take the ith columns of $\mathbf{A}$ and $\mathbf{A}^{(i)}$ as two random permutations of these m values. This makes $\mathbf{A}$, $\mathbf{A}^{(i)}$ $(i = 1, \ldots, k)$ as Latin hypercube designs (McKay et al., 1979).

Since there are many choices for LHDs, we can try to choose the one that improves the identification of important factors. One possibility is to choose the LHDs so as to maximize the total Sobol' index or μ_i^* because large values of them will help identify the important factors. However, this depends on the underlying response surface. Since the response surface is unknown to us before the experiment, we should design the experiment in such a way that it will work well irrespective of the response surface. Thus, to evaluate the effectiveness of a design, we may assume that the response surface to be a realization from a stochastic process.

Assume that $h(\mathbf{x})$ is a Brownian motion (Zhang and Apley, 2014) with

$$E\{h(\mathbf{u}) - h(\mathbf{v})\} = 0 \text{ and } var\{h(\mathbf{u}) - h(\mathbf{v})\} = \tau^2 ||\mathbf{u} - \mathbf{v}||_{\mathbf{w}}, \tag{6.12}$$

where $||\mathbf{u}-\mathbf{v}||_{\mathbf{w}} = \sqrt{\sum_{i=1}^{p} w_i(u_i - v_i)^2}$ and $\sum_{i=1}^{k} w_i = 1$, $w_i \geq 0$ for $i = 1, \ldots, p$. Brownian motion is a special case of the Gaussian process discussed in Section 2.1.4, but with a nonstationary covariance function. Here, w_i can be viewed as a measure of importance of the ith factor. Suppose that the experiment is performed according to the design in (6.11). Then, from (6.7)

$$\begin{aligned} E(\widehat{V}_i^{tot}) &= \frac{1}{2m}\sum_{j=1}^{m} E\{h(\mathbf{A}_j) - h(\mathbf{A}_j^{(i)})\}^2 \\ &= \frac{1}{2m}\sum_{j=1}^{m} var\{h(\mathbf{A}_j) - h(\mathbf{A}_j^{(i)})\} \\ &= \frac{\tau^2}{2m}\sum_{j=1}^{m} ||\mathbf{A}_j - \mathbf{A}_j^{(i)}||_{\mathbf{w}} \\ &= \frac{\tau^2\sqrt{w_i}}{2m}\sum_{j=1}^{m} |\mathbf{A}_{ji} - \mathbf{A}_{ji}^{(i)}|, \end{aligned} \tag{6.13}$$

where the last step follows from the fact that the jth rows of $\mathbf{A}$ and $\mathbf{A}^{(i)}$ differ only for the factor i, which is a unique property of the OFAT design. Since $\widehat{T}_i$ is proportional to $\widehat{V}_i^{tot}$, we can maximize the total Sobol' index by maximizing $\sum_{j=1}^{m} |\mathbf{A}_{ji} - \mathbf{A}_{ji}^{(i)}|$, which is the L_1 distance between the ith columns of $\mathbf{A}$ and $\mathbf{A}^{(i)}$. Define the Maximum One-Factor-At-a-Time (MOFAT) design as the design $\mathbf{D}$ that maximizes

$$\sum_{j=1}^{m} |\mathbf{A}_{ji} - \mathbf{A}_{ji}^{(i)}| \text{ for all } i = 1, \ldots, p,$$

where $\mathbf{A}$, $\mathbf{A}^{(i)}$ $(i = 1, \ldots, p)$ are all LHDs. This definition leads to the following optimality result for MOFAT design.

Theorem 6.1 *If the response surface is a realization of a Brownian motion field with mean and variance functions given in (6.12), then a MOFAT design maximizes the expected value of* $\widehat{V}_i^{tot}$ *for* $i = 1, \ldots, p$.

Finding a MOFAT design requires optimization over $2mp$ variables (i.e., the elements of matrices $\mathbf{A}$ and $\mathbf{B}$), which can be expensive for large m and p. This optimization can be simplified as follows. Define the following transformation:

$$\psi_m(x) = x - \left\lfloor \frac{m}{2} \right\rfloor + mI\left(x < \frac{m+1}{2}\right), \tag{6.14}$$

where $\lfloor x \rfloor$ is the largest integer not exceeding x and the indicator function $I(x < (1+m)/2)$ equals 1 if $x < (1+m)/2$ and 0 otherwise. Now, given an LHD $\mathbf{A}$, construct the matrix $\mathbf{B}$ using $\mathbf{B}_{ij} = \psi_m(\mathbf{A}_{ij})$, where the m levels of $\mathbf{A}$ and $\mathbf{B}$ are coded in $\{1, 2, \ldots, m\}$. Xiao et al. (2023) showed that the design $\mathbf{D}$ in (6.11) constructed using these two matrices $\mathbf{A}$ and $\mathbf{B} = \psi(\mathbf{A})$ will be a MOFAT design. So, the only computational expense is to construct the design $\mathbf{A}$ for which we can use a Maximin or MaxPro LHD.

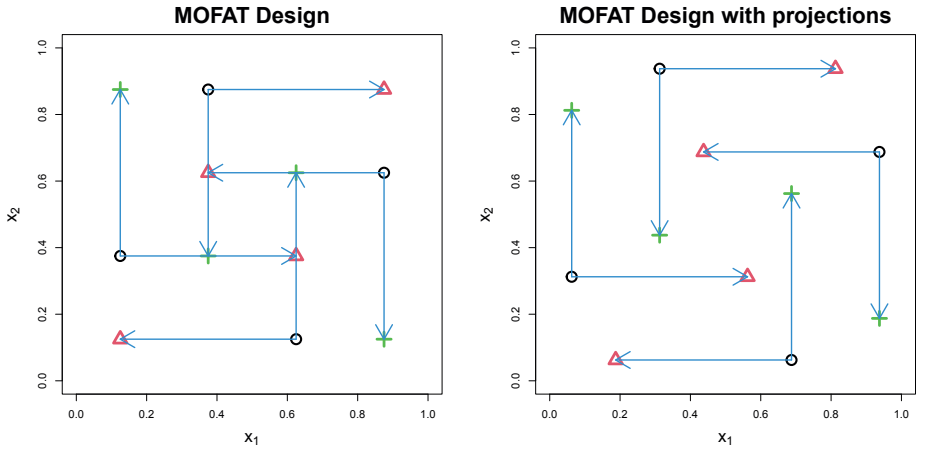


FIGURE 6.7: MOFAT design (left) and MOFAT design after incorporating projections (right). The OFAT structure is shown using arrows.

The left panel of Figure 6.7 shows an example of a MOFAT design with $m = 4$ and $p = 2$, which can be generated using the R package `MOFAT` (Xiao and Joseph, 2022). We can see that the MOFAT design tends to explore the experimental region in a much better way than the Morris designs in Figure 6.4 and Sobol' designs in Figure 6.6.

A MOFAT design has only m levels, $\{.5/m, 1.5/m, \ldots, (m-.5)/m\}$, for each factor. In addition to factor screening, if response surface modeling is also of interest, we may try to improve the projections of the MOFAT design by adjusting the values in $\mathbf{A}$ and $\mathbf{B}$ as follows:

$$x_i \leftarrow \begin{cases} x_i - \frac{.25}{m}, & \text{if } x_i < .5 \\ x_i + \frac{.25}{m}, & \text{if } x_i \geq .5 \\ x_i, & \text{otherwise,} \end{cases} \quad \text{and} \tag{6.15}$$

$$x_i \leftarrow \begin{cases} x_i + \frac{.25}{m}, & \text{if } x_i < .5 \\ x_i - \frac{.25}{m}, & \text{if } x_i \geq .5 \\ x_i, & \text{otherwise,} \end{cases} \tag{6.16}$$

for $i = 1, \ldots, m$. If (6.15) is used for $\mathbf{A}$, then (6.16) should be used for $\mathbf{B}$, and vice versa. This choice can be made using maximin L_1-distance criterion. The resulting MOFAT design is shown in the right panel of Figure 6.7. We can see that now both factors have eight levels as opposed to only four levels in the original design.

Consider again the borehole function. We use $m = 4$, thus a MOFAT design will have $4 \times (8 + 1) = 36$ runs. Figure 6.8 shows the boxplots of μ^* from 100 repetitions. We can see that the MOFAT design clearly identifies variables $\{x_1, x_4, x_6, x_7, x_8\}$ as important and $\{x_2, x_3, x_5\}$ as unimportant. MOFAT designs are almost deterministic, but the non-uniqueness of Maximin LHD used in the `MOFAT` package introduces some randomness. The results agree with those of the Morris screening design in Figure 6.5. However, the results of MOFAT have much less variability, which helps to distinguish important variables from unimportant variables with much less ambiguity.

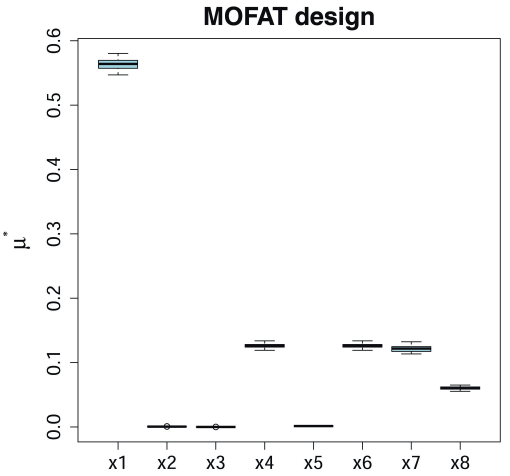


FIGURE 6.8: Results of using MOFAT designs on the borehole function. Boxplots of μ^* from 100 repetitions are shown.

Exercises

1. Suppose that
$$h(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p,$$
where x_i's are independent with mean μ_i and variance σ_i^2 for $i = 1, \ldots, p$. Show that
$$S_i = T_i = \frac{\beta_i^2 \sigma_i^2}{\sum_{j=1}^p \beta_j^2 \sigma_j^2},\ i = 1, \ldots, p.$$

2. Consider the g-function of Sobol' (2001):
$$h(\mathbf{x}) = \prod_{i=1}^p \frac{|4x_i - 2| + a_i}{1 + a_i}.$$
Show that (Santner et al., 2018, p. 270)
$$V_i = \frac{1}{3(1+a_i)^2}, V_i^{tot} = \frac{V_i}{(V_i+1)} \prod_{j=1}^p (V_j + 1), \text{ and } V = \prod_{j=1}^p (V_j + 1) - 1.$$

3. Suppose that we fitted a GP model: $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$ to the data $\{\mathbf{x}_i, y_i)\}_{i=1}^n$. The predictor is given by
$$\hat{y}(\mathbf{x}) = \hat{\mu} + \sum_{i=1}^n c_i R(\mathbf{x} - \mathbf{x}_i),$$
where $\hat{\mu} = \mathbf{1}'\mathbf{R}^{-1}\mathbf{y}/\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}$ and $\mathbf{c} = \mathbf{R}^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1})$. Assume a Gaussian correlation function $R(\boldsymbol{\Delta}) = \exp\{-\sum_{k=1}^p \Delta_k^2\}$. Then, show that (Oakley and O'Hagan, 2004; Chen

et al., 2005)

$$\hat{h}_0 = \hat{\mu} + \sum_{j=1}^{n} c_j \prod_{k=1}^{p} Q_{jk},$$

where

$$Q_{jk} = \int_0^1 e^{-\theta_k (x_k - x_{jk})^2} dx_k = \sqrt{\frac{\pi}{\theta_k}} \left\{ \Phi(\sqrt{2\theta_k}(1 - x_{jk})) - \Phi(-\sqrt{2\theta_k} x_{jk}) \right\}$$

and $\Phi(\cdot)$ is the standard normal distribution function. Also, show that the main effects can be analytically obtained as

$$\hat{h}_i(x_i) = \hat{\mu} + \sum_{j=1}^{n} c_j e^{-\theta_i (x_i - x_{ji})^2} \prod_{k \neq i} Q_{jk} - \hat{h}_0, \quad i = 1, \ldots, p. \tag{6.17}$$

4. Show that if m is even, then for the transformation in (6.14), $|\psi_m(x) - x| = m/2$ for all $x \in \{1, 2, \ldots, m\}$.

5. Show that in a MOFAT design,

$$\min_i |\mathbf{A}_{ij} - \mathbf{A}_{ij}^{(j)}| = \lfloor m/2 \rfloor, \ \forall j = 1, \ldots, p,$$

where the levels of the $\mathbf{A}$ matrix are coded in $\{1, 2, \ldots, m\}$.

6. Show that in a MOFAT design,

$$\sum_{i=1}^{m} |\mathbf{A}_{ij} - \mathbf{A}_{ij}^{(j)}| = \lfloor m^2/2 \rfloor, \ \forall j = 1, \ldots, p,$$

where the levels of the $\mathbf{A}$ matrix are coded in $\{1, 2, \ldots, m\}$.

7. A cyclone is used for separating solid mass and gaseous mass (Li and Wu, 1999). The critical parameter of particles (y) is related to seven input variables by

$$y = 174.42 \left(\frac{x_1}{x_5} \right) \left(\frac{x_3}{x_2 - x_1} \right)^{0.85} \left(\frac{1 - 2.62[1 - 0.36(\frac{x_4}{x_2})^{-0.56}]^{\frac{3}{2}} (\frac{x_4}{x_2})^{1.16}}{x_6 x_7} \right)^{\frac{1}{2}}.$$

The ranges of the variables are $x_1 \in [0.075, 0.125]$, $x_2 \in [0.225, 0.375]$, $x_3 \in [0.075, 0.125]$, $x_4 \in [0.075, 0.125]$, $x_5 \in [1.125, 1.875]$, $x_6 \in [12.0, 20.0]$, and $x_7 \in [0.5625, 0.9375]$. Answer the following questions.

   (a) Plot the main effects of the seven input variables.

   (b) Compute the first order and total sensitivity indices of the seven variables. (i) Order the seven variables according to their importance. (ii) Do you suspect any interactions (if yes, explain why)?

8. Fluidized-bed microencapsulation processes are used in the food industry to coat certain food products with additives. Three thermodynamic computer models were developed for predicting the steady-state thermodynamic operation point of a Glatt GPCG-1 fluidized-bed unit in the top-spray configuration (Reese et al., 2004). Several variables potentially affect the steady-state thermodynamic operating point: V_f (fluid velocity of the fluidization air), Ta (temperature of the air from the pump), R_f (flow rate of the coating

solution), P_a (pressure of atomization air), T_r (ambient room temperature), and Hr (ambient room humidity). A computer experiment was performed using a 28-run design. The data are given in the file `fluid data.txt`. The outputs from the three computer models are denoted by T_{21}, T_{22}, and T_{23} (ignore the column on T_2exp for the moment; we will use it later for an exercise in Chapter 9).

(a) Fit a GP model to the data using a Gaussian correlation function for each of the computer models.

(b) Compute the first-order and total Sobol indices of the six factors for the three computer models, and comment.

(c) Plot the main effects of the third computer model (output T_{23}) using (6.17).

9. Consider the Friedman function (Friedman, 1991) with $p = 10$ input factors, i.e., $x_1, \ldots, x_{10} \in [0, 1]$, where only the first five factors are significant:

$$h(\mathbf{x}) = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5. \tag{6.18}$$

Generate the following three designs:

(a) Sobol' Monte Carlo design with $m = 4$,

(b) Morris screening design with $m = L = 4$, and

(c) MOFAT design with $m = 4$.

Compute the μ^* measure and repeat the procedure 30 times. Show the results for the three designs using box plots.

10. Consider the Morris function (Morris, 1991) involving $p = 20$ input factors, $x_1, \ldots, x_{20} \in [0, 1]$:

$$h(\mathbf{x}) = \beta_0 + \sum_{i=1}^{20} \beta_i w_i + \sum_{i<j}^{20} \beta_{ij} w_i w_j + \sum_{i<j<l}^{20} \beta_{ijl} w_i w_j w_l + \sum_{i<j<l<s}^{20} \beta_{ijls} w_i w_j w_l w_s,$$

where

$$w_i = \begin{cases} 2\left(\frac{1.1x_i}{x_i+0.1} - 0.5\right), & \text{for } i = 3, 5, 7, \\ 2(x_i - 0.5), & \text{otherwise,} \end{cases}$$

and the coefficients are assigned as $\beta_i = 20$ for $i = 1, \ldots, 10$, $\beta_{ij} = -15$ for $i, j = 1, \ldots, 6$, $\beta_{ijl} = -10$ for $i, j, l = 1, \ldots, 5$ and $\beta_{ijls} = 5$ for $i, j, l, s = 1, \ldots, 4$. The remaining first- and second-order coefficients are independently generated from a standard Gaussian distribution, and the remaining third- and fourth-order coefficients are set to zero. This simulation model is available as `morris.fun` in the R package `sensitivity` (Iooss et al., 2024). Generate the following three designs:

(a) Sobol' Monte Carlo design with $m = 4$,

(b) Morris screening design with $m = L = 4$, and

(c) MOFAT design with $m = 4$.

Compute the total Sobol' indices and repeat the procedure 30 times. Show the results for the three designs using box plots.

7

Sequential Designs

The previous chapters discussed designs that can be used when we do not know anything about the input-output relationship. We developed space-filling and screening designs that will perform well over a wide class of functions, but they may not be optimal for a given function. In some cases, we can perform the experiment sequentially and learn about the underlying function as the experiment progresses. This may allow us to design future experiments that are specific to the function under study, which is the topic of this chapter. The sequential framework will also help us to introduce the objective of the experiment and thus design the experiment that is specific to the objective, such as optimizing the function. Sequential designs are also known by the name *active learning* in the machine learning literature.

7.1 Emulation

In this section, we will look at the objective of accurately approximating (or emulating) the underlying input-output function. To solve this problem, in Chapter 3, we tried to find optimal designs to minimize the integrated prediction variance or maximize the entropy of the design under a Gaussian process model. However, we quickly realized that these designs are not practical because of two reasons: (i) the designs depend on the unknown correlation parameters and (ii) the optimization is computationally expensive. In Chapters 4–6, we tried to overcome these two issues by finding space-filling designs that work for a wide class of functions and simplifying the design criteria to make the optimization faster by sidestepping the computation of $\mathbf{R}^{-1}$ or $|\mathbf{R}|$.

Sequential design is an alternative way to overcome the issues with the model-based optimal designs. We will do the experiments and observe the data one by one (or in small batches). This generates the design and the data sequentially: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. Since the data are observed sequentially, we can estimate the correlation parameters and use the estimated parameters to decide the next experiment. This overcomes the first issue. Since we only need to find the settings for the next experiment $\mathbf{x}_{n+1}$, it is a much smaller optimization problem. Moreover, the computation of $\mathbf{R}_{n+1}^{-1}$ and $|\mathbf{R}_{n+1}|$ can be simplified because we already know $\mathbf{R}_n^{-1}$ and $|\mathbf{R}_n|$. This overcomes the second issue.

Compared to space-filling designs, sequential designs have certain advantages. In space-filling designs, we first need to choose the size of the design, n. Loeppky et al. (2009) suggests $n = 10p$, where p is the number of input factors. However, the optimal choice of n depends on the complexity of the underlying function. If the function is linear, we need only $n = p+1$ experiments to approximate the function. However, if the function is non-smooth or very wiggly, then even $n = 100p$ may not be sufficient to accurately approximate the function. This is not an issue with sequential designs because the experiments are performed one by

DOI: 10.1201/9781003661535-7

This chapter has been made available under a CC-BY-NC-ND license 4.0.

one and we can stop when we are happy with the approximation, e.g., when the absolute cross-validation errors fall below a specified tolerance level.

Another advantage of sequential design is that it can place the points more intelligently in places where we really need data. For example, after some experiments with 10 factors $x_1, \ldots, x_{10}$, we may find that only x_1 and x_4 are important to the output. Then, the next experiment can focus only on the two-dimensional subspace of (x_1, x_4). On the other hand, in space-filling designs such as MaxPro designs, we try to place the points that give good projections to all possible subspaces of the experimental region. Such strategies make compromises, and therefore, the space filling in the space of (x_1, x_4) would be sacrificed to gain space-fillingness in the other subspaces. This is a waste of resources.

Although sequential design seems like a better strategy than space-filling designs, performing experiments sequentially is not always possible. Mak et al. (2018) reports an experiment using large eddy simulation codes to optimize the design of an injector that goes into a rocket. They conducted a 30-run experiment using MaxPro design. Each run of the simulation code takes about one week. However, the simulations can be run in parallel. Therefore, they were able to complete the entire experiment in less than six weeks. If we were to perform this experiment using sequential design, the whole experiment would have taken more than seven or eight months! Thus, sequential designs cannot always replace space-filling designs. In fact, both types of experiments go hand in hand. The most common strategy is to start with a space-filling design and then perform sequential experiments. See Figure 7.1. Having a space-filling design as the initial design will also ensure that the sequential design is not abruptly terminated due to poor estimation of the response surface.

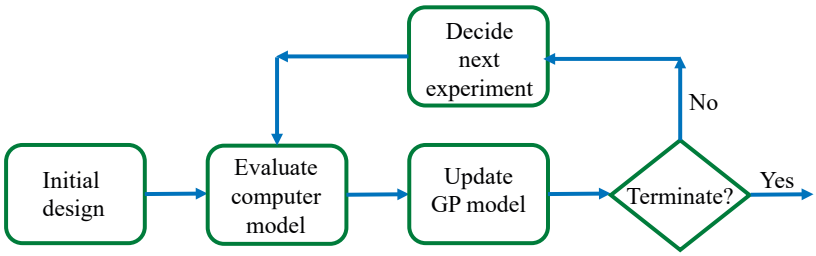


FIGURE 7.1: Sequential design of experiments.

7.1.1 Prediction-based Sequential Design

Suppose that we have performed n experiments sequentially using the design $D_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathcal{X} = [0,1]^p$, and observed the data $\mathbf{y}_n = (y_1, \ldots, y_n)'$. Our aim is to optimally choose where to perform the next experiment, that is, $\mathbf{x}_{n+1}$. Let $y = h(\mathbf{x})$ be the computer model. Assume a Gaussian process prior for the underlying function: $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$. If we were to observe y_{n+1} at the next experiment, then the posterior distribution would be

$$h(\mathbf{x})|D_{n+1}, \mathbf{y}_{n+1}, \boldsymbol{\theta} \sim \mathcal{N}(\hat{y}_{n+1}(\mathbf{x}; \boldsymbol{\theta}), s^2_{n+1}(\mathbf{x}; \boldsymbol{\theta})),$$

where

$$\begin{aligned} \hat{y}_{n+1}(\mathbf{x}; \boldsymbol{\theta}) &= \mu + \mathbf{r}_{n+1}(\mathbf{x})' \mathbf{R}^{-1}_{n+1}(\mathbf{y}_{n+1} - \mu \mathbf{1}_{n+1}), \\ s^2_{n+1}(\mathbf{x}; \boldsymbol{\theta}) &= \tau^2 \{1 - \mathbf{r}_{n+1}(\mathbf{x})' \mathbf{R}^{-1}_{n+1} \mathbf{r}_{n+1}(\mathbf{x})\}, \end{aligned}$$

where $\mathbf{r}_{n+1}(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^{n+1}$, $\mathbf{R}_{n+1} = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^{n+1}$, and $\boldsymbol{\theta}$ are the correlation parameters.

A sensible approach to choose $\mathbf{x}_{n+1}$ is to minimize $s^2_{n+1}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X} = [0,1]^p$. Due to the normality assumption, $s^2_{n+1}(\mathbf{x})$ is independent of y_{n+1}. This provides a great simplification for our sequential strategy because we do not know y_{n+1} until we perform the experiment. In Chapter 3, we discussed two cases: minimizing the average variance (IMSE) or the maximum variance (MMSE) over $\mathcal{X}$. Let us first consider the case of IMSE. Then, the sequential design strategy is to

$$\mathbf{x}_{n+1} \leftarrow \underset{\mathbf{x}_{n+1}}{\operatorname{argmin}} \int s^2_{n+1}(\mathbf{x}) d\mathbf{x}, \tag{7.1}$$

where

$$s^2_{n+1}(\mathbf{x}) = E\{s^2(\mathbf{x}_{n+1}; \boldsymbol{\theta}) | D_{n+1}, \mathbf{y}_{n+1}\} + var\{\hat{y}_{n+1}(\mathbf{x}; \boldsymbol{\theta}) | D_{n+1}, \mathbf{y}_{n+1}\}.$$

Obtaining the posterior distribution of $\boldsymbol{\theta}$ is computationally expensive. A common strategy is to ignore these uncertainties and fix $\boldsymbol{\theta}$ at its estimated value after the n experiments. This gives us the following approximation:

$$s^2_{n+1}(\mathbf{x}) \approx s^2(\mathbf{x}_{n+1}; \hat{\boldsymbol{\theta}}_n).$$

The computation of the objective function in (7.1) can be expensive because (i) we need to invert $\mathbf{R}_{n+1}$ and (ii) the integral can be high-dimensional. We can overcome the first issue as follows. By (2.19), the joint distribution of $h(\mathbf{x})$ and y_{n+1} given the data is a bivariate normal with variance

$$\begin{aligned} &var\left\{ \begin{pmatrix} h(\mathbf{x}) \\ y_{n+1} \end{pmatrix} | D_n, \mathbf{y}_n \right\} \\ &= \tau^2 \begin{bmatrix} 1 - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}) & R(\mathbf{x} - \mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1}) \\ R(\mathbf{x} - \mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1}) & 1 - \mathbf{r}_n(\mathbf{x}_{n+1})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1}) \end{bmatrix}. \end{aligned}$$

Now using Lemma 2.2, we obtain

$$s^2_{n+1}(\mathbf{x}) = s^2_n(\mathbf{x}) - \tau^2 \frac{\{R(\mathbf{x} - \mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})\}^2}{1 - \mathbf{r}_n(\mathbf{x}_{n+1})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})}, \tag{7.2}$$

which can be easily computed. Thus, (7.1) simplifies to (Gramacy and Lee, 2009)

$$\begin{aligned} \mathbf{x}_{n+1} &\leftarrow \underset{\mathbf{x}_{n+1}}{\operatorname{argmin}} \left\{ \int s^2_n(\mathbf{x}) d\mathbf{x} - \tau^2 \int \frac{\{R(\mathbf{x} - \mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})\}^2}{1 - \mathbf{r}_n(\mathbf{x}_{n+1})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})} d\mathbf{x} \right\} \\ &\equiv \underset{\mathbf{x}_{n+1}}{\operatorname{argmax}} \frac{\int \{R(\mathbf{x} - \mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})\}^2 d\mathbf{x}}{1 - \mathbf{r}_n(\mathbf{x}_{n+1})' \mathbf{R}_n^{-1} \mathbf{r}_n(\mathbf{x}_{n+1})}. \end{aligned} \tag{7.3}$$

Seo et al. (2000) call this strategy Active Learning-Cohn (ALC) because of its connections to a similar strategy used in neural networks (Cohn, 1996). Explicit expressions for the high-dimensional integral in the numerator can be obtained for certain correlation functions such as Gaussian, but in general it can be approximated using a Monte Carlo, Quasi-Monte Carlo, or a uniform design as in Section 5.1.2. See Binois et al. (2019) for the details on its efficient implementation. It is important to note that the sequential design points depend on the previously observed data $\mathbf{y}_n$ through their dependence on the estimated correlation parameters $\hat{\boldsymbol{\theta}}_n$. This is what makes these sequential designs different from the sequential construction of designs discussed in Chapter 4 such as (4.20) and (4.27).

Consider the Branin function given in (4.28), but with two redundant variables x_3 and x_4. Thus, the experimental region is a four-dimensional space, but only x_1 and x_2 affect the output. Suppose that we use a 20-run MaxPro design as the initial design. A Gaussian process model with Gaussian correlation function is fitted to the data using the R package `rkriging` (Huang and Joseph, 2024). Then, the next design point is chosen using (7.3), where the integral in the numerator is approximated using a candidate set of $100p$-Sobol' points. Maximization in (7.3) is performed approximately by choosing the best point out of the same candidates. Continuing with this procedure, we added a total of 10 points with the Gaussian process model updated at each step. The left and middle panels of Figure 7.2 show the points on the active (x_1, x_2) and inactive (x_3, x_4) subspaces, respectively. We can see that the points are more space-filling in the active dimensions than in the inactive dimensions. The root mean squared prediction error (RMSE) computed at 4,000 points is shown in the right panel of the same figure, which shows an improvement in the prediction as we add more points to the design.

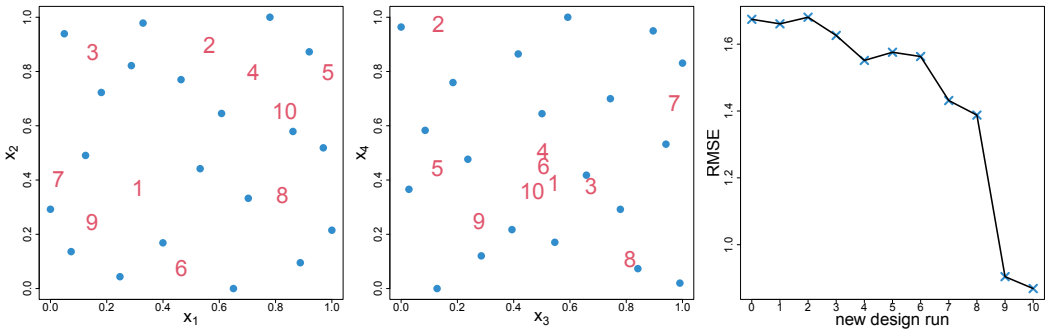


FIGURE 7.2: Sequential design using IMSE criterion (Active learning-Cohn) on the Branin function with two inactive variables.

We can also consider adding the next design point to minimize the maximum prediction variance (MMSE):

$$\mathbf{x}_{n+1} \leftarrow \underset{\mathbf{x}_{n+1}}{\operatorname{argmin}} \max_{\mathbf{x}\in\mathcal{X}} s^2_{n+1}(\mathbf{x}), \tag{7.4}$$

$$= \underset{\mathbf{x}_{n+1}}{\operatorname{argmin}} \max_{\mathbf{x}\in\mathcal{X}} \left\{ 1 - \mathbf{r}_n(\mathbf{x})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x}) - \frac{\{R(\mathbf{x}-\mathbf{x}_{n+1}) - \mathbf{r}_n(\mathbf{x})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x}_{n+1})\}^2}{1 - \mathbf{r}_n(\mathbf{x}_{n+1})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x}_{n+1})} \right\}.$$

The design points and RMSE using this strategy are shown in Figure 7.3. Compared to the sequential design using the IMSE criterion, the MMSE criterion seems to push the points toward the boundaries.

7.1.2 Entropy-based Sequential Design

Let us now consider the maximum entropy designs discussed in Section 3.2. Under the Gaussian process model, the design criterion for finding n points is to maximize $|\mathbf{R}_n|$. Therefore, the sequential design strategy for finding the next design point becomes

$$\mathbf{x}_{n+1} \leftarrow \underset{\mathbf{x}_{n+1}}{\operatorname{argmax}} |\mathbf{R}_{n+1}|. \tag{7.5}$$

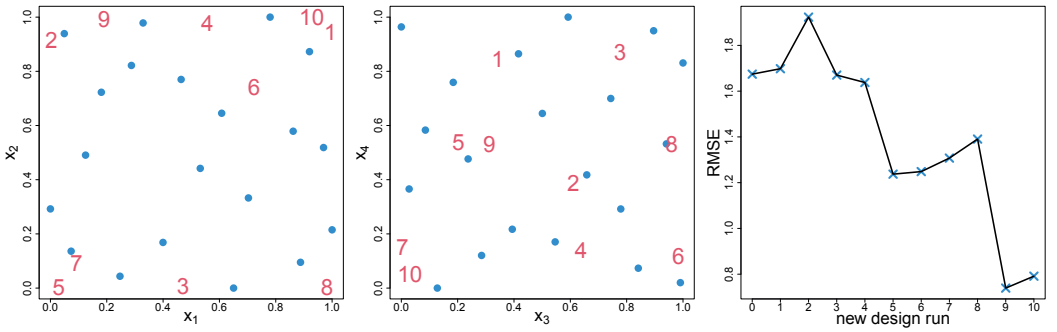


FIGURE 7.3: Sequential design using MMSE criterion on the Branin function with two inactive variables.

Using the result of determinant for partitioned matrices (Lemma 2.1), we have

$$|\mathbf{R}_{n+1}| = \begin{vmatrix} \mathbf{R}_n & \mathbf{r}_n(\mathbf{x}_{n+1}) \\ \mathbf{r}_n(\mathbf{x}_{n+1})' & 1 \end{vmatrix} = |\mathbf{R}_n|\{1 - \mathbf{r}_n(\mathbf{x}_{n+1})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x}_{n+1})\}.$$

Thus, (7.5) becomes

$$\mathbf{x}_{n+1} \leftarrow \underset{\mathbf{x}_{n+1}}{\operatorname{argmax}} \left\{1 - \mathbf{r}_n(\mathbf{x}_{n+1})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x}_{n+1})\right\}. \tag{7.6}$$

That is, the next point is chosen where the prediction uncertainty is the largest. This strategy was originally proposed for active learning by MacKay (1992) and is therefore known by the name Active Learning-MacKay (ALM) (Seo et al., 2000). This idea is also closely related to the algorithms used to find D-optimal designs (Fedorov, 1972).

Consider again the Branin function with two redundant variables discussed in the previous section. The design points and RMSE using (7.6) are shown in Figure 7.4. We can see that the design points are pushed toward the boundaries. This is expected because, as shown in Section 4.3, the maximum entropy design with Gaussian correlation function leads to the maximin design, which is known to push the points toward the boundaries. Song and Joseph (2025) recently showed that this issue can be mitigated by using multiplicative inverse multiquadric (MIM) kernels in (7.15) which encourages better low-dimensional projections. They also found improved results using a MOFAT design as the initial design instead of a space-filling design like MaxPro. This could be due to MOFAT design's ability to accurately estimate the length-scale parameters.

An advantage of ALM over prediction-based sequential designs is its computational speed. Generating 10 sequential design points using ALC (with 400 QMC points to approximate the integral) took 44.12 seconds, and the MMSE-based design took 57.28 seconds, while ALM took only 0.27 seconds on a 2.1 GHz desktop. However, all of these methods tend to spread the points uniformly in the experimental region, which is due to the constant-variance assumption of the Gaussian process model (ordinary kriging). Better sequential designs that place more points in the highly varying and interesting regions of the response surface can be obtained by using heteroskedastic Gaussian process models, such as the composite (Ba and Joseph, 2012) or the deep (Sauer et al., 2023) Gaussian process. A recent attempt that uses a heteroskedastic version of rational kriging in (2.52) can be found in Wang and Joseph (2025a).

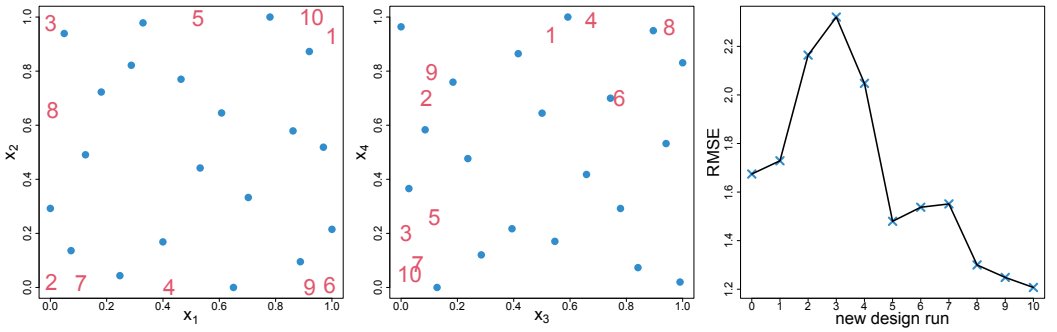


FIGURE 7.4: Sequential design using entropy-based criterion (Active Learning-MacKay) on the Branin function with two inactive variables.

7.2 Optimization

Suppose that our objective is to optimize a computationally expensive black-box function $h(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$. Without loss of generality, we will take the optimization to be a minimization problem and $\mathcal{X} = [0,1]^p$. Thus, the objective is to

$$\min_{\mathbf{x}\in[0,1]^p} h(\mathbf{x}). \tag{7.7}$$

This is a nonlinear optimization problem with bound constraints. In the optimization literature, there are many algorithms to solve this problem, which can be broadly classified as local and global. A local optimization algorithm such as Newton's method or Nelder-Mead simplex algorithm starts with a point in the feasible region and converges to a local optimum near the starting point. On the other hand, a global optimization algorithm, such as a simulated annealing or genetic algorithm, can converge to the global optimum under certain conditions. Both local and global algorithms require a large number of evaluations of $h(\mathbf{x})$, with global-type algorithms typically requiring more evaluations than a local-type optimization algorithm. Therefore, such algorithms cannot be applied directly to solve (7.7) where $h(\mathbf{x})$ is an expensive computer model.

We need a global optimization method that can solve (7.7) with as few evaluations of $h(\mathbf{x})$ as possible. One approach is to start with a small experimental design $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $[0,1]^p$, evaluate $y_i = h(\mathbf{x}_i)$, for $i = 1, \ldots, n$, and then approximate it using an easy-to-evaluate surrogate model $\hat{h}_n(\mathbf{x})$ such as the posterior mean of a Gaussian process (GP) model. We can then find the next design point by minimizing the surrogate model, that is, $\mathbf{x}_{n+1} = \text{argmin}_{\mathbf{x}} \hat{h}_n(\mathbf{x})$. This procedure can be continued until convergence with the surrogate model updated after each step. However, the new points are likely to be clustered around optimum obtained after the initial experimental design, leading to little improvements in the surrogate model approximation. Therefore, if the initial surrogate model approximation is not good, then this method is likely to fail.

A nice feature of the GP model is that we can probabilistically quantify the approximation error at each step of the sequential design using its variance estimate $s^2(\mathbf{x})$. Thus, if we can develop a method that can balance the dual objectives of optimization and surrogate model approximation, we can steer the method to converge to the global optimum. Such methods exist and are called Bayesian optimization methods because GP uses a Bayesian

framework. The idea of using stochastic processes for global optimization can be traced back to Kushner (1964). More details of Bayesian optimization can be found in the tutorial paper by Frazier (2018) or in the book by Garnett (2023).

The literature on Bayesian optimization is huge. Here, we will focus on one of the most commonly used Bayesian optimization methods, known as expected improvement. This method was originally proposed by Jones et al. (1998), which builds on an earlier work by Mockus (1978). See the review paper by Jones (2001) for more details.

Suppose that we have performed n experiments sequentially using the design $D_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and observed the data $\mathbf{y}_n = (y_1, \ldots, y_n)'$. Assume a Gaussian process prior for the underlying function: $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$. Then the posterior distribution would be

$$h(\mathbf{x})|D_n, \mathbf{y}_n \sim \mathcal{N}(\hat{y}_n(\mathbf{x}), s_n^2(\mathbf{x})), \tag{7.8}$$

where, as before

$$\begin{aligned} \hat{y}_n(\mathbf{x}) &= \mu + \mathbf{r}_n(\mathbf{x})'\mathbf{R}_n^{-1}(\mathbf{y}_n - \mu\mathbf{1}_n), \\ s_n^2(\mathbf{x}) &= \tau^2\{1 - \mathbf{r}_n(\mathbf{x})'\mathbf{R}_n^{-1}\mathbf{r}_n(\mathbf{x})\}, \end{aligned}$$

where $\mathbf{r}_n(\mathbf{x}) = \{R(\mathbf{x} - \mathbf{x}_i)\}_{i=1}^n$ and $\mathbf{R}_n = \{R(\mathbf{x}_i - \mathbf{x}_j)\}_{i,j=1}^n$. Let $h_{min}^{(n)} = \min \mathbf{y}_n$. Define the improvement function by

$$I(\mathbf{x}) = \max\{h_{min}^{(n)} - h(\mathbf{x}), 0\}.$$

Thus, if we choose an $\mathbf{x}$ with $h(\mathbf{x}) < h_{min}^{(n)}$, then we have an improvement; otherwise there is no improvement. Thus, we should choose an $\mathbf{x}$ to maximize the improvement. However, we cannot do this directly because we do not know $h(\mathbf{x})$ for all $\mathbf{x}$. But we can predict it with some uncertainties using (7.8), which in turn induces a distribution for $I(\mathbf{x})$. Since $I(\mathbf{x})$ is evaluated with uncertainties, a sensible approach to choose the next point is to maximize the expected value of $I(\mathbf{x})$, that is,

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}}{\operatorname{argmax}}\, E\{I(\mathbf{x})|D_n, \mathbf{y}_n\}.$$

Jones et al. (1998) provided an explicit expression for the expected improvement (EI), which makes optimization faster. It can be derived as follows:

$$\begin{aligned} EI = E\{I(\mathbf{x})|D_n, \mathbf{y}_n\} &= \int_{-\infty}^{\infty} \max\{h_{min}^{(n)} - y, 0\}\phi(y; \hat{h}_n(\mathbf{x}), s^2(\mathbf{x}))dy \\ &= \int_{-\infty}^{h_{min}^{(n)}} \{h_{min}^{(n)} - y\}\phi(y; \hat{h}_n(\mathbf{x}), s^2(\mathbf{x}))dy \\ &= \int_{-\infty}^{(h_{min}^{(n)} - \hat{h}_n(\mathbf{x}))/s_n(\mathbf{x})} \{h_{min}^{(n)} - \hat{h}_n(\mathbf{x}) - s_n(\mathbf{x})z\}\phi(z)dz \\ &= (h_{min}^{(n)} - \hat{h}_n(\mathbf{x}))\Phi\left(\frac{h_{min}^{(n)} - \hat{h}_n(\mathbf{x})}{s_n(\mathbf{x})}\right) + s_n(\mathbf{x})\phi\left(\frac{h_{min}^{(n)} - \hat{h}_n(\mathbf{x})}{s_n(\mathbf{x})}\right), \end{aligned}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal density and distribution functions, respectively. Thus, the EI algorithm becomes

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}}{\operatorname{argmax}} \left\{ s_n(\mathbf{x}) \left[u_n(\mathbf{x})\Phi(u_n(\mathbf{x})) + \phi(u_n(\mathbf{x})) \right] \right\}, \tag{7.9}$$

where $u_n(\mathbf{x}) = \{h_{min}^{(n)} - \hat{h}_n(\mathbf{x})\}/s_n(\mathbf{x})$. Taking derivatives, we obtain

$$\frac{\partial EI}{\partial \hat{h}_n(\mathbf{x})} = -\Phi\left(u_n(\mathbf{x})\right) < 0 \;\text{ and }\; \frac{\partial EI}{\partial s_n(\mathbf{x})} = \phi\left(u_n(\mathbf{x})\right) > 0,$$

which shows that EI monotonically decreases with $\hat{h}_n(\mathbf{x})$ and increases with $s_n(\mathbf{x})$. Therefore, EI can be maximized by choosing a point that minimizes $\hat{h}_n(\mathbf{x})$ or that maximizes $s_n(\mathbf{x})$; The former helps with optimization (exploitation) and the latter with the surrogate model approximation (exploration), as we saw in Section 7.1.2. In fact, EI nicely balances these two objectives, which helps the sequential design to converge to the global optimum (Bull, 2011).

As an example, consider the minimization of the one-dimensional function in (2.1). Let $\{(i-.5)/n\}_{i=1}^n$ be the initial design with $n = 8$ points. The design points, the true function, and the GP model fitted using `rkriging` (Huang and Joseph, 2024) are shown in the left panel of Figure 7.5. The EI is also shown in the same figure with its axis on the right side of the plot. The EI is maximized at $x = 0.25$, which will be the next point (x_9) to add to the design. Unfortunately, this is the value that maximizes the function! Such errors are likely to occur because our surrogate approximation can be poor. The GP model after adding this point is shown in the middle panel of Figure 7.5. Now we can see that the EI algorithm correctly identifies the global optimum, which is around $x = 0.16$. One more step is shown in the right panel of the same figure, which identifies the second-best minima. It is also possible to add two or more points at each step, which is called a batch sequential design. This is especially useful when parallel computing is possible. See Ginsbourger et al. (2010) for some useful heuristics for extending the single-point expected improvement algorithm for batch sequential design.

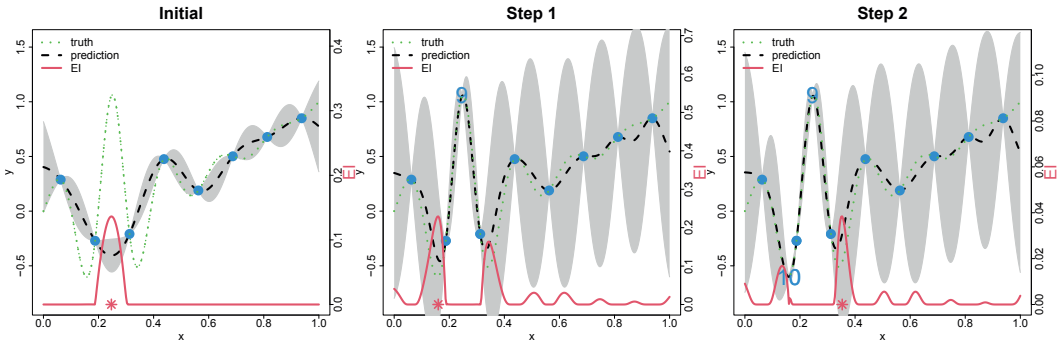


FIGURE 7.5: Two iterations of the expected improvement algorithm for minimizing the one-dimensional function in (2.1).

It is important to mention a numerical implementation issue with the EI algorithm. At some iteration, the new point can get very close to an existing point which can make the correlation matrix $\mathbf{R}_n$ ill-conditioned. As mentioned earlier in Section 2.2.3, a simple method to circumvent this issue is to add a small nugget $\lambda > 0$ to the diagonal elements of the $\mathbf{R}_n$ matrix. That is, we will use $\mathbf{R}_n + \lambda \mathbf{I}_n$ in place of $\mathbf{R}_n$. However, this will result in nonzero variance $s_n^2(\mathbf{x})$ at the design points, which can make the new design points very close to the existing design points. One way to avoid this situation is to use a candidate set for optimization in (7.9) that is spread out and away from the existing design points (Gramacy et al., 2022). Another approach is to use a more stable computation of $\mathbf{R}_n^{-1}$ as described in Ranjan et al. (2011). Let

$$\mathbf{A} = (\mathbf{R} + \lambda I)^{-1},$$

which can be accurately computed. However, our aim is to find $\mathbf{B} = \mathbf{R}^{-1}$. Note that

$$\begin{aligned}\mathbf{B} &= (\mathbf{R} + \lambda\mathbf{I} - \lambda\mathbf{I})^{-1} \\ &= \mathbf{A}(\mathbf{I} - \lambda\mathbf{A})^{-1} \\ &= \mathbf{A}(\mathbf{I} + \lambda\mathbf{A} + \lambda^2\mathbf{A}^2 + \cdots) \\ &= \mathbf{A}(\mathbf{I} + \lambda\mathbf{A}\{\mathbf{I} + \lambda\mathbf{A} + \cdots\}) \\ &= \mathbf{A}(\mathbf{I} + \lambda\mathbf{A}\{I - \lambda\mathbf{A}\}^{-1}) \\ &= \mathbf{A}(\mathbf{I} + \lambda\mathbf{B}).\end{aligned}$$

This suggests that we can compute $\mathbf{R}^{-1}$ using iterations

$$\mathbf{B}_{k+1} = \mathbf{A}(\mathbf{I} + \lambda\mathbf{B}_k),\ k = 1, 2, \ldots,$$

with $\mathbf{B}_1 = \mathbf{A}$. The convergence of $\mathbf{B}_k \to \mathbf{R}^{-1}$ occurs in a few iterations for small λ, which is implemented in the R package `rkriging` (Huang and Joseph, 2024).

7.3 Inverse Designs

In many engineering problems, the aim is to create a design that achieves certain properties for the product. Note that in this section, we will use the term "design" interchangeably – in engineering terminology to refer to a product design and in statistical terminology to refer to the experimental design. Their difference will be clear from the context. Suppose that the performance of the product is assessed using q properties (or outputs) $\mathbf{y} = (y_1, \ldots, y_q)'$ and let $\mathbf{x} = (x_1, \ldots, x_p)'$ be the p design variables (or inputs). Suppose that the ith output is related to the input variables by the relation $y_i = h_i(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X} = [0, 1]^p$ for $i = 1, \ldots, q$. The aim is to find the input design $\mathbf{x}^*$ that achieves a given output $\mathbf{y}^*$. Let w_i denote the weight assigned to the ith output to quantify its importance. Then, the inverse design can be obtained by solving the following optimization problem:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y}^* - \mathbf{h}(\mathbf{x})\|_{\mathbf{w}}, \tag{7.10}$$

where $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_q(\mathbf{x}))'$, $\|\mathbf{u} - \mathbf{v}\|_{\mathbf{w}} = \sqrt{\sum_{i=1}^q w_i(u_i - v_i)^2}$, $\sum_{i=1}^q w_i = 1$, and $w_i \geq 0$ for $i = 1, \ldots, q$.

Optimization in (7.10) will be prohibitively expensive if one or more of the $h_i(\mathbf{x})$'s is computationally expensive to evaluate. In such a case, we can use the Bayesian optimization techniques discussed in the previous section. Now suppose that we want to find inverse designs to achieve several different properties $\mathbf{y}_1^*, \ldots, \mathbf{y}_N^*$. Performing N Bayesian optimizations will be an arduous task when N is large. Moreover, it is not an efficient approach because the sequential designs used in each of the optimizations can overlap, and therefore, it will be a waste of computational resources. A more efficient approach might be to find the N local optima of

$$\min_{\mathbf{x}} \max_{i=1:N} \|\mathbf{y}_i^* - \mathbf{h}(\mathbf{x})\|_{\mathbf{w}}.$$

However, finding multiple optima is not an easy problem.

The foregoing problem becomes even more difficult if the user can specify any target in a continuous space $\mathcal{Y}$. In fact, this happens quite often in engineering design. To give a concrete example, consider the acoustic metasurface design problem studied by Krishna et al. (2022)

to independently control the amplitude and phase of the reflected and transmitted acoustic waves. The experiments can be performed by using the full-wave simulations in COSMOL, which solve a Helmholtz wave equation with some boundary conditions. In applications such as underwater communications or holography, the user may need different values for the amplitude and phase of both the reflected and transmitted acoustic waves, and the user will immediately have to come up with a design of the metasurface to achieve those targets. Krishna et al. (2022) tried to solve this problem by finding N designs to minimize

$$\max_{\mathbf{y}\in\mathcal{Y}} \min_{i} \|\mathbf{y} - \mathbf{h}(\mathbf{x}_i)\|_{\mathbf{w}} \tag{7.11}$$

with respect to $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. This is equivalent to finding a minimax design $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ in the output space $\mathcal{Y}$, where $\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i)$. However, the problem here is much more difficult because $\mathcal{Y}$ is unknown and $\mathbf{y}$ can only be observed by evaluating the expensive function $\mathbf{h}(\mathbf{x})$. Once we have the minimax design in the output space, for any given target $\mathbf{y}^*$, we can find $i^* = \text{argmin}_i \|\mathbf{y}^* - \mathbf{y}_i\|_{\mathbf{w}}$ and use the design $\mathbf{x}_{i^*}$ as a solution. In simple terms, $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ can be viewed as a look-up table.

Krishna et al. (2022) proposed a sequential perturbation algorithm to find the inverse design. The idea is to start with an initial design and perform full-wave simulations to obtain the outputs. The next step is to identify the largest gap in the output space, which is taken to be slightly larger than the convex hull of the observed outputs. Once the largest gap is identified, a perturbation is done on the corresponding point in the input space in the hope that the perturbed point will produce an output to fill in the gap. This procedure is continued until the largest gap in the output space is less than a specified tolerance. Improvements to this algorithm are proposed in Wang et al. (2024). A Gaussian process model-based approach for inverse designs is proposed in Bettinger et al. (2008). On the other hand, the sequential perturbation algorithm does not require any model fitting, and therefore is very fast and suitable for generating large designs. Krishna et al. (2022) had to perform 24,000 full-wave simulations to achieve the desired accuracy for the acoustic outputs, but it was only a small fraction of the 563×10^{12} possible acoustic metasurface designs in a 7×7 grid. This clearly shows the benefit of using sequential designs.

For illustration, consider a simple example with $p = q = 2$ taken from Wang et al. (2024):

$$y_1 = \frac{1}{\sqrt{x_1^2 + x_2^2 + .01}} \quad \text{and} \quad y_2 = \tan^{-1}\left(\frac{x_2}{x_1}\right).$$

We start with a 10-point MaxPro LHD in the input space. The corresponding points in the output are shown in the right panel of Figure 7.6 as solid blue circles. The biggest gap is identified, and the corresponding point in the input space is perturbed. See Wang et al. (2024) for details on how the biggest gap is identified and how the perturbation is performed. Their algorithm is implemented in the R package `OSFD` (Wang and Joseph, 2024). This is repeated for 40 steps. The points are shown as red crosses in the same figure. We can see that the points in the output space have expanded quite a bit from the space covered by the initial design. To achieve this, the input points have to be added mostly in the lower left corner of the input space. There is no easy way to identify this without using a sequential design algorithm.

We can see that although the output points fill the space in Figure 7.6, the input points are highly clustered. If our aim is to develop both forward and inverse models simultaneously, then it would be good to have space-fillingness in both the input and output spaces. This can be done by making some compromises on the space-fillingness of input and output spaces. Lu and Anderson-Cook (2021) and Barton and Morris (2025) explain how to achieve this using a two-stage model-based approach. Wang and Joseph (2025b) modified the approach

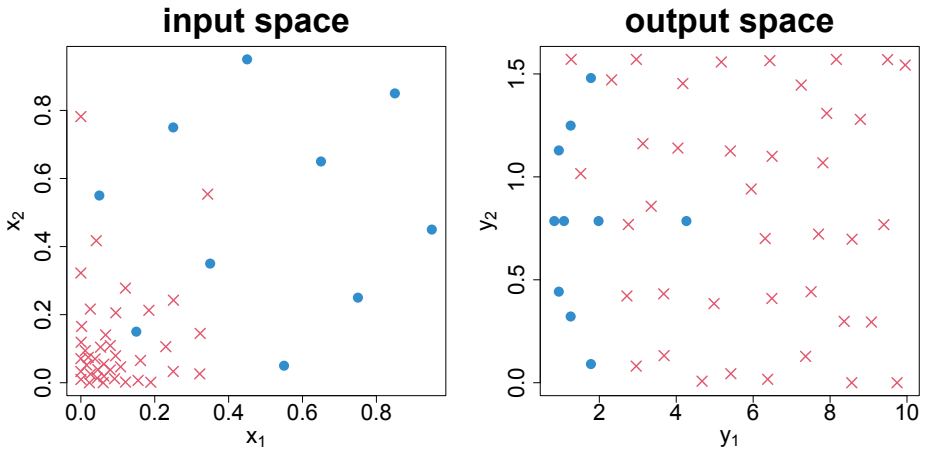


FIGURE 7.6: 40 design points (red crosses) added using the sequential perturbation algorithm starting with a 10-point MaxPro LHD (blue solid circles) in the input space for solving the inverse design problem.

of Barton and Morris (2025) to develop a fully sequential model-independent input-output space-filling design.

Interestingly, the sequential perturbation algorithm developed for the inverse designs has other applications. Consider the crystal structure prediction example from Krishna et al. (2024) described in Chapter 1. The input space is the Cartesian coordinates of the atomic configurations of a single crystal structure, and the output is the potential energy computed using Density Functional Theory (DFT). However, since the potential energy is invariant to translational, rotational, and permutational operations of the atoms, the Cartesian coordinate system is not suitable for model building. Therefore, the Cartesian coordinates are converted to a set of features using a fingerprinting technique. Since the Gaussian process model for the potential energy is fitted using the features, we need the points in the feature space to be space-filling. The fingerprinting is fast compared to DFT computations, but has non-negligible cost, which makes developing space-filling points in the feature space a difficult task. Therefore, we can apply the sequential perturbation algorithm by treating the feature space as the output space and the Cartesian coordinate space as the input space.

Exercises

1. Consider the improvement function (Schonlau et al., 1998)

$$I_2(\mathbf{x}) = \left(\max\left\{h_{min}^{(n)} - h(\mathbf{x}), 0\right\}\right)^2.$$

Then show that, the expected improvement function is given by

$$EI_2(\mathbf{x}) = s_n^2(\mathbf{x})\left\{(u_n^2(\mathbf{x})+1)\Phi(u_n(\mathbf{x})) + u_n(\mathbf{x})\phi(u_n(\mathbf{x}))\right\}, \tag{7.12}$$

where the notation is as defined in (7.9).

2. Ranjan et al. (2008) describes a problem of locating the contour that achieves a specified value a for the output $y = h(\mathbf{x})$:

$$S(a) = \{\mathbf{x} \in [0,1]^p : h(\mathbf{x}) = a\}.$$

The contour can be accurately estimated if we have design points in an ϵ-neighborhood, that is, in the constrained region $h(\mathbf{x}) \in [a-\epsilon, a+\epsilon]$. To do this in a sequential design framework, define the improvement function to be

$$I(\mathbf{x}) = \epsilon^2(\mathbf{x}) - \min\left\{(h(\mathbf{x}) - a)^2, \epsilon^2(\mathbf{x})\right\},$$

where ϵ is allowed to vary with $\mathbf{x}$. With this definition, improvement will be $\epsilon^2(\mathbf{x}) - (h(\mathbf{x}) - a)^2$ if $h(\mathbf{x}) \in [a - \epsilon(\mathbf{x}), a + \epsilon(\mathbf{x})]$ and 0 otherwise. Given the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we have the posterior distribution of $h(\mathbf{x})$ as in (7.8). Ranjan et al. (2008) recommends to choose $\epsilon(\mathbf{x}) = \alpha s_n(\mathbf{x})$, where $\alpha > 0$ is a pre-specified constant. Then, show that the expected improvement function is given by

$$\begin{aligned} E[I(x)] = &\left\{\alpha^2 s_n^2(x) - (\hat{y}_n(x) - a)^2\right\}\left[\Phi\left(\frac{a - \hat{y}_n(x)}{s_n(x)} + \alpha\right) - \Phi\left(\frac{a - \hat{y}_n(x)}{s_n(x)} - \alpha\right)\right] \\ &+ 2\{\hat{y}_n(x) - a\} s_n^2(x)\left[\phi\left(\frac{a - \hat{y}_n(x)}{s_n(x)} + \alpha\right) - \phi\left(\frac{a - \hat{y}_n(x)}{s_n(x)} - \alpha\right)\right] \\ &- \int_{a - \alpha s_n(x)}^{a + \alpha s_n(x)} \{y - \hat{y}_n(x)\}^2 \phi\left(\frac{y - \hat{y}_n(x)}{s_n(x)}\right) dy. \end{aligned}$$

3. Consider the isotropic Rational Quadratic (RQ) correlation function given by

$$R(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\theta^2}\right)^{-\alpha}, \alpha > 0 \tag{7.13}$$

Consider fitting a Gaussian process (ordinary kriging) with this correlation function and the design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Then, show that for all $\mathbf{x} \notin D$ (Song and Joseph, 2025):

$$\lim_{\theta \to 0} \frac{\tau^2 - s_n^2(\mathbf{x})}{\theta^{4\alpha}\tau^2} = \sum_{i=1}^n \frac{1}{\|\mathbf{x} - \mathbf{x}_i\|^{4\alpha}}. \tag{7.14}$$

4. Consider the Multiplicative Inverse Multiquadric (MIM) correlation function given by

$$R(\mathbf{x}_i, \mathbf{x}_j) = \prod_{k=1}^p \left(1 + \frac{(x_{ik} - x_{jk})^2}{\theta_k^2}\right)^{-\alpha}, \alpha > 0. \tag{7.15}$$

Consider fitting a Gaussian process (ordinary kriging) with this correlation function and the design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Then, show that for all $\mathbf{x} \notin D$ (Song and Joseph, 2025):

$$\lim_{\{\theta_k\}_{k=1}^d \to 0} \frac{\tau^2 - s_n^2(\mathbf{x})}{\tau^2} \prod_{k=1}^d \frac{1}{\theta_k^{4\alpha}} = \sum_{i=1}^n \frac{1}{\prod_{k=1}^d |x_k - x_{ik}|^{4\alpha}}. \tag{7.16}$$

Which correlation function (7.13) or (7.15) will produce better projection of points in sequential designs?

5. Consider the Branin function in (4.28) with two redundant variables x_3 and x_4. Find its global minima using the expected improvement algorithm. Start with a MaxPro design in 20 runs and add 20 more runs using the EI algorithm in (7.9). Plot the points in the (x_1, x_2)-space overlayed on the image of the Branin function. Now try the algorithm in (7.12) and comment.

6. Sequential designs require a p-dimensional optimization for generating each new point, which can be time consuming. The computational time needed for this optimization can be reduced by carefully selecting a finite set of candidates and then choosing the best. To test some strategies for candidate generation, consider the minimization of Levy function evaluated on $[-10, 10]^p$:

$$h(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{k=1}^{p-1} (w_k - 1)^2 \left[1 + 10\sin^2(\pi w_k + 1)\right] + (w_p - 1)^2 \left[1 + \sin^2(2\pi w_p)\right],$$
$$\text{where } w_k = 1 + \frac{x_k - 1}{4}, \text{ for all } k = 1, \ldots, p.$$

   Let $p = 6$ and start with $n = 10p$ points generate using `MaxPro`. Add $5p$ points using the EI algorithm, where the optimization in (7.9) is done using three candidate generation strategies:

   (a) 2^{15} Sobol' points generated using `spacefillr` (Morgan-Wall, 2025).

   (b) Points generated using `spanfill` function in `OSFD` (Wang and Joseph, 2024).

   (c) Points generated using `tricands` (Gramacy et al., 2022).

   Compare the three strategies using the minima achieved and the total time taken using 10 simulations.

7. So far we have been using space-filling designs such as MaxPro as the initial designs for sequential designs. Song and Joseph (2025) recently pointed out that MOFAT designs work much better than space-filling designs. To test this, consider again the minimization of the six-dimensional Levy function used in the previous exercise. Generate initial designs using

   (a) $4(p + 1)$-run `MaxPro` (Ba and Joseph, 2018) design.

   (b) `MOFAT` (Xiao and Joseph, 2022) design with $m = 4$.

   Compare the two initial design strategies using the minima achieved using 10 simulations.



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Part III

Physical Experiments



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

8

Fractional Factorial Designs

Physical experiments differ from computer experiments in many aspects. First of all, they are very expensive. For example, it is extremely expensive to build rockets and study their performance than to perform simulations on a computer. However, many physical systems are too complex to realistically model in a computer, and therefore physical experiments are inevitable. Second, the outputs of a physical experiment need to be measured using an instrument, which cannot offer arbitrary precision. Therefore, the data from a physical experiment are corrupted by measurement noise, unlike the output from a computer experiment, which can be computed with good precision. Moreover, there are many known or unknown variables in the physical system that cannot be controlled during the experiment. They can change during the experiment and can cause variations in the output, which again appear to us as noise in the output. Therefore, an experimenter must pay special attention to taking care of this noise using techniques such as replication, randomization, and blocking, which are irrelevant to computer experiments. Third, physical experiments are difficult and time consuming to perform. For example, to change the temperature in a furnace, we may have to wait for several hours, whereas in a computer experiment we only need to plug in a new value for the temperature, and the change happens instantaneously. For all these reasons, the number of runs in a physical experiment tends to be small. Linear models can do a good job for estimating the response surface from such small noisy data. Therefore, most of the literature on physical experiments focuses on linear regression models. However, as pointed out in Section 1.3, a linear regression model does not do well in capturing the model uncertainties and, therefore, has some limitations in developing optimal experimental designs. In this chapter, we will start with a brief review of experimental designs based on linear models and see how a Gaussian process model can help in the design and analysis of physical experiments.

8.1 Two-level Designs

There are many excellent textbooks on physical experiments that use linear regression models: Box et al. (1978); Mee (2009); Morris (2010); Lawson (2014); Dean et al. (2017); Montgomery (2017); Wu and Hamada (2021), to name a few. Therefore, we will only review some of the basic concepts that will help us to make some connections to the designs for computer experiments and develop an approach for physical experiments using Gaussian process regression. The readers are referred to one of these books for more details on physical experiments. For a more theoretical treatment of the subject, see Mukerjee and Wu (2007) and Cheng (2016).

Most physical experimental designs use two or three levels for each factor. This is because, as mentioned earlier, the run-size of the experiment is usually small and the data are noisy. Therefore, models with linear or quadratic effects might be able to approximate

DOI: 10.1201/9781003661535-8

This chapter has been made available under a CC-BY-NC-ND license 4.0.

the response surface well. For this approximation to work well, we can choose the range of factors to be not too large. Another reason for choosing two or three levels in physical experiments is that changing the factor levels in an experiment is not easy, and there can be errors in their settings. In this section, we will focus on two-level designs.

Suppose that we have p factors $x_1, \ldots, x_p$ each experimented at two levels using a n-run design D. A full factorial design would require $n = 2^p$ runs, which can be unaffordable in most practical scenarios when p is large. Therefore, we are mainly interested in designs with $n << 2^p$, that is, we will use only a fraction of the full factorial design. Such designs are called fractional factorial designs.

Since each variable has only two levels, we could consider fitting a linear regression model with only main effects:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon, \tag{8.1}$$

where $\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. Consider an unreplicated experiment and let $\mathbf{y} = (y_1, \ldots, y_n)'$ be the data. Then, as in (2.32), we can obtain the estimate of the parameters as

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y},$$

where $\mathbf{X} = [\mathbf{1}, D]$ is the model matrix. How should we choose the design D so that this is a good estimator?

In Section 2.2.1, we introduced several criteria for choosing an optimal design. Suppose that we use the D-optimal design:

$$\max_D |\mathbf{X}'\mathbf{X}|.$$

In general, we can use an optimization algorithm such as the Fedorov exchange algorithm (Fedorov, 1972) discussed in Section 2.2.1 to find the optimal design. However, for certain choices of n, the optimal design can be constructed in a much simpler way. The following is a useful result (Rao, 1973, p.236).

Theorem 8.1 $|\mathbf{X}'\mathbf{X}|$ *is maximized when the columns of* $\mathbf{X}$ *are orthogonal.*

To see this, partition the model matrix as follows: $\mathbf{X} = [\mathbf{x}_i \;\; \mathbf{X}_{(i)}]$. Then, using Lemma 2.1

$$\begin{aligned} |\mathbf{X}'\mathbf{X}| &= \begin{vmatrix} \mathbf{x}_i'\mathbf{x}_i & \mathbf{x}_i'\mathbf{X}_{(i)} \\ \mathbf{X}_{(i)}'\mathbf{x}_i & \mathbf{X}_{(i)}'\mathbf{X}_{(i)} \end{vmatrix} \\ &= |\mathbf{X}_{(i)}'\mathbf{X}_{(i)}|\{\mathbf{x}_i'\mathbf{x}_i - \mathbf{x}_i'\mathbf{X}_{(i)}(\mathbf{X}_{(i)}'\mathbf{X}_{(i)})^{-1}\mathbf{X}_{(i)}'\mathbf{x}_i\} \\ &\leq |\mathbf{X}_{(i)}'\mathbf{X}_{(i)}|\mathbf{x}_i'\mathbf{x}_i, \end{aligned}$$

where the inequality in the last step follows from the fact that $\mathbf{x}_i'\mathbf{X}_{(i)}(\mathbf{X}_{(i)}'\mathbf{X}_{(i)})^{-1}\mathbf{X}_{(i)}'\mathbf{x}_i \geq 0$. Therefore, the upper bound is achieved when $\mathbf{x}_i'\mathbf{X}_{(i)}(\mathbf{X}_{(i)}'\mathbf{X}_{(i)})^{-1}\mathbf{X}_{(i)}'\mathbf{x}_i = 0$. Since $\mathbf{X}_{(i)}'\mathbf{X}_{(i)}$ is positive definite, this happens when $\mathbf{x}_i'\mathbf{X}_{(i)} = 0$, which implies that $\mathbf{x}_i'\mathbf{x}_j = 0$ for all $i \neq j$.

Without loss of generality, we can take the two levels as -1 and 1. Then, to make each column orthogonal to the intercept column $\mathbf{1}$, we must have the same number of -1's and 1's in each column of the design. Moreover, to make two columns of the design matrix orthogonal, we must have pairs $\{(-1,-1),(-1,1),(1,-1),(1,1)\}$ appearing equal numbers of times.

Consider an example with four factors x_1, x_2, x_3, and x_4. Suppose that the experimenter can afford only 8 runs. Thus, we need a half fraction of the full factorial design, which will be called a 2^{4-1} design. To generate such a design, we can start with a full factorial design in x_1, x_2, and x_3, and then assign the column of x_4 to a product among the columns of

the three factors: $\{x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3\}$. This type of construction produces a *regular* design; see Wu and Hamada (2021, Ch. 8) for the construction of *nonregular* designs. Let $\mathbf{d}_i$ denote the ith column of the design matrix D and let $\mathbf{d}_i\mathbf{d}_j$ denote the element-wise product of the ith and jth columns. The design obtained by letting $\mathbf{d}_4 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3$ is shown in Table 8.1. It is easy to verify that the model matrix $\mathbf{X} = [\mathbf{1}\ D]$ is orthogonal and therefore this design is D-optimal for the estimation of the model in (8.1). If we used the assignment $\mathbf{d}_4 = \mathbf{d}_1\mathbf{d}_2$, we would get another 2^{4-1} design which is equally good as the previous design in terms of the D-optimality criterion.

TABLE 8.1: A 2^{4-1} design and data.

run	x_1	x_2	x_3	x_4	y
1	−1	−1	−1	−1	504
2	−1	−1	1	1	984
3	−1	1	−1	1	928
4	−1	1	1	−1	808
5	1	−1	−1	1	992
6	1	−1	1	−1	784
7	1	1	−1	−1	464
8	1	1	1	1	976

What if the model in (8.1) is wrong? One possibility is to consider the full model given by

$$\begin{aligned} y = {} & \alpha_0 + \alpha_1 x_1 + \cdots + \alpha_p x_p \\ & + \alpha_{12} x_1 x_2 + \cdots + \alpha_{p-1,p} x_{p-1} x_p \\ & + \alpha_{123} x_1 x_2 x_3 + \cdots + \alpha_{p-2,p-1,p} x_{p-2} x_{p-1} x_p \\ & + \cdots \\ & + \alpha_{12\cdots p} x_1 x_2 \cdots x_p + \delta, \end{aligned} \tag{8.2}$$

where $\delta \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$. Although many more terms are added to this model, it could still be wrong because the model does not contain any nonlinear terms such as quadratic or cubic effects. However, since we are restricting our design to two-level designs, such nonlinear terms cannot be entertained.

Unfortunately, the full model contains 2^p coefficients, much larger than the number of runs in a fractional factorial design. Therefore, this model cannot be estimated from the data. There are many ways to handle this issue. We can use a penalized regression method such as lasso (Tibshirani, 1996) or a Bayesian regression by placing an informative prior on the coefficients (Chipman et al., 1997). Before considering these more sophisticated approaches, we will review some of the traditional analysis techniques to gain some insight that is essential for developing optimal experimental designs.

In a 2^{p-k} regular fractional factorial design, a full model in $p-k$ factors is estimable. Consider the 2^{4-1} design in Table 8.1. We can fit a full model in any of the three factors, say $\{x_1, x_2, x_3\}$:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \beta_{123} x_1 x_2 x_3 + \epsilon. \tag{8.3}$$

This model gives

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = (805, -1, -11, 83, -73, -7, 15, 165)'.$$

The half-normal plot of the absolute coefficients is shown in Figure 8.1. We can see that the coefficients $\hat{\beta}_{123}$, $\hat{\beta}_3$, and $\hat{\beta}_{12}$ appear to be significant, which can be confirmed using Lenth's method (Lenth, 1989).

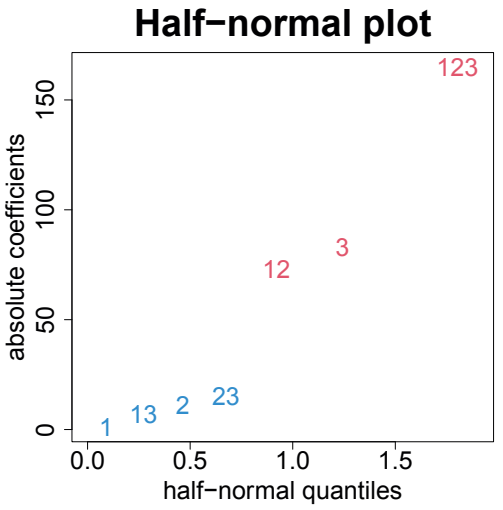


FIGURE 8.1: Half-normal plot of the absolute coefficients of the model in (8.3) that does not include x_4.

However, the model we were interested in fitting was

$$\mathbf{y} = \mathbf{F}\boldsymbol{\alpha} + \boldsymbol{\delta},$$

where $\mathbf{F}$ is the model matrix corresponding to the full model of the four factors $\{x_1, x_2, x_3, x_4\}$. By pre-multiplying both sides of the equation using $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$, we obtain

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{F}\boldsymbol{\alpha} + \tilde{\boldsymbol{\delta}},$$

where $\tilde{\boldsymbol{\delta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\delta}$. Simplifying this gives the following set of equations for the significant coefficients:

$$\begin{aligned}
\hat{\beta}_{123} &= \alpha_4 + \alpha_{123} + \tilde{\delta}_{123} = 165 \\
\hat{\beta}_3 &= \alpha_3 + \alpha_{124} + \tilde{\delta}_3 = 83 \\
\hat{\beta}_{12} &= \alpha_{12} + \alpha_{34} + \tilde{\delta}_{12} = -73.
\end{aligned}$$

How can we find the six coefficients from the above three equations? Since $\tilde{\delta}$'s are zero-mean error terms, we can ignore them. However, we cannot separate α_4 from α_{123} or α_3 from α_{124} or α_{12} from α_{34}. These coefficients are unidentifiable or in design terminology, they are *aliased*. We can try to break these aliases using some of the empirical principles such as effect hierarchy. The *effect hierarchy principle* states that (Wu and Hamada, 2021, p.168) (i) lower-order effects are more likely to be important than higher-order effects and (ii) effects of the same order are equally likely to be important. Using this principle, we obtain

$$\begin{aligned}
\hat{\alpha}_4 &= 165 \\
\hat{\alpha}_3 &= 83 \\
\hat{\alpha}_{12} + \hat{\alpha}_{34} &= -73.
\end{aligned}$$

Unfortunately, the effect hierarchy principle could not resolve the last alias because both α_{12} and α_{34} are of the same order. Here we can use another principle called effect heredity (Hamada and Wu, 1992). *Effect heredity principle* states that in order for an interaction to be important, at least one of its parent effects should be important. Since the main effects of x_1 and x_2 are not important and the main effects of x_3 and x_4 are important, it is quite likely that the observed effect of -73 is due to the two-factor interaction between x_3 and x_4. Thus, we obtain $\hat{\alpha}_{34} = -73$. Therefore, the final model is

$$\hat{y} = 805 + 165x_4 + 83x_3 - 73x_3x_4.$$

Although this model looks reasonable, it is somewhat unsettling that we have completely ignored α_{12}, as well as α_{123} and α_{124} from our analysis. We will come back to this issue in the next section and try to resolve the aliases in a different but more convincing way.

The foregoing analysis would be accurate if there were no aliases among the effects that are likely to be important. Therefore, we should design the experiment in such a way that we can avoid or minimize aliases among important effects. For regular 2^{p-k} fractional factorial designs, there is an easy way to understand the aliasing relationships without computing the aliasing matrix $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{F}$. Consider again the 2^{4-1} design in Table 8.1. We constructed this design using the *design generator* $\mathbf{d}_4 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3$. Since $\mathbf{d}_4\mathbf{d}_4 = \mathbf{1}$, by multiplying $\mathbf{d}_4$ on both sides, we obtain the *defining relation* of the design as $\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4$. This relation shows that the columns of $\mathbf{F}$ corresponding to α_0 and α_{1234} are identical. Therefore, these two coefficients are aliased. The remaining seven aliases of the 2^{4-1} design can be obtained as follows.

$$\begin{aligned}
\mathbf{d}_1 &= \mathbf{d}_2\mathbf{d}_3\mathbf{d}_4 \\
\mathbf{d}_2 &= \mathbf{d}_1\mathbf{d}_3\mathbf{d}_4 \\
\mathbf{d}_3 &= \mathbf{d}_2\mathbf{d}_3\mathbf{d}_4 \\
\mathbf{d}_4 &= \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3 \\
\mathbf{d}_1\mathbf{d}_2 &= \mathbf{d}_3\mathbf{d}_4 \\
\mathbf{d}_1\mathbf{d}_3 &= \mathbf{d}_2\mathbf{d}_4 \\
\mathbf{d}_2\mathbf{d}_3 &= \mathbf{d}_1\mathbf{d}_4.
\end{aligned}$$

From these, we can deduce that α_1 and α_{234} will be aliased, α_2 and α_{134} will be aliased, and so on. Now consider another 2^{4-1} design constructed using the design generator $\mathbf{d}_4 = \mathbf{d}_1\mathbf{d}_2$. Its defining relation is $\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_4$. We can obtain the remaining seven aliases as before and compare them with the aliases of the previous design to determine which design is better. Since all aliases are generated from the defining relation, it suffices to compare $\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4$ with $\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_4$. We can see that the first design has a "word" of length 4 and the second design has a word of length 3. Therefore, the first design will alias the main effects with three-factor interactions, whereas the second design will alias the main effects with two-factor interactions. By effect hierarchy principle, two-factor interactions are more likely to be important than three-factor interactions. Therefore, the aliases of the first design are less severe than those of the second design, which implies that we should prefer the first design.

As another example, consider a 2^{6-2} design with six factors, which is a quarter fraction of a full factorial 2^6 design. To construct this design, we can start with a 16-run full factorial design in the first four factors and assign the last two factors to the columns of the product of the first four factors. Suppose that we use the generators $\mathbf{d}_5 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4$ and $\mathbf{d}_6 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3$. Then, the defining relation is $\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4\mathbf{d}_5 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_6$. Since $\mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4\mathbf{d}_5$ and $\mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_6$ are both $\mathbf{1}$, their product will also be $\mathbf{1}$. Thus, taking their product, we

obtain the *defining contrast subgroup*:

$$\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4\mathbf{d}_5 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_6 = \mathbf{d}_4\mathbf{d}_5\mathbf{d}_6.$$

We can obtain the 15 remaining aliasing relationships from this defining contrast subgroup. The smaller words in the defining contrast subgroup are the "dangerous" words because they will lead to aliasing among lower-order effects. Therefore, define the *resolution* of the design as the minimum length of a word in the defining contrast subgroup. Thus, the above design has resolution 3. Obviously, we want to choose a design with maximum resolution, a criterion proposed by Box and Hunter (1961). Consider another 2^{6-2} design with generators $\mathbf{d}_5 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3$ and $\mathbf{d}_6 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_4$. Its defining contrast subgroup is given by

$$\mathbf{1} = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_5 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_4\mathbf{d}_6 = \mathbf{d}_3\mathbf{d}_4\mathbf{d}_5\mathbf{d}_6.$$

This design has resolution 4 and is therefore better than the previous design.

For larger designs, there will be many designs with the same resolution. Therefore, we need a further refinement of the maximum resolution criterion to choose optimal designs. For this define the *wordlength pattern* of a design D as

$$W(D) = (A_1(D), A_2(D), A_3(D), \ldots, A_p(D)), \tag{8.4}$$

where $A_i(D)$ is the number of words of length i. Consider two 2^{7-2} designs:

$$\begin{aligned} D_1 &: \mathbf{d}_6 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_4,\ \mathbf{d}_7 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3\mathbf{d}_5 \\ D_2 &: \mathbf{d}_6 = \mathbf{d}_1\mathbf{d}_2\mathbf{d}_3, \quad\ \mathbf{d}_7 = \mathbf{d}_1\mathbf{d}_4\mathbf{d}_5. \end{aligned} \tag{8.5}$$

Their wordlength patterns are given by $W(D_1) = (0, 0, 0, 1, 2, 0, 0)$ and $W(D_2) = (0, 0, 0, 2, 0, 1, 0)$. Thus, both designs have the same resolution of 4. However, we can see that the first design has only one word of length 4 as opposed to two words of length 4 for the second design. Intuitively, a smaller number of words will produce a smaller number of aliases with lower-order effects, and therefore the first design seems to be better. Fries and Hunter (1980) called the first design to have less aberration than the second design. If no other design has less aberration than a given design, then that design is said to have minimum aberration. In other words, a minimum aberration design can be obtained by sequentially minimizing $A_1, A_2, A_3, \ldots, A_p$.

In a 2^{p-k} design, there are $n - 1 = 2^{p-k} - 1$ alias sets excluding the defining contrast subgroup. Order them in such a way that the first p are alias sets that involve the main effects and the remaining sets that involve a two-factor interaction or higher. Let $m_i(D)$ be the number of two-factor interactions in the ith alias set, $i = 1, \ldots, n-1$. Cheng et al. (1999) proved the following result for a design of resolution 3 or higher (i.e., designs with $A_1(D) = A_2(D) = 0$):

$$\begin{aligned} A_3(D) &= \frac{1}{3}\left\{\binom{p}{2} - \sum_{i=p+1}^{n-1} m_i(D)\right\}, \\ A_4(D) &= \frac{1}{6}\left\{\sum_{i=1}^{n-1} m_i^2(D) - \binom{p}{2}\right\}. \end{aligned}$$

Since a minimum aberration design sequentially minimizes $A_3(D)$ and $A_4(D)$, it maximizes the number of two-factor interactions which are not aliases of main effects and, subject to that condition, minimizes the sum of squares of the sizes of alias sets of two-factor interactions. In other words, a minimum aberration design minimizes the contamination of

main effects from two-factor interactions and tries to uniformly distribute the two-factor interactions among the alias sets. This makes a minimum aberration design robust and a good choice to use in practice when we do not know which two-factor interactions are going to be important.

Minimum aberration designs can be found through combinatorial search, which can be computationally expensive (Xu, 2002). Minimum aberration designs of smaller sizes are tabulated in Wu and Hamada (2021, Ch. 5) or they can be quickly retrieved from the R package `FrF2` (Groemping, 2023).

8.2 Bayesian-inspired Designs

Table 8.2 shows the design and data from a cast fatigue experiment (Hamada and Wu, 1992). This design is an example of a *nonregular design*, which is obtained using the first seven columns of a 12-run orthogonal array, also known as the Plackett-Burman design (Wu and Hamada, 2021, Ch. 8). Let $\mathbf{X}$ be the model matrix corresponding to the main effects model in (8.1) and $\mathbf{F}$ corresponding to the full model in (8.2). Thus, $\mathbf{X}$ is a 12×8 matrix and $\mathbf{F}$ is a 12×128 matrix. The alias matrix $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{F}$ in this case is too complex with many non-zero entries. Therefore, it is hopeless to entertain two-factor and higher-order interactions in the analysis method that we followed in the case of a regular design. We need a completely different strategy to estimate them. Hamada and Wu (1992) proposed a forward variable selection strategy that incorporates the effect hierarchy and heredity principles. Extension of their strategy using the LARS, lasso, and Gauss-Dantzig selector algorithms can be found in Yuan et al. (2007), Choi et al. (2010), and Singh and Stufken (2024), respectively. See Kane and Mandal (2020) for a comparison of different variable selection strategies.

TABLE 8.2: Design matrix and data for cast fatigue experiment.

Run	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
1	1	1	−1	1	1	1	−1	6.058
2	1	−1	1	1	1	−1	−1	4.733
3	−1	1	1	1	−1	−1	−1	4.625
4	1	1	1	−1	−1	−1	1	5.899
5	1	1	−1	−1	−1	1	−1	7.000
6	1	−1	−1	−1	1	−1	1	5.752
7	−1	−1	−1	1	−1	1	1	5.682
8	−1	−1	1	−1	1	1	−1	6.607
9	−1	1	−1	1	1	−1	1	5.818
10	1	−1	1	1	−1	1	1	5.917
11	−1	1	1	−1	1	1	1	5.863
12	−1	−1	−1	−1	−1	−1	−1	4.809

Another approach to estimating the full model is to use a Bayesian framework because effect hierarchy and heredity principles can be viewed as prior information available to us before performing the data analysis. Let $\boldsymbol{\alpha} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the prior on $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{12\cdots p})$ in the full model (8.2), where $\boldsymbol{\mu}$ is a vector of length 2^p and $\boldsymbol{\Sigma}$ is a $2^p \times 2^p$ matrix. The entries of μ and $\boldsymbol{\Sigma}$ can be chosen carefully to reflect the effect hierarchy and heredity priors. However, this is an almost impossible task because there are too many parameters to

specify. Another way to think about this problem is to recognize that placing a prior on $\boldsymbol{\beta}$ induces a prior on the underlying input-output function $h(\mathbf{x}) \sim \mathcal{N}(\mathbf{f}(\mathbf{x})'\boldsymbol{\mu}, \mathbf{f}(\mathbf{x})'\boldsymbol{\Sigma}\mathbf{f}(\mathbf{x}))$, where $\mathbf{f}(\mathbf{x}) = (1, x_1, \ldots, x_1 x_2 \cdots x_p)$. But we know a much better way to place a prior on a function − the Gaussian process (GP) prior that we used extensively in Parts I and II of the book. Then, we can start with a GP prior on the function and go in the other direction to induce a prior for $\boldsymbol{\beta}$. This is much easier to do because for the GP prior we only need to specify a mean parameter μ, a variance parameter τ^2, and a few correlation parameters $\boldsymbol{\theta}$.

Let

$$h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot)),$$

where $R(\cdot)$ is the correlation function. Assume that the full model in (8.2) is the true model:

$$h(\mathbf{x}) = \mu + \mathbf{f}(\mathbf{x})'\boldsymbol{\alpha}, \tag{8.6}$$

where $\mathbf{f}(\mathbf{x}) = (1, x_1, \ldots, x_1 x_2 \cdots x_p)'$. Note that we have separated μ from α_0, which is only to get a simplified expression for $\boldsymbol{\alpha}$. Let $\mathbf{h} = (h_1, \ldots, h_{2^p})'$ be the function evaluations at the 2^p full factorial design points. Then, $\mathbf{h} = \mu\mathbf{1} + \mathbf{F}\boldsymbol{\alpha}$. Therefore,

$$\boldsymbol{\alpha} = \mathbf{F}^{-1}(\mathbf{h} - \mu\mathbf{1}).$$

Since $\mathbf{h} \sim \mathcal{N}(\mu\mathbf{1}, \tau^2\mathbf{R})$, we obtain (Joseph, 2006a):

$$\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \tau^2\mathbf{F}^{-1}\mathbf{R}(\mathbf{F}^{-1})'), \tag{8.7}$$

where $\mathbf{0}$ is a vector of zeros of length 2^p and $\mathbf{R}$ as defined in (2.13). This can be used as the prior for $\boldsymbol{\alpha}$.

Assume a product correlation function:

$$R(\mathbf{u} - \mathbf{v}) = \prod_{i=1}^{p} R_i(u_i - v_i). \tag{8.8}$$

Then, (8.7) can be simplified as follows. Since a 2^p full factorial design can be obtained by crossing the designs for the p factors

$$\mathbf{F} = \mathbf{F}_1 \otimes \mathbf{F}_2 \otimes \cdots \otimes \mathbf{F}_p = \bigotimes_{i=1}^{p} \mathbf{F}_i,$$

where $\otimes$ denotes the Kronecker product and

$$\mathbf{F}_i = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Let

$$\mathbf{R}_i = \begin{bmatrix} 1 & R_i(2) \\ R_i(2) & 1 \end{bmatrix}, \; i = 1, \ldots, p.$$

Under the product correlation structure

$$\mathbf{R} = \mathbf{R}_1 \otimes \mathbf{R}_2 \otimes \cdots \otimes \mathbf{R}_p = \bigotimes_{i=1}^{p} \mathbf{R}_i.$$

Using the properties of Kronecker products (see Lemma 9.1), we obtain

$$\begin{aligned}\tau^2\mathbf{F}^{-1}\mathbf{R}(\mathbf{F}^{-1})' &= \tau^2\left(\bigotimes_{i=1}^{p}\mathbf{F}_i\right)^{-1}\bigotimes_{i=1}^{p}\mathbf{R}_i\left(\left(\bigotimes_{i=1}^{p}\mathbf{F}_i\right)^{-1}\right)' \\ &= \tau^2\bigotimes_{i=1}^{p}\mathbf{F}_i^{-1}\mathbf{R}_i\left(\mathbf{F}_i^{-1}\right)' \\ &= \tau^2\bigotimes_{i=1}^{p}\frac{1}{2}\begin{bmatrix}1+R_i(2) & 0\\ 0 & 1-R_i(2)\end{bmatrix} \\ &= \xi^2\bigotimes_{i=1}^{p}\begin{bmatrix}1 & 0\\ 0 & \gamma_i\end{bmatrix},\end{aligned}$$

where

$$\xi^2 = \frac{\tau^2}{2^p}\prod_{i=1}^{p}\{1+R_i(2)\} \text{ and } \gamma_i = \frac{1-R_i(2)}{1+R_i(2)}.$$

Thus, we have the following result (Joseph, 2006a).

Theorem 8.2 *For two-level designs, the prior for* $\boldsymbol{\alpha}$ *in (8.6) induced from a GP under the product correlation structure is given by*

$$\begin{aligned}\alpha_0 &\sim \mathcal{N}(0,\xi^2)\\ \alpha_i &\sim \mathcal{N}(0,\xi^2\gamma_i)\ \forall i\\ \alpha_{ij} &\sim \mathcal{N}(0,\xi^2\gamma_i\gamma_j)\ \forall i,j\\ &\vdots\\ \alpha_{12\cdots p} &\sim \mathcal{N}(0,\xi^2\gamma_1\gamma_2\cdots\gamma_p)\end{aligned}$$

and they are all independent.

Since $\gamma_i \in (0,1)$ for all i, the variance of the main effects will be smaller than the variance of α_0. Similarly, the variances of two-factor interactions will be smaller than those of the main effects, and so on. Thus, this prior obeys the effect hierarchy principle. Moreover, if any of the γ_i or γ_j is small, then $var\{\alpha_{ij}\} = \xi^2\gamma_i\gamma_j$ will also be small. This means that a two-factor interaction will be significant only if either of its main effects is significant. Thus, the prior also obeys the effect heredity principle. In other words, the effect hierarchy and heredity principles are embedded in a GP that has a product correlation structure.

Thus, we have the Bayesian linear model

$$\begin{aligned}\mathbf{y}|\boldsymbol{\alpha} &\sim \mathcal{N}(\mu\mathbf{1}+\mathbf{X}\boldsymbol{\alpha},\sigma^2\mathbf{I}),\\ \boldsymbol{\alpha} &\sim \mathcal{N}(\mathbf{0},\xi^2\boldsymbol{\Gamma}),\end{aligned}$$

where $\mathbf{X}$ is the $n\times 2^p$ model matrix constructed based on the n-run fractional factorial design D and $\boldsymbol{\Gamma}$ is a diagonal matrix with diagonal elements 1 for α_0, γ_i for the ith main effect, etc. as given in Theorem 8.2. The posterior distribution of $\boldsymbol{\alpha}$ is given by

$$\begin{aligned}\boldsymbol{\alpha}|\mathbf{y} &\sim \mathcal{N}\left(\left\{\mathbf{X}'\mathbf{X}+\lambda\boldsymbol{\Gamma}^{-1}\right\}^{-1}\mathbf{X}'(\mathbf{y}-\mu\mathbf{1}),\sigma^2\left\{\mathbf{X}'\mathbf{X}+\lambda\boldsymbol{\Gamma}^{-1}\right\}^{-1}\right)\\ &= \mathcal{N}\left(\boldsymbol{\Gamma}\mathbf{X}'\left\{\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}'+\lambda\mathbf{I}\right\}^{-1}(\mathbf{y}-\mu\mathbf{1}),\xi^2\boldsymbol{\Gamma}-\xi^2\boldsymbol{\Gamma}\mathbf{X}'\left\{\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}'+\lambda\mathbf{I}\right\}^{-1}\mathbf{X}\boldsymbol{\Gamma}\right),\end{aligned} \tag{8.9}$$

where $\lambda = \sigma^2/\xi^2$. The two equivalent forms given here for the posterior distribution follow from Woodbury's matrix identity (Petersen and Pedersen, 2006, p. 18):

Lemma 8.1 *For conformable matrices* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U}$, *and* $\mathbf{V}$,

$$(\mathbf{A}+\mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{B}^{-1}+\mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}.$$

Consider again the 2^{4-1} design and data given in Table 8.1. For simplicity, let $\sigma^2 = 0$. Now fitting a GP with a Gaussian correlation function $R(\mathbf{u}-\mathbf{v}) = \exp\{-.5\sum_{i=1}^{p}(u_i - v_i)^2/\theta_i^2\}$ to the data using `rkriging` (Huang and Joseph, 2024) gives $\mu = 805$, $\tau^2 = 408413$, and $\boldsymbol{\theta} = (100, 26.8, 2.2, 4.1)'$. Thus, we obtain $\boldsymbol{\gamma} = (.01, .04, .43, .24)'$. The posterior means of $\boldsymbol{\alpha}$ from (8.9) excluding the intercept are plotted as a half-normal plot in Figure 8.2. We can see that α_4, α_3, and α_{34} are significant, which agrees with our analysis in Section 8.1.

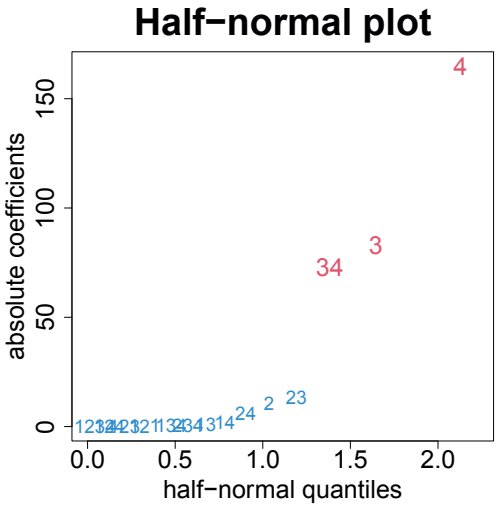


FIGURE 8.2: Half-normal plot of the absolute posterior mean of the coefficients in (8.9).

We can see that the aliasing is automatically resolved in the Bayesian analysis. To understand what is happening, consider the aliasing between α_4 and α_{123}:

$$\alpha_4 + \alpha_{123} = 165,$$

where the error term $\tilde{\delta}_{123}$ is ignored because σ^2 is assumed to be 0. The prior variance of α_4 is proportional to $\gamma_4 = 0.24$ and that of α_{123} is proportional to $\gamma_1\gamma_2\gamma_3 = 0.00016$. The posterior mean divides 165 into the two terms according to their prior variances:

$$\alpha_4 = \frac{0.24}{0.24+0.00016} \times 165 \quad \text{and} \quad \alpha_{123} = \frac{0.00016}{0.24+0.00016} \times 165.$$

Thus, α_4 receives the larger chunk of 165 making it a significant effect in the Bayesian analysis. The same happens to the other 15 aliases.

More generally, consider a 2^{p-k} design. Let $(\hat{\beta}_0, \ldots, \hat{\beta}_{n-1})'$ be the estimated coefficients of the full model containing $p-k$ factors, where $n = 2^{p-k}$. Let $J_0(D)$ denote the set of indices of the effects in the defining contrast subgroup and let $J_j(D)$, $j = 1, \ldots, n-1$ denote the set of indices of the other aliasing relationships. Then for a positive fraction (all design generators have positive signs) of a 2^{p-k} design, Joseph (2006a) showed that for an $I \in J_j(D)$

$$\alpha_I \sim \mathcal{N}\left(\frac{\prod_{i\in I}\gamma_i}{\sum_{I\in J_j(D)}\prod_{i\in I}\gamma_i + \lambda/n}\hat{\beta}_j, \xi^2\prod_{i\in I}\gamma_i - \frac{\xi^2\prod_{i\in I}\gamma_i^2}{\sum_{I\in J_j(D)}\prod_{i\in I}\gamma_i + \lambda/n}\right), \tag{8.10}$$

where $\gamma_0 = 1$. We can see from the posterior mean that the estimated aliased effect gets split into the effects in the aliasing relationship proportional to their prior variances. Our estimate will be accurate if the posterior variance is small. Therefore, we should design the experiment in such a way that the posterior variances are small. Since all aliasing relationships of a 2^{p-k} design are derived from the defining contrast subgroup, we can just focus on the posterior variance of α_0:

$$var\{\alpha_0|\mathbf{y}\} = \xi^2 - \frac{\xi^2}{\sum_{I\in J_0(D)}\prod_{i\in I}\gamma_i + \lambda/n}.$$

Consider the case with $\gamma_1 = \cdots = \gamma_p = \gamma$, which is obtained when the GP has an isotropic correlation function. Then

$$\sum_{i\in J_0(D)}\prod_{i\in I}\gamma_i = 1 + \sum_{i=1}^{p}\gamma^i A_i(D),$$

where $(A_1(D), A_2(D), \ldots, A_p(D))$ is the wordlength pattern of the design D. Thus, minimizing $var\{\alpha_0|\mathbf{y}\}$ is equivalent to minimizing $\sum_{i=1}^{p}\gamma^i A_i(D)$, which is a weighted average of the wordlength pattern with higher weights given to the lower-order effects. This is closely related to the minimum aberration design which is obtained by sequentially minimizing $A_1(D), A_2(D), \ldots, A_p(D)$. In fact, for small γ, it can be shown that a minimum aberration design minimizes $\sum_{i=1}^{p}\gamma^i A_i(D)$. Thus,

$$D^*(\gamma) = \underset{D}{\text{argmin}}\sum_{i=1}^{p}\gamma^i A_i(D)$$

can be viewed as a Bayesian version of the minimum aberration design. A drawback of this approach is that the design can only be obtained after specifying a value for γ. Based on the analysis of several real experiments, Kang and Joseph (2009) recommended $\gamma = 1/3$ as a good choice if no other prior information is available.

Let us return to the analysis of the 2^{4-1} design. We had neglected the correlation of the effects and therefore half-normal plot analysis can go wrong. Joseph (2006a) tried to overcome this issue using a forward variable selection strategy. Yu and Joseph (2025a) recently proposed to select important effects using nonnegative garrote (Breiman, 1995) with heredity constraints (Yuan et al., 2009). The nonnegative garrote requires an initial estimate for the coefficients, which can be chosen as the Bayesian posterior mean. This approach is implemented in the R package `HiGarrote` (Yu and Joseph, 2025b). It gives the model

$$\hat{y} = 805 + 150.1x_4 + 53.2x_3 - 39.1x_3x_4.$$

This is very similar to the model identified earlier in Section 8.1, but has coefficients shrunk toward 0 which is a property of nonnegative garrote.

Now consider the cast-fatigue experiment given in Table 8.2. The traditional main effect analysis identifies only x_6 as important with $R^2 = 0.45$, while HiGarrote identifies $\{x_6, x_6x_7\}$ as highly significant, and $\{x_4, x_7, x_4x_7\}$ as potentially significant giving $R^2 = 0.96$. Although Bayesian analysis looks superior to traditional analysis, a caveat of the approach is that it works well only if the estimation of the correlation parameters is good. This is not an easy task because the likelihood in (2.46) can be multimodal with small data. The package `HiGarrote` uses multiple starts and gradient-based optimization to find the global optimum of (2.46), which is found to work well in practice.

8.3 Designs with More than Two Levels

Experiments with more than two levels are very common, especially with three levels. This is because experiments are usually used to improve a system that is currently operating at a value m for a factor x. The engineer may not be sure whether to decrease or increase the factor. Therefore, the engineer may wish to try taking two levels $m - \Delta$ and $m + \Delta$ for the experiment. However, it is important to include the value m also in the experiment to get a baseline for comparisons with existing conditions. This leads to three levels $\{m - \Delta, m, m + \Delta\}$. In this chapter, we denote the three levels as $\{1, 2, 3\}$ or $\{0, 0.5, 1\}$.

A factor with three levels can be represented using two dummy variables: d_1 and d_2. Their values can be specified using several coding systems, such as treatment coding, Helmert coding, and orthogonal polynomial coding, which are shown below (Faraway, 2015, p.219).

x	d_1	d_2
1	0	0
2	1	0
3	0	1

x	d_1	d_2
1	−1	−1
2	1	−1
3	0	2

x	d_1	d_2
1	−1	1
2	0	−2
3	1	1

Similarly, a factor with four levels can be represented using three dummy variables. The three coding schemes (treatment, Helmert, and orthogonal polynomial) for a four-level factor are shown below.

x	d_1	d_2	d_3
1	0	0	0
2	1	0	0
3	0	1	0
4	0	0	1

x	d_1	d_2	d_3
1	−1	−1	−1
2	1	−1	−1
3	0	−1	−1
4	0	2	3

x	d_1	d_2	d_3
1	−3	1	−1
2	−1	−1	3
3	1	−1	−3
4	3	1	1

Xu and Wu (2001) has generalized the minimum aberration criterion for the optimal selection of a fractional factorial design with different number of levels for each factor, which builds on an earlier work by Tang and Deng (1999) to find minimum aberration designs for two-level nonregular designs. Suppose that the factor x_i has L_i levels, where $L_i = 2, 3, \ldots$. The ith factor can be represented using $L_i - 1$ dummy variables. Now we can write the full ANOVA model as in (8.2) with the dummy variables instead of the x_i's, omitting the interactions between the dummy variables within each factor. Thus, for a given n-run design D, we obtain

$$\begin{aligned}\mathbf{y} &= \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\delta} \\ &= \mathbf{1}\alpha_0 + \mathbf{X}^{(1)}\boldsymbol{\alpha}_1 + \cdots + \mathbf{X}^{(p)}\boldsymbol{\alpha}_p + \boldsymbol{\delta},\end{aligned}$$

where $\mathbf{X}$ is the $n \times \prod_{i=1}^{p} L_i$ full model matrix, $\mathbf{X}^{(1)}$ is the submatrix of $\mathbf{X}$ that contains the columns of main effects, and $\mathbf{X}^{(i)}$, $i = 2, \ldots, p$ are the submatrices corresponding to the i-factor interactions.

Suppose that we use an orthogonal coding scheme to obtain the dummy variables such as Helmert or orthogonal polynomial coding. Scale the dummy variables so that the columns of $\mathbf{X}^{(1)}$ are of length n. Define

$$A_j(D) = \frac{1}{n^2}\|\mathbf{1}'\mathbf{X}^{(j)}\|^2, \; j = 1, \ldots, p.$$

The vector $(A_1(D), \ldots, A_p(D))$ is called the generalized wordlength pattern of the design D. The generalized minimum aberration criterion of Xu and Wu (2001) is to sequentially minimize $A_1(D), \ldots, A_p(D)$. This criterion reduces to the original minimum aberration criterion for two-level regular designs and minimum G_2-aberration criterion of Tang and Deng (1999) for two-level nonregular designs.

Traditionally, experiments with more than two levels are designed using orthogonal arrays (Rao, 1947), which were popularized in industrial experimentation by Taguchi (1987). See Hedayat et al. (1999) for details on orthogonal arrays.

Definition 8.1 *An orthogonal array* $OA(n, L_1^{m_1}, \ldots, L_r^{m_r}, t)$ *of strength* t *is an* $n \times (m_1 + \cdots + m_r)$ *matrix in which* m_i *columns have* L_i *levels such that for any* t *columns, all possible combinations of levels appear equally often in the matrix.*

The two-level regular design in Table 8.1 and the nonregular design in Table 8.2 are all special cases of orthogonal arrays. Xu and Wu (2001) proved the following result.

Theorem 8.3 *A design* D *is an orthogonal array of strength* t *if and only if* $A_1(D) = \cdots = A_t(D) = 0$.

Suppose that an engineer is interested in experimenting with four 3-level factors and has a budget of 18 experiments. Tables 8.3 and 8.4 show two orthogonal arrays $OA(9, 3^4)$ and $OA(18, 2^1 3^7)$ of run sizes 9 and 18, respectively (by convention, the notation on t is omitted for strength-2 orthogonal arrays). These arrays are generated using the function `oa.design` in the R package `DoE.base` (Groemping, 2025). Since the engineer has a budget of 18 runs, one option is to use $OA(9, 3^4)$ with two replications or assign the four factors to any of the four columns 2–8 in $OA(18, 2^1 3^7)$. Which option is better for a precise estimation of the effects? To answer this, we can compute the generalized wordlength pattern of the designs. Using the `GWLP` function in `DoE.base` package, we obtain $W(D_1) = (0, 0, 8, 0)$ for the first option and $W(D_2) = (0, 0, 3.5, 0)$ if we assign columns 2–5 for the four factors in the 18-run orthogonal array. Clearly, the second design is much better. However, the 18-run orthogonal

TABLE 8.3: $OA(9, 3^4)$.

Run	1	2	3	4
1	1	1	1	1
2	1	2	3	2
3	1	3	2	3
4	2	1	3	3
5	2	2	2	1
6	2	3	1	2
7	3	1	2	2
8	3	2	1	3
9	3	3	3	1

TABLE 8.4: $OA(18, 2^1 3^7)$.

Run	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	3	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	2	1
16	2	3	1	3	2	3	1	2
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1

array has seven 3-level columns, and therefore, we can create many four-factor designs. Suppose that we assign the factors to columns 3-6, then the generalized wordlength pattern becomes $W(D_3) = (0, 0, 2, 1.5)$, which, in fact, is the generalized minimum aberration design (Xu and Wu, 2001).

As we have seen in Section 4.5.2, a factor can be qualitative or quantitative. This distinction does not affect two-level designs but affects designs with more than two levels. This is because the order of the levels does not matter for qualitative factors, but the geometric structure of the design can change with different ordering of the levels of quantitative factors (Cheng and Ye, 2004). Consider the 18-run generalized minimum aberration design D_3. Assume that all four factors are continuous. Then, several designs from D_3 can be obtained by reordering the three levels. Since reordering of rows and columns does not change the design, we only need to do the level permutations of the last three columns. This will give us $(3!)^3$ designs. However, half of these designs are geometrically the same because orders $(1, 2, 3)$ and $(3, 2, 1)$ will generate the same geometrically isomorphic designs. Thus, we need to generate only $(3!/2)^3$ designs. To understand the space-filling properties of these designs, we can compute the MaxPro criterion for discrete factors as in (4.42):

$$MP(D) = \left(\frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{k=1}^{p} \left\{ \left[\frac{x_{ik} - x_{jk}}{1/L_k} \right]^2 + 1 \right\}} \right)^{1/p},$$

which can be obtained using the `maxpro.crit` function in the R package `SFDesign` (Wang and Joseph, 2025c). Of the 27 values, only 4 are unique, which are plotted as solid vertical lines in the left panel of Figure 8.3. We can see that the permutation of the levels can affect the space-fillingness of the design. For comparison, we also plotted $MP(D_2)$ and $MP(D_1)$ as dashed and dotted vertical lines in the same figure, which are much larger than the MaxPro measures of the generalized minimum aberration designs. We have also generated one million

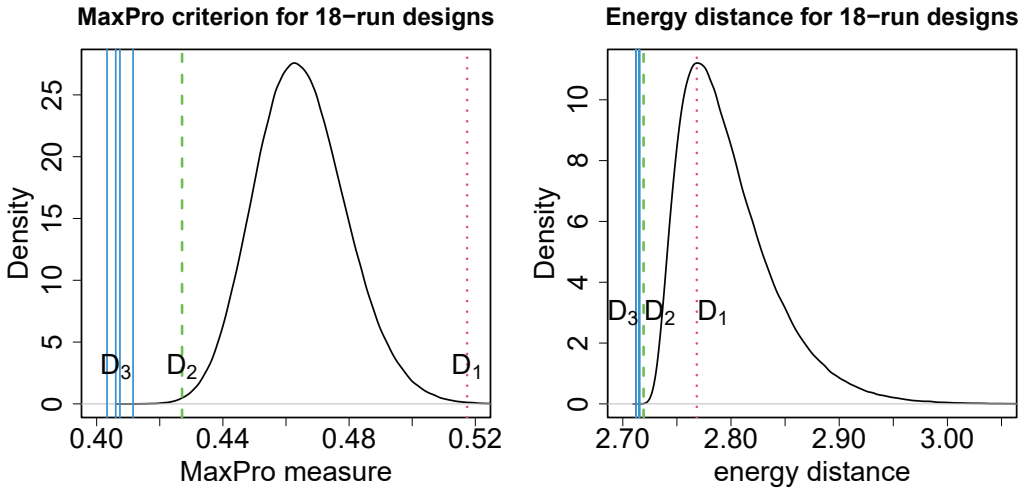


FIGURE 8.3: The density shows the MaxPro (left) and energy distance (right) criteria of one million randomly generated 18-run designs. The criterion values of orthogonal arrays D_1, D_2, and D_3 are shown using dotted, dashed, and solid vertical lines, respectively. The different values obtained using reordering of levels of D_3 are also shown.

18-run designs by sampling 18 rows of a full factorial 3^4 design. Their MaxPro measures are shown as a density in the same figure. We can see that the four unique generalized minimum aberration designs are superior to these one million randomly generated designs. The energy distance (5.10) is also calculated for these designs with respect to the full factorial design and is shown in the right panel of Figure 8.3, which gives the same conclusion.

Interestingly, the design D_1, which is the $OA(9, 3^4)$ with each run replicated twice, is worse than most of the 18-run designs generated randomly. This shows that replications are not useful in fractional factorial experiments, which is somewhat counterintuitive because replication is considered a fundamental principle in experiments (Wu and Hamada, 2021, Sec. 1.3). This could be because we had assumed that the noise variance σ^2 was known and our focus was on accurately estimating the mean model. However, σ^2 is only a single parameter, which can be estimated by replicating a single run. Replications for every run will be useful if we also need to model the variance as a function of the input variables as in the case of robust parameter design experiments (Wu and Hamada, 2021, Ch. 11). Binois et al. (2019) investigate the use of replication in the context of fitting heteroskedastic Gaussian process models.

Zhou and Xu (2014) have made the following theoretical connection between generalized wordlength pattern and space-filling measures.

Theorem 8.4 *Suppose*

$$\phi(D) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \prod_{k=1}^{p} f(x_{ik}, x_{jk})$$

is a space-filling measure, where $f(\cdot, \cdot)$ *satisfies*

$$f(x, y) \geq 0 \text{ and } f(x, x) + f(y, y) > f(x, y) + f(y, x) \text{ for any } x \neq y \in [0, 1].$$

Then for an n*-run* p*-factor design with* L *levels, the average value of* $\phi(D)$ *computed over all level permutations*

$$\bar{\phi}(D) = \left(\frac{c_1(c_2 + L - 1)}{L^2(L-1)} \right)^p \sum_{i=0}^{p} \left(\frac{c_2 - 1}{c_2 + L - 1} \right)^i A_i(D),$$

where $c_1 = \sum_{k=0}^{L-1} \sum_{l \neq k} f(k, l)$, $c_2 = (L-1) \sum_{k=0}^{L-1} f(k, k)/c_1$, *and* $A_0(D) = 1$.

This result shows that sequentially minimizing $A_1(D), A_2(D), \ldots$ will produce designs with good space-filling measures. Thus, to find good space-filling fractional factorial designs, Zhou and Xu (2014) proposed the following two-step procedure:

(i) find the generalized minimum aberration design and

(ii) reorder the levels of the design to improve its space-filling property.

Generalized minimum aberration designs can be obtained from the R package `DoE.base` (Groemping, 2025).

Traditionally, response surface designs (Myers et al., 2016) are commonly used for continuous factors such as the central composite design (CCD) (Box and Wilson, 1951). A small CCD for four factors can be constructed as follows. Start with a fractional factorial design of resolution 3 that does not contain four-letter words known as cube points, and add two axial points for each factor and then add some center points to the design. For an experimental region $[-1, 1]^p$, the two axial points of a factor x_i can be obtained as $(0, \ldots, \pm\alpha, \ldots, 0)$, where $\pm\alpha$ appears at the ith position. Usually, $\alpha = n_c^{1/4}$ is used, where n_c is the number

of cube points. However, such a choice is good only if the experimental region is spherical. For CCDs in cuboidal regions, we can let $\alpha = 1$, which are called face-centered CCDs. The center point is a vector of 0's having length p. Thus, consider a face-centered CCD constructed using a 2^{4-1} design with generator $x_4 = x_1x_2$ and two center points. This CCD has $8+8+2 = 18$ runs. We can obtain its MaxPro measure and energy distance as 0.47 and 2.73, respectively, which are larger than those of D_2 and D_3 shown in Figure 8.3. Thus, the space-filling properties of CCDs appear to be inferior to those of space-filling orthogonal arrays. Morris (2000) proposed a class of three-level designs, which are obtained by augmenting a small two-level fractional factorial design using sequentially generated maximin design points. They are likely to have better space-filling measures than CCDs, however, they may not possess the model robustness of space-filling orthogonal arrays obtained using Zhou and Xu (2014)'s two-step procedure.

Another approach to accommodate quantitative and qualitative factors in the designs is to follow the Bayesian approach introduced in the previous section. Joseph and Delaney (2007) developed priors for the model parameters using (8.7) for factors with more than two levels. Analysis of experiments using these priors and nonnegative garrote can be found in Yu and Joseph (2025a). Joseph et al. (2009) explain how these priors can be used to generate Bayesian-inspired minimum aberration designs for factors with two and four levels. Kang and Joseph (2009) used these priors for the optimal selection of single arrays for robust parameter design experiments.

8.4 Mixture Designs

Many products are formed by mixing or blending several components or ingredients, such as a shampoo or paint. Details about the design and analysis of mixture experiments can be found in the books by Cornell (1990), Lawson (2014), and Myers et al. (2016). Although most of the mixture experiments reported in the literature are physical experiments (Piepel, 2006), we will soon see that the design techniques of Part II are very useful for them.

Let there be p components in the mixture and $x_i \in [0, 1]$ denote the proportion of the ith component. Then, we must have

$$\sum_{i=1}^{p} x_i = 1. \tag{8.11}$$

Therefore, our aim is to design an experiment in the region

$$\mathcal{S} = \{\mathbf{x} \in [0, 1]^p : \sum_{i=1}^{p} x_i = 1.\}$$

Although we have discussed methods to obtain space-filling designs in constrained regions in Sections 4.5.1 and 4.6.2, those methods work well only with inequality constraints. The equality constraint in (8.11) makes $\mathcal{S}$ a $(p-1)$-dimensional manifold in a p-dimensional space, making the candidate generation challenging.

A simplex lattice design (Scheffé, 1958) in p components denoted by $SLD(p, m)$ is constructed by taking $m+1$ equally spaced values

$$\left\{0, \frac{1}{m}, \frac{2}{m}, \ldots, 1\right\}$$

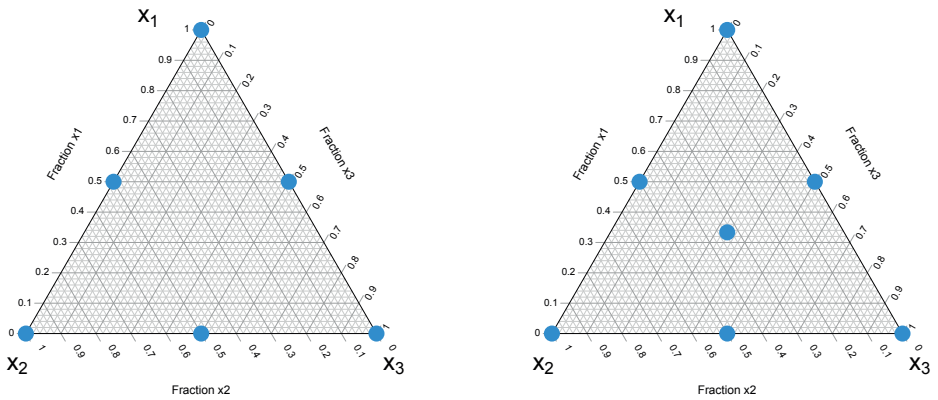


FIGURE 8.4: (left) Simplex lattice design with two levels and (right) simplex centroid design with two levels.

for each factor. We can form a full factorial design with these levels, which contains $(m+1)^p$ points. However, many of these design points will be outside the feasible region $\mathcal{S}$. Therefore, only the points that fall in $\mathcal{S}$ are accepted, which gives us the simplex lattice design. For example, consider an SLD for a three-component mixture with three equally spaced values, that is, $p = 3$ and $m = 2$. To construct an $SLD(3, 2)$, we start with the 3^3 full factorial design. Rejecting all the points that fall outside $\mathcal{S}$ gives us six points:

$$SLD(3, 2): \{(1, 0, 0), (0, 1, 0), (0, 0, 1), (.5, .5, 0), (.5, 0, .5), (0, .5, .5)\}.$$

This is shown in a simplex coordinate system (ternary plot) in the left panel of Figure 8.4. In general, an $SLD(p, m)$ will have

$$\binom{p+m-1}{m}$$

design points. This design can be generated using the R package `mixexp` (Lawson and Willden, 2024).

Scheffe (1963) also proposed another design known as simplex-centroid design. A simplex-centroid design with p components denoted by $SCD(p)$ contains p pure blends, $\binom{p}{2}$ binary mixtures, $\binom{p}{3}$ ternary mixtures, etc., and the overall centroid $(1/p, \ldots, 1/p)$. Thus, there will be a total of $2^p - 1$ design points. Thus, a three-component SCD will have seven points:

$$SCD(3): \{(1, 0, 0), (0, 1, 0), (0, 0, 1), (.5, .5, 0), (.5, 0, .5), (0, .5, .5), (1/3, 1/3, 1/3)\}.$$

and are shown in the right panel of Figure 8.4. This design can also be generated using the `mixexp` package.

For large p, the number of runs in SLD and SCD can be huge and unaffordable. Moreover, the points of the SLD and SCD are mainly in the boundaries and not uniformly distributed within the simplex $\mathcal{S}$. Therefore, Wang and Fang (1990) proposed to use uniform designs on the simplex. Using the inverse probability transform method, they showed that if $\mathbf{u} \in$

$\mathcal{U}(0,1)^{p-1}$, then $\mathbf{x}$ obtained using the transformation (see Fang and Wang (1993, Ch. 5))

$$\begin{aligned} x_1 &= 1 - u_1^{1/(p-1)} \\ x_i &= \prod_{j=1}^{i-1} u_j^{1/(p-j)} \{1 - u_i^{1/(p-i)}\}, \; i = 2, \ldots, p-1 \\ x_p &= \prod_{j=1}^{p-1} u_j^{1/(p-j)} \end{aligned} \tag{8.12}$$

is uniform on the simplex. Therefore, we can generate a uniform design in $[0,1]^p$ using the techniques discussed in Section 5.1 and then use the transformation in (8.12) to obtain a uniform design on $\mathcal{S}$.

However, as pointed out by Borkowski and Piepel (2009), the transformation method may not work well when the components have additional constraints. See the eight constraints in the 16-component glass mixture experiment in Table 8.6. The constraints can become even more complex in more complicated experiments, such as the mixture-of-mixtures experiments (Kang et al., 2011). Therefore, a more flexible approach to developing uniform designs is needed. Ning et al. (2011) proposed to generate candidates in the constrained region using (8.12) and then use clustering-based methods to come up with the final design. However, the clustering step can distort the distribution and the design points need not be uniform (see Exercise 2 in Chapter 5). We can use support points discussed in Section 5.2.2 to obtain a good uniform design.

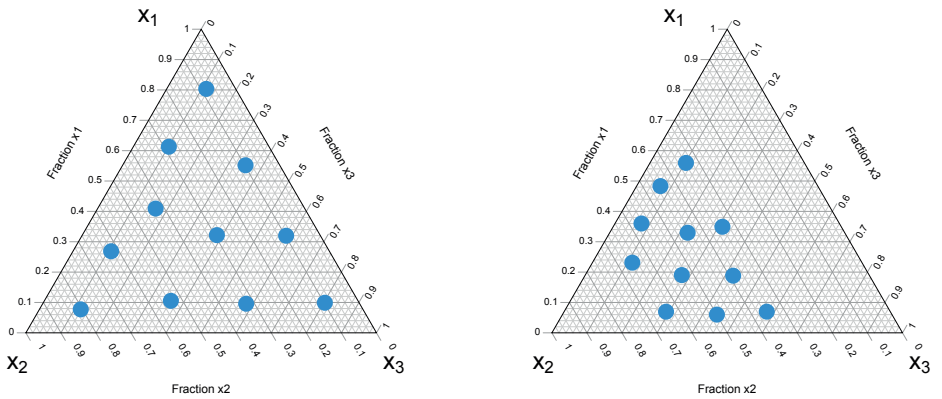


FIGURE 8.5: 11 support points in the simplex without any additional constraints (left) and with constraints (right).

We will explain this approach using a simple example. First, consider a three-component mixture design problem without any constraints in which we want to generate 11 points. We start by generating 10,000 Sobol' points using the R package `spacefillr` (Morgan-Wall, 2025). They are then transformed into uniform points on $\mathcal{S}$ using (8.12). Now we can compute the support points using the R package `support` (Mak, 2018). They are shown on the left panel of Figure 8.5. Now consider the constraints:

$$0.3 \leq x_1 + x_2 \leq 0.7.$$

We can reject the points outside the constrained region, which gives about 4,010 feasible candidate points. The support points obtained from these candidate points are shown on the right panel of Figure 8.5, which looks like a good design.

Exercises

1. Fang and Mukerjee (2000) made a beautiful connection between uniform designs and minimum aberration designs:

   Theorem 8.5 *If the design D is a regular 2^{p-k} fractional factorial design, then*

   $$\mathcal{CL}_2(D) = \left(\frac{13}{12}\right)^p - 2\left(\frac{35}{32}\right)^p + \left(\frac{9}{8}\right)^p \left\{1 + \sum_{i=1}^{p} \frac{A_i(D)}{9^i}\right\},$$

   where $\mathcal{CL}_2$ is the centered L_2 discrepancy in (5.6) and $(A_1(D), A_2(D), \ldots, A_p(D))$ is the wordlength pattern of the design D.

   Consider the two 2^{7-2} designs in (8.5). Calculate the centered L_2 discrepancy of the two designs using the R package `DiceDesign` (Franco et al., 2023) and check the validity of the above result.

2. Suppose that we have p qualitative factors with levels $L_1, \ldots, L_p$. Assume the product correlation structure in (8.8) with

   $$R_j(u_j - v_j) = \begin{cases} 1, & \text{if } u_j = v_j \\ \rho_j, & \text{if } u_j \neq v_j \end{cases}, \; j = 1, \ldots, p.$$

   Show that if we use orthogonal coding for each factor, then (8.7) can be simplified to (Joseph and Delaney, 2007)

   $$\alpha_i \overset{ind.}{\sim} \mathcal{N}\left(0, \xi^2 \prod_{j=1}^{p} \gamma_j^{\delta_{ij}}\right), \quad i = 0, 1, \ldots, \prod_{j=1}^{p} L_j - 1, \tag{8.13}$$

   where

   $$\xi^2 = \tau^2 \prod_{j=1}^{p} \frac{1 + (L_j - 1)\rho_j}{L_j}, \text{ and } \gamma_j = \frac{1 - \rho_j}{1 + (L_j - 1)\rho_j},$$

   and $\delta_{ij} = 1$ if α_i corresponds to an effect involving a dummy variable of the jth factor and 0 otherwise.

3. Consider fitting a GP model $y = h(\mathbf{x}) + \epsilon$, where $h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot))$ and $\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ to replicated data. That is, each run in the design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is replicated m times. Then, show that the predictive equations in (2.43) simplify to (Binois et al., 2018):

   $$\hat{y}(\mathbf{x}) = \mu + \mathbf{r}(\mathbf{x})' \left(\mathbf{R} + \frac{\lambda}{m}\mathbf{I}\right)^{-1} (\bar{\mathbf{y}} - \mu\mathbf{1})$$

   $$s^2(\mathbf{x}) = \tau^2 \left\{1 + \frac{\lambda}{m} - \mathbf{r}(\mathbf{x})' \left(\mathbf{R} + \frac{\lambda}{m}\mathbf{I}\right)^{-1} \mathbf{r}(\mathbf{x})\right\},$$

where $\lambda = \sigma^2/\tau^2$ and $\bar{\mathbf{y}} = (\bar{y}_1, \ldots, \bar{y}_n)'$ is the $n \times 1$ vector of averages. In other words, we can replace the replicated observations in each run with their average, reduce the error variance to σ^2/m, and fit the GP model to the averages. This simplifies the computations from $\mathcal{O}(m^3n^3)$ to $\mathcal{O}(n^3)$. Replications can be used to obtain an estimate of the error variance:

$$\hat{\sigma}^2 = \frac{1}{n(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m} (y_{ij} - \bar{y}_i)^2.$$

Therefore, the GP likelihood should be maximized with respect to μ, τ^2, and $\boldsymbol{\theta}$, where an explicit expression for τ^2 is no longer available.

4. Show that the transformation in (8.12) produces a uniform random variable in the simplex (Fang and Wang, 1993, p. 152).

5. Maruthi and Joseph (1999) reported a case study on improving the yield of printed circuit board (PCB) manufacturing. Eight factors were selected from the inner layer fabrication process and are shown in Table 8.5. The first two factors are qualitative and the remaining six factors quantitative. The eight factors were assigned to the eight columns of $OA(18, 2^1 3^7)$ in the same order as shown in Table 8.4. Compute the MaxProQQ measure of this design. Is there any other assignment possible to improve the measure? Can we further improve it using level permutations of the six continuous factors?

6. Consider the 2^{9-5} pharmaceutical drug experiment reported by Raghavarao and Altan (2003), where the nine factors are labeled A to J, excluding I. The design matrix and data are given in the file `drug.csv`.

   (a) First perform the traditional analysis of the data using half-normal plot and Lenth's method.

   (b) Now perform the analysis using `HiGarrote` and compare the results with the traditional analysis.

7. Suppose that we have six continuous factors and a budget of about 30 runs. The experimenter is contemplating several choices for the design.

   (a) a 2^{6-1} design which has 32 runs

   (b) a 3^{6-3} design which has 27 runs

   (c) a face-centered CCD with 29 runs.

   Which design is better according to the energy distance criterion computed with respect to the full factorial 3^6 design?

TABLE 8.5: Factors and levels for the PCB experiment.

Factors	**Notation**	**1**	**2**	**3**
Preheat	x_1	No	Yes	-
Surface preparation	x_2	Scrub	Pumice	Chemical
Lamination speed	x_3	1.2 mpm	1.5 mpm	1.8 mpm
Lamination pressure	x_4	20 psi	40 psi	60 psi
Lamination temperature	x_5	95°C	105°C	115°C
Exposure energy	x_6	14	17	20
Developer speed	x_7	3 fpm	4 fpm	5 fpm
ORP	x_8	500	530	560

TABLE 8.6: Coefficients for the Constraints in the 16-Component Waste-Glass Mixture experiment.

i	ith Glass Component	Lower Limit A_i	Upper Limit B_i	Coefficients of Eight Multiple-Component Constraints							
				1	2	3	4	5	6	7	8
1	SiO_2	.38	.53		.1485	−.0164	−.0080	−.0155	.0127	−.1369	.0003
2	B_2O_3	.05	.14		−.0222	.0236	.1231	.07598	.1071	.02075	.0046
3	Na_2O	.04	.15		−.0442	.1143	.2310	.2226	.2142	.1060	.0123
4	Fe_2O_3	.08	.14	1	.0391	−.0197	−.0049	−.0154	−.0253	−.1427	−.0082
5	ZrO_2	0	.06	1	.0952	−.0719	−.0161	−.0411	−.0641	−.1521	−.0060
6	MnO	0	.05				.1157	.0825	.1446		−.0033
7	SrO	0	.05							−.0232	
8	Al_2O_3	.04	.085	1	.1866	−.0173	−.0572	−.0636	−.0565	−.1775	−.0089
9	Li_2O	.02	.06		−.2900	.2062	.2443	.1844	.2907	.1549	.0161
10	CdO	.0005	.015								
11	Spike	.0015	.015								−.0120
12	NiO	.001	.01				.13	.1572	.0318		−.0386
13	Tl_2O_3	.0002	.002								
14	Sb_2O_3	.0002	.002								
15	SeO_2	.0002	.002								
16	Other	.042	.042								
			$D \rightarrow$	.180	.0550	.0016					.0336
			$E \rightarrow$	.225	.0780	.0141	.0694	.0777	.0695	−.0526	

8. Jones and Lanzerath (2021) described a 21-run experiment aimed at creating a special resin for car vents designed to prevent moisture accumulation within a car's housing. The experiment is carried out using a definitive screening design (Jones and Nachtsheim, 2011) with nine continuous factors, labeled A through J, excluding I. Data are provided in the file `resin.csv`.

   (a) Analyze the data using traditional methods with the response taken as the logarithm of impurity.

   (b) Analyze the data using `HiGarrote` and compare the results with the traditional analysis.

9. Generate a candidate set of 10,000 points in the simplex $\mathcal{S}$. Starting with $(1, 0, 0)$ generate an additional nine points using sequential maximin design as in (4.20) with rectangular and Euclidean distances. Compare the designs to an $SLD(3, 3)$ visually by plotting them on a ternary plot using the R package `Ternary` (Smith, 2025).

10. Cooley and Piepel (2003) reported a case study on designing a 16-component waste-glass mixture experiment. See also Borkowski and Piepel (2009). These components have the following constraints:

$$0 \leq A_i \leq x_i \leq B_i \leq 1,\ i = 1, \ldots, 16$$

$$D_i \leq \sum_{j=1}^{p} C_{ij} x_j \leq E_i, i = 1, \ldots, 8.$$

    The coefficients are given in Table 8.6. Design a 51-run experiment.

9

Model Calibration

In Part II, we discussed experiments with computer models and in the last chapter, we saw experiments on the physical system. In this chapter, we will see how to design and analyze physical experiments when we also have a computer model that describes the physical system. Even when we have a computer model, we will need physical experiments for several reasons. First, the computer model may contain unknown parameters that need to be specified before a prediction can be made. To make the predictions meaningful, we need to choose the values of those parameters so that the predictions from the computer model match the real observations, which can only be assessed by performing physical experiments. Second, the computer model, which is a numerical implementation of the mathematical model describing the system, can be too simplistic to represent the complex physical system. Therefore, computer model predictions may have to be adjusted based on the data from physical experiments so that the predictions are closer to reality.

9.1 Nonlinear Optimal Designs

Consider the problem of making hybrid membranes for chemical separations mentioned in Chapter 1. Metal particles are infused into a polymer membrane using the vapor phase infiltration (VPI) process as depicted in Figure 1.2. The amount of metal particles trapped in the polymer membrane are governed by the following partial differential equations (Ren et al., 2021)

$$\begin{cases} \frac{\partial C_{\text{free}}}{\partial t} = D\frac{\partial^2 C_{\text{free}}}{\partial s^2} - kC_{\text{free}}C_{\text{polymer}} \\ \frac{\partial C_{\text{product}}}{\partial t} = kC_{\text{free}}\, C_{\text{polymer}} \\ D = D_0 \exp\left(-K' C_{\text{product}}\right) \\ \frac{\partial C_{\text{polymer}}}{\partial t} = -kC_{\text{free}}C_{\text{polymer}} \end{cases} \tag{9.1}$$

with the boundary and initial conditions:

$$\begin{cases} C_{\text{free}} = 0, & 0 < x < l, t = 0 \\ C_{\text{product}} = 0, & 0 < x < l, t = 0 \\ C_{\text{polymer}} = C^0_{\text{polymer}}, & 0 < x < l, t = 0 \\ \frac{\partial C_{\text{free}}}{\partial s} = 0, & x = 0, t > 0 \\ C_{\text{free}} = C_s, & x = l, 0 < t < \tau \\ C_{\text{free}} = 0, & x = l, t > \tau, \end{cases} \tag{9.2}$$

where $C_{free}(\text{mol/cm}^3)$ is the concentration of the free diffusing vapor-phase precursor, $C_{polymer}(\text{mol/cm}^3)$ is the concentration of the accessible reactive polymeric functional

DOI: 10.1201/9781003661535-9

This chapter has been made available under a CC-BY-NC-ND license 4.0.

groups, $C_{product}(\text{mol/cm}^3)$ is the concentration of immobilized product from the reaction between the free diffusing vapor-phase precursor and the polymeric functional groups, and τ is the time when the desorption begins. The first differential equation describes how the free diffusing vapor-phase precursor diffuses into the polymer while being consumed via reaction; the second equation describes the formation of the immobilized product; the third equation captures how the diffusivity decreases exponentially with the formation of the immobilized product; and the last equation describes the consumption of the polymer reactive groups. There are five unknown parameters in the governing equations and boundary conditions: $\boldsymbol{\eta} = \{D_0, C_s, C^0_{polymer}, K', k\}$ that will directly impact the transport process, where $D_0(\text{cm}^2/\text{s})$ is initial diffusivity of the free diffusing vapor-phase precursor, $C_s(\text{mol/cm}^3)$ is the surface concentration of the free diffusing vapor-phase precursor, $C^0_{polymer}(\text{mol/cm}^3)$ is the initial concentration of accessible reactive polymeric functional groups, $K'(\text{cm}^3/\text{mol})$ is the hindering factor that describes how the immobilized product $C_{product}$ slows down the diffusivity of free diffusing vapor, and $k(\text{cm}^3/\text{mol}\cdot\text{s})$ is the associated reaction rate. There are three input variables: polymer thickness l, temperature, and vapor pressure, that can be experimentally controlled. The total mass uptake $y = C_{free} + C_{product}$ can be measured in the physical system and is plotted on the right side of Figure 1.2 as a solid black line at temperature 130°C and 8.7 Torr vapor pressure using a 483 nm thick polymer film.

In general, let

$$y = h(\mathbf{x}; \boldsymbol{\eta}) \tag{9.3}$$

be the computer model, where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ are the input variables and $\boldsymbol{\eta} \in \mathbb{R}^q$ the unknown parameters in the model. The unknown parameters are known as *calibration parameters* because they can be chosen to calibrate the computer model to match the real data. In the VPI process, $\mathbf{x}$ includes the thickness of the membrane l and the time t, and $\boldsymbol{\eta} = \{D_0, C_s, C^0_{polymer}, K', k\}$. Assuming that there is no *experimental bias*, we can model the data from the physical experiment as

$$y = \mu(\mathbf{x}) + \epsilon, \quad \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2), \tag{9.4}$$

where $\mu(\mathbf{x})$ is the true response surface and ϵ is the random error in the measured output. Note that unlike the computer model, this relationship does not contain any calibration parameters because they take some true values in the physical system, say $\boldsymbol{\eta}^*$. The assumption of no experimental bias can sometimes be violated, which could happen due to the instrument bias for measuring the output or due to problems with the experimental setup. Deng et al. (2009) reported an example of estimating the elastic modulus of a nanobelt using a simply supported beam model. Severe experimental bias was observed in their experiment due to initial bending of the nanobelt, occasional stick-slip events that occur at the ends of the nanobelt, wear and tear of the instrument, and lateral shifting and sliding of the nanobelt. These issues are problem-specific, and special care should be taken to identify and remove them. An effective approach to mitigate the effect of experimental biases is to randomize the experiments if their source is unknown and to block if the source is known (Wu and Hamada, 2021, Ch. 1). Statistical modeling can also help identify experimental bias (Chang and Joseph, 2014), and in some cases their identification can lead to new discoveries (Joseph et al., 2024).

If there is no *model bias*, we can let

$$\mu(\mathbf{x}) = h(\mathbf{x}; \boldsymbol{\eta}^*). \tag{9.5}$$

This assumption of no model bias is often violated because it is too difficult to capture the entire physics of a complex process. However, for the moment, we will assume that there is no bias and develop the modeling techniques and revisit this case in the next section. Thus, putting (9.4) and (9.5) together, we obtain

$$y = h(\mathbf{x}; \boldsymbol{\eta}^*) + \epsilon, \quad \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2). \tag{9.6}$$

Suppose that we conducted a physical experiment using a design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and obtained data $\mathbf{y} = (y_1, \ldots, y_n)'$. The calibration parameters can then be estimated using least squares

$$\hat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - h(\mathbf{x}_i; \boldsymbol{\eta})\}^2. \tag{9.7}$$

This appears to be the same as fitting a nonlinear regression model to the data, which is a well-studied problem in statistics (Bates and Watts, 1988). However, there is a crucial difference here: the computer model $h(\mathbf{x}; \boldsymbol{\eta})$ can be expensive to evaluate and therefore the optimization in (9.7) is not easy to perform.

However, we will start by assuming that the computer model is easy to evaluate so that we can invoke the results from nonlinear regression and optimal design theory (Fedorov and Leonov, 2013). We will then extend the results to the case of expensive computer models.

9.1.1 Analytical Models

First consider the case where the computer model is a simple analytical model. As an example, consider the sulfur dioxide (SO_2) mixing process described in Pal and Joseph (1998). In this process, SO_2 is continuously mixed with soft water in a tank to produce the SO_2 solution, which is continuously removed from the tank to bleach the wood pulp. Let x_1 be the flow rate of SO_2 gas and x_2 be the flow rate of soft water. Then, the concentration of SO_2 in the solution at time t is given by

$$h(\mathbf{x}; \eta) = h_0 + \left(\frac{\eta_2 + \eta_3 x_1}{\eta_2 + \eta_3 x_1 + x_2} - h_0 \right) \left\{ 1 - e^{-(\eta_2 + \eta_3 x_1 + x_2)\frac{t}{\eta_1}} \right\}, \tag{9.8}$$

where h_0 is the concentration at time $t = 0$, η_1 is the mass of the solution in the mixing tank, and (η_2, η_3) are instrument calibration parameters for the SO_2 gas flow meter.

The dataset `so2.csv` can be accessed from the book's GitHub site, which contains 30 data points. The nonlinear regression can be performed as follows. Let $\boldsymbol{\eta}^0$ be an initial guess for $\hat{\boldsymbol{\eta}}$. Using a first-order Taylor series approximation:

$$h(\mathbf{x}; \boldsymbol{\eta}) \approx h(\mathbf{x}; \boldsymbol{\eta}^0) + \nabla_\eta h(\mathbf{x}; \boldsymbol{\eta}^0)'(\boldsymbol{\eta} - \boldsymbol{\eta}^0),$$

where $\nabla_\eta h(\mathbf{x}; \boldsymbol{\eta}) = (\partial h/\partial \eta_1, \ldots, \partial h/\partial \eta_q)'$. Let

$$\mathbf{h}(\boldsymbol{\eta}) = \begin{pmatrix} h(\mathbf{x}_1; \boldsymbol{\eta}) \\ \vdots \\ h(\mathbf{x}_n; \boldsymbol{\eta}) \end{pmatrix}$$

and

$$\mathbf{S}(\boldsymbol{\eta}) = \begin{pmatrix} \nabla_\eta h(\mathbf{x}_1; \boldsymbol{\eta})' \\ \vdots \\ \nabla_\eta h(\mathbf{x}_n; \boldsymbol{\eta})' \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial \eta_1} h(\mathbf{x}_1; \boldsymbol{\eta}) & \ldots & \frac{\partial}{\partial \eta_q} h(\mathbf{x}_1; \boldsymbol{\eta}) \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial \eta_1} h(\mathbf{x}_n; \boldsymbol{\eta}) & \ldots & \frac{\partial}{\partial \eta_q} h(\mathbf{x}_n; \boldsymbol{\eta}) \end{pmatrix}.$$

$\mathbf{S}(\boldsymbol{\eta})$ is called a sensitivity matrix (Smith, 2024). Then, the sum of squares of the errors in (9.7) becomes

$$\begin{aligned} SSE &= (\mathbf{y} - \mathbf{h}(\boldsymbol{\eta}))'(\mathbf{y} - \mathbf{h}(\boldsymbol{\eta})) \\ &\approx \left(\mathbf{y} - \mathbf{h}(\boldsymbol{\eta}^0) - \mathbf{S}(\boldsymbol{\eta}^0)(\boldsymbol{\eta} - \boldsymbol{\eta}^0)\right)' \left(\mathbf{y} - \mathbf{h}(\boldsymbol{\eta}^0) - \mathbf{S}(\boldsymbol{\eta}^0)(\boldsymbol{\eta} - \boldsymbol{\eta}^0)\right), \end{aligned}$$

which is quadratic in $\boldsymbol{\eta}$. Differentiating and equating to 0, we obtain

$$\mathbf{S}(\boldsymbol{\eta}^0)'\mathbf{S}(\boldsymbol{\eta}^0)(\boldsymbol{\eta} - \boldsymbol{\eta}^0) = \mathbf{S}(\boldsymbol{\eta}^0)'(\mathbf{y} - \mathbf{h}(\boldsymbol{\eta}^0)).$$

If $\mathbf{S}(\boldsymbol{\eta}^0)'\mathbf{S}(\boldsymbol{\eta}^0)$ is invertible, then we obtain a *unique* solution

$$\boldsymbol{\eta}^1 = \boldsymbol{\eta}^0 + \left\{\mathbf{S}(\boldsymbol{\eta}^0)'\mathbf{S}(\boldsymbol{\eta}^0)\right\}^{-1}\mathbf{S}(\boldsymbol{\eta}^0)'(\mathbf{y} - h(\mathbf{x};\boldsymbol{\eta}^0)),$$

which can be an improvement to the initial solution $\boldsymbol{\eta}^0$. This iteration can be continued until convergence, which is known as the Gauss-Newton algorithm (Bates and Watts, 1988, p. 40).

The invertibility assumption of the $\mathbf{S}(\boldsymbol{\eta})'\mathbf{S}(\boldsymbol{\eta})$ matrix needs further clarification, where $\boldsymbol{\eta}$ is in the neighborhood of $\boldsymbol{\eta}^*$. In linear regression models, the invertibility of the model matrix can be ensured by requiring some conditions for the columns of the design matrix. The situation is a bit more complex with nonlinear regression because the invertibility assumption can fail when the parameters are unidentifiable. This issue does not arise with linear regression models because it is easy to ensure that the parameters in a linear model are identifiable. However, this is not easy to check with nonlinear regression models, especially when the model is a black box (Smith, 2024, Ch.7).

To understand the identifiability issue, consider a simple nonlinear model

$$y = x_1^{\eta_1}x_2^{\eta_2\eta_3}.$$

Clearly, we cannot estimate η_2 and η_3 uniquely because they appear in the model as a product. Now suppose that $x_{i1} = x_{i2} = c$ for $i = 1, \ldots, n$ in the design. Then

$$y_i = x_{i1}^{\eta_1}x_{i2}^{\eta_2\eta_3} = c^{\eta_1+\eta_2\eta_3},\ i = 1, \ldots, n.$$

In this case, we can identify only $\eta_1 + \eta_2\eta_3$ from the data. The former identifiability issue is due to the structure of the model, whereas the latter is due to a poor experimental design in addition to the structural nonidentifiability.

The $\mathbf{S}(\boldsymbol{\eta})'\mathbf{S}(\boldsymbol{\eta})$ matrix is invertible if $rank\{\mathbf{S}(\boldsymbol{\eta})'\mathbf{S}(\boldsymbol{\eta})\} = rank\{\mathbf{S}(\boldsymbol{\eta})\} = q$. Let $\tilde{\boldsymbol{\eta}}$ be a minimizer of SSE. Now suppose that $rank(\mathbf{S}(\tilde{\boldsymbol{\eta}})) = q$, where $\mathbf{S}(\tilde{\boldsymbol{\eta}}) = (\nabla_\eta h(\mathbf{x}_i;\tilde{\eta}_j))_{n\times q}$. Then $\hat{\boldsymbol{\eta}} = \tilde{\boldsymbol{\eta}}$ is the unique (global) minimizer of SSE. If $rank(\mathbf{S}(\tilde{\boldsymbol{\eta}})) < q$, then there exist infinitely many minima. This can happen because of structural nonidentifiability or problems with the experimental design. The structural identifiability of analytical models can be detected as follows. Let $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ be a large uniform sample from $\mathcal{X}$ and $\mathbf{G}(\boldsymbol{\eta}) = (\nabla_\eta h(\mathbf{x}^i;\eta_j))_{N\times q}$. By taking a large sample, we can avoid the confounding effect of experimental design that is used for collecting the data. If

$$rank\{\mathbf{G}(\tilde{\boldsymbol{\eta}})\} < q,$$

then there is structural nonidentifiability. We can overcome structural nonidentifiability by fixing some parameters at their nominal values or through reparametrization. Dadashova et al. (2024) provides some algorithms to do this using the singular value decomposition of $\mathbf{G}(\tilde{\boldsymbol{\eta}})$. On the other hand, if $rank\{\mathbf{G}(\tilde{\boldsymbol{\eta}})\} = q$, then the issue is with the experimental design. In such a case, performing more experiments is an option to overcome the nonidentifiability issue.

Consider the SO_2 mixing model in (9.8), which has three calibration parameters. By minimizing the sum of squares of errors using a derivative-free optimization method, we obtain $\tilde{\boldsymbol{\eta}} = (-1.53, 0.91, 759.04)'$. Now using the `rankMatrix` function in the R package `Matrix` (Bates et al., 2025), we obtain $rank\{\mathbf{S}(\tilde{\boldsymbol{\eta}})\} = 3$. Thus, the three parameters in (9.8) are identifiable and therefore $\hat{\boldsymbol{\eta}} = \tilde{\boldsymbol{\eta}}$ is the unique minimizer of SSE.

Now that we know how to obtain a unique minimizer $\hat{\boldsymbol{\eta}}$ of SSE, we will now turn our attention to quantifying the uncertainties in the estimate. First, we will use a frequentist approach. Under the independence and normality assumption of ϵ in (9.6), we obtain the likelihood of the data as

$$L = \frac{1}{(\sqrt{2\pi})^n \sigma^n} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \{y_i - h(\mathbf{x}_i; \boldsymbol{\eta}^*)\}^2 \right).$$

The log-likelihood is

$$\log L = -\frac{n}{2} - \frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \{y_i - h(\mathbf{x}_i; \boldsymbol{\eta}^*)\}^2.$$

Thus,

$$\hat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta}}{\operatorname{argmax}} \log L = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - h(\mathbf{x}_i; \boldsymbol{\eta})\}^2,$$

which shows that the maximum likelihood estimate (MLE) of $\boldsymbol{\eta}$ is the same as the least squares estimate in (9.7). Differentiating $\log L$ with respect to η_j^*

$$\frac{\partial \log L}{\partial \eta_j^*} = \frac{1}{2\sigma^2} \sum_{i=1}^{n} 2\{y_i - h(\mathbf{x}_i; \boldsymbol{\eta}^*)\} \frac{\partial h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_j^*}.$$

Differentiating once more

$$\frac{\partial^2 \log L}{\partial \eta_k^* \partial \eta_j^*} = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left[\{y_i - h(\mathbf{x}_i; \boldsymbol{\eta}^*)\} \frac{\partial^2 h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_k^* \partial \eta_j^*} - \frac{\partial h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_k^*} \frac{\partial h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_j^*} \right].$$

Since $E\{y_i\} = h(\mathbf{x}_i; \boldsymbol{\eta}^*)$,

$$E\left\{ \frac{\partial^2 \log L}{\partial \eta_k^* \partial \eta_j^*} \right\} = -\frac{1}{\sigma^2} \sum_{i=1}^{n} \frac{\partial h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_k^*} \frac{\partial h(\mathbf{x}_i; \boldsymbol{\eta}^*)}{\partial \eta_j^*}.$$

Thus, the Fisher information matrix (Lehmann and Casella, 2006, p. 125) is given by

$$\mathcal{I}(\boldsymbol{\eta}) = \frac{1}{\sigma^2} \mathbf{S}(\boldsymbol{\eta})' \mathbf{S}(\boldsymbol{\eta}). \tag{9.9}$$

We know that for a maximum likelihood estimator (Lehmann and Casella, 2006, p. 463)

$$\hat{\boldsymbol{\eta}} \Rightarrow \mathcal{N}(\boldsymbol{\eta}^*, \mathcal{I}(\boldsymbol{\eta}^*)^{-1})$$

as $n \to \infty$. Thus,

$$\hat{\boldsymbol{\eta}} \overset{a}{\sim} \mathcal{N}\left(\boldsymbol{\eta}^*, \sigma^2 \{\mathbf{S}(\boldsymbol{\eta}^*)' \mathbf{S}(\boldsymbol{\eta}^*)\}^{-1}\right).$$

This reduces to (2.33) for linear regression models, except that the distribution in (2.33) is exact and not an approximation.

Now, we can use one of the optimal design criteria to construct the experimental design $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The D-optimal criterion is to choose the design to minimize the volume of the confidence ellipsoid of $\boldsymbol{\eta}^*$. As in (2.34), it can be obtained as

$$D^* = \underset{D}{\operatorname{argmax}} |\mathbf{S}(\boldsymbol{\eta}^*)' \mathbf{S}(\boldsymbol{\eta}^*)|.$$

Unfortunately, this optimization can only be obtained if we know the true value $\boldsymbol{\eta}^*$, but the whole purpose of the experimental design was to accurately estimate this unknown parameter! This was not an issue for the linear regression models because the sensitivity matrix was independent of the parameters. This is not the case with nonlinear regression models, which is a major drawback of nonlinear optimal designs. An approach to circumvent this issue is to guess the unknown parameter and design the experiment based on this guessed value. Let $\boldsymbol{\eta}^0$ be our initial guess. Then, we can obtain the *locally* D-optimal design as (Chernoff, 1953)

$$D^* = \underset{D}{\operatorname{argmax}} |\mathbf{S}(\boldsymbol{\eta}^0)'\mathbf{S}(\boldsymbol{\eta}^0)|.$$

We should bear in mind that this design might be good only if the true value is close to $\boldsymbol{\eta}^0$. Therefore, a sequential design strategy might make more sense when dealing with nonlinear regression models (Box and Hunter, 1965).

Since we assume that there is no model bias, there is no need to spread out the design points in the experimental region $\mathcal{X}$. It is very likely that there exist only m distinct optimal points in $\mathcal{X}$, where $m < n$. Therefore, the remaining $n - m$ runs can be allocated to replicate some of the m design points. In the literature on optimal design, these m unique points are called support points of the design, but we will not use this terminology here to avoid confusion with the support points of a distribution introduced in Chapter 5. Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be the m unique optimal design points and $n_1, \ldots, n_m$ be the number of their replications at those points, where $n_1 + \cdots + n_m = n$. Thus, the design can be represented as $D = \{(\mathbf{u}_i, n_i)\}_{i=1}^m$. The D-optimal optimization problem becomes

$$D^* = \underset{D}{\operatorname{argmax}} \left| \sum_{i=1}^m n_i \nabla_\eta h(\mathbf{u}_i; \boldsymbol{\eta}^0) \nabla_\eta h(\mathbf{u}_i; \boldsymbol{\eta}^0)' \right|, \tag{9.10}$$

where $\sum_{i=1}^m n_i = n$. This formulation provides a great simplification to the optimization problem when $m << n$. However, it is a mixed integer programming problem, which is still difficult to solve. The problem can be converted to a continuous optimization problem by relaxing the integral requirements of n_i (Kiefer and Wolfowitz, 1959):

$$D^* = \underset{D}{\operatorname{argmax}} \left| \sum_{i=1}^m w_i \nabla_\eta h(\mathbf{u}_i; \boldsymbol{\eta}^0) \nabla_\eta h(\mathbf{u}_i; \boldsymbol{\eta}^0)' \right| \tag{9.11}$$

where $D = \{(\mathbf{u}_i, w_i)\}_{i=1}^m$, $w_i \in [0, 1]$, and $\sum_{i=1}^m w_i = 1$. This design is called an *approximate design* because in practice we will not be able to perform a fraction of an experiment. The approximate design can be converted to an *exact design* by carefully rounding $nw_1, \ldots, nw_n$ to the nearest integers (Pukelsheim and Rieder, 1992).

To find the optimal design, the first step is to choose the number of unique design points, m. Clearly $m \geq q$ because we need at least q data points to estimate q parameters. If $\mathcal{X}$ is convex, then an application of Carathéodory's theorem gives $m \leq q(q+1)/2$ (Silvey, 1980, p. 16). Recent investigations have shown that the optimal number of unique design points is close to the lower bound q (Yang, 2010; Dette et al., 2022).

Consider a simple example of finding the D-optimal design of the model

$$h(\mathbf{x}; \boldsymbol{\eta}) = \eta_0 + e^{-\eta_1 x_1} + e^{-\eta_2 x_2}. \tag{9.12}$$

Suppose the initial guess of the parameters is $\boldsymbol{\eta}^0 = (0, 2, 2)'$. We start with $m = q = 3$. The optimal design and weights obtained using the R package `ICAOD` (Masoudi et al., 2022) are shown in the next page (left). The D-optimal objective function value is 4.23×10^{-5}.

Now, let $m = 4$. The results are shown in the middle panel. This design has an objective function value of 7.15×10^{-5}, which is an improvement. The right panel shows the results

run	x_1	x_2	w
1	0.45	0.00	0.333
2	0.50	0.50	0.333
3	0.00	0.50	0.333

run	x_1	x_2	w
1	0.50	0.00	0.251
2	0.50	0.50	0.246
3	0.00	0.49	0.251
4	0.00	0.00	0.252

run	x_1	x_2	w
1	0.50	0.00	0.250
2	0.54	0.50	0.002
3	0.00	0.49	0.249
4	0.00	0.00	0.253
5	0.50	0.51	0.246

with $m = 5$, which gives the same value 7.15×10^{-5}. Thus, there is no improvement in the objective function when we increase m to 5. Moreover, we can see that the weight of its second point is close to 0, which essentially makes it a 4-point design. Thus, the design obtained with $m = 4$ can be considered optimal. This agrees with the theoretical analysis of Dette et al. (2022). Suppose that we have a budget for 20 experiments. Then, the number of replications for each run is about 5. In general, special care should be taken for rounding nw_i to obtain the exact design (Pukelsheim and Rieder, 1992). Since the exact design can be viewed as an optimal discrete representation of the approximate design, we can also use the importance support points (Section 5.2.4) to obtain the exact design (Krishna et al., 2022).

Let us now consider a Bayesian approach. The Bayesian model is

$$\begin{aligned} y_i|\boldsymbol{\eta} &\overset{ind.}{\sim} f(y_i|\boldsymbol{\eta}) = \mathcal{N}(h(\mathbf{x}_i;\boldsymbol{\eta}), \sigma^2),\ i = 1, \ldots, n \\ \boldsymbol{\eta} &\sim f(\boldsymbol{\eta}), \end{aligned}$$

where $f(\boldsymbol{\eta})$ is the prior distribution of $\boldsymbol{\eta}$. The prior distribution captures the uncertainty in the parameters before seeing the data. After obtaining the data, the updated uncertainty is captured by the posterior distribution $f(\boldsymbol{\eta}|\mathbf{y})$, where $\mathbf{y} = (y_1, \ldots, y_n)'$. The Shannon information gain from prior to posterior is given by

$$IG = \int f(\boldsymbol{\eta}|\mathbf{y}) \log f(\boldsymbol{\eta}|\mathbf{y}) d\boldsymbol{\eta} - \int f(\boldsymbol{\eta}) \log f(\boldsymbol{\eta}) d\boldsymbol{\eta},$$

which is the same as the reduction in the entropy of $\boldsymbol{\eta}$ (see Section 3.2). Our aim is to design the experiment so that the information gain is maximized (Lindley, 1956). However, IG depends on the data $\mathbf{y}$, which is not known before doing the experiment. Therefore, we can take the expectation of the IG with respect to the marginal distribution of the data to obtain the expected information gain:

$$\begin{aligned} EIG &= \int \left\{ \int f(\boldsymbol{\eta}|\mathbf{y}) \log f(\boldsymbol{\eta}|\mathbf{y}) d\boldsymbol{\eta} \right\} f(\mathbf{y}) d\mathbf{y} - \int f(\boldsymbol{\eta}) \log f(\boldsymbol{\eta}) d\boldsymbol{\eta} \\ &= \int \int f(\boldsymbol{\eta}|\mathbf{y}) f(\mathbf{y}) \log f(\boldsymbol{\eta}|\mathbf{y}) d\boldsymbol{\eta} d\mathbf{y} - \int \int f(\mathbf{y}, \boldsymbol{\eta}) d\mathbf{y} \log f(\boldsymbol{\eta}) d\boldsymbol{\eta}. \end{aligned}$$

It can be simplified as

$$\begin{aligned} EIG &= \int \int \log \frac{f(\boldsymbol{\eta}|\mathbf{y})}{f(\boldsymbol{\eta})} f(\mathbf{y}, \boldsymbol{\eta}) d\boldsymbol{\eta} d\mathbf{y} \\ &= \int \int \log \frac{f(\mathbf{y}|\boldsymbol{\eta})}{f(\mathbf{y})} f(\mathbf{y}, \boldsymbol{\eta}) d\boldsymbol{\eta} d\mathbf{y}. \end{aligned}$$

The Bayesian optimal experimental design can now be obtained by maximizing EIG with respect to $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. However, this is an extremely difficult objective function because of the high-dimensional integrals. The integrals can be approximated using Monte

Carlo (MC) method. Let $(\mathbf{y}^i, \boldsymbol{\eta}^i) \overset{iid}{\sim} f(\mathbf{y}, \boldsymbol{\eta}) = f(\mathbf{y}|\boldsymbol{\eta})f(\boldsymbol{\eta})$, $i = 1, \ldots, N$. Then, EIG can be approximated using

$$EIG \approx \frac{1}{N}\sum_{i=1}^{N} \log \frac{f(\mathbf{y}^i|\boldsymbol{\eta}^i)}{f(\mathbf{y}^i)},$$

where the evidence term can be estimated using

$$f(\mathbf{y}^i) \approx \frac{1}{N}\sum_{j=1}^{N} f(\mathbf{y}^i|\boldsymbol{\eta}^j),\ i = 1, \ldots, N.$$

This double loop makes the computations $\mathcal{O}(N^2)$, making the objective function very expensive and difficult to optimize. Several algorithms are proposed in the literature to make this computation efficient (Huan and Marzouk, 2013; Overstall and Woods, 2017; Rainforth et al., 2024). The optimal design can be obtained using the function `acenlm` in the R package `acebayes` (Overstall et al., 2024), which implements an approximate coordinate exchange algorithm. However, since EIG depends on the noise variance, we need to also specify a prior for σ^2.

Here we will describe an algorithm to optimize EIG using a normal approximation of the posterior. Unlike simulation-based algorithms, it can provide only approximate results, but it is much easier to implement and does not need a prior specification for σ^2. Let $\hat{\boldsymbol{\eta}} = \text{argmax}_{\boldsymbol{\eta}} f(\mathbf{y}|\boldsymbol{\eta})$ be the maximum likelihood estimate and $\hat{\mathcal{I}}(\hat{\boldsymbol{\eta}}) = -\nabla^2_{\eta} \log f(\mathbf{y}|\hat{\boldsymbol{\eta}})$ be the observed Fisher information. Then, the following posterior approximation can be used (Berger, 2013, p. 224)

$$\boldsymbol{\eta}|\mathbf{y} \overset{a}{\sim} \mathcal{N}\left(\hat{\boldsymbol{\eta}}, \hat{\mathcal{I}}(\hat{\boldsymbol{\eta}})^{-1}\right),$$

which can be viewed as a Bayesian version of the asymptotic distribution of maximum likelihood estimate and is closely related to the Laplace approximation widely used in Bayesian computation. Under this approximation, we can obtain

$$\begin{aligned}
\max_D EIG &= \max_D \left\{ \int\int f(\boldsymbol{\eta}|\mathbf{y}) \log f(\boldsymbol{\eta}|\mathbf{y}) d\boldsymbol{\eta} f(\mathbf{y}) d\mathbf{y} - \int f(\boldsymbol{\eta}) \log f(\boldsymbol{\eta}) d\boldsymbol{\eta} \right\} \\
&\equiv \max_D \int\int f(\boldsymbol{\eta}|\mathbf{y}) \log f(\boldsymbol{\eta}|\mathbf{y}) d\boldsymbol{\eta} f(\mathbf{y}) d\mathbf{y} \\
&\approx \max_D \int -\log |\hat{\mathcal{I}}(\hat{\boldsymbol{\eta}})^{-1}| f(\mathbf{y}) d\mathbf{y} \quad \because \ (3.4) \\
&= \max_D \int \log |\hat{\mathcal{I}}(\hat{\boldsymbol{\eta}})| f(\mathbf{y}) d\mathbf{y} \\
&\approx \max_D \int \log |\mathcal{I}(\boldsymbol{\eta})| f(\boldsymbol{\eta}) d\boldsymbol{\eta},
\end{aligned}$$

where the last step is based on an approximation proposed by Chaloner and Larntz (1989). Thus, using (9.9), we can obtain the (pseudo) Bayesian D-optimal design as

$$\begin{aligned}
D^* &= \underset{D}{\text{argmax}} \int \log |\mathbf{S}(\boldsymbol{\eta})' \mathbf{S}(\boldsymbol{\eta})| f(\boldsymbol{\eta}) d\boldsymbol{\eta} \\
&= \underset{\{(\mathbf{u}_i, w_i)\}_{i=1}^m}{\text{argmax}} \int \log \left| \sum_{i=1}^{m} w_i \nabla_{\eta} h(\mathbf{u}_i; \boldsymbol{\eta}) \nabla_{\eta} h(\mathbf{u}_i; \boldsymbol{\eta})' \right| f(\boldsymbol{\eta}) d\boldsymbol{\eta}, \qquad (9.13)
\end{aligned}$$

where we have also used our arguments earlier to convert the exact design to an approximate design.

Now consider the two-factor exponential model with $\eta_0 \sim U(-10, 10)$ and $\eta_i \sim U(1, 3)$ for $i = 1, 2$. The Bayesian D-optimal design for a given m can be obtained using the `bayes` function in the R package `ICAOD` (Masoudi et al., 2022). In this particular example, we obtain the same design as the locally D-optimal design with $\boldsymbol{\eta}^0 = (0, 2, 2)'$, which is expected because (9.11) is a reasonably good approximation of (9.13) if $f(\boldsymbol{\eta})$ is centered around $\boldsymbol{\eta}^0$.

9.1.2 Computationally Expensive Models

When the physics-based model h is computationally expensive such as the reaction-diffusion model in (9.1), optimization in (9.11) or (9.13) can become too time consuming and unaffordable. Therefore, we need to perform a computer experiment and approximate h using a surrogate model before trying to find the optimal design.

Unlike in the physical experiment, we can change both the input variables and the calibration parameters in the computer experiment. Let $\mathbf{z} = (\mathbf{x}', \boldsymbol{\eta}')'$ and $D_1 = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ be the design for the computer experiment. We can use the techniques described in Chapters 3-7 to design this experiment. Let $\mathbf{y} = (y_1, \ldots, y_N)'$ be the output of the experiment. As before, assume $h(\mathbf{z}) \sim GP(\mu, \tau^2 R(\cdot))$. Then, the posterior mean is given by

$$\begin{aligned}\hat{h}(\mathbf{x}; \boldsymbol{\eta}) &= \hat{\mu} + \mathbf{r}(\mathbf{z})' \mathbf{R}^{-1} (\mathbf{y} - \hat{\mu} \mathbf{1}_N) \\ &= \hat{\mu} + \sum_{i=1}^{N} c_i R(\mathbf{z} - \mathbf{z}_i),\end{aligned}$$

where $\mathbf{c} = \mathbf{R}^{-1}(\mathbf{Y} - \hat{\mu}\mathbf{1}_N)$. Under a product correlation structure: $R(\cdot) = \prod_{l=1}^{p} R_l^x(\cdot) \prod_{l=1}^{q} R_l^{\eta}(\cdot)$, the partial derivatives with respect to the calibration parameters can be easily obtained as

$$\begin{aligned}\frac{\partial}{\partial \eta_j} \hat{h}(\mathbf{x}; \boldsymbol{\eta}) &= \sum_{i=1}^{N} c_i \frac{\partial}{\partial \eta_j} R(\mathbf{z} - \mathbf{z}_i), \\ &= \sum_{i=1}^{N} c_i R(\mathbf{z} - \mathbf{z}_i) \frac{\partial}{\partial \eta_j} \log R_j^{\eta}(\eta_j - \eta_{ij}) \; j = 1, \ldots, q.\end{aligned}$$

The partial derivatives can also be computed numerically through finite differencing and Richardson's extrapolation, which is implemented in the R package `numDeriv` (Gilbert and Varadhan, 2019). Now, the optimal design can be generated efficiently using (9.11) or (9.13).

Consider the reaction diffusion model in (9.1). For simplicity, we will see how to optimally design the physical experiment using only the time variable, i.e., $\mathbf{x} = t$. First, a computer experiment needs to be performed to approximate the reaction diffusion model. Huang et al. (2021) use a 50-run MaxPro design on the five calibration parameters $\boldsymbol{\eta} = (\log D_0, C_s, C^0_{polymer}, K', \log k)'$ for this purpose, where $D_0 \in [10^{-12}, 10^{-9}]$, $C_s \in [4 \times 10^{-3}, 5 \times 10^{-3}]$, $C^0_{polymer} \in [5 \times 10^3, 6 \times 10^{-3}]$, $K' \in [500, 2500]$, and $k \in [10^{-3}, 10]$. Solving (9.1) for each run produces a functional response of mass uptake with respect to time. The outputs are shown in the left panel of Figure 9.1, where $\tau = 62,500$ seconds. The design and data are given in https://github.com/BillHuang01/BOFO, which can also be accessed from the book's GitHub site. There are 6,601 time values for each functional output. Thus, the total number of data points is $50 \times 6601 = 330,050$. It is computationally demanding to fit a GP model on such big data. In Part IV we will discuss techniques for

data compression and ways to fit approximate GP models with big data. However, here we can use a more efficient technique due to the functional nature of the output (Hung et al., 2015).

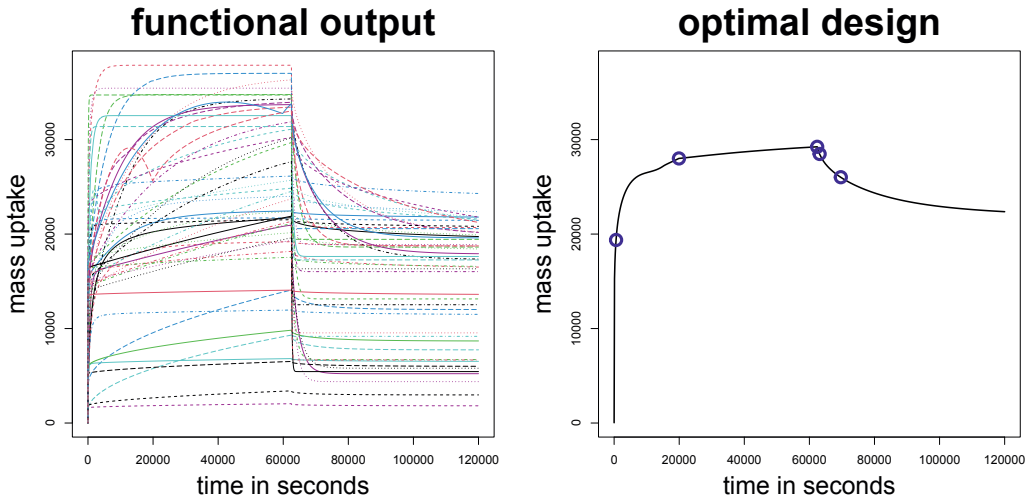


FIGURE 9.1: (left) Outputs from the reaction diffusion model for the 50 MaxPro runs. (right) Output for $\boldsymbol{\eta} = \boldsymbol{\eta}^0$ and the locally D-optimal design (blue circles).

Let $N = 50$, $M = 6601$, and $\mathbf{Y}$ be the $N \times M$ output matrix. The main issue for fitting a GP model is due to the size of the correlation matrix $\mathbf{R}$ which is $NM \times NM$. Using Kronecker products, we can reduce computational complexity from $\mathcal{O}(M^3N^3)$ to $\mathcal{O}(M^3 + N^3)$, which is a substantial savings. The following results from matrix algebra are useful (Petersen and Pedersen, 2006, p. 60), some of which we have already used in Chapter 8.

Lemma 9.1 *Let* $\mathbf{A}$ *be an* $N \times N$ *matrix,* $\mathbf{B}$ *an* $M \times M$ *matrix, and* $\mathbf{Y}$ *an* $N \times M$ *matrix. Define the Kronecker product and the vec operator as*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & \cdots & A_{1N}\mathbf{B} \\ \vdots & \vdots & \vdots \\ A_{N1}\mathbf{B} & \cdots & A_{NN}\mathbf{B} \end{bmatrix} \quad \textit{and} \quad vec(\mathbf{Y}) = \begin{pmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_M \end{pmatrix},$$

where $\mathbf{Y}_i$ *is the* i*th column of* $\mathbf{Y}$*. Then,*

1. $(\mathbf{A} \otimes \mathbf{B})' = \mathbf{A}' \otimes \mathbf{B}'$.
2. $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$.
3. $|\mathbf{A} \otimes \mathbf{B}| = |\mathbf{A}|^M |\mathbf{B}|^N$.
4. $(\mathbf{B}' \otimes \mathbf{A})vec(\mathbf{Y}) = vec(\mathbf{AYB})$.

Let $\mathbf{R}_\eta$ be the $N \times N$ correlation matrix corresponding to the calibration parameters and $\mathbf{R}_t$ be the $M \times M$ correlation matrix corresponding to the time variable. Then, under the

product correlation structure, $\mathbf{R} = \mathbf{R}_t \otimes \mathbf{R}_\eta$, where $\mathbf{R}$ is the correlation matrix of $\text{vec}(\mathbf{Y})$. Now, as described in Chapter 2, we can estimate the GP model parameters as

$$\begin{aligned}\hat{\mu} &= \frac{\mathbf{1}'_{NM}\mathbf{R}^{-1}\text{vec}(\mathbf{Y})}{\mathbf{1}'_{NM}\mathbf{R}^{-1}\text{vec}(\mathbf{1}_N\mathbf{1}'_M)} \\ &= \frac{\mathbf{1}'_{NM}(\mathbf{R}_t^{-1}\otimes\mathbf{R}_\eta^{-1})\text{vec}(\mathbf{Y})}{\mathbf{1}'_{NM}(\mathbf{R}_t^{-1}\otimes\mathbf{R}_\eta^{-1})\text{vec}(\mathbf{1}_N\mathbf{1}'_M)} \\ &= \frac{\mathbf{1}'_{NM}\text{vec}(\mathbf{R}_\eta^{-1}\mathbf{Y}\mathbf{R}_t^{-1})}{\mathbf{1}'_{NM}\text{vec}(\mathbf{R}_\eta^{-1}\mathbf{1}_N\mathbf{1}'_M\mathbf{R}_t^{-1})},\end{aligned}$$

$$\begin{aligned}\hat{\tau}^2 &= \frac{1}{NM}(\text{vec}(\mathbf{Y}) - \hat{\mu}\mathbf{1}_{NM})'(\mathbf{R}_t^{-1}\otimes\mathbf{R}_\eta^{-1})(\text{vec}(\mathbf{Y}) - \hat{\mu}\mathbf{1}_{NM}) \\ &= \frac{1}{NM}(\text{vec}(\mathbf{Y}) - \hat{\mu}\mathbf{1}_{NM})'\text{vec}\{\mathbf{R}_\eta^{-1}(\mathbf{Y} - \hat{\mu}\mathbf{1}_N\mathbf{1}'_M)\mathbf{R}_t^{-1}\},\end{aligned}$$

and the correlation parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}'_\eta, \theta_t)'$ as

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{NM\log\tau^2 + \log|\mathbf{R}|\right\} \\ &= \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{NM\log\tau^2 + M\log|\mathbf{R}_\eta| + N\log|\mathbf{R}_t|\right\}.\end{aligned}$$

The predictor can be simplified to

$$\begin{aligned}\hat{h}(t;\boldsymbol{\eta}) &= \hat{\mu} + (\mathbf{r}_t(t)\otimes\mathbf{r}_\eta(\boldsymbol{\eta}))'(\mathbf{R}_t^{-1}\otimes\mathbf{R}_\eta^{-1})(\text{vec}(\mathbf{Y}) - \hat{\mu}\mathbf{1}_{NM}) \\ &= \hat{\mu} + (\mathbf{r}_t(t)'\otimes\mathbf{r}_\eta(\boldsymbol{\eta})')\text{vec}\{\mathbf{R}_\eta^{-1}(\mathbf{Y} - \hat{\mu}\mathbf{1}_N\mathbf{1}'_M)\mathbf{R}_t^{-1}\} \\ &= \hat{\mu} + \mathbf{r}_\eta(\boldsymbol{\eta})'\mathbf{R}_\eta^{-1}(\mathbf{Y} - \hat{\mu}\mathbf{1}_N\mathbf{1}'_M)\mathbf{R}_t^{-1}\mathbf{r}_t(t). \qquad (9.14)\end{aligned}$$

Although the computations have been substantially reduced, it can still be demanding when M is large (here $M = 6,601$). If the functional output is observed at equally spaced time points, then this computation can be further simplified using an exponential correlation function (Hung et al., 2015):

$$R_t(t_i - t_j) = \exp\left\{-\frac{|t_i - t_j|}{\theta_t}\right\}.$$

Let the M time points be $\{0, 1/(M-1), \ldots, 1\}$ and let $\rho = \exp\{-1/((M-1)\theta_t)\}$. Then

$$\mathbf{R}_t = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{M-1} \\ \rho & 1 & \rho & \cdots & \rho^{M-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho^{M-1} & \rho^{M-2} & \rho^{M-3} & \cdots & 1 \end{bmatrix}.$$

For this matrix, the inverse and determinant can be computed explicitly:

$$\mathbf{R}_t^{-1} = \begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{and} \quad |\mathbf{R}_t| = (1-\rho^2)^M.$$

This reduces the computational complexity from $\mathcal{O}(N^3 + M^3)$ to $\mathcal{O}(N^3 + MN^2)$.

Unfortunately, the 6,601 time points are not equally spaced because more values are taken in the interval $[0, 5000]$ to properly capture the rapid change in the response. To simplify the calculations, we sampled 1,001 equally spaced points from $[0, 120000]$ and obtained the corresponding responses for each run using a nearest-neighbor predictor. Now we can use the explicit expression of $\mathbf{R}_t^{-1}$ in (9.14) and predict the output efficiently.

Since we have five calibration parameters to estimate, a five-point design is likely to be optimal. Let $\boldsymbol{\eta}^0 = (0.5, \ldots, 0.5)'$ be the initial guess, where the experimental region is scaled in $[0, 1]^5$. Then, the locally D-optimal design can be obtained from (9.11) using the `ICAOD` package (Masoudi et al., 2022). The design points are shown on the right panel of Figure 9.1 on top of the response generated with $\boldsymbol{\eta} = \boldsymbol{\eta}^0$. For this design $w_i \approx 0.2$ for $i = 1, \ldots, 5$. We can see that the location of the design points makes intuitive sense and seems to be capable of capturing some of the salient features of the functional output. However, in this particular application, the engineers were able to measure the mass uptake for the entire time domain. In other applications, careful choice of time points will be useful. For example, in the SO_2 mixing process, a sample must be collected, taken to the laboratory, and the concentration measured using titration, which is labor intensive and time consuming. Therefore, the concentration of SO_2 can only be observed at a selected few time points.

The calibration parameters can be estimated using

$$\hat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - \hat{h}(t_i; \boldsymbol{\eta})\}^2,$$

where $\hat{h}(t; \boldsymbol{\eta})$ is given in (9.14) and $\mathbf{y} = (y_1, \ldots, y_n)'$ is the output from the physical experiment. However, since $\hat{h}(t; \boldsymbol{\eta})$ is obtained using only 50 runs, the function approximation may not be accurate. Huang et al. (2021) use Bayesian optimization to find $\hat{\eta}$ by sequentially adding more runs to the design. Unlike the techniques described in Section 7.2, they use the Gaussian process for $h(t; \boldsymbol{\eta})$ and not for the sum of squares of errors. In fact, they develop expected improvement algorithm to efficiently minimize $\int \{y(t) - h(t; \eta)\}^2 dt$ using functional principal component analysis. See Huang et al. (2025) for another real application of this approach.

9.2 Robust Designs

We had to make an initial guess of the unknown calibration parameters to find the optimal design. If our guess is wrong, the resulting design can be suboptimal. We also assumed that the physics-based model is the true model, which is rarely the case in practice. Therefore, the optimal design that we found by assuming the model to be true can be far from optimal. In this section, we will see how to obtain robust designs that take into account the uncertainties of the parameter and the model.

9.2.1 Parameter Uncertainties

Consider the two-factor exponential model in (9.12). Dette et al. (2022) show that the D-optimal design is given by

$$D = \begin{bmatrix} 0 & 0 \\ 1/\eta_1 & 0 \\ 0 & 1/\eta_2 \\ 1/\eta_1 & 1/\eta_2 \end{bmatrix}$$

with $w_i = 1/4$ for $i = 1, \ldots, 4$. In the previous section, we had generated the locally D-optimal design assuming $\boldsymbol{\eta} = (0, 2, 2)'$. Clearly, the choice of η_0 does not affect the result, but the choice of η_1 and η_2 is critical.

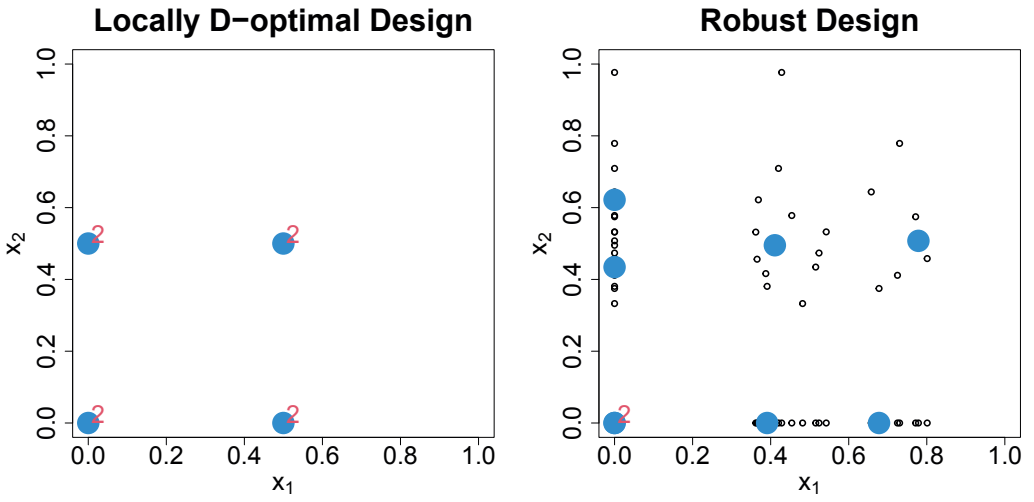


FIGURE 9.2: (left) The 8-run locally D-optimal design for the two-factor exponential model, where each point is replicated twice. (right) An 8-run design that is robust to the parameter uncertainties.

Suppose that we have a budget of eight runs. Then, the locally D-optimal design with initial guess $\boldsymbol{\eta}^0 = (0, 2, 2)'$ will have four points each replicated twice, as shown in the left panel of Figure 9.2. Now suppose that the following distributions capture our uncertainty of the parameters: $\eta_1 \sim \mathcal{U}(1, 3)$ and $\eta_2 \sim \mathcal{N}(2, 1/2^2)$. We can obtain N samples from their uncertainty distributions and obtain the corresponding locally D-optimal designs. Since finding a locally D-optimal design in general can be computationally cumbersome, we can use support points in (5.11) to obtain a small number of good representative samples to reduce computational burden. The N locally D-optimal design points are shown as black circles in the right panel of Figure 9.2. We can see that they vary quite a lot, showing that the locally D-optimal design generated with $\boldsymbol{\eta}^0 = (0, 2, 2)'$ can be poor.

How can we find optimal designs robust to the misspecifications of the parameters? Dror and Steinberg (2006) proposed a simple idea to find the robust design by clustering the $4N$ points using `kmeans`. In this case, all the points have equal weight and therefore the clustering would work. However, in general, each point in the locally D-optimal design can have a different weight. Moreover, clustering can distort the underlying distribution and may not produce good representative points of the distribution. Therefore, Krishna et al. (2022) proposed to use the importance support points (Huang et al., 2022) in (5.17) to find the robust design with weights adjusted using $\mathbf{w} \leftarrow \mathbf{w}/N$. The eight points thus obtained are shown in the right panel of Figure 9.2, which can be considered robust to parameter uncertainties.

9.2.2 Model Uncertainties

As George Box famously said "All models are wrong, but some are useful", it is very likely that the model $h(\mathbf{x}; \boldsymbol{\eta}^*)$ is not the true model $\mu(\mathbf{x})$. This is because several assumptions

are made to simplify the modeling and make it mathematically tractable. Some of these assumptions may not hold true in practice and therefore, the resulting model may not agree with the reality even with the best possible choice of the calibration parameters. Take, for example, the vapor phase infiltration process. The measured mass uptake at 8.7 torr vapor pressure and 130^oC temperature on a 483 nm thick polymer film shown in Figure 1.2 is reproduced in Figure 9.3. The prediction from the reaction-diffusion model with parameters estimated using the Bayesian functional optimization method of Huang et al. (2021) is also plotted in the same figure. Although the predictions generally agree with the measurements, significant departure can also be observed, especially during the initial part of the sorption.

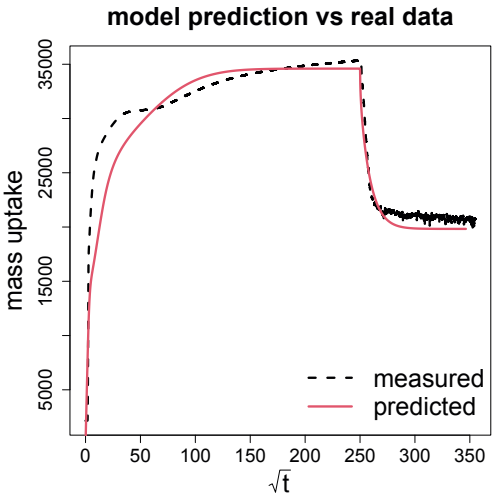


FIGURE 9.3: The measured mass uptake from the physical experiment and the calibrated reaction-diffusion model prediction.

Ren et al. (2021) point out several possible reasons for the model discrepancy. A key assumption used for deriving the reaction-diffusion model is that the swelling and relaxation of the polymer can be ignored. However, *in situ* spectroscopic ellipsometry studies performed clearly show thickness variations during the infiltration process. Nevertheless, this effect is not easy to model mathematically and thus is ignored. Another modeling assumption is that the heat generated during the sorption can be ignored, which can again be violated in practice.

One can investigate the possible reasons for the model discrepancy, fix the wrong assumptions, and try to develop a better physics-based model. This is the usual engineering approach, but it can be too time-consuming. A simpler option is to fit a statistical model to the observed discrepancy. Kennedy and O'Hagan (2001) proposed to model the discrepancy using a Gaussian process:

$$\mu(\mathbf{x}) = h(\mathbf{x}; \boldsymbol{\eta}^*) + \delta(\mathbf{x}),\ \delta(\mathbf{x}) \sim GP(0, \tau^2 R(\cdot)).$$

Thus, the data from the physical experiment can be modeled using

$$y = h(\mathbf{x}; \boldsymbol{\eta}) + \delta(\mathbf{x}) + \epsilon,\ \delta(\mathbf{x}) \sim GP(0, \tau^2 R(\cdot)),\ \ \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2). \tag{9.15}$$

Using this model, the calibration parameters and the discrepancy can be estimated simultaneously from the data. Note that this model contains two sources of uncertainty in the data:

$\delta(\mathbf{x})$ captures the uncertainties introduced due to the missing physics and ϵ captures the unknown random effects. In the uncertainty quantification literature (Der Kiureghian and Ditlevsen, 2009), the former is called *epistemic* and the latter *aleatoric*. Epistemic uncertainty can be reduced by adding more physics into the model or collecting more data, where as aleatoric uncertainties are irreducible unless we can find the sources of the randomness (Joseph et al., 2024).

Although the predictions from the Kennedy-O'Hagan model are good, Tuo and Wu (2015, 2016) showed that the estimate of calibration parameters can be poor due to the non-identifiability between $\boldsymbol{\eta}$ and $\delta(\mathbf{x})$. Plumlee (2017) proposed to use an orthogonal Gaussian process (Plumlee and Joseph, 2018) to overcome the identifiability issue. Other approaches to deal with the identifiability issue in model calibration are proposed in Gu and Wang (2018), Tuo (2019), and Zhang and Wang (2025). See Sung and Tuo (2024) for a recent review. A much simpler approach is to use a two-stage procedure for model fitting: first, estimate the calibration parameters using least squares, and then fit a Gaussian process to the residuals (Joseph and Melkote, 2009; Wong et al., 2017). To make this approach clear, first consider the case of analytical models. Then, the least squares estimate of $\boldsymbol{\eta}$ can be obtained as in (9.7). The residuals are given by

$$e_i = y_i - h(\mathbf{x}_i; \hat{\boldsymbol{\eta}}),\ i = 1, \ldots, n.$$

We can now fit a Gaussian process regression with zero mean to the data $\{(\mathbf{x}_i, e_i)\}_{i=1}^n$ as described in Section 2.2.3, which gives

$$\begin{aligned} \hat{\tau}^2 &= \frac{1}{n}\mathbf{e}'(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{e}, \\ (\hat{\lambda}, \hat{\boldsymbol{\theta}}) &= \underset{\lambda, \boldsymbol{\theta}}{\operatorname{argmin}} \left\{ n \log \hat{\tau}^2 + \log |\mathbf{R} + \lambda\mathbf{I}| \right\}, \end{aligned}$$

where $\lambda = \sigma^2/\tau^2$. Then, the prediction at a new location $\mathbf{x}$ is given by

$$\mu(\mathbf{x})|\mathbf{y} \sim \mathcal{N}(\hat{y}(\mathbf{x}), s^2(\mathbf{x})),$$

where

$$\begin{aligned} \hat{y}(\mathbf{x}) &= h(\mathbf{x}; \hat{\boldsymbol{\eta}}) + \mathbf{r}(\mathbf{x})'(\mathbf{R} + \hat{\lambda}\mathbf{I})^{-1}(\mathbf{y} - \mathbf{h}(\hat{\boldsymbol{\eta}})) \\ s^2(\mathbf{x}) &= \hat{\tau}^2 \left\{ 1 + \hat{\lambda} - \mathbf{r}(\mathbf{x})'(\mathbf{R} + \hat{\lambda}\mathbf{I})^{-1}\mathbf{r}(\mathbf{x}) \right\}. \end{aligned}$$

Although simple, this approach ignores the uncertainties in the estimation of $\boldsymbol{\eta}$. However, it is a less serious issue than obtaining a wrong estimate of $\boldsymbol{\eta}$ with a nonidentifiable model.

If h is a computationally expensive model, we can first obtain a surrogate model $\widehat{h}(\mathbf{x}; \boldsymbol{\eta})$ and proceed as before with the two-stage procedure. However, such an approach will ignore the uncertainties in the surrogate model predictions. Kennedy and O'Hagan (2001) explains how to incorporate them in the final prediction using a Gaussian process for h. Chang and Joseph (2014) and Joseph and Yan (2015) propose an approximate way to account for these uncertainties which can be seamlessly integrated into the two-stage procedure.

Let us return to the experimental design problem. Motivated by the two-stage procedure, Krishna et al. (2022) proposed a simple idea to find robust designs against model uncertainties. In the previous section, we have already learned about finding robust designs for estimating calibration parameters. Denote this design by D_η. In Chapter 4, we learned about space-filling designs for fitting a GP model, which are robust to misspecifications of the correlation parameters. Suppose that we have a budget of n runs, of which we have used n_1 runs in D_η. Then we can find a space-filling design in $n - n_1$ runs to *augment* the design D_η. We will illustrate this approach using an example.

Consider again the two-factor exponential model in (9.12). Suppose that we have a budget of eight runs. Figure 9.4 shows four runs (solid blue circles) obtained using importance support points from locally D-optimal designs generated using samples from $\eta_1 \sim \mathcal{U}(1,3)$ and $\eta_2 \sim \mathcal{N}(2, 1/2^2)$. The remaining four runs can be obtained using sequential MaxPro in (4.27), which can be easily generated using the `MaxProAugment` function in the `MaxPro` package (Ba and Joseph, 2018). They are shown as red crosses in Figure 9.4. This design can be considered robust to both parameter and model uncertainties.

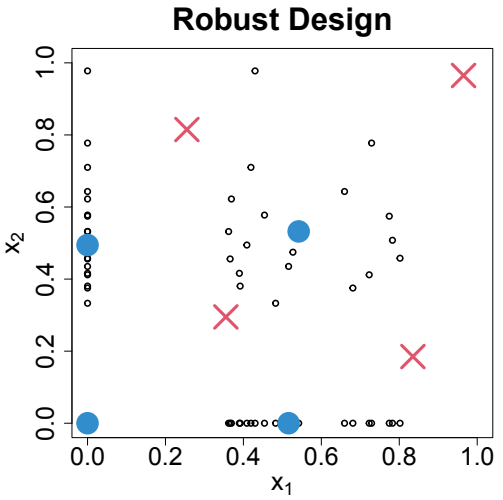


FIGURE 9.4: An eight-run robust design for the two-factor exponential model. The solid blue circles are the points obtained using importance support points on the locally D-optimal designs, which are robust to parameter uncertainties. The red crosses are the points obtained using sequential MaxPro, which protect against model uncertainties.

Sequential design is an attractive option in model calibration. See Williams et al. (2011), Ranjan et al. (2011), Castillo et al. (2019), and Sürer et al. (2024) for details.

Exercises

1. Suppose that the model for predicting the SO_2 concentration in the mixing process also includes calibration parameters for the soft water flow meter:

$$h(\mathbf{x};\eta) = h_0 + \left(\frac{\eta_2 + \eta_3 x_1}{\eta_2 + \eta_3 x_1 + \eta_4 + \eta_5 x_2} - h_0\right)\left\{1 - e^{-(\eta_2+\eta_3 x_1+\eta_4+\eta_5 x_2)\frac{t}{\eta_1}}\right\}.$$

   Check the structural identifiability of this model.

2. Suppose that the data is binary:

$$y_i \overset{ind.}{\sim} Bernoulli(\pi(x_i)),\ i = 1, \ldots, n,$$

where the probability $\pi(x)$ is related to the input variable by

$$\pi(x) = \frac{e^{\frac{x-\mu}{\sigma}}}{1 + e^{\frac{x-\mu}{\sigma}}}.$$

This is called a logistic regression.

(a) Let $\boldsymbol{\theta} = (\mu, \sigma)'$. Show that the Fisher information of $\boldsymbol{\theta}$ per observation is given by

$$\frac{\pi(x)\{1-\pi(x)\}}{\sigma^2} \begin{bmatrix} 1 & \frac{x-\mu}{\sigma} \\ \frac{x-\mu}{\sigma} & \frac{(x-\mu)^2}{\sigma^2} \end{bmatrix}.$$

(b) Consider a two-point symmetric design $x_1 = \mu - \kappa\sigma$ and $x_2 = \mu + \kappa\sigma$. Show that the determinant of the Fisher information matrix is given by

$$\begin{aligned} |\mathcal{I}(\boldsymbol{\theta})| &= \frac{4\kappa^2}{\sigma^4} \frac{e^{-2\kappa}}{(1+e^{-\kappa})^4} \\ &= \frac{4}{\sigma^4} \left\{ q(1-q) \log \frac{q}{1-q} \right\}^2, \end{aligned} \tag{9.16}$$

where $q = \pi(x_1)$.

(c) Show that (9.16) is maximized at $q^* = 0.176$ and therefore, the two-point D-optimal design is given by

$$\{\mu \pm F^{-1}(q^*)\sigma\},$$

where $F(z) = e^z/(1+e^z)$ is the standard logistic distribution function. Sitter and Wu (1993) show that this is indeed the D-optimal design for logistic regression, where the two points have equal weights.

3. Consider a probit regression with a single predictor, where the probability of $y = 1$ is modeled using

$$\pi(x) = \Phi\left(\frac{x-\mu}{\sigma}\right),$$

where $\Phi(\cdot)$ is the standard normal distribution function. Following the steps in the previous exercise, show that the two-point D-optimal design is given by (Sitter and Wu, 1993)

$$\{\mu \pm \Phi^{-1}(q^*)\sigma\},$$

where $q^* = 0.13$.

4. Consider the nonlinear regression model $y = h(\mathbf{x}; \eta) + \epsilon$, $\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$, where

$$h(\mathbf{x}; \eta) = \exp\{-\eta(x_1 - 1.5x_2)^2\} + \exp\{-2\eta(x_1 + x_2 - 0.7)^2\},\ \mathbf{x} \in [0,1]^2.$$

Find the one-point locally D-optimal design analytically for $\eta^0 = 0.5$.

5. Zhu et al. (2014) proposed several models to characterize nanowire growth synthesized using a vapor-liquid-solid process. One of the models considered is an exponential growth model given by

$$h(t; \boldsymbol{\eta}) = \eta_1 \exp\left(-\frac{\eta_2}{t}\right),$$

where $t \in [t_{min}, t_{max}]$ is the time. The weight of the nanowire can be measured at different time points and can be modeled as $y = h(t; \boldsymbol{\eta}) + \epsilon$, where $\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. They

have shown that the D-optimal design for estimating the parameters $\boldsymbol{\eta} = (\eta_1, \eta_2)$ is a balanced two-point design given by

$$\left(\max \left\{ \frac{\eta_2 t_{max}}{\eta_2 + t_{max}}, t_{min} \right\}, t_{max} \right).$$

Use the `ICAOD` package (Masoudi et al., 2022) to numerically obtain a two-point D optimal design with $\eta_1^0 = 10$, $\eta_2^0 = 1.2$, $t_{min} = 0.5$ and $t_{max} = 40$ and compare it with the theoretical result of Zhu et al. (2014).

6. Another model considered by Zhu et al. (2014) to characterize nanowire growth is the exponential-linear growth model with a change point given by

$$h(t; \boldsymbol{\eta}) = \begin{cases} \eta_1 \exp\left(-\frac{\eta_2}{t}\right), & t_{min} \le t < \eta_3 \\ a + bt, & \eta_3 \le t \le t_{max} \end{cases},$$

where $a = \eta_1(1 - \eta_2/\eta_3)\exp(-\eta_2/\eta_3)$ and $b = \eta_1\eta_2/\eta_3^2 \exp(-\eta_2/\eta_3)$. Find a three-point locally D-optimal design using the R package `ICAOD` with $\eta_1^0 = 10$, $\eta_2^0 = 1.2$, $\eta_3^0 = 5$, $t_{min} = 0.5$ and $t_{max} = 40$, and compare it to the analytical solution obtained by Zhu et al. (2014):

$$\left(\max \left\{ \frac{\eta_2 t_{max}}{\eta_2 + t_{max}}, t_{min} \right\}, \eta_3, t_{max} \right).$$

7. In the fluidized-bed experiment (Reese et al., 2004), the average outlet air temperature (the steady-state response of interest), T_{2exp}, is also measured and is given in the file `fluid data.txt`. See the Exercise in Chapter 6 for details. Use the third computer model (output T_{23}) and the experimental data to build a predictive model for the average outlet air temperature using Kennedy-O'Hagan Bayesian model calibration method. Show a scatter plot of T_{23} vs T_{2exp}, and add the predictions from the Kennedy-O'Hagan model to this plot using a different plotting character/color.

8. A computer experiment using a thermomechanical finite element model to simulate the friction drilling process was performed by varying the friction coefficient (η) and observing the force output (y) over the tool travel distance (x). A physical experiment was also conducted by measuring the force output over the tool travel distance. See Miller and Shih (2007) for details. The computer experiment data are given in `drillcomp.csv` and physical experiment data in `drillphy.csv`.

   (a) Fit the computer experiment data using an ordinary kriging model. Use a log-transform of the force to avoid negative values in the prediction.

   (b) Estimate the friction coefficient using least squares.

   (c) Plot the force against the tool travel distance from the physical experiment data. Overlay this plot with the predictions from the calibrated computer model (that is, fix η in the surrogate model at the estimated value from (b)).

   (d) Suppose that we are able to measure the force only at a few tool travel distances. Find the locally D-optimal design assuming $\eta^0 = 0.75$.

9. The file `LAMM.txt` contains the data from a laser-assisted micro-machining process. Four variables are used in the experiment: nominal depth of cut (x_1), cutting speed (x_2), speed (x_3), and laser location (x_4). The cutting force output from the computer model (f) and the measured output (y) from a real experiment from Prof. Melkote's lab at Georgia Tech are also given.

(a) Singh et al. (2011) postulate the following model to approximate the computer model:

$$g(\mathbf{x}) = \beta_0 x_1^{\beta_1} \exp\left\{\beta_2 x_2 - \beta_3 x_3 e^{-\beta_4 x_4}\right\}, \tag{9.17}$$

where $\beta_i > 0$ for all i. Fit this model to the computer experimental data using nonlinear regression.

(b) Joseph and Yan (2015) proposed the following adjustment model to capture the model discrepancy:

$$y = g(\gamma_1 x_1, \gamma_2 x_2, \gamma_3 x_3, \gamma_4 x_4),$$

where $\gamma_i = 1$ for all i if there is no model discrepancy. Find the locally D-optimal design to accurately estimate $\boldsymbol{\gamma}$ assuming $\boldsymbol{\gamma}^0 = \mathbf{1}$.

(c) Suppose that we have a budget for 20 runs. Find the design that is robust to parameter and model uncertainties assuming $\gamma_i \overset{iid}{\sim} \mathcal{N}(1, 0.1^2)$ for $i = 1, \ldots, 4$. Compare this design to the actual 48-run design and comment.

10. Consider the spot welding experiment discussed in Bayarri et al. (2007). Three controllable variables are studied using a computationally expensive finite element model: load (x_1), current (x_2), and gage (x_3). In addition, the computer model contains an unknown calibration parameter (η). The response of the experiment is the nugget diameter of the weld (y). A computer experiment is carried out using a 52-run maximin Latin hypercube design. After removing 17 runs that did not produce clear outputs, the data for the remaining 35 runs are given in the file `weldcomp.csv`. The physical experimental design has 12 runs with 10 replicates per run. The data from the physical experiment is given in the file `weldphy.csv`.

(a) Fit an ordinary kriging model to the computer experimental data.

(b) Find the locally D-optimal design for estimating the calibration parameter assuming $\eta^0 = 4$.

(c) Find a 12-run design that is robust to parameter and model uncertainties assuming $\eta \sim \mathcal{U}(1, 7)$. Compare this design to the actual 12-run design and comment.

(d) Find the nested-robust design by choosing the points in (b) and (c) from the computer experimental design projected onto the $\mathbf{x}$-space.

Part IV

Data Science



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

10

Data Subsampling

So far we have been looking at the problem of where to collect the data. In this chapter, we will look at a different problem–the data have already been collected, but they are too big for storage or analysis and, therefore, we need to choose a subset of the data. We want to choose the subset in such a way that we lose very little information that is contained in the big data. This is very similar to the experimental design problem. However, there are two main differences. First, in the experimental design problem, the experimental space is generally continuous, whereas in the subset data selection problem, the space is discrete because the big data restrict our choices to a finite number of cases. Second, the response is not available while designing the experiment, but in the subset data selection problem, we already have the responses.

10.1 Support Points-based Subsampling

Since the data we have in hand is a sample from the population, the subset of the data can also be viewed as a subsample. A common technique to obtain a subsample is to choose some of the rows of the data matrix uniformly at random. However, as we have observed earlier in the experimental design problem, the random sample can sometimes be good and sometimes very poor. We need to use a better sampling technique to get more consistent and robust results. For fitting a linear regression model, Wang et al. (2019) proposed choosing samples by maximizing the D-optimality criterion. However, such a subsample is optimal only for the assumed linear model. If we need to fit a different model, say a random forest or a Gaussian process model, then the chosen subsample will no longer be optimal. There is a growing literature on coresets (or core-sets) in machine learning (Feldman, 2019) which again seems to focus on a specific parametric model. Since statistical modeling is an iterative process, it is imperative to choose a subsample without relying on a specific model.

One possible approach to do this is to choose a subsample which has the same distribution as the original dataset. Then, all the properties that we compute using the subsample will approximately match with those of the dataset. If we can optimally match the distribution, then we will lose very little information in using the subsample. Thus, we can use the techniques described in Chapter 5 to obtain the best possible representative sample (Chen et al., 2012; Mak and Joseph, 2018c,b; Shang et al., 2023).

Let $\mathcal{D} = \{\mathbf{Z}_i = (\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^N$ be the dataset, where there are p input variables and q output variables. Our aim is to obtain a subsample $\mathcal{S} = \{\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $n < N$ so that the empirical distribution of the subsample is as close to the empirical distribution of the dataset. First, standardize all variables to have unit variance so that Euclidean distance makes sense. Then, the distance between these two empirical distributions can be measured

DOI: 10.1201/9781003661535-10

This chapter has been made available under a CC-BY-NC-ND license 4.0.

using the energy distance (Székely et al., 2004):

$$ED(\mathcal{S},\mathcal{D}) = \frac{2}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N}\|\mathbf{z}_i-\mathbf{Z}_j\| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{z}_i-\mathbf{z}_j\| - \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\|\mathbf{Z}_i-\mathbf{Z}_j\|.$$

Thus, the support points-based subsample can be obtained using (Mak and Joseph, 2018c):

$$\mathcal{S}^* = \underset{\mathbf{z}_1,\cdots,\mathbf{z}_n\in\mathcal{D}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n}\sum_{j=1}^{N}\|\mathbf{z}_i-\mathbf{Z}_j\| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{z}_i-\mathbf{z}_j\| \right\}. \tag{10.1}$$

The major difference between (5.13) and (10.1) is that the former performs a continuous optimization, whereas the latter performs a discrete optimization. Therefore, we cannot use the difference-of-convex programming technique that we used for solving (5.13). However, discrete optimization is generally a much more difficult problem. Our main motivation to obtain a subsample was to save the computational burden of handling the big dataset. Therefore, if it is time-consuming to obtain the subsample, then the whole purpose is defeated. One approach to circumvent this issue is to first use the difference-of-convex programming to quickly find the support points in the continuous space spanned by the dataset. Then, we can use a nearest neighbor algorithm to choose the closest point in the dataset. However, a naive nearest neighbor assignment can lead to duplicates. Therefore, Joseph and Vakayil (2022) proposed a one-at-a-time sequential nearest-neighbor algorithm, where a point is removed from the dataset if it is already selected. This is implemented in the R function `subsample` in the R package `SPlit` (Vakayil et al., 2022), which uses `kd-tree` for fast nearest-neighbor assignment.

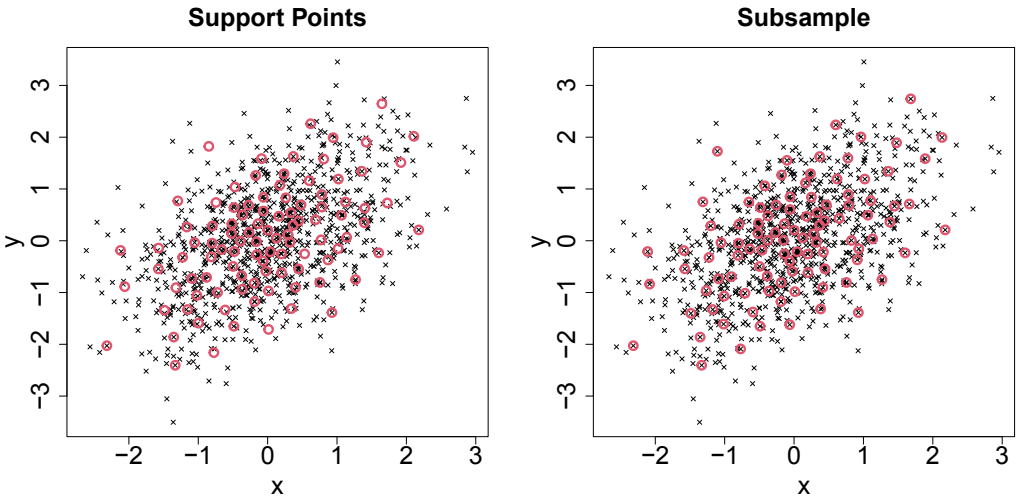


FIGURE 10.1: (left) Support points obtained using continuous optimization. (right) Subsample obtained using sequential nearest-neighbor algorithm.

As an example, consider the 1,000 points (black crosses) in Figure 10.1. The 100 support points obtained using the difference-of-convex program are shown in the left panel. We can see that they do not fall on the data points. The right panel shows the points obtained using the sequential nearest-neighbor algorithm, which is a subsample of the data. The

nearest-neighbor assignment may cause a slight distortion to the distribution, which can be neglected.

Now consider the problem of predicting power from wind turbines based on wind speed (Ding, 2019). The dataset[1] contains 39,195 data collected in 10-minute time intervals. They are plotted in Figure 10.2. Suppose that we are interested in fitting a Gaussian process model to this data. However, this data size is too large for a GP. Therefore, we sampled 1% of the data using the support points-based subsampling method and fitted a GP (ordinary kriging) with a Gaussian correlation function using `rkriging` (Huang and Joseph, 2024). The data reduction took about two seconds and the model fitting on 392 data points took less than one second. Thus, the overall fitting could be done in less than three seconds. This is a great saving on computational time because fitting on the original dataset would have taken several days. The predictions are shown in Figure 10.2. Unfortunately, they do not look good for low and high speeds, which is a well-known issue with stationary GPs. The constant mean assumption of a stationary GP causes a "mean reversion", that is, predictions will go back to the prior mean in data-sparse regions (Joseph, 2006b). We will revisit this problem later in Chapter 11 and explain how the GP can be modified to improve the predictions with large datasets.

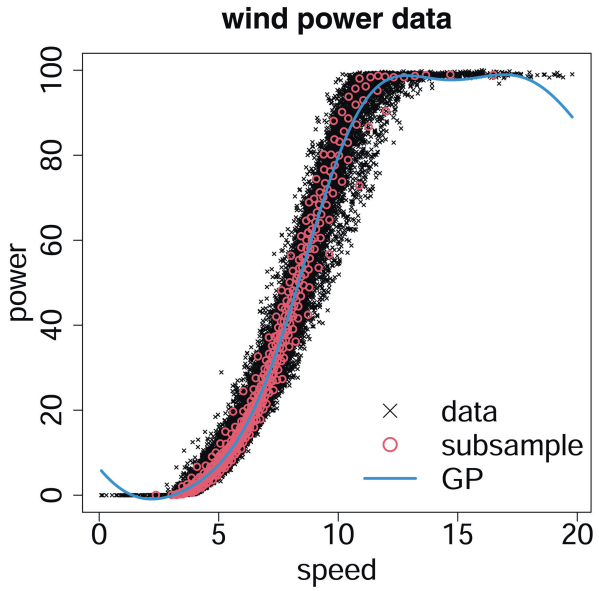


FIGURE 10.2: The plot shows 392 subsamples taken from the wind power dataset of size 39,195 and the GP model fitted using the subsample.

Another application of support points-based sampling is for scenario reduction in stochastic programming (Dupačová et al., 2003). Let $\{\mathbf{\Psi}\}_{i=1}^{N}$ be the set of scenarios for the random variable $\mathbf{\Psi}$. Then, the stochastic optimization problem can be simplified as

$$\min_{\mathbf{x}} \{E[h(\mathbf{x}; \mathbf{\Psi})]\} \approx \min_{\mathbf{x}} \left\{ \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{x}; \mathbf{\Psi}_i) \right\} \approx \min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}; \boldsymbol{\psi}_i) \right\},$$

[1] https://sites.google.com/view/yuding/book-dswe

where $\{\boldsymbol{\psi}_i\}_{i=1}^n$ is a subsample of the set of scenarios. Since $n << N$, the optimization can be done much faster using the subsample of scenarios than using the full set of scenarios. Barry et al. (2022) use this approach for the optimization of electric power grids and Knapp et al. (2023) for the optimization of plasma radiation detectors.

10.2 Data Splitting

Suppose that we have several models that explain the data well. Then, which model should we deploy in the field? The best way to answer this question is to deploy all the models and then choose the one that performs well, but this is almost impossible to do in practice. A common way to circumvent this problem is to split the data into two parts: training and testing. Then, use only the training data to estimate the models and use the testing data to choose the best model. Under the assumption that the distribution of the data remains the same when we deploy the models to the field (i.e., there is no covariate shift), we should obtain the testing data that have the same distribution as the entire dataset. Randomly sampling some rows from the data matrix is one way to do this. For example, if we are interested in splitting the data in an 80:20 ratio, then we can randomly sample 20% of the rows to obtain the testing set and the remaining 80% will become the training set. However, random sampling causes too much variability in the test results, making the choice of model unreliable.

Clearly, we could use ideas from space-filling designs to select a good testing set that uniformly covers the entire dataset. Interestingly, one of the earliest applications of space-filling designs was to choose a testing set from a given dataset. Snee (1977) proposed a modification of the sequential maximin designs in (4.20) (Kennard and Stone, 1969) to find an optimal testing set. As we have observed earlier, the maximin design pushes the points to the boundaries, and therefore the testing set would have a distribution that is much more dispersed than the original dataset. Snee's idea was to alternate the one-point-at-a-time greedy selection of the points in (4.20) between the training and testing sets. This algorithm, known as DUPLEX, produces a much nicer testing set. However, as shown in Joseph and Vakayil (2022), the distribution of the DUPLEX testing set can still be very different from that of the original dataset. To maintain the distribution and obtain a well-distributed testing set, Joseph and Vakayil (2022) proposed to use the support points-based subsampling algorithm discussed in the previous section. They termed this algorithm as SPlit. Although the algorithm is the same as the one used for data subsampling, the motivation here is completely different, and it can also be applied to small datasets.

For illustration, consider a synthetic dataset of 100 points from Joseph and Vakayil (2022). There are two input variables (x_1 and x_2) and one categorical output variable y that has three levels. The data is plotted in Figure 10.3 with the three categories shown in red-triangles, green-pluses, and blue-crosses. Suppose that we are interested in splitting this dataset in an 80:20 ratio. The `SPlit` function in the R package `SPlit` (Vakayil et al., 2022) automatically codes the categorical variable using Helmert coding mentioned in Section 8.3, computes the support points using the difference-of-convex program, and selects the subsample using the sequential nearest neighbor algorithm. The 20 testing points thus obtained are shown in the right panel of Figure 10.3. The left panel shows a random testing set. We can see that the `SPlit` testing set is much more balanced and representative of the original dataset than that of the random testing set. This will help compute the generalization error more accurately and will help select the best model in a more consistent way.

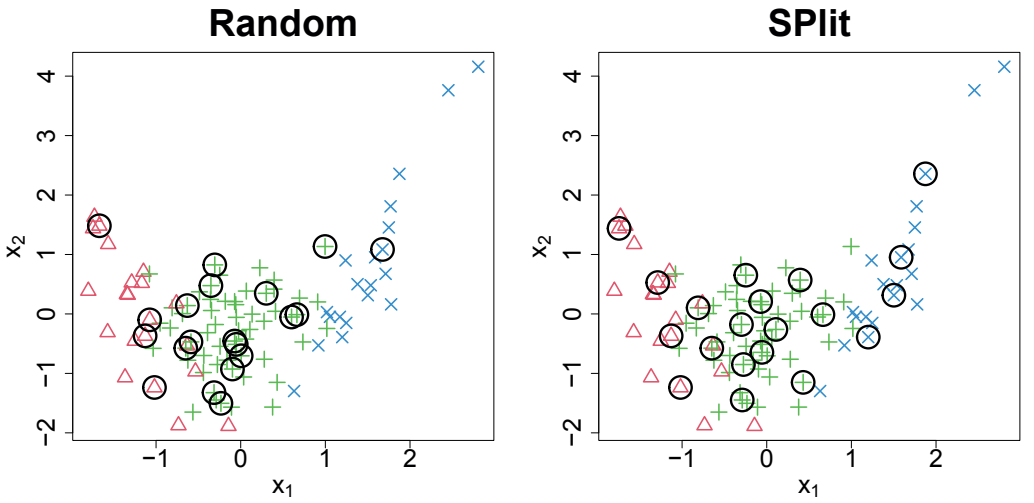


FIGURE 10.3: Testing points (circles) selected using random sampling (left) and `SPlit` (right). The three categories of the response are shown in red-triangles, green-pluses, and blue-crosses.

10.3 Data Twinning

The computational complexity of finding n support points from N data points is $\mathcal{O}(pn(n+N)m/P)$, where m is the number of iterations used in the difference-of-convex program and P the number of processor cores available for parallel computation. The average complexity of building a `kd-tree` is $\mathcal{O}(pN \log N)$. Thus, the average complexity of support points-based subsampling or splitting is $\mathcal{O}(pN \log N + pn(n+N)m/P)$. Let $\gamma = n/N$ be the splitting ratio, which is usually in $(0, 0.5]$ (Joseph, 2022). Therefore, the overall complexity can be simplified to

$$\mathcal{O}\left(pN \log N + \frac{mp\gamma N^2}{P}\right).$$

This grows quadratic in N and therefore `SPlit` cannot be used for large datasets. Vakayil and Joseph (2022a) proposed an algorithm to overcome this computational bottleneck, which is described next.

So far we have been discussing about choosing a small subset $\mathcal{S}$ of size n from a large dataset $\mathcal{D}$ of size N. The complement of $\mathcal{S}$ is the dataset $\overline{\mathcal{S}}$ which is of size $N-n$. The energy distance between the empirical distributions of these two sets can be calculated as

$$ED(\mathcal{S}, \overline{\mathcal{S}}) = \frac{2}{n(N-n)} \sum_{i=1}^{n} \sum_{j=1}^{N-n} \|\mathbf{z}_i - \mathbf{v}_j\| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{z}_i - \mathbf{z}_j\| - \frac{1}{(N-n)^2} \sum_{i=1}^{N-n} \sum_{j=1}^{N-n} \|\mathbf{v}_i - \mathbf{v}_j\|,$$

where $\mathcal{S} = \{\mathbf{z}_i\}_{i=1}^{n}$ and $\overline{\mathcal{S}} = \{\mathbf{v}_i\}_{i=1}^{N-n}$. Now, instead of trying to minimize $ED(\mathcal{S}, \mathcal{D})$, we can think of minimizing $ED(\mathcal{S}, \overline{\mathcal{S}})$. This will make the two sets $\mathcal{S}$ and $\overline{\mathcal{S}}$ statistically similar and, therefore, Vakayil and Joseph (2022a) call them *twins*. Interestingly, the following identity holds:

$$ED(\mathcal{S}, \mathcal{D}) = \left(\frac{N-n}{N}\right)^2 ED(\mathcal{S}, \overline{\mathcal{S}}). \tag{10.2}$$

Therefore, minimization of $ED(\mathcal{S}, \overline{\mathcal{S}})$ and $ED(\mathcal{S}, \mathcal{D})$ will produce identical results. Nevertheless, the idea of twinning provides a new perspective of the optimization problem, which can be used to develop a heuristic algorithm to solve the problem.

First, note that $ED(\mathcal{S}, \overline{\mathcal{S}})$ contains three terms. Therefore, minimization of $ED(\mathcal{S}, \overline{\mathcal{S}})$ can be achieved by (i) bringing $\mathcal{S}$ and $\overline{\mathcal{S}}$ closer, (ii) spreading $\mathcal{S}$, and (iii) spreading $\overline{\mathcal{S}}$. We will explain how these conflicting objectives can be achieved using a simple example. Consider the 50 data points in Figure 10.4. Suppose that we want to split this dataset into 40 training points and 10 testing points. Start with the point marked "1" as the first testing point. Choose the nearest four points to it and assign them to the training set. This is to bring $\mathcal{S}$ and $\overline{\mathcal{S}}$ closer together. Now find the farthest point from "1" among the four points and choose the closest point to it as "2", which will be the second testing point. This ensures spreading of $\overline{\mathcal{S}}$. This algorithm can be continued until we obtain 10 points in the testing set, which is shown in the right panel of the same figure. All of these computations can be done using a `kd-tree` with an average computational complexity of $\mathcal{O}(pN \log N)$. This is a huge saving over the `SPlit` algorithm. However, as seen in the right panel, the convex hull of points starts to overlap in later iterations, making the selection of testing points not so good. Nevertheless, Vakayil and Joseph (2022a) have shown that the twinning algorithm produces testing sets as good as the `SPlit` algorithm or even better sets when N is large.

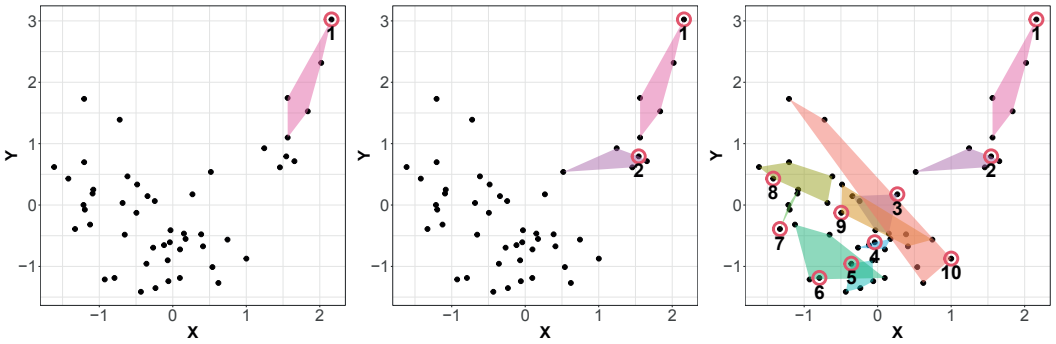


FIGURE 10.4: First, second, and tenth iteration of the `twinning` algorithm for selecting 10 points out of 50 points.

Consider again the wind power dataset that contains 39,195 data points. The twinning algorithm is implemented in the R package `twinning` (Vakayil and Joseph, 2022b) and in a Python package[2]. Choosing 392 points from the 39,195 points using `twinning` took only about 0.02 seconds, which is 100 times faster than the `SPlit` algorithm.

Because `twinning` is so fast that it can be efficiently applied to other data science applications such as partitioning the dataset into multiple sets for cross-validation or divide-and-conquer procedures. See Joseph et al. (2024) for an application of `twinning` to speed-up a computationally expensive cross-validation-based estimation procedure.

In survey sampling, for populations with spatial trends, it is of interest to select samples that are well spread over the population. A well-spread sample is known as *spatially balanced*, which is synonymous with space-filling designs in the experimental design literature. There are many methods available in the survey sampling literature to obtain a spatially balanced sample. See the book by Tillé (2006) for details. To make a comparison with these methods, consider a simulated dataset $\mathbf{X}_i \overset{iid}{\sim} \mathcal{N}_2(\mathbf{0}, \mathbf{I})$ for $i = 1, \ldots, 1000$. Suppose that we are interested in sampling $n = 100$ points $\{\mathbf{x}_i\}_{i=1}^{n}$ from the dataset where each point has

[2] https://github.com/avkl/twinning

equal inclusion probabilities. Consider the local pivotal methods 1 and 2 proposed by Grafström et al. (2012), which are implemented in the R package `BalancedSampling` (Grafström and Prentius, 2024). We will compare them to `SPlit` and `twinning` on two metrics: energy distance and a spatial balance measure defined as

$$SB = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{n_i}{(N/n)} - 1\right\}^2,$$

where n_i is the number of points in the Voronoi region around the point $\mathbf{x}_i$ and $\sum_{i=1}^{n} n_i = N = 1000$. If the sample is spatially balanced, then $n_i = N/n$ for all $i = 1, \ldots, n$, and therefore $SB = 0$. This may not be possible to achieve in practice, but a smaller value indicates better spatial balancing. Figure 10.5 shows the boxplots of these two measures over 30 repetitions of the algorithms. We can see that `SPlit` and `twinning` provide a much better spatially balanced sample than both the local pivotal methods. To be fair, the local pivotal methods can handle data points with unequal inclusion probabilities, whereas `SPlit` and `twinning` cannot. If we need to use unequal inclusion probabilities in support points-based subsampling, it can be implemented using the importance support points in (5.17).

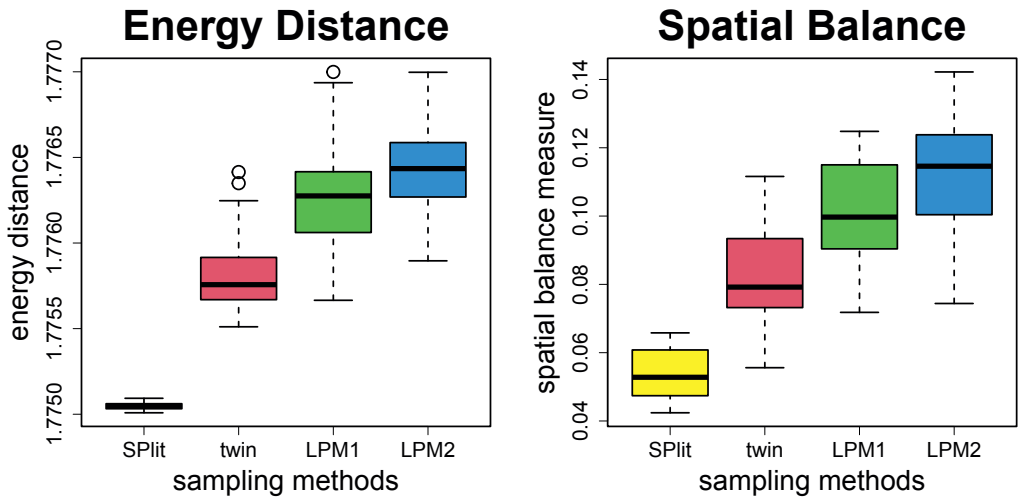


FIGURE 10.5: Comparison of `SPlit` and `twinning` with local pivotal methods for spatially balanced sampling. Smaller values of energy distance and spatial balance measure show well-spread samples that are representative of the population.

10.4 Supervised Compression

Although the datasets include response information, we have not explicitly used them to select the subsample, except to ensure that the join distribution of the subsample $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ matches the join distribution of the dataset $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{n}$. However, if the purpose of obtaining the subsample is to develop a predictive model, then it makes sense to choose samples based on the information in the responses. For example, consider the wind power data in Figure 10.2. Since power has little variation for speeds less than 3

m/s or more than 13 m/s, it might be enough to get the subsample for speed in the range [3, 13]. This is done under the assumption that our modeling method is capable of modeling this non-stationary behavior of the input-output relationship. As shown in Figure 10.2, the stationary Gaussian process was clearly unable to do a good job with this dataset. We might need to use Gaussian processes that can handle non-stationarity such as the Treed Gaussian process (Gramacy and Lee, 2008) or the composite Gaussian process (Ba and Joseph, 2012).

Suppose that we have a continuous scalar output. Denote the fitted model using the subsample by $y = \hat{h}(\mathbf{x}|\mathcal{S})$. Assuming a squared-error loss, our objective is to find the optimal subsample using

$$\min_{\mathcal{S}} \quad L = \sum_{i=1}^{N} \{Y_i - \hat{h}(\mathbf{X}_i|\mathcal{S})\}^2. \tag{10.3}$$

We need to make a choice of $\hat{h}$ before this optimization can be performed. If we choose it to be a linear regression model, then the optimal subsample will be optimal only for that chosen linear model and may not work well with the other models. Therefore, to make the subsample robust to the modeling choices, we need to choose a nonparametric regression model for $\hat{h}$. However, if we choose a treed or composite Gaussian process, then its estimation and computation will be expensive, and therefore the time to find an optimal subsample will be too high for it to be of any use. Thus, we need to choose a simple non-parametric model that is flexible and fast to evaluate. Joseph and Mak (2021) proposed to use a nearest neighbor method (1-NN) for this purpose.

The nearest neighbor predictor is given by

$$\hat{h}(\mathbf{x}|\mathcal{S}) = y_i, \quad \text{where} \quad i = \underset{j=1:n}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_j\|.$$

Thus, for a given set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we can partition the index set of the dataset, namely $\{1, \ldots, N\}$, into index sets I_i, $i = 1, \cdots, n$, which collect all indices of points in the dataset closest to point $\mathbf{x}_i$:

$$I_i = \{j : ||\mathbf{X}_j - \mathbf{x}_i|| < ||\mathbf{X}_j - \mathbf{x}_{i'}|| \text{ for all } i' \neq i\}, \quad i = 1, \cdots, n. \tag{10.4}$$

Clearly, $\bigcup_{i=1}^{n} I_i = \{1, \ldots, N\}$ and $I_i \cap I_j = \emptyset$ for $i \neq j$. The nearest-neighbor regions in (10.4) are the same as the Voronoi regions that we have used extensively in Chapter 4. Then, the loss in (10.3) can be simplified as

$$\begin{aligned} L &= \sum_{i=1}^{n} \sum_{j \in I_i} \{Y_j - \hat{h}(\mathbf{X}_j|\mathcal{S})\}^2 \\ &= \sum_{i=1}^{n} \sum_{j \in I_i} \{Y_j - y_i\}^2. \end{aligned}$$

Minimization of L with respect to $\{y_1, \ldots, y_n\}$ gives the familiar solution

$$y_i = \frac{\sum_{j \in I_i} Y_j}{N_i} = \overline{Y}_i,$$

where N_i is the number of points in cluster i. Thus, the problem in (10.3) simplifies to

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_n} \quad L = \sum_{i=1}^{n} \sum_{j \in I_i} \{Y_j - \overline{Y}_i\}^2. \tag{10.5}$$

Although this optimization looks similar to a `kmeans` clustering, it is a very different problem because here we need to find the optimal partitions in the input space and not the output space.

Joseph and Mak (2021) proposed a sequential algorithm to approximately solve (10.5). The algorithm starts with some points (usually two) generated using `kmeans`. Then, it identifies the cluster that gives the largest sum of squares of errors $\sum_{j \in I_i} \{Y_j - \overline{Y}_i\}^2$. This cluster is then split into two, and the iteration continues until we have the desired number of clusters.

Consider a simulated dataset generated using the Michaelwicz function with some noise as follows:

$$Y_i = -\sum_{k=1}^{2} \sin(\pi X_{ik}) \sin^{20}(k\pi X_{ik}^2) + \epsilon_i,$$

where $\mathbf{X}_i \overset{iid}{\sim} \mathcal{U}(0,1)^2$ and $\epsilon_i \sim^{iid} \mathcal{N}(0, 0.01^2)$, $i = 1, \ldots, N$. Let $N = 10,000$ and suppose that we want to compress this dataset to a size $n = 100$. The algorithm of Joseph and Mak (2021) is implemented in the R package `supercompress` (Huang and Joseph, 2022). The 100 points and the Voronoi regions are shown in the right panel of Figure 10.6. For comparison, the unsupervised selection of points using `kmeans` is shown in the left panel of the same figure. We can see that `kmeans` uniformly spreads out the points in the input region, whereas the `supercompress` algorithm chooses more points in the region where the response has steep changes and fewer points in the flat regions. As a result, Voronoi regions with `kmeans` are roughly of the same size, while those with `supercompress` are small in the region of high variation and large in the region of low variation. Clearly, a model capable of capturing the non-stationarity of the function will do much better with the points selected using `supercompress` than with those of `kmeans`. Quantitatively, the root mean squared prediction error of a nearest neighbor predictor is 0.1556 with the unsupervised set of points and 0.0996 with the supervised set of points, showing a 36% improvement in prediction accuracy. However, the computational complexity of the `supercompress` algorithm is of the

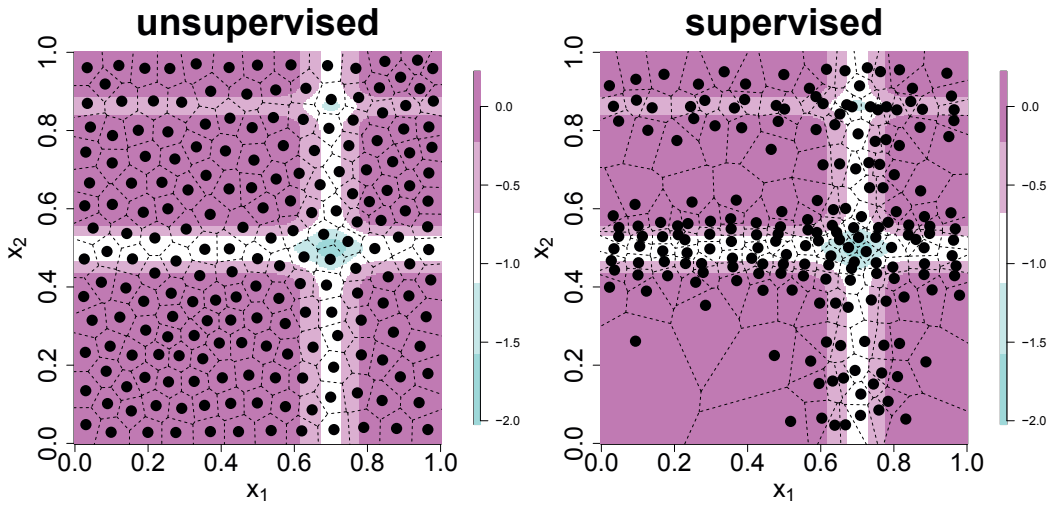


FIGURE 10.6: 100 points selected using `kmeans` (left) and `supercompress` (right) out of 10,000 points. The points are overlayed on the image of the Michaelwicz function, which is used for generating the data.

order $\mathcal{O}(pnN \log N)$, which makes it much slower than the unsupervised `twinning` algorithm for data reduction.

Exercises

1. Suppose the dataset $\mathcal{D} = \{\mathbf{Z}_i\}_{i=1}^{N}$ is split into $\mathcal{S} = \{\mathbf{z}_i\}_{i=1}^{n}$ and $\overline{\mathcal{S}} = \{\mathbf{v}_i\}_{i=1}^{N-n}$.
   (a) Prove the identity in (10.2).
   (b) Show that $ED(\mathcal{S}, \mathcal{D}) + ED(\overline{\mathcal{S}}, \mathcal{D})$ is minimized when $n = N/2$.

2. Most statistical and machine learning models have some hyper-parameters or regularization parameters, which are commonly estimated from the training set by holding out a validation set, say of size m. Thus, we want to split the dataset into three sets: training, validation, and testing. Consider the following two approaches to doing this.

   Approach 1: Split the dataset $\mathcal{D}$ of size N into two sets $\mathcal{S}$ and $\overline{S}$ of size n and $N - n$, respectively using `SPlit`. Then split $\overline{S}$ into two sets of size m and $N - n - m$ again using `SPlit`. Use the set of size m as the validation test and the other as the training set.

   Approach 2: Split the dataset $\mathcal{D}$ of size N into two sets $\mathcal{S}$ and $\overline{S}$ of size $n + m$ and $N - n - m$, respectively using `SPlit`. Then split S into two sets of size n and m again using `SPlit`. Use the set of size m as the validation test and the other as the testing set.

   Which approach creates a better validation set so that the testing performance is not influenced by the model estimation/validation step?

3. Suppose that we want to split a dataset into four representative sets of 25% points each. For example, this could be used for a four-fold cross-validation. Vakayil and Joseph (2022a) describes three strategies to do this as shown in Figure 10.7, which is implemented in the function `multiplet` in the R package `twinning`. To assess the performance of these strategies, randomly generate a dataset of size $N = 1,000$ from

$$\mathcal{N}_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right),$$

   with $\rho = 0.5$. Calculate the maximum energy distance among the four sets for each strategy. Repeat this 100 times and compare the three strategies in terms of the maximum energy distance and computational time.

4. Consider the population density in various zip code locations of the United States without Alaska and Hawaii (Rozzi, 2022) shown in Figure 4.22. Assign a weight to each zip code proportional to $\log(y + 1)$, where y is the population density. Now sample 200 zip codes out of the 41,877 zip codes using three methods: importance support points (without replacement), LPM1, and LPM2. Compare them using energy distance and spatial balance measures.

5. Consider the dataset released by the New York Taxi and Limosine Commission in 2016, which was made publicly available for a popular Kaggle predictive modeling competition in 2017 (Risdal, 2017). We will use the augmented dataset provided in Benmeida (2020), which includes a total of $p = 8$ input features: Trip distance, Motorway proportion, Trunk proportion, Primary proportion, Secondary proportion, Tertiary proportion, Unclassified proportion, and Residential proportion. The response is the trip duration in seconds. The

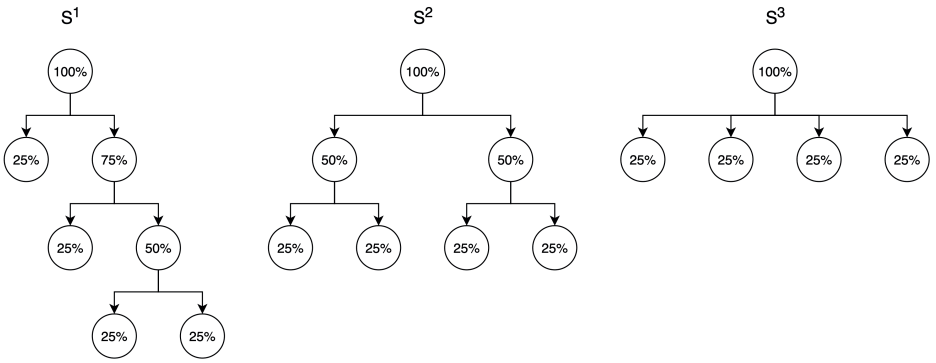


FIGURE 10.7: Different multiplet strategies.

dataset contains 2,074,291 points. Joseph and Mak (2021) sampled 80% of the points randomly and used it as the training set and the remaining as testing set. They are given in the files `taxitain.csv` and `taxitest.csv`. Sample 1% of the training set using the following methods

(a) random

(b) twinning

(c) supercompress.

Fit a Gaussian process model and a random forest model to the subsample and compare their prediction performance using the testing set.

6. Consider NASA airfoil self-noise dataset[3] (Brooks et al., 1989), obtained from a series of aerodynamic and acoustic tests of two- and three-dimensional airfoil blades conducted in a wind tunnel. The dataset has 1,503 rows and 5 input variables. The response is the scaled sound pressure in dB. Split the dataset 80:20 using (i) random and (ii) SPlit. Fit a Gaussian process model with Gaussian correlation function and a random forest model to the training data and compute the root mean squared prediction error on the testing set. Repeat this procedure 30 times and compare the prediction performance of the two modeling methods.

[3] https://archive.ics.uci.edu/dataset/291/airfoil+self+noise

11

Data Analysis

A major goal of data analysis is to identify the important input variables that affect the output and to build a predictive model using the important variables. Traditionally, experimental design is not viewed as a technique for data analysis but as a tool for data collection and planning. The purpose of this chapter is to show that the experimental design or its related techniques that we discussed in earlier chapters can be used for data analysis and predictive modeling. We will see that a careful integration of these techniques into data analysis and modeling can yield better and faster results, especially when dealing with big data.

11.1 Factor Selection and Ranking

Variable selection is considered one of the most important topics in statistics, which has recently seen renewed interest due to the need for interpreting and explaining black-box statistical and machine learning models. This topic has been in the literature for a long time and the fact that the research on it has not been settled speaks to its difficulty. Fortunately, we have already laid the necessary groundwork for developing variable selection techniques in Chapter 6. We can use the global sensitivity indices for quantifying the importance of variables and use them for selection and ranking. However, in data analysis problems, we only have a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^p$. This means that we do not have access to the computer model or know the distribution of the variables, which substantially increases the complexity of the problem.

To understand the challenges associated with measuring factor importance in data analysis problems, let us first recall some of the concepts discussed in Chapter 6. One of the main measures that we used for factor screening was the total Sobol' index (Homma and Saltelli, 1996). Let $y = h(\mathbf{x})$, $\mathbf{x} \in [0,1]^p$ be the input-output relationship which we assume to be known or that at least we have access to a computer code which can be used for obtaining the output given the input values. By treating $\mathbf{x}$ to be a uniformly distributed random variable, i.e., $\mathbf{x} \sim \mathcal{U}(0,1)^p$, the total Sobol' index for the ith factor is defined as

$$T_i = \frac{V_i^{tot}}{V} = \frac{E\{var[h(\mathbf{x})|\mathbf{x}_{\sim i}]\}}{var\{h(\mathbf{x})\}}. \tag{11.1}$$

We developed Monte Carlo and experimental design-based techniques to estimate them and then declared a factor to be important if T_i is large and unimportant if T_i is small. Unfortunately, many things are different in a data analysis problem: (i) we do not know the input-output relationship, (ii) the joint and conditional distributions of the inputs are

DOI: 10.1201/9781003661535-11

This chapter has been made available under a CC-BY-NC-ND license 4.0.

unknown and can be far from uniform, (iii) the data are noisy and can be unstructured with no option to collect new data using experimental design, and (iv) the input variables need not be independent.

We can try to overcome the first issue by estimating the unknown input-output relationship from data using nonparametric regression or machine learning techniques. However, such an approach can be difficult and time-consuming, especially with large amounts of data. Moreover, errors in estimation can bias the importance measures. Similarly, we can try to overcome the second issue through density estimation, but it can be an arduous task by itself, especially in high-dimensional problems. If we are able to solve the first two issues, we can solve the third issue by using the estimated function and distribution.

Here we will describe another approach to simultaneously take care of the first three issues. However, the fourth one on dependent inputs is a serious issue because it questions the validity of Sobol' indices, which requires some discussions before we proceed with this approach. If the variables are not independent, then the orthogonality that we used to decompose the variance does not hold true anymore, and therefore the functional ANOVA which formed the basis of Sobol' indices breaks down. As an example, let

$$y = 2x_1 + \frac{2}{\sqrt{3}}x_2 + x_3, \tag{11.2}$$
$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right), \; x_3 \sim \mathcal{N}(0,1).$$

Since the model is additive, it is easy to visualize the effect of each variable as shown in the left panel of Figure 11.1. If we ignore the correlations, clearly x_1 is the most important variable, x_2 the second most important, and x_3 the least important. However, if the correlation $|\rho|$ is high, then adding x_2 to the model when x_1 is already in the model is not useful and vice versa. Since x_1 is more important than x_2, it would make sense to develop a predictive model using (x_1, x_3) and ignore x_2.

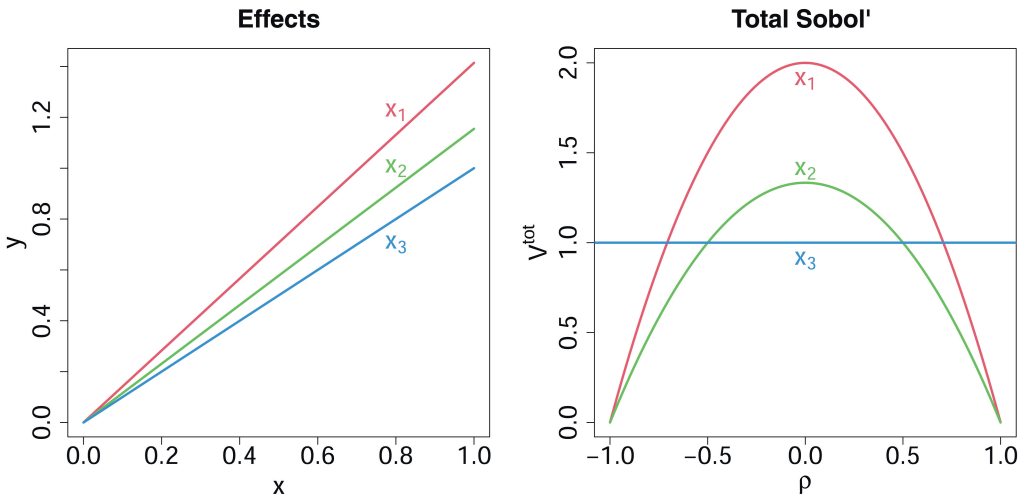


FIGURE 11.1: The effects of the three variables for the model in (11.2) (left) and their total Sobol' indices (right).

Now let us look at the total Sobol' indices for this simple linear model. Since the denominator in (11.1) is a constant for all the variables, we only need to consider the numerator V_i^{tot}. They can be obtained as

$$V_1^{tot} = 4(1-\rho^2), \; V_2^{tot} = \frac{4}{3}(1-\rho^2), \text{ and } V_3^{tot} = 1,$$

and are plotted in the right panel of Figure 11.1. When the correlation is small, specifically when $|\rho| < 0.5$, the total Sobol' index agrees with our intuition that $x_1 \succ x_2 \succ x_3$, where "$\succ$" means "more important than". However, when correlation is high, the total Sobol' index does not seem to make any sense. In fact, as $|\rho| \to 1$, the total Sobol index for both x_1 and x_2 goes to zero, indicating that both of these variables are unimportant. This is clearly not the case!

Many proposals exist in the literature that attempt to overcome the deficiency of total Sobol' indices with dependent inputs. Among them, a prominent approach is to compute the importance measure using game-theoretic Shapley values (Owen, 2014; Song et al., 2016), which can be viewed as a weighted average of total Sobol' indices computed over all possible subsets of variables (Verdinelli and Wasserman, 2024). Another interesting approach is the conditional density-based sensitivity measure proposed by Borgonovo (2007). We refer to the book by Da Veiga et al. (2021, Ch. 5) and the references therein for details.

Hart and Gremaud (2018) provided a new perspective of the total Sobol' index that it can be viewed as the L_2 approximation error when a variable is removed from the model keeping the other variables fixed. To see this, first note that the best model that approximates a function $y = h(\mathbf{x})$ when a variable x_i is removed can be obtained as

$$h_{\sim i}(\mathbf{x}_{\sim i}) = \underset{g(\mathbf{x}_{\sim i})}{\operatorname{argmin}} E\{h(\mathbf{x}) - g(\mathbf{x}_{\sim i})\}^2 = E\{h(\mathbf{x})|\mathbf{x}_{\sim i}\}.$$

With this optimal choice, the approximation error becomes

$$\begin{aligned} E\{h(\mathbf{x}) - h_{\sim i}(\mathbf{x}_{\sim i})\}^2 &= E\{h(\mathbf{x}) - E[h(\mathbf{x})|\mathbf{x}_{\sim i}]\}^2 \\ &= E\left(E\{h(\mathbf{x}) - E[h(\mathbf{x})|\mathbf{x}_{\sim i}]|\mathbf{x}_{\sim i}\}^2\right) \\ &= E\left(var\{h(\mathbf{x})|\mathbf{x}_{\sim i}\}\right) \\ &= V_i^{tot}. \end{aligned}$$

Huang and Joseph (2025a) showed that this interpretation of total Sobol' index agrees with the leave-out-covariate (LOCO) approach proposed in Lei et al. (2018), Williamson et al. (2021), and Verdinelli and Wasserman (2024). In fact, they proved the following result.

Theorem 11.1 *The total Sobol' index $T_i = 0$ if and only if $h(\mathbf{x}) \perp \mathbf{x}_i \mid \mathbf{x}_{\sim i}$, that is, $h(\mathbf{x})$ and $\mathbf{x}_i$ are independent conditional on $\mathbf{x}_{\sim i}$.*

Thus, although the absolute value of total Sobol' index may not make sense in the case of dependent variables, it can be used as a measure to select factors.

Before the total Sobol' index can be used for factor selection, we need to develop a method to estimate it from unstructured, dependent, and noisy data. As mentioned earlier, we can use a nonparametric regression method to estimate the underlying model from the data. To reduce the chances of potential bias caused by a poor fitting model, Williamson et al. (2021) proposed using an ensemble of models. However, this substantially increases the computational complexity of the approach. To overcome the computational burden and at the same time achieve model robustness, Huang and Joseph (2025a) proposed a model-independent approach to the estimation of total Sobol' indices. Their approach is built on the nearest neighbor-based estimation procedure of Broto et al. (2020) for clean data.

First note that the numerator of total Sobol' index can be estimated using a double-Monte Carlo estimator (Song et al., 2016):

$$\hat{V}_i^{tot} = \frac{1}{N_O} \sum_{m=1}^{N_O} \left\{ \frac{1}{N_I - 1} \sum_{j=1}^{N_I} \left(h(\mathbf{x}_{\sim i}^{(m)}, x_i^{(m,j)}) - \frac{1}{N_I} \sum_{l=1}^{N_I} h(\mathbf{x}_{\sim i}^{(m)}, x_i^{(m,l)}) \right)^2 \right\}, \tag{11.3}$$

where $\{\mathbf{x}_{\sim i}^{(m)}\}_{m=1}^{N_O}$ are i.i.d. samples from the distribution of $\mathbf{x}_{\sim i}$ and $\{x_i^{(m,j)}\}_{j=1}^{N_I}$ conditionally to $\mathbf{x}_{\sim i}^{(m)}$ are i.i.d. with respect to the distribution of $x_i|\mathbf{x}_{\sim i} = \mathbf{x}_{\sim i}^{(m)}$. When $N_I = 2$, this reduces to the Jansen-Sobol estimator in (6.7). Since $h(\cdot)$ is unknown, Broto et al. (2020) proposed to replace the inner loop designs $\{(\mathbf{x}_{\sim i}^{(m)}, x_i^{(m,j)})\}_{j=1}^{N_I}$ by N_I samples in $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ with the values of their $\sim i$ entries closest to $\mathbf{x}_{\sim i}^{(m)}$. More formally, let $k_{\sim i}^{(m)}(j)$ be the index such that $\mathbf{x}_{\sim i}^{(k_{\sim i}^{(m)}(j))}$ is the j-th closest element to $\mathbf{x}_{\sim i}^{(m)}$ in $\{\mathbf{x}_{\sim i}^{(n)}\}_{n=1}^{N}$, with ties broken by random selection. Then the double Monte Carlo estimator for V_i^{tot} becomes

$$\hat{V}_i^{tot} = \frac{1}{N_O}\sum_{m=1}^{N_O}\left\{\frac{1}{N_I-1}\sum_{j=1}^{N_I}\left(y^{(k_{\sim i}^{(m)}(j))} - \frac{1}{N_I}\sum_{l=1}^{N_I} y^{(k_{\sim i}^{(m)}(l))}\right)^2\right\}, \tag{11.4}$$

where $\{\mathbf{x}_{\sim i}^{(m)}\}_{m=1}^{N_O}$ for the outer loop Monte Carlo can be either all the samples $\{\mathbf{x}_{\sim i}^{(n)}\}_{n=1}^{N}$ or a random subsample (with or without replacement) of $\{\mathbf{x}_{\sim i}^{(n)}\}_{n=1}^{N}$ when N is large.

Now assume that the output is contaminated by additive random error: $y^{(i)} = h(\mathbf{x}^{(i)}) + \epsilon_i$, where $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. An estimate of σ^2 can be obtained as in (11.4) using the full space for the nearest neighbor search:

$$\hat{\sigma}^2 = \frac{1}{N_O}\sum_{m=1}^{N_O}\left\{\frac{1}{N_I-1}\sum_{j=1}^{N_I}\left(y^{(k_{1:p}^{(m)}(j))} - \frac{1}{N_I}\sum_{l=1}^{N_I} y^{(k_{1:p}^{(m)}(l))}\right)^2\right\}. \tag{11.5}$$

Since

$$E[var\{y|\mathbf{x}_{\sim i}\}] = E[var\{h(\mathbf{x}) + \epsilon|\mathbf{x}_{\sim i}\}] = V_i^{tot} + \sigma^2,$$

we can get an unbiased estimate of V_i^{tot} from (11.4) by

$$V_i^{tot} = \frac{1}{N_O}\sum_{m=1}^{N_O}\left\{\frac{1}{N_I-1}\sum_{j=1}^{N_I}\left(y^{(k_{\sim i}^{(m)}(j))} - \frac{1}{N_I}\sum_{l=1}^{N_I} y^{(k_{\sim i}^{(m)}(l))}\right)^2\right\} - \hat{\sigma}^2. \tag{11.6}$$

This estimator is not guaranteed to be nonnegative. Therefore, if it becomes negative, we can take $V_i^{tot} = 0$.

The foregoing estimator can be used to develop a forward selection algorithm as follows. The variance of $h(\mathbf{x})$ explained by a set of factors u is given by

$$\begin{aligned} V_u &= var[E\{h(\mathbf{x})|\mathbf{x}_u\}] \\ &= var\{h(\mathbf{x})\} - E[var\{h(\mathbf{x})|\mathbf{x}_u\}] \\ &= var(y) - E[var\{h(\mathbf{x}) + \epsilon|\mathbf{x}_u\}]. \end{aligned}$$

Thus, V_u can be estimated by

$$\hat{V}_u = \widehat{var}(y) - \frac{1}{N_O}\sum_{m=1}^{N_O}\left\{\frac{1}{N_I-1}\sum_{j=1}^{N_I}\left(y^{(k_u^{(m)}(j))} - \frac{1}{N_I}\sum_{l=1}^{N_I} y^{(k_u^{(m)}(l))}\right)^2\right\}. \tag{11.7}$$

We can now choose factors one-by-one that maximizes $\hat{V}_u$. Huang and Joseph (2025a) also uses a backward elimination to further enhance the selection procedure. The forward and backward elimination strategies also mitigate the curse of dimensionality with nearest-neighbor methods. They named the overall algorithm as FIRST, which stands for Factor Importance Ranking and Selection using Total indices. The algorithm is implemented in the

R package `first` (Huang and Joseph, 2025b) and the Python package `pyfirst`[1]. `FIRST` gave the best selection performance and was the fastest among the state-of-the-art techniques for factor selection in the extensive simulations performed by Huang and Joseph (2025a).

The ranking for the selected factors can be done by removing the factor with the smallest FIRST index one-by-one, and reversing the order of removal. For example, consider the linear model in (11.2). We generate 1,000 data points with $\rho = 0.9$. The total Sobol' indices can be obtained using the `first` function in the R package `first`: $\hat{T}_1^{tot} = 0.060$, $\hat{T}_2^{tot} = 0.026$, $\hat{T}_3^{tot} = 0.114$. Thus, x_2 is the least important and is removed. Now using only x_1 and x_3, we obtain $\hat{T}_1 = 0.886$ and $\hat{T}_3^{tot} = 0.109$. Therefore, x_3 should be removed, and now only x_1 remains. Thus, the ranking is $x_1 \succ x_3 \succ x_2$, just as expected.

Now consider a real dataset: the concrete compressive strength dataset from Yeh (1998) which can be obtained from the UCI Machine Learning Repository (Dua and Graff, 2017). This dataset has $N = 1,030$ rows with eight continuous predictors pertaining to concrete ingredients and age, and one response: compressive strength. First, we split the dataset into training and testing sets in the ratio 95:5 using `SPlit` (Vakayil et al., 2022). Then we fit a random forest model and a Gaussian process model (OK) with Gaussian correlation function to the training data with the full set of eight predictors and using the predictors selected using `FIRST` (about four factors). The random forest was fitted using the R package `randomForest` (Liaw and Wiener, 2002) and the GP model using `rkriging` (Huang and Joseph, 2024). The root mean squared prediction errors (RMSE) are then computed on the testing set. This process is repeated 30 times and the resulting RMSEs are shown as boxplots in Figure 11.2. Although the RMSEs are higher with the selected set of predictors compared to the full set of predictors, this might be a reasonable performance considering that only about half the predictors have been used.

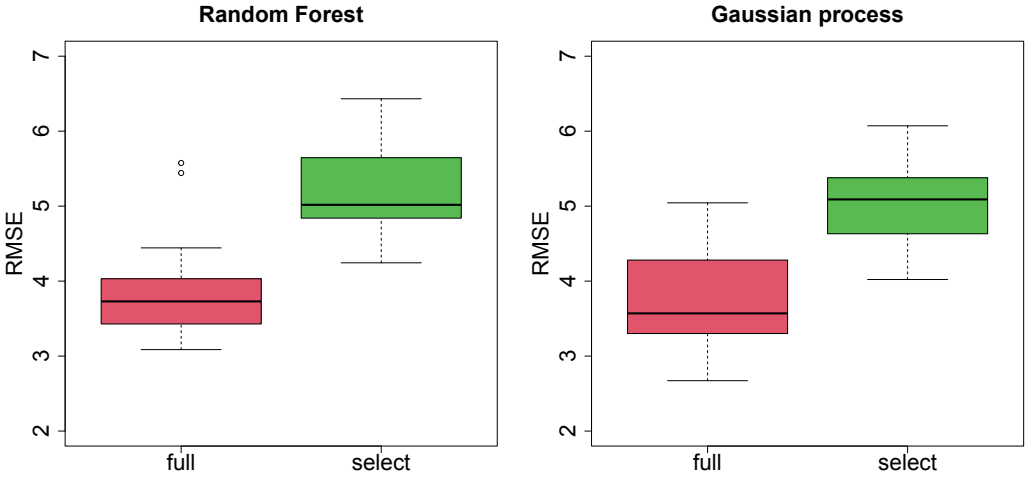


FIGURE 11.2: The root mean squared prediction errors on the testing set of the concrete dataset using random forest (left) and Gaussian process (right) using the full set of eight predictors and the selected set of factors using `FIRST`.

[1] https://github.com/BillHuang01/pyfirst

11.2 Twin-Gaussian Process

Consider the Gaussian process regression problem that we discussed in Section 2.2.3:

$$y = h(\mathbf{x}) + \epsilon,\ h(\mathbf{x}) \sim GP(\mu, \tau^2 R(\cdot)),\ \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2),$$

but this time with large amounts of data $(\mathbf{X}_N, \mathbf{Y}_N) = \{(\mathbf{X}^{(i)}, Y^{(i)})\}_{i=1}^N$. For example, in Chapter 10, we saw the wind power data (Ding, 2019) that had $N = 39,195$ points. It was too big for a Gaussian process model, and therefore, we obtained a subsample of 392 points using support points and fitted the GP model. The fitting on the subsample using `rkriging` took only about 0.3 seconds in a 2.10 GHz desktop computer. Since we know that the computational time for fitting a GP scales at $\mathcal{O}(N^3)$, we can estimate the time it would have taken to fit on the full dataset as $0.3 \times (39195/392)^3$, which is approximately 3.5 days! Thus, it is computationally expensive to fit a GP on this full dataset. However, as shown in Figure 10.2, the fitted GP model on the subsample is not good. Therefore, there is a need to develop GP to fit on large datasets, which has been an active area of research in statistics and machine learning (Gramacy, 2020, Ch. 9).

The computational bottleneck of GP comes from the need to take the inverse and determinant of the $N \times N$ correlation matrix $\mathbf{R}$. Therefore, some approaches try to simplify these computations using sparse matrix techniques with compactly-supported correlation functions (Furrer et al., 2006; Kaufman et al., 2011), but it compromises GP's prediction performance. Another approach is to partition the data region into smaller regions, fit GPs in each of the subregions, and then combine the predictions (Tresp, 2000; Gramacy and Lee, 2008; Cao and Fleet, 2014), but they can have discontinuities in the prediction. Park and Apley (2018) try to overcome the discontinuity issue using pseudo-observations at the boundaries of the subregions at the expense of increased computational cost. Another approach to dealing with the large dataset is to choose a small representative subsample from the full region, and fit a *global* GP on the subsample, which is the idea we used for the wind power dataset. This approach is closely related to sparse GP methods (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005; Titsias, 2009). An improvement to this approach is to build GPs on several subsamples and average them (Tresp, 2000; Deisenroth and Ng, 2015; Li and Mak, 2025). Gramacy and Apley (2015) proposed choosing a subset of data points depending on the prediction location and fit a *local* GP. Another popular approach uses Vecchia's approximation (Vecchia, 1988), where likelihood is factorized into a product of univariate conditional distributions based on some ordering of the data (Katzfuss and Guinness, 2021).

Since the global and local subset of points can be obtained using experimental design methods, here we will explain the approach of Vakayil and Joseph (2022a), which combines the global and local GP approximation methods. We can use `twinning` discussed in Section 10.3 to get a good and fast subsample from a large dataset. We can then build a global GP using this subsample. However, prediction errors can be large, as shown in Figure 10.2. Suppose that we want to predict the power when the wind speed is around 20 m/s. There are many data points around this value and we could have used them to develop a better prediction model. One way to do this is to obtain the residuals of the global GP and fit a local GP using only the points around 20 m/s. We can then add the prediction from the local GP to the global GP prediction to get the final prediction. Ba and Joseph (2012) have shown that such a two-stage GP predictor is equivalent to modeling the underlying function as the sum of two GPs, which is again equivalent to fitting a single GP with a correlation function of the form:

$$R(\boldsymbol{\Delta}) = (1 - \lambda)G(\boldsymbol{\Delta}) + \lambda L(\boldsymbol{\Delta}), \tag{11.8}$$

where $G(\cdot)$ is the correlation function for the global GP, $L(\cdot)$ the correlation function for the local GP, and $\lambda \in [0,1]$ is a weight factor to balance the predictions from the global and local GPs. Due to the twin datasets D_g and D_l, and twin correlation functions $G(\cdot)$ and $L(\cdot)$, Vakayil and Joseph (2024a) call this method TwinGP.

Denote the n_g global training points obtained using `twinning` as $D_g = \{\mathbf{x}_i\}_{i=1}^{n_g}$ and the l local training points obtained around the prediction point $\mathbf{x}$ as $D_l(\mathbf{x}) = \{\mathbf{X}_{\mathcal{L}(\mathbf{x})}\}$, where $\mathcal{L}(\mathbf{x})$ denotes the set of local point indices. Gramacy and Apley (2015) describe sequential design methods to carefully select local points, but we avoid this step to save computational time. Instead, given the testing location $\mathbf{x}$, we obtain D_l as the l nearest neighbors to $\mathbf{x}$ in $\{\mathbf{X}_i\}_{i=1}^{N} \setminus D_g$, a task that can be performed efficiently with kd-tree in $\mathcal{O}(l \log l)$. We will now fit a GP on the combined set $D_c(\mathbf{x}) = D_g \cup D_l(\mathbf{x})$, which contains $n = n_g + n_l$ points. For notational simplicity, we will denote $D_l(\mathbf{x})$ by D_l and $D_c(\mathbf{x})$ by D_c.

Let $\mathbf{Y}_c$ be the $n \times 1$ vector of response values corresponding to D_c. Thus, given $\mathbf{Y}_c$, we can obtain the posterior distribution as

$$h(\mathbf{x})|\mathbf{Y}_c \sim \mathcal{N}(\hat{y}(\mathbf{x}), s^2(\mathbf{x})),$$

where

$$\begin{aligned}
\hat{y}(\mathbf{x}) &= \hat{\mu}_n + \mathbf{r}_c(\mathbf{x})'(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}(\mathbf{Y}_c - \hat{\mu}_n\mathbf{1}_n),\\
s^2(\mathbf{x}) &= \tau_n^2\left\{1 + \eta - \mathbf{r}_c(\mathbf{x})'(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}\mathbf{r}_c(\mathbf{x})\right\},\\
\hat{\mu}_n &= \frac{\mathbf{1}_n'(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}\mathbf{Y}_c}{\mathbf{1}_n'(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}\mathbf{1}_n},\\
\hat{\tau}_n^2 &= \frac{1}{n}(\mathbf{Y}_c - \hat{\mu}_n\mathbf{1}_n)'(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}(\mathbf{Y}_c - \hat{\mu}_n\mathbf{1}_n),
\end{aligned}$$

where $\mathbf{r}_c(\mathbf{x})$ is the $n \times 1$ correlation vector, $\mathbf{R}_{cc}$ the $n \times n$ correlation matrix of the combined set $\mathbf{Y}_c$ obtained using the correlation function $R(\cdot)$, and $\eta = \sigma^2/\tau_n^2$. The difficulty with this approach is that we need to compute $(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1}$ at each prediction location $\mathbf{x}$, which can become expensive if the testing set is large. However, this computation can be simplified as follows. First, partition the matrix $\mathbf{R}_{cc}$ as

$$\begin{aligned}
\mathbf{R}_{cc} &= \begin{bmatrix} \mathbf{R}_{gg} & \mathbf{R}_{gl} \\ \mathbf{R}_{lg} & \mathbf{R}_{ll} \end{bmatrix}\\
&= \begin{bmatrix} (1-\lambda)\mathbf{G}_{gg} + \lambda\mathbf{L}_{gg} & (1-\lambda)\mathbf{G}_{gl} + \lambda\mathbf{L}_{gl} \\ (1-\lambda)\mathbf{G}_{lg} + \lambda\mathbf{L}_{lg} & (1-\lambda)\mathbf{G}_{ll} + \lambda\mathbf{L}_{ll} \end{bmatrix},
\end{aligned}$$

where $\mathbf{R}_{gg}$ is the $n_g \times n_g$ correlation matrix corresponding to $\mathbf{Y}_g$, $\mathbf{R}_{ll}$ is the $n_l \times n_l$ correlation matrix corresponding to $\mathbf{Y}_l$, and $\mathbf{R}_{gl} = \mathbf{R}_{lg}'$ is the $n_g \times n_l$ cross-correlation matrix. The matrices $\mathbf{G}_{gg}$, $\mathbf{L}_{gg}$, etc. are defined similarly using the correlation functions $G(\cdot)$ and $L(\cdot)$ in (11.8). Now, using the formula for the inverse of partitioned matrices (Lemma 2.1), we obtain

$$(\mathbf{R}_{cc} + \eta\mathbf{I}_n)^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{gg}^{-1} + \boldsymbol{\Sigma}_{gg}^{-1}\mathbf{R}_{gl}\mathbf{S}^{-1}\mathbf{R}_{lg}\boldsymbol{\Sigma}_{gg}^{-1} & -\boldsymbol{\Sigma}_{gg}^{-1}\mathbf{R}_{gl}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{R}_{lg}\boldsymbol{\Sigma}_{gg}^{-1} & \mathbf{S}^{-1} \end{bmatrix}, \tag{11.9}$$

where $\boldsymbol{\Sigma}_{gg} = \mathbf{R}_{gg} + \eta\mathbf{I}_g$, and $\mathbf{S} = \mathbf{R}_{ll} - \mathbf{R}_{lg}\boldsymbol{\Sigma}_{gg}^{-1}\mathbf{R}_{gl}$. Since $\mathbf{R}_{gg} = (1-\lambda)\mathbf{G}_{gg} + \lambda\mathbf{L}_{gg}$ is independent of the testing location $\mathbf{x}$, $\boldsymbol{\Sigma}_{gg}^{-1}$ can be predetermined before testing. Therefore, we only need to invert the small $n_l \times n_l$ matrix $\mathbf{S}$ per testing location, which can be done efficiently.

Vakayil and Joseph (2024a) proposed to use power exponential correlation function (see Table 2.1) for $G(\cdot)$. Its correlation parameters can be computed as in (2.22) using only the

global set. For the local correlation function, they proposed to use Wendland's compactly supported radial function given by(Wendland, 2004),

$$L(\boldsymbol{\Delta}) = \left(q\frac{\|\boldsymbol{\Delta}\|}{\theta_l} + 1\right) \max\left\{0, \left(1 - \frac{\|\boldsymbol{\Delta}\|}{\theta_l}\right)\right\}^q, \; q = \lfloor p/2 \rfloor + 3,$$

where $\lfloor u \rfloor$ is the largest integer less than or equal to u. The local correlation function has only one hyperparameter θ_l which can be set as the fill distance of D_g as defined in (4.8). This choice ensures that for any testing location $\mathbf{x}$, there exists at least one global training point such that the local kernel is active between them, i.e., there exists an $\mathbf{x}_i \in D_g$ such that $L(\mathbf{x} - \mathbf{x}_i) > 0$ for all $\mathbf{x}$.

Consider again the wind power dataset with $N = 39,195$ points. `TwinGP` is implemented in the R package `twingp` (Vakayil and Joseph, 2024b). By default, it uses $n_g \approx \sqrt{N}$ and $n_l = 25$ so that the computational cost of fitting the GP is $\mathcal{O}(N^{1.5})$. The predictions on a 1,000-point equally-spaced testing set is shown in Figure 11.3. Although not as smooth as the predictions in Figure 10.2, the accuracy is much better. The overall fitting and prediction took less than 0.20 seconds, which clearly shows that TwinGP can be used efficiently to fit GPs with large datasets.

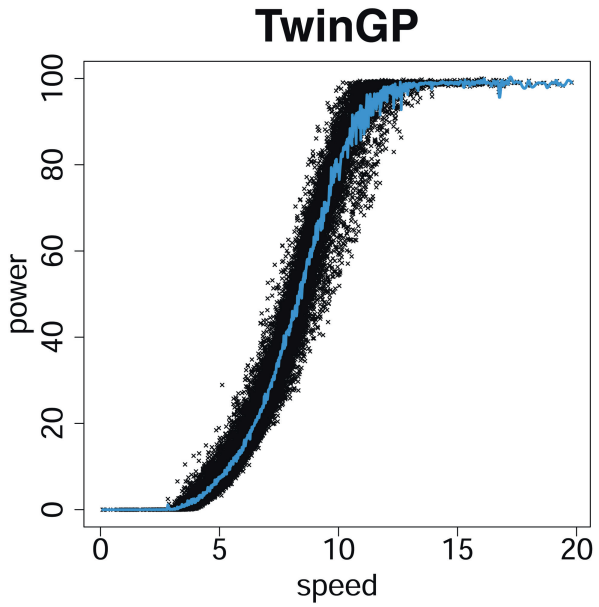


FIGURE 11.3: Prediction from `TwinGP` is shown as a solid line for the wind power dataset.

Exercises

1. Let $\mathbf{X} = (X_1, \ldots, X_p) \in \mathcal{X} \subseteq \mathbb{R}^p$ be a p-dimensional random vector with joint distribution $\mathbb{P}_{\mathcal{X}}$ and $Y = h(\mathbf{X}_{A^*}) \in \mathcal{Y} \subseteq \mathbb{R}$ be a scalar output depending only on the input $\mathbf{X}_{A^*}$ where $A^* \subseteq 1{:}p$ is the set of indices of the important factors, i.e., $S_i^{\text{tot}} > 0$ for $i \in A^*$ and

$S_j^{\text{tot}} = 0$ for $j \notin A^*$. Let $A \subseteq A^*$ and V_A be the model variance that can be explained by $\mathbf{X}_A$, i.e., $V_A = \text{Var}[\mathbb{E}\{h(\mathbf{X})|\mathbf{X}_A\}]$. Show that if $V_{A\cup\{i\}} > V_A$, then $S_i^{\text{tot}} > 0$ (Huang and Joseph, 2025a). This justifies the use of forward selection using $\hat{V}_u$ in (11.7).

2. Consider the coefficient proposed by Azadkia and Chatterjee (2021) to quantify the degree of conditional dependence between Y and Z given X:

$$\xi(Y, Z|X) = \frac{\int \mathbb{E}\{\text{Var}(\mathbb{E}[1_{Y\geq t}|Z, X]|X)\}d\mu(t)}{\int \mathbb{E}[\text{Var}(1_{Y\geq t}|X)]d\mu(t)},$$

where μ is the law of Y. This coefficient is bounded between 0 and 1, with 0 indicating that Y and Z are conditionally independent given X, and 1 if and only if Y is some measurable function of Z given X. Consider the case of binary output $Y = h(\mathbf{X}_u, \mathbf{X}_{\sim u}) \in \{0, 1\}$. Show that the numerator of $\xi(Y, \mathbf{X}_u|\mathbf{X}_{\sim u})$ is the same as the numerator of the total Sobol' index (Huang and Joseph, 2025a):

$$\int \mathbb{E}\{\text{Var}(\mathbb{E}[1_{Y\geq t}|\mathbf{X}_u, \mathbf{X}_{\sim u}]|\mathbf{X}_{\sim u})\}d\mu(t) = V_u^{tot}.$$

This result shows that FIRST can be used for factor selection and ranking in classification problems as well.

3. Consider the meat spectroscopy data of 215 samples given in the R package `faraway` (Faraway, 2025). For each sample, the fat content was measured along with a 100-channel spectrum of absorbances. Since determining the fat content via analytical chemistry is time consuming, it will be good to build a model to predict the fat content of new samples using the 100 absorbances which can be measured more easily. Split the dataset in 80:20 ratio and fit a random forest and XGBoost on the training set. Compare their prediction performance on the testing set with the full set of 100 factors and using the factors selected by FIRST. Repeat this procedure 30 times and show the results using boxplots.

4. Consider the ozone column dataset which contains measurement of total column of ozone by the NIMBUS-7/TOMS satellite on October 1, 1988, at different latitudes and longitudes over the globe. There are $182,591$ observations with $d = 2$ inputs, latitude and longitude. The response is the total column of ozone. The data is available in the file `ozone.csv`. Split the data in 90:10 ratio and fit the following GP models on the training set.

   (a) laGP (Gramacy and Apley, 2015)

   (b) TwinGP (Vakayil and Joseph, 2024a).

   Compute the root mean squared prediction error and the interval score in (2.53) with $\alpha = 0.05$ on the testing set. Repeat the procedure 30 times and compare the performance of the two methods, including their computational time.

Bibliography

An, J. and A. Owen (2001). Quasi-regression. *Journal of Complexity 17*(4), 588–607.

Atkinson, A., A. Donev, and R. Tobias (2007). *Optimum experimental designs, with SAS*. Oxford University Press, Oxford.

Audze, P. and V. Eglais (1977). New approach for planning out of experiments. *Problems of Dynamics and Strengths 35*, 104–107.

Azadkia, M. and S. Chatterjee (2021). A simple measure of conditional dependence. *The Annals of Statistics 49*(6), 3070–3102.

Ba, S. (2015). *SLHD: maximin-distance (sliced) Latin hypercube designs*. R package version 2.1-1.

Ba, S. and V. R. Joseph (2012). Composite Gaussian process models for emulating expensive functions. *The Annals of Applied Statistics 6*, 1838–1860.

Ba, S. and V. R. Joseph (2018). *MaxPro: Maximum Projection Designs*. R package version 4.1-2.

Ba, S., W. R. Myers, and W. A. Brenneman (2015). Optimal sliced Latin hypercube designs. *Technometrics 57*(4), 479–487.

Banerjee, S., B. P. Carlin, and A. E. Gelfand (2003). *Hierarchical modeling and analysis for spatial data*. Chapman and Hall/CRC.

Barry, N., M. Chatzos, W. Chen, D. Han, C. Huang, V. R. Joseph, M. Klamkin, S. Park, M. Tanneau, P. Van Hentenryck, et al. (2022). Risk-aware control and optimization for high-renewable power grids. *arXiv preprint arXiv:2204.00950*.

Barton, R. R. and M. D. Morris (2025). Experimental design and modeling for forward-inverse maps. *Technometrics* 67(3), 367-381.

Bates, D., M. Maechler, M. Jagan, et al. (2025). *Matrix: Sparse and dense matrix classes and methods*. R package version 1.7-3.

Bates, D. M. and D. G. Watts (1988). *Nonlinear regression analysis and its applications*. Wiley, New York.

Bates, R. A., B. Giglio, and H. P. Wynn (2003). A global selection procedure for polynomial interpolators. *Technometrics 45*(3), 246–255.

Bayarri, M. J., J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu (2007). A framework for validation of computer models. *Technometrics 49*(2), 138–154.

Ben-Ari, E. N. and D. M. Steinberg (2007). Modeling data from computer experiments: an empirical comparison of kriging with mars and projection pursuit regression. *Quality Engineering 19*(4), 327–338.

Benmeida, M. (2017 (accessed October 17, 2020)). *NYC Taxi Trip Durations: Data augmentation using OSRM*.

Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

Bettinger, R., P. Duchêne, L. Pronzato, and E. Thierry (2008). Design of experiments for response diversity. In *Journal of Physics: Conference Series*, Volume 135, pp. 012017. IOP Publishing.

Binois, M., R. B. Gramacy, and M. Ludkovski (2018). Practical heteroscedastic Gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics 27*(4), 808–821.

Binois, M., J. Huang, R. B. Gramacy, and M. Ludkovski (2019). Replication or exploration? sequential design for stochastic simulation experiments. *Technometrics 61*(1), 7–23.

Borgonovo, E. (2007). A new uncertainty importance measure. *Reliability Engineering & System Safety 92*(6), 771–784.

Borkowski, J. J. and G. F. Piepel (2009). Uniform designs for highly constrained mixture experiments. *Journal of Quality Technology 41*(1), 35–47.

Borodachov, S., D. Hardin, and E. Saff (2008a). Asymptotics for discrete weighted minimal riesz energy problems on rectifiable sets. *Transactions of the American Mathematical Society 360*(3), 1559–1580.

Borodachov, S., D. Hardin, and E. Saff (2008b). Asymptotics of weighted best-packing on rectifiable sets. *Sbornik: Mathematics 199*(11), 1579.

Borodachov, S. V., D. P. Hardin, and E. B. Saff (2019). *Discrete energy on rectifiable sets*, Volume 4. Springer.

Bowman, V. E. and D. C. Woods (2013). Weighted space-filling designs. *Journal of Simulation 7*(4), 249–263.

Box, G. E. P. and D. R. Cox (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology 26*(2), 211–243.

Box, G. E. P. and J. S. Hunter (1961). The 2^{k-p} fractional factorial designs. *Technometrics 3*(3), 311–351.

Box, G. E. P., S. Hunter, and W. Hunter (1978). *Statistics for experimenters*. John Wiley and sons New York.

Box, G. E. P. and W. G. Hunter (1965). Sequential design of experiments for nonlinear models. In *Proc. IBM Scientific Computing Symposium in Statistics, 1965*, pp. 113–137.

Box, G. E. P. and K. B. Wilson (1951). On the experimental attainment of optimum conditions. *Journal of Royal Statistical Society–Series B 13*, 1–45.

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics 37*(4), 373–384.

Briol, F.-X., C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic (2019). Probabilistic integration. *Statistical Science 34*(1), 1–22.

Brooks, T. F., D. S. Pope, and M. A. Marcolini (1989). *Airfoil self-noise and prediction*, Volume 1218. Washington, DC: NASA.

Broto, B., F. Bachoc, and M. Depecker (2020). Variance reduction for estimation of shapley effects and adaptation to unknown input distribution. *SIAM/ASA Journal on Uncertainty Quantification 8*(2), 693–716.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research 12*(10), 2879–2904.

Cameron, A., J. Casey, and G. Simpson (1986). *Benchmark tests for therminal analyses (summary)*. NAFEMS.

Campolongo, F., J. Cariboni, and A. Saltelli (2007). An effective screening design for sensitivity analysis of large models. *Environmental modelling & software 22*(10), 1509–1518.

Campolongo, F., A. Saltelli, and J. Cariboni (2011). From screening to quantitative sensitivity analysis. a unified approach. *Computer physics communications 182*(4), 978–988.

Cao, Y. and D. J. Fleet (2014). Generalized product of experts for automatic and principled fusion of Gaussian process predictions. *arXiv preprint arXiv:1410.7827*.

Cappé, O., A. Guillin, J.-M. Marin, and C. P. Robert (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics 13*(4), 907–929.

Cardot, H. (2022). *Gmedian: Geometric Median, k-Medians Clustering and Robust Median PCA*. R package version 1.2.7. available at https://CRAN.R-project.org/package=Gmedian.

Castillo, A. R., V. R. Joseph, and S. R. Kalidindi (2019). Bayesian sequential design of experiments for extraction of single-crystal material properties from spherical indentation measurements on polycrystalline samples. *JOM 71*(8), 2671–2679.

Chaloner, K. and K. Larntz (1989). Optimal Bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inference 21*(2), 191–208.

Chang, C.-J. and V. R. Joseph (2014). Model calibration through minimal adjustments. *Technometrics 56*(4), 474–482.

Chen, W., R. Jin, and A. Sudjianto (2005). Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty. *Journal of mechanical design 127*(5), 875–886.

Chen, Y., M. Welling, and A. Smola (2012). Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*.

Cheng, C.-S. (2016). *Theory of factorial design*. Chapman and Hall/CRC Boca Raton, FL, USA.

Cheng, C.-S., D. M. Steinberg, and D. X. Sun (1999). Minimum aberration and model robustness for two-level fractional factorial designs. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61*(1), 85–93.

Cheng, S.-W. and K. Q. Ye (2004). Geometric isomorphism and minimum aberration for factorial designs with quantitative factors. *The Annals of Statistics 32*, 2168–2185.

Chernoff, H. (1953). Locally optimal designs for estimating parameters. *The Annals of Mathematical Statistics 24*(4), 586–602.

Chipman, H., M. Hamada, and C. F. J. Wu (1997). A Bayesian variable-selection approach for analyzing designed experiments with complex aliasing. *Technometrics 39*(4), 372–381.

Chitre, A., D. Semochkina, D. C. Woods, and A. A. Lapkin (2025). Machine learning-guided space-filling designs for high throughput liquid formulation development. *Computers & Chemical Engineering*, 109007.

Choi, N. H., W. Li, and J. Zhu (2010). Variable Selection With the Strong Heredity Constraint and Its Oracle Property. *Journal of the American Statistical Association 105*(489), 354–364.

Cohn, D. A. (1996). Neural network exploration using optimal experiment design. *Neural networks 9*(6), 1071–1083.

Conway, J. H. and N. J. A. Sloane (2013). *Sphere packings, lattices and groups*, Volume 290. Springer Science & Business Media.

Cook, R. D. and C. J. Nachtsheim (1980). A comparison of algorithms for constructing exact D-optimal designs. *Technometrics 22*(3), 315–324.

Cooley, S. K. and G. F. Piepel (2003). A two-stage layered mixture experiment design for a nuclear waste glass application- parts 1 and 2. In *Proceedings of the American Statistical Association*, pp. 1044–1051.

Cornell, J. A. (1990). *Experiments with mixtures: designs, models, and the analysis of mixture data, 2nd Edition*. John Wiley & Sons.

Cover, T. M. and J. A. Thomas (1991). *Elements of information theory*. John Wiley & Sons.

Cox, D. R. (1957). Note on grouping. *Journal of the American Statistical Association 52*(280), 543–547.

Cressie, N. (1990). The origins of kriging. *Mathematical geology 22*, 239–252.

Cressie, N. (1993). *Statistics for spatial data*. John Wiley & Sons.

Currin, C., T. Mitchell, M. Morris, and D. Ylvisaker (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association 86*(416), 953–963.

Da Veiga, S., F. Gamboa, B. Iooss, and C. Prieur (2021). *Basics and Trends in Sensitivity Analysis: Theory and Practice in R*. SIAM.

Dadashova, K., R. C. Smith, and M. A. Haider (2024). Local identifiability analysis, parameter subset selection and verification for a minimal brain pbpk model. *Bulletin of Mathematical Biology 86*(2), 12.

Dalenius, T. (1950). The problem of optimum stratification. *Scandinavian Actuarial Journal 1950*(3-4), 203–213.

Damianou, A. and N. D. Lawrence (2013). Deep Gaussian processes. In *Artificial intelligence and statistics*, pp. 207–215. PMLR.

Dancik, G. M. (2022). *mlegp: Maximum Likelihood Estimates of Gaussian Processes*. R package version 3.1.9.

Daniel, C. (1973). One-at-a-time plans. *Journal of the American statistical association 68*(342), 353–360.

Dasgupta, T., B. Weintraub, and V. R. Joseph (2011). A physical–statistical model for density control of nanowires. *IIE Transactions 43*(4), 233–241.

Dean, A., D. Voss, and D. Draguljić (2017). *Design and analysis of experiments, Second Edition*. Springer.

Deisenroth, M. and J. W. Ng (2015). Distributed Gaussian processes. In *International Conference on Machine Learning*, pp. 1481–1490. PMLR.

Deng, X., Y. Hung, and C. D. Lin (2015). Design for computer experiments with qualitative and quantitative factors. *Statistica Sinica*, 1567–1581.

Deng, X., V. R. Joseph, W. Mai, Z. L. Wang, and C. F. J. Wu (2009). Statistical approach to quantifying the elastic deformation of nanomaterials. *Proceedings of the National Academy of Sciences 106*(29), 11845–11850.

Der Kiureghian, A. and O. Ditlevsen (2009). Aleatory or epistemic? does it matter? *Structural safety 31*(2), 105–112.

Dette, H., X. Liu, and R.-X. Yue (2022). Design admissibility and de la garza phenomenon in multifactor experiments. *The Annals of Statistics 50*(3), 1247–1265.

Dette, H. and A. Pepelyshev (2010). Generalized Latin hypercube design for computer experiments. *Technometrics 52*(4), 421–429.

Dette, H. and W. Studden (1992). On a new characterization of the classical orthogonal polynomials. *Journal of approximation theory 71*(1), 3–17.

Diaconis, P. (1988). Bayesian numerical analysis. *Statistical decision theory and related topics IV 1*, 163–175.

Diggle, P. J. and P. J. Ribeiro Jr (2007). *Model-based Geostatistics*. Springer.

Ding, Y. (2019). *Data science for wind energy*. Chapman and Hall/CRC.

Draguljić, D., T. J. Santner, and A. M. Dean (2012). Noncollapsing space-filling designs for bounded nonrectangular regions. *Technometrics 54*(2), 169–178.

Dror, H. A. and D. M. Steinberg (2006). Robust experimental design for multivariate generalized linear models. *Technometrics 48*(4), 520–529.

Dua, D. and C. Graff (2017). *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml.

Dupačová, J., N. Gröwe-Kuska, and W. Römisch (2003). Scenario reduction in stochastic programming. *Mathematical programming 95*, 493–511.

Elfving, G. (1952). Optimum allocation in linear regression theory. *The Annals of Mathematical Statistics*, 255–262.

Fampa, M. and J. Lee (2022). *Maximum-entropy sampling: algorithms and application.* Springer Nature.

Fang, K.-T., R. Li, and A. Sudjianto (2005). *Design and modeling for computer experiments.* Chapman and Hall/CRC.

Fang, K.-T. and R. Mukerjee (2000). A connection between uniformity and aberration in regular fractions of two-level factorials. *Biometrika 87*(1), 193–198.

Fang, K.-T. and Y. Wang (1993). *Number-theoretic methods in statistics*, Volume 51. CRC Press.

Fang, K.-T., H. Ye, and Y. Zhou (2025). *Representative Points of Statistical Distributions: Applications in Statistical Inference.* Chapman & Hall.

Faraway, J. J. (2015). *Linear Models with R, Second Edition.* Chapman and Hall/CRC.

Faraway, J. J. (2016). *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models, Second Edition.* Chapman and Hall/CRC.

Faraway, J. J. (2025). *faraway: Datasets and Functions for Books by Julian Faraway.* R package version 1.0.9.

Fasshauer, G. E. (2007). *Meshfree approximation methods with MATLAB*, Volume 6. World Scientific.

Fasshauer, G. E. and M. J. McCourt (2015). *Kernel-based approximation methods using Matlab*, Volume 19. World Scientific Publishing Company.

Fedorov, V. V. (1972). *Theory of optimal experiments (Edited and Translated by W. J. Studden and E. M. Klimko).* Academic Press.

Fedorov, V. V. and S. L. Leonov (2013). *Optimal design for nonlinear response models.* CRC Press.

Feldman, D. (2019). Core-sets: Updated survey. In *Sampling techniques for supervised or unsupervised tasks*, pp. 23–44. Springer.

Flury, B. A. (1990). Principal points. *Biometrika 77*(1), 33–41.

Franco, J., D. Dupuy, O. Roustant, P. Kiener, G. Damblin, and B. Iooss (2023). *DiceDesign: Designs of Computer Experiments.* R package version 1.10.

Frazier, P. I. (2018). Bayesian optimization. *INFORMS TutORials in Operations Research*, 255–278.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 1–67.

Fries, A. and W. Hunter (1980). Minimum aberration 2^{k-p} designs. *Technometrics 22*(4), 601–608.

Furrer, R., M. G. Genton, and D. Nychka (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics 15*(3), 502–523.

Garnett, R. (2023). *Bayesian optimization.* Cambridge University Press.

Gilbert, P. and R. Varadhan (2019). *numDeriv: Accurate Numerical Derivatives*. R package version 2016.8-1.1.

Ginsbourger, D., R. Le Riche, and L. Carraro (2010). Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems*, pp. 131–162. Springer.

Gneiting, T. and A. E. Raftery (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association 102*(477), 359–378.

Godichon-Baggioni, A. and S. Surendran (2023). *Kmedians: K-Medians*. R package version 2.2.0. available at https://CRAN.R-project.org/package=Kmedians.

Golchi, S. and J. L. Loeppky (2015). Monte Carlo based designs for constrained domains. *arXiv preprint arXiv:1512.07328*.

Golub, G. H., M. Heath, and G. Wahba (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics 21*(2), 215–223.

Goos, P. and B. Jones (2011). *Optimal design of experiments: a case study approach*. John Wiley & Sons.

Graf, S. and H. Luschgy (2007). *Foundations of quantization for probability distributions*. Springer.

Grafström, A., N. L. P. Lundström, and L. Schelin (2012). Spatially balanced sampling through the pivotal method. *Biometrics 68*(2), 514–520.

Grafström, A. and W. Prentius (2024). *BalancedSampling: Balanced and Spatially Balanced Sampling*. R package version 2.1.1.

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC.

Gramacy, R. B. and D. W. Apley (2015). Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics 24*(2), 561–578.

Gramacy, R. B. and H. K. H. Lee (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association 103*(483), 1119–1130.

Gramacy, R. B. and H. K. H. Lee (2009). Adaptive design and analysis of supercomputer experiments. *Technometrics 51*(2), 130–145.

Gramacy, R. B. and H. K. H. Lee (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing 22*, 713–722.

Gramacy, R. B., A. Sauer, and N. Wycoff (2022). Triangulation candidates for Bayesian optimization. *Advances in Neural Information Processing Systems 35*, 35933–35945.

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola (2012). A kernel two-sample test. *The Journal of Machine Learning Research 13*(1), 723–773.

Groemping, U. (2023). *FrF2: Fractional Factorial Designs with 2-Level Factors*. R package version 2.3-3.

Groemping, U. (2025). *DoE.base: Full Factorials, Orthogonal Arrays and Base Utilities for DoE Packages*. R package version 1.2-5.

Gu, M. and L. Wang (2018). Scaled Gaussian stochastic process for computer model calibration and prediction. *SIAM/ASA Journal on Uncertainty Quantification 6*(4), 1555–1583.

Gul, E., V. R. Joseph, H. Yan, and S. N. Melkote (2018). Uncertainty quantification of machining simulations using an in situ emulator. *Journal of Quality Technology 50*(3), 253–261.

Haario, H., E. Saksman, and J. Tamminen (1999). Adaptive proposal distribution for random walk metropolis algorithm. *Computational statistics 14*, 375–395.

Hamada, M. and C. F. J. Wu (1992). Analysis of designed experiments with complex aliasing. *Journal of quality technology 24*(3), 130–137.

Hart, J. and P. A. Gremaud (2018). An approximation theoretic perspective of sobol'indices with dependent variables. *International Journal for Uncertainty Quantification 8*(6).

He, X. (2017a). Interleaved lattice-based minimax distance designs. *Biometrika 104*(3), 713–725.

He, X. (2017b). Rotated sphere packing designs. *Journal of the American Statistical Association 112*(520), 1612–1622.

Hedayat, A. S., N. J. A. Sloane, and J. Stufken (1999). *Orthogonal arrays: theory and applications*. Springer, New York.

Hennig, P., M. A. Osborne, and H. P. Kersting (2022). *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press.

Heo, J. and C.-L. Sung (2021). *RNAmf: Recursive Non-Additive Emulator for Multi-Fidelity Data*. R package version 1.1.1.

Heo, J. and C.-L. Sung (2025). Active learning for a recursive non-additive emulator for multi-fidelity computer experiments. *Technometrics 67*(1), 58–72.

Hickernell, F. J. (1998a). A generalized discrepancy and quadrature error bound. *Mathematics of computation 67*(221), 299–322.

Hickernell, F. J. (1998b). Lattice rules: how well do they measure up? In *Random and quasi-random point sets*, pp. 109–166. Springer.

Hickernell, F. J. (1999). Goodness-of-fit statistics, discrepancies and robust designs. *Statistics & probability letters 44*(1), 73–78.

Hildebrand, F. B. (1987). *Introduction to numerical analysis*. Dover Publications, New York.

Hoerl, A. E. and R. W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*(1), 55–67.

Homma, T. and A. Saltelli (1996). Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety 52*(1), 1–17.

Huan, X. and Y. M. Marzouk (2013). Simulation-based optimal Bayesian experimental design for nonlinear systems. *Journal of Computational Physics 232*(1), 288–317.

Huang, C., S. Aduloju, J. Fritz, V. R. Joseph, M. Messner, and T. Truster (2025). Calibration of rafm micromechanical model for creep using Bayesian optimization for functional output. *Journal of Computing and Information Science in Engineering* 25(3): 031007.

Huang, C. and V. R. Joseph (2022). *supercompress: Supervised Compression of Big Data*. R package version 1.1.

Huang, C. and V. R. Joseph (2024). *rkriging: Kriging Modeling*. R package version 1.0.

Huang, C. and V. R. Joseph (2025a). Factor importance ranking and selection using total indices. *Technometrics 67*(4), 573–589.

Huang, C. and V. R. Joseph (2025b). *first: Factor Importance Ranking and Selection using Total Indices*. R package version 2.0.

Huang, C., V. R. Joseph, and S. Mak (2022). Population quasi-Monte Carlo. *Journal of Computational and Graphical Statistics 31*(3), 695–708.

Huang, C., V. R. Joseph, and D. M. Ray (2021). Constrained minimum energy designs. *Statistics and Computing 31*(6), 80.

Huang, C., Y. Ren, E. K. McGuinness, M. D. Losego, R. P. Lively, and V. R. Joseph (2021). Bayesian optimization of functional output in inverse problems. *Optimization and Engineering 22*, 2553–2574.

Hung, Y., V. R. Joseph, and S. N. Melkote (2009). Design and analysis of computer experiments with branching and nested factors. *Technometrics 51*(4), 354–365.

Hung, Y., V. R. Joseph, and S. N. Melkote (2015). Analysis of computer experiments with functional response. *Technometrics 57*(1), 35–44.

Iooss, B., S. Da Veiga, A. Janon, and G. Pujol (2024). *Sensitivity: global sensitivity analysis of model outputs*. R package version 1.30.1.

Jansen, M. J. (1999). Analysis of variance designs for model output. *Computer Physics Communications 117*(1-2), 35–43.

Ji, Y., S. Mak, D. Soeder, J.-F. Paquet, and S. A. Bass (2024). A graphical multi-fidelity Gaussian process model, with application to emulation of heavy-ion collisions. *Technometrics 66*(2), 267–281.

Ji, Y., H. S. Yuchi, D. Soeder, J.-F. Paquet, S. A. Bass, V. R. Joseph, C. F. J. Wu, and S. Mak (2024). Conglomerate multi-fidelity Gaussian process modeling, with application to heavy-ion collisions. *SIAM/ASA Journal on Uncertainty Quantification 12*(2), 473–502.

Jin, R., W. Chen, and A. Sudjianto (2005). An efficient algorithm for constructing optimal design of computer experiments. *Journal of statistical planning and inference 134*(1), 268–287.

Joe, S. and F. Y. Kuo (2008). Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing 30*(5), 2635–2654.

Johnson, M. E., L. M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference 26*(2), 131–148.

Jones, B. and M. Lanzerath (2021). A Novel Application of a Definitive Screening Design: A Case Study. *Quality Engineering 33*(3), 563–569.

Jones, B. and C. J. Nachtsheim (2011). A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology 43*(1), 1–15.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization 21*, 345–383.

Jones, D. R., M. Schonlau, and W. J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization 13*, 455–492.

Joseph, V. R. (2006a). A Bayesian approach to the design and analysis of fractionated experiments. *Technometrics 48*(2), 219–229.

Joseph, V. R. (2006b). Limit kriging. *Technometrics 48*(4), 458–466.

Joseph, V. R. (2012). Bayesian computation using design of experiments-based interpolation technique. *Technometrics 54*(3), 209–225.

Joseph, V. R. (2013). A note on nonnegative doit approximation. *Technometrics 55*(1), 103–107.

Joseph, V. R. (2016). Space-filling designs for computer experiments: A review (with discussions). *Quality Engineering 28*(1), 28–44.

Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal 15*(4), 531–538.

Joseph, V. R. (2025). Rational kriging. *Journal of the American Statistical Association 120*(549), 548–558.

Joseph, V. R., M. Ai, and C. F. J. Wu (2009). Bayesian-inspired mixed two-and four-level designs. *Biometrika 96*, 95–106.

Joseph, V. R., T. Dasgupta, R. Tuo, and C. F. J. Wu (2015). Sequential exploration of complex surfaces using minimum energy designs. *Technometrics 57*(1), 64–74.

Joseph, V. R. and J. D. Delaney (2007). Functionally induced priors for the analysis of experiments. *Technometrics 49*(1), 1–11.

Joseph, V. R., L. Gu, S. Ba, and W. R. Myers (2019). Space-filling designs for robustness experiments. *Technometrics 61*(1), 24–37.

Joseph, V. R., E. Gul, and S. Ba (2015). Maximum projection designs for computer experiments. *Biometrika 102*(2), 371–380.

Joseph, V. R., E. Gul, and S. Ba (2020). Designing computer experiments with multiple types of factors: The MaxPro approach. *Journal of Quality Technology 52*(4), 343–354.

Joseph, V. R. and Y. Hung (2008). Orthogonal-maximin Latin hypercube designs. *Statistica Sinica*, 171–186.

Joseph, V. R., Y. Hung, and A. Sudjianto (2008). Blind kriging: A new method for developing metamodels. *ASME Journal of Mechanical Design 130*, 031102.

Joseph, V. R. and L. Kang (2011). Regression-based inverse distance weighting with applications to computer experiments. *Technometrics 53*(3), 254–265.

Joseph, V. R., W. E. Lewis, H. S. Yuchi, and K. A. Maupin (2024). Discovering the unknowns: A first step. *SIAM/ASA Journal on Uncertainty Quantification 12*(4), 1336–1348.

Joseph, V. R. and S. Mak (2021). Supervised compression of big data. *Statistical Analysis and Data Mining: The ASA Data Science Journal 14*(3), 217–229.

Joseph, V. R. and S. N. Melkote (2009). Statistical adjustments to engineering models. *Journal of Quality Technology 41*(4), 362–375.

Joseph, V. R. and A. Vakayil (2022). SPlit: An optimal method for data splitting. *Technometrics 64*(2), 166–176.

Joseph, V. R., D. Wang, L. Gu, S. Lyu, and R. Tuo (2019). Deterministic sampling of expensive posteriors using minimum energy designs. *Technometrics 61*(3), 297–308.

Joseph, V. R. and H. Yan (2015). Engineering-driven statistical adjustment and calibration. *Technometrics 57*(2), 257–267.

Kanagawa, M., P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur (2018). Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*.

Kane, A. and A. Mandal (2020). A new analysis strategy for designs with complex aliasing. *The American Statistician 74*(3), 274–281.

Kang, L. and V. R. Joseph (2009). Bayesian optimal single arrays for robust parameter design. *Technometrics 51*(3), 250–261.

Kang, L., V. R. Joseph, and W. A. Brenneman (2011). Design and modeling strategies for mixture-of-mixtures experiments. *Technometrics 53*(2), 125–136.

Katzfuss, M. and J. Guinness (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*.

Kaufman, C. G., D. Bingham, S. Habib, K. Heitmann, and J. A. Frieman (2011). Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics 5*(4), 2470 – 2492.

Kenett, R. S. and S. Zacks (2021). *Modern industrial statistics: With applications in R, MINITAB, and JMP*. John Wiley & Sons.

Kennard, R. W. and L. A. Stone (1969). Computer aided design of experiments. *Technometrics 11*(1), 137–148.

Kennedy, M. C. and A. O'Hagan (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika 87*(1), 1–13.

Kennedy, M. C. and A. O'Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63*(3), 425–464.

Kiefer, J. and J. Wolfowitz (1959). Optimum designs in regression problems. *The annals of mathematical statistics 30*(2), 271–294.

Kiefer, J. and J. Wolfowitz (1960). The equivalence of two extremum problems. *Canadian Journal of Mathematics 12*, 363–366.

Kleijnen, J. P. (2008). *Design and analysis of simulation experiments*. Springer.

Knapp, P., W. Lewis, V. R. Joseph, C. Jennings, and M. Glinsky (2023). Optimizing the configuration of plasma radiation detectors in the presence of uncertain instrument response and inadequate physics. *Journal of Plasma Physics 89*(1), 895890101.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy 52*(6), 119–139.

Krishna, A., S. R. Craig, C. Shi, and V. R. Joseph (2022). Inverse design of acoustic metasurfaces using space-filling points. *Applied Physics Letters 121*(7), 071701.

Krishna, A., V. R. Joseph, S. Ba, W. A. Brenneman, and W. R. Myers (2022). Robust experimental designs for model calibration. *Journal of Quality Technology 54*(4), 441–452.

Krishna, A., H. Tran, C. Huang, R. Ramprasad, and V. R. Joseph (2024). Adaptive exploration and optimization of materials crystal structures. *INFORMS Journal on Data Science 3*(1), 68–83.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering 86*, 97–106.

Lange, K. (2016). *MM optimization algorithms*. SIAM.

Laurent, S. (2024). *uniformly: Uniform Sampling*. R package version 0.5.0.

Lawson, J. (2014). *Design and Analysis of Experiments with R*. CRC press.

Lawson, J. and C. Willden (2024). *mixexp: Design and Analysis of Mixture Experiments*. R package version 1.2.7.1.

Leatherman, E. R., T. J. Santner, and A. M. Dean (2018). Computer experiment designs for accurate prediction. *Statistics and Computing 28*, 739–751.

Lehmann, E. L. and G. Casella (2006). *Theory of point estimation, Second Edition*. Springer Science & Business Media.

Lei, J., M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association 113*(523), 1094–1111.

Lekivetz, R. and B. Jones (2015). Fast flexible space-filling designs for nonrectangular regions. *Quality and Reliability Engineering International 31*(5), 829–837.

Lenth, R. V. (1989). Quick and easy analysis of unreplicated factorials. *Technometrics 31*(4), 469–473.

Leobacher, G. and F. Pillichshammer (2014). *Introduction to quasi-Monte Carlo integration and applications*. Springer.

Li, K. and S. Mak (2025). Prospar-gp: scalable Gaussian process modeling with massive non-stationary datasets. *Journal of Computational and Graphical Statistics 34*(4), 1742–1759.

Li, W. and C. F. J. Wu (1999). An integrated method of parameter design and tolerance design. *Quality Engineering 11*(3), 417–425.

Li, Y., L. Kang, and F. J. Hickernell (2020). Is a transformed low discrepancy design also low discrepancy? *Contemporary Experimental Design, Multivariate Analysis and Data Mining: Festschrift in Honour of Professor Kai-Tai Fang*, 69–92.

Liaw, A. and M. Wiener (2002). Classification and regression by randomforest. *R News 2*(3), 18–22.

Lindley, D. V. (1956). On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics 27*(4), 986–1005.

Lloyd, S. P. (1957). Least squares quantization in PCM's. *Bell Telephone Laboratories Article, Murray Hill, NJ*.

Loeppky, J. L., L. M. Moore, and B. J. Williams (2010). Batch sequential designs for computer experiments. *Journal of Statistical Planning and Inference 140*(6), 1452–1464.

Loeppky, J. L., J. Sacks, and W. J. Welch (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics 51*(4), 366–376.

López-Fidalgo, J. (2023). *Optimal experimental design: A Concise Introduction for Researchers*. Springer.

Lu, L. and C. M. Anderson-Cook (2021). Input-response space-filling designs. *Quality and Reliability Engineering International 37*(8), 3529–3551.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural computation 4*(4), 590–604.

Mak, S. (2018). *support: Support Points*. R package version 0.1.6.

Mak, S. (2021). *minimaxdesign*. R package version 0.1.5.

Mak, S. and V. R. Joseph (2018a). Minimax and minimax projection designs using clustering. *Journal of Computational and Graphical Statistics 27*(1), 166–178.

Mak, S. and V. R. Joseph (2018b). Projected support points: a new method for high-dimensional data reduction. *arXiv:1708.06897*.

Mak, S. and V. R. Joseph (2018c). Support points. *The Annals of Statistics 46*(6A), 2562–2592.

Mak, S. and V. R. Joseph (2025). Robust designs for Gaussian process modeling in computer experiments. *arXiv:2507.09156*.

Mak, S., C.-L. Sung, X. Wang, S.-T. Yeh, Y.-H. Chang, V. R. Joseph, V. Yang, and C. F. J. Wu (2018). An efficient surrogate model for emulation and physics extraction of large eddy simulations. *Journal of the American Statistical Association 113*(524), 1443–1456.

Maruthi, B. and V. R. Joseph (1999). Improving the yield of printed circuit boards using design of experiments. *Quality Engineering 12*(2), 259–265.

Masoudi, E., H. Holling, W. K. Wong, and S. Kim (2022). Icaod: An r package for finding optimal designs for nonlinear statistical models by imperialist competitive algorithm. *The R journal 14*(3), 20.

Matheron, G. (1963). Principles of geostatistics. *Economic geology 58*(8), 1246–1266.

Max, J. (1960). Quantizing for minimum distortion. *IRE Transactions on Information Theory 6*(1), 7–12.

McKay, M. D., R. J. Beckman, and W. J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*(2), 239–245.

Mee, R. W. (2009). *A comprehensive guide to factorial two-level experimentation*. Springer.

Miller, S. F. and A. J. Shih (2007). Thermo-mechanical finite element modeling of the friction drilling process. *Journal of Manufacturing Science and Engineering 129*, 531–538.

Mitchell, T. J. (1974). An algorithm for the construction of "D-optimal" experimental designs. *Technometrics 16*(1), 203–210.

Mitchell, T. J. and M. D. Morris (1992). Bayesian design and analysis of computer experiments: two examples. *Statistica Sinica*, 359–379.

Mockus, J. (1978). The application of Bayesian methods for seeking the extremum. *Towards global optimization (Eds. L.C.W. Dixon and G.P. Szego) 2*, 117–129.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Morgan-Wall, T. (2025). *spacefillr: Space-Filling Random and Quasi-Random Sequences*. R package version 0.4.0.

Morris, M. (2010). *Design of experiments: An introduction based on linear models*. Chapman and Hall/CRC.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics 33*(2), 161–174.

Morris, M. D. (2000). A class of three-level experimental designs for response surface modeling. *Technometrics 42*(2), 111–121.

Morris, M. D. and T. J. Mitchell (1995). Exploratory designs for computational experiments. *Journal of statistical planning and inference 43*(3), 381–402.

Morris, M. D., T. J. Mitchell, and D. Ylvisaker (1993). Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics 35*(3), 243–255.

Mukerjee, R. and C. F. J. Wu (2007). *A modern theory of factorial design*. Springer Science & Business Media.

Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook (2016). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons.

Nguyen, N.-K. and A. J. Miller (1992). A review of some exchange algorithms for constructing discrete d-optimal designs. *Computational Statistics & Data Analysis 14*(4), 489–498.

Niederreiter, H. (1992). *Random number generation and quasi-Monte Carlo methods*. SIAM.

Ning, J.-H., K.-T. Fang, and Y.-D. Zhou (2011). Uniform design for experiments with mixtures. *Communications in Statistics-Theory and Methods 40*(10), 1734–1742.

Oakley, J. E. and A. O'Hagan (2004). Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society Series B: Statistical Methodology 66*(3), 751–769.

Oates, C. J. (2021). Minimum discrepancy methods in uncertainty quantification. *arXiv preprint arXiv:2109.06075*.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society: Series B (Methodological) 40*(1), 1–24.

O'Hagan, A. (1987). Monte Carlo is fundamentally unsound. *The Statistician*, 247–249.

O'Hagan, A. (1991). Bayes–Hermite quadrature. *Journal of statistical planning and inference 29*(3), 245–260.

Overstall, A. M. and D. C. Woods (2017). Bayesian design of experiments using approximate coordinate exchange. *Technometrics 59*(4), 458–470.

Overstall, A. M., D. C. Woods, M. Adamou, and D. Michaelides (2024). *acebayes: Optimal Bayesian Experimental Design using the ACE Algorithm*. R package version 1.10.

Owen, A. B. (1994). Controlling correlations in Latin hypercube samples. *Journal of the American Statistical Association 89*(428), 1517–1522.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/.

Owen, A. B. (2014). Sobol' indices and shapley value. *SIAM/ASA Journal on Uncertainty Quantification 2*(1), 245–251.

Pal, B. and V. R. Joseph (1998). Developing objective strategies for monitoring multi input/single output chemical process in Statistical Methods for Quality Improvement, ed. B. Abraham, Boston: Birkhauser, 151–162.

Park, C. and D. W. Apley (2018). Patchwork kriging for large-scale Gaussian process regression. *The Journal of Machine Learning Research 19*(1), 269–311.

Park, J.-S. (1994). Optimal Latin-hypercube designs for computer experiments. *Journal of statistical planning and inference 39*(1), 95–111.

Peng, C.-Y. and C. F. J. Wu (2014). On the choice of nugget in kriging modeling for deterministic computer experiments. *Journal of Computational and Graphical Statistics 23*(1), 151–168.

Petersen, K. and M. Pedersen (2006). *The Matrix Cookbook*. Technical University of Denmark. Version 20051003.

Phadke, M. S. (1995). *Quality engineering using robust design*. Prentice Hall PTR.

Piepel, G. F. (2006). 50 years of mixture experiment research: 1955–2004. In *Response Surface Methodology and Related Topics, Ed. A. I. Khuri, Ch 12*, pp. 283–327. World Scientific.

Plackett, R. L. (1972). Studies in the history of probability and statistics. xxix: The discovery of the method of least squares. *Biometrika 59*(2), 239–251.

Plumlee, M. (2014). Fast prediction of deterministic functions using sparse grid experimental designs. *Journal of the American Statistical Association 109*(508), 1581–1591.

Plumlee, M. (2017). Bayesian calibration of inexact computer models. *Journal of the American Statistical Association 112*(519), 1274–1285.

Plumlee, M. and V. R. Joseph (2018). Orthogonal Gaussian process models. *Statistica Sinica* 28, 601–619.

Pronzato, L. and W. G. Müller (2012). Design of computer experiments: space filling and beyond. *Statistics and Computing 22*, 681–701.

Pujol, G. (2009). Simplex-based screening designs for estimating metamodels. *Reliability Engineering & System Safety 94*(7), 1156–1160.

Pukelsheim, F. (1993). *Optimal design of experiments*. John Wiley & Sons, New York.

Pukelsheim, F. and S. Rieder (1992). Efficient rounding of approximate designs. *Biometrika*, 763–770.

Qian, P. Z. (2009). Nested Latin hypercube designs. *Biometrika 96*(4), 957–970.

Qian, P. Z. (2012). Sliced Latin hypercube designs. *Journal of the American Statistical Association 107*(497), 393–399.

Qian, P. Z., M. Ai, and C. F. J. Wu (2009). Construction of nested space-filling designs. *The Annals of Statistics 37*, 3616–3643.

Qian, P. Z., B. Tang, and C. F. J. Wu (2009). Nested space-filling designs for computer experiments with two levels of accuracy. *Statistica Sinica 19*, 287–300.

Qian, P. Z. G., H. Wu, and C. F. J. Wu (2008). Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics 50*(3), 383–396.

Qian, Z., C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. J. Wu (2006). Building surrogate models based on detailed and approximate simulations. *ASME Journal of Mechanical Design 128*, 668–677.

Quiñonero-Candela, J. and C. E. Rasmussen (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research 6*(65), 1939–1959.

Raghavarao, D. and S. Altan (2003). A Heuristic Analysis of Highly Fractionated 2^n Factorial Experiments. *Metrika 58*, 185–191.

Rainforth, T., A. Foster, D. R. Ivanova, and F. Bickford Smith (2024). Modern Bayesian experimental design. *Statistical Science 39*(1), 100–114.

Ranjan, P., D. Bingham, and G. Michailidis (2008). Sequential experiment design for contour estimation from complex computer codes. *Technometrics 50*(4), 527–541.

Ranjan, P., R. Haynes, and R. Karsten (2011). A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data. *Technometrics 53*(4), 366–378.

Ranjan, P., W. Lu, D. Bingham, S. Reese, B. J. Williams, C.-C. Chou, F. Doss, M. Grosskopf, and J. P. Holloway (2011). Follow-up experimental designs for computer models and physical processes. *Journal of Statistical Theory and Practice 5*(1), 119–136.

Rao, C. R. (1947). Factorial experiments derivable from combinatorial arrangements of arrays. *Journal of the Royal Statistical Society 9*(1), 128–139.

Rao, C. R. (1973). *Linear statistical inference and its applications*, Volume 2. Wiley New York.

Rasmussen, C. E. and C. K. Williams (2006). *Gaussian processes for machine learning*. Cambridge, MA: The MIT Press.

Reese, C. S., A. G. Wilson, M. Hamada, H. F. Martz, and K. J. Ryan (2004). Integrated analysis of computer and physical experiments. *Technometrics 46*(2), 153–164.

Ren, Y., E. K. McGuinness, C. Huang, V. R. Joseph, R. P. Lively, and M. D. Losego (2021). Reaction–diffusion transport model to predict precursor uptake and spatial distribution in vapor-phase infiltration processes. *Chemistry of Materials 33*(13), 5210–5222.

Risdal, M. (2017). New York City taxi trip duration. https://kaggle.com/competitions/nyc-taxi-trip-duration. Kaggle.

Robert, C. P., G. Casella, and G. Casella (2010). *Introducing Monte Carlo methods with R*, Volume 18. Springer.

Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The annals of mathematical statistics 23*(3), 470–472.

Roustant, O., D. Ginsbourger, and Y. Deville (2021). *DiceKriging: Kriging Methods for Computer Experiments*. R package version 1.6.0.

Rozzi, G. (2022). *zipcodeR: Data & Functions for Working with US ZIP Codes*. R package version 0.3.5.

Sacks, J., S. B. Schiller, and W. J. Welch (1989). Designs for computer experiments. *Technometrics 31*(1), 41–47.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and Analysis of Computer Experiments. *Statistical Science 4*(4), 409 – 423.

Sacks, J. and D. Ylvisaker (1970). Statistical designs and integral approximation. *Proc. 12th Bienn. Semin. Can. Math. Congr.(R. Pyke, ed.)*, 115–136.

Saltelli, A. (2008). *Global Sensitivity Analysis: the Primer*. John Wiley & Sons.

Saltelli, A., P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola (2010). Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer physics communications 181*(2), 259–270.

Santner, T. J., B. J. Williams, and W. I. Notz (2018). *The design and analysis of computer experiments, Second Edition*. Springer.

Sauer, A., R. B. Gramacy, and D. Higdon (2023). Active learning for deep Gaussian process surrogates. *Technometrics 65*(1), 4–18.

Scheffé, H. (1958). Experiments with mixtures. *Journal of the Royal Statistical Society: Series B (Methodological) 20*(2), 344–360.

Scheffe, H. (1963). The simplex-centroid design for experiments with mixtures. *Journal of the Royal Statistical Society: Series B (Methodological) 25*(2), 235–251.

Scheidegger, A. (2024). *adaptMCMC: Implementation of a Generic Adaptive Monte Carlo Markov Chain Sampler*. R package version 1.5.

Schonlau, M., W. J. Welch, and D. R. Jones (1998). Global versus local search in constrained optimization of computer models. *New Developments and Applications in Experimental Design, Lecture notes-monograph series 34*, 11–25.

Seo, S., M. Wallat, T. Graepel, and K. Obermayer (2000). Gaussian process regression: active data selection and test point rejection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, Volume 3, pp. 241–246 vol.3.

Shang, B. and D. W. Apley (2021). Fully-sequential space-filling design algorithms for computer experiments. *Journal of Quality Technology 53*(2), 173–196.

Shang, B., D. W. Apley, and S. Mehrotra (2023). Diversity subsampling: Custom subsamples from large data sets. *INFORMS Journal on Data Science 2*(2), 161–182.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal 27*(3), 379–423.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524.

Shewry, M. C. and H. P. Wynn (1987). Maximum entropy sampling. *Journal of applied statistics 14*(2), 165–170.

Silvey, S. D. (1980). *Optimal design*. Chapman & Hall, London.

Singh, R. and J. Stufken (2024). Factor selection in screening experiments by aggregation over random models. *Computational Statistics & Data Analysis 194*, 107940.

Singh, R. K., V. R. Joseph, and S. N. Melkote (2011). A statistical approach to the optimization of a laser-assisted micromachining process. *The International Journal of Advanced Manufacturing Technology 53*, 221–230.

Sitter, R. and C. F. J. Wu (1993). Optimal designs for binary response experiments: Fieller, d, and a criteria. *Scandinavian Journal of Statistics*, 329–341.

Smith, M. R. (2025). *Ternary: Create Ternary and Holdridge Plots*. R package version 2.3.4.

Smith, R. C. (2024). *Uncertainty quantification: theory, implementation, and applications*. SIAM.

Smolyak, S. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Soviet Math. Dokl.*, Volume 4, pp. 240–243.

Snee, R. D. (1977). Validation of regression models: methods and examples. *Technometrics 19*(4), 415–428.

Snelson, E. and Z. Ghahramani (2005). Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems 18*.

Sobol', I. M. (1990). On sensitivity estimation for nonlinear mathematical models. *Matematicheskoe modelirovanie (in Russian) 2*(1), 112–118.

Sobol', I. M. (1993). On sensitivity estimation for nonlinear mathematical models. *Mathematical Modeling and Computational Experiments 1*(4), 407–414.

Sobol', I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and computers in simulation 55*(1-3), 271–280.

Sobol', I. M. and S. Kucherenko (2009). Derivative based global sensitivity measures and their link with global sensitivity indices. *Mathematics and Computers in Simulation 79*(10), 3009–3017.

Song, D. and V. R. Joseph (2025). Efficient active learning strategies for computer experiments. *Technometrics,* DOI: 10.1080/00401706.2025.2546366.

Song, E., B. L. Nelson, and J. Staum (2016). Shapley effects for global sensitivity analysis: Theory and computation. *SIAM/ASA Journal on Uncertainty Quantification 4*(1), 1060–1083.

Stein, M. L. (1999). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

Sun, F. and R. B. Gramacy (2024). *maximin: Space-Filling Design under Maximin Distance*. R package version 1.0-6.

Sun, F., T. J. Santner, and A. M. Dean (2013). One-at-a-time designs for estimating elementary effects of simulator experiments with non-rectangular input regions. *Stat Appl 11*, 15–32.

Sung, C.-L., Y. Ji, S. Mak, W. Wang, and T. Tang (2024). Stacking designs: Designing multifidelity computer experiments with target predictive accuracy. *SIAM/ASA Journal on Uncertainty Quantification 12*(1), 157–181.

Sung, C.-L. and R. Tuo (2024). A review on computer model calibration. *Wiley Interdisciplinary Reviews: Computational Statistics 16*(1), e1645.

Sürer, Ö., M. Plumlee, and S. M. Wild (2024). Sequential Bayesian experimental design for calibration of expensive simulation models. *Technometrics 66*(2), 157–171.

Surjanovic, S. and D. Bingham (2013). *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved July 11, 2025.

Székely, G. J. and M. L. Rizzo (2013). Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference 143*(8), 1249–1272.

Székely, G. J. and M. L. Rizzo (2023). *The energy of data and distance correlation*. Chapman and Hall/CRC.

Székely, G. J., M. L. Rizzo, et al. (2004). Testing for equal distributions in high dimension. *InterStat 5*(16.10), 1249–1272.

Taguchi, G. (1987). *System of Experimental Design*. UNIPUB/Kraus International Publications, Whiteplains.

Tan, M. H. (2013). Minimax designs for finite design regions. *Technometrics 55*(3), 346–358.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American statistical association 88*(424), 1392–1397.

Tang, B. and L.-Y. Deng (1999). Minimum G_2-aberration for nonregular fractional factorial designs. *The Annals of Statistics 27*(6), 1914–1926.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology 58*(1), 267–288.

Tillé, Y. (2006). *Sampling algorithms*. Springer.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR.

Tresp, V. (2000, 11). A Bayesian committee machine. *Neural Computation 12*(11), 2719–2741.

Tuo, R. (2019). Adjustments to computer models via projected kernel calibration. *SIAM/ASA Journal on Uncertainty Quantification 7*(2), 553–578.

Tuo, R. and C. F. J. Wu (2015). Efficient calibration for imperfect computer models. *The Annals of Statistics 43*(6), 2331–2352.

Tuo, R. and C. F. J. Wu (2016). A theoretical framework for calibration in computer models: Parametrization, estimation and convergence properties. *SIAM/ASA Journal on Uncertainty Quantification 4*(1), 767–795.

Tuo, R., C. F. J. Wu, and D. Yu (2014). Surrogate modeling of computer experiments with different mesh densities. *Technometrics 56*(3), 372–380.

Vakayil, A. and V. R. Joseph (2022a). Data twinning. *Statistical Analysis and Data Mining: The ASA Data Science Journal 15*(5), 598–610.

Vakayil, A. and V. R. Joseph (2022b). *twinning: Data Twinning*. R package version 1.0.

Vakayil, A. and V. R. Joseph (2024a). A global-local approximation framework for large-scale Gaussian process modeling. *Technometrics 66*(2), 295–305.

Vakayil, A. and V. R. Joseph (2024b). *twingp: A Fast Global-Local Gaussian Process Approximation*. R package version 1.0.0.

Vakayil, A., V. R. Joseph, and S. Mak (2022). *SPlit: Split a Dataset for Training and Testing*. R package version 1.0.

van Dam, E. R., G. Rennen, and B. Husslage (2009). Bounds for maximin latin hypercube designs. *Operations Research 57*(3), 595–608.

Vázquez, A. R. and H. Xu (2024). An integer programming algorithm for constructing maximin distance designs from good lattice point sets. *Statistica Sinica 34*, 1347–1366.

Vecchia, A. V. (1988). Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society Series B: Statistical Methodology 50*(2), 297–312.

Verdinelli, I. and L. Wasserman (2024). Feature importance: A closer look at shapley values and loco. *Statistical Science 39*(4), 623–636.

Vihola, M. (2012). Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and computing 22*, 997–1008.

Wahba, G. (1978). Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology 40*(3), 364–372.

Wahba, G. (1990). *Spline models for observational data*. SIAM.

Wald, A. (1943). On the efficient design of statistical investigations. *The Annals of Mathematical Statistics 14*(2), 134–140.

Wang, D. and V. R. Joseph (2022). *mined: Minimum Energy Designs*. R package version 1.0-3.

Wang, H., Q. Xiao, and A. Mandal (2022). *LHD: Latin Hypercube Designs (LHDs)*. R package version 1.3.3.

Wang, H., M. Yang, and J. Stufken (2019). Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association 114*(525), 393–405.

Wang, L., Q. Xiao, and H. Xu (2018). Optimal maximin l_1-distance Latin hypercube designs based on good lattice point designs. *The Annals of Statistics 46*(6B), 3741–3766.

Wang, S., A. P. Generale, S. R. Kalidindi, and V. R. Joseph (2024). Sequential designs for filling output spaces. *Technometrics 66*(1), 65–76.

Wang, S. and V. R. Joseph (2024). *OSFD: Output Space-Filling Design*. R package version 3.1.

Wang, S. and V. R. Joseph (2025a). Active learning via heteroskedastic rational kriging. SIAM/ASA Journal on Uncertainty Quantification. *arXiv: 2507.10952*.

Wang, S. and V. R. Joseph (2025b). Comment: A model-free method for input-output space-filling design. *Technometrics* 67(3), 384–387.

Wang, S. and V. R. Joseph (2025c). *SFDesign: Space-Filling Designs*. R package version 0.1.2.

Wang, Y. and K.-T. Fang (1990). Number theoretic methods in applied statistics (ii). *Chinese Annals of Mathematics, Series B 11*, 859–914.

Warnock, T. T. (1972). Computational investigations of low-discrepancy point sets. In *Applications of number theory to numerical analysis*, pp. 319–343. Elsevier.

Wasserman, L. (2006). *All of nonparametric statistics*. Springer Science & Business Media.

Weber, A. (1929). *The Theory of the Location of Industries*. Chicago University Press, Chicago.

Wendland, H. (2004). *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge, United Kingdom: Cambridge University Press.

Wheeler, B. and J. Braun (2024). *AlgDesign: Algorithmic Experimental Design*. R package version 1.2.1.1.

Williams, B. J., J. L. Loeppky, L. M. Moore, and M. S. Macklem (2011). Batch sequential design to achieve predictive maturity with calibrated computer models. *Reliability Engineering & System Safety 96*(9), 1208–1219.

Williamson, B. D., P. B. Gilbert, M. Carone, and N. Simon (2021). Nonparametric variable importance assessment using machine learning techniques. *Biometrics 77*(1), 9–22.

Wong, R. K., C. B. Storlie, and T. C. Lee (2017). A frequentist approach to computer model calibration. *Journal of the Royal Statistical Society Series B: Statistical Methodology 79*(2), 635–648.

Wu, C. F. J. and M. S. Hamada (2021). *Experiments: planning, analysis, and optimization*. John Wiley & Sons.

Xiao, Q. and V. R. Joseph (2022). *MOFAT: Maximum One-Factor-at-a-Time Designs*. R package version 1.0.

Xiao, Q., V. R. Joseph, and D. M. Ray (2023). Maximum one-factor-at-a-time designs for screening in computer experiments. *Technometrics 65*(2), 220–230.

Xiu, D. (2010). *Numerical methods for stochastic computations: a spectral method approach*. Princeton university press.

Xu, H. (2002). An algorithm for constructing orthogonal and nearly-orthogonal arrays with mixed levels and small runs. *Technometrics 44*(4), 356–368.

Xu, H. and C. F. J. Wu (2001). Generalized minimum aberration for asymmetrical fractional factorial designs. *The Annals of Statistics 29*(4), 1066–1077.

Yang, J.-F., C. D. Lin, P. Z. Qian, and D. K. J. Lin (2013). Construction of sliced orthogonal Latin hypercube designs. *Statistica Sinica*, 1117–1130.

Yang, M. (2010). On the de la garza phenomenon. *The Annals of Statistics 38*, 2499–2524.

Ye, K. Q. (1998). Orthogonal column Latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association 93*(444), 1430–1439.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research 28*(12), 1797–1808.

Yin, Y., D. K. J. Lin, and M.-Q. Liu (2014). Sliced Latin hypercube designs via orthogonal arrays. *Journal of Statistical Planning and Inference 149*, 162–171.

Ylvisaker, D. (1975). Designs on random fields. *A survey of statistical design and linear models 37*(6), 593–607.

Yu, W.-Y. and V. R. Joseph (2025a). Automated analysis of experiments using hierarchical garrote. *Journal of Quality Technology* 57(4), 316–330.

Yu, W.-Y. and V. R. Joseph (2025b). *HiGarrote: Nonnegative Garrote Method Incorporating Hierarchical Relationships*. R package version 2.0.0.

Yuan, M., V. R. Joseph, and Y. Lin (2007). An Efficient Variable Selection Approach for Analyzing Designed Experiments. *Technometrics 49*(4), 430–439.

Yuan, M., V. R. Joseph, and H. Zou (2009). Structured variable selection and estimation. *The Annals of Applied Statistics*, 1738–1757.

Yuchi, S. H., V. R. Joseph, and C. F. J. Wu (2023). Design and analysis of multifidelity finite element simulations. *Journal of Mechanical Design 145*(6), 061703.

Yuille, A. L. and A. Rangarajan (2003). The concave-convex procedure. *Neural computation 15*(4), 915–936.

Zador, P. (1982). Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory 28*(2), 139–149.

Zhang, B., D. A. Cole, and R. B. Gramacy (2021). Distance-distributed design for Gaussian process surrogates. *Technometrics 63*(1), 40–52.

Zhang, N. and D. W. Apley (2014). Fractional Brownian fields for response surface metamodeling. *Journal of Quality Technology 46*(4), 285–301.

Zhang, Q. and W. Wang (2025). Sobolev calibration of imperfect computer models. *Journal of the American Statistical Association 120*(549), 408–418.

Zhou, Y.-D. and H. Xu (2014). Space-filling fractional factorial designs. *Journal of the American Statistical Association 109*(507), 1134–1144.

Zhu, L., T. Dasgupta, and Q. Huang (2014). A D-optimal design for estimation of parameters of an exponential-linear growth curve of nanostructures. *Technometrics 56*(4), 432–442.



Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Index

Aberration, 148
  G_2, 155
  Bayesian, 153, 158
  generalized, 155–157
  minimum aberration designs, 9, 148, 149, 153, 154, 161
Active learning
  Cohn, 129
  MacKay, 131
  See also Sequential
Aliasing, 146–148, 152, 153
  complex, 149
  matrix, 147

Bayesian
  computation, 78, 104
  optimization, 133
  quadrature, 108
Brownian motion, 99, 100, 121, 122
  kernel, 82

Candidate set, 26, 66, 80, 130
Chebyshev, 11–13, 27, 41, 42, 59, 62
Clustering, 48
  L_q-quantization, 107
  `kmeans`, 48, 108, 176, 193
Clustering-based design, 47
  `SFDesign`, 48
  `kmeans`, 48
Coding
  Helmert, 154
  orthogonal polynomial, 154
  treatment, 154
Constrained region, 65, 66, 80, 81, 108, 109, 158, 160
Coresets, 185
  See also Subsample
Correlation function, 16
  exponential, 17, 174
  Gaussian, 16, 32, 45, 48, 49, 52, 61, 152
  Matérn, 17
  multiplicative inverse multiquadric, 131, 138
  power exponential, 17, 202
  rational quadratic, 17, 53, 138
  Wendland, 203
Covering design, 51
  See also Minimax
Cross-validation, 14, 16, 19, 28, 30, 32, 80, 128, 190
  generalized, 28

Data splitting, 188
  `SPlit`, 188–191
  DUPLEX, 188
  See also Twinning
Defining contrast subgroup, 148
Defining relation, 147
Design generator, 147
Designs
  A-optimal, 25
  approximate, 169
  Bayesian optimal, 170
  Bayesian-inspired, 149
  branching, 70
  central composite, 157
  clustering-based, 47
  cross array, 46
  D-optimal, 25, 144
  definitive screening, 163
  exact, 169
  four-level, 154
  fractional factorial, 143
  G-optimal, 25
  generalized minimum aberration, 155
  I-optimal, 25
  IMSE optimal, 40
  inverse, 135
  IRMSE optimal, 40
  Latin hypercube, 56
  marginally coupled, 68
  maximin, 52
  maximin Latin hypercube, 58
  maximum entropy, 43
  maximum resolution, 148
  MaxPro, 60

minimax, 49
minimum aberration, 9, 148
minimum energy, 75
mixture, 158
MMSE optimal, 42
MOFAT, 120
Morris, 118
nested, 71
nested BFS, 74
nonregular, 145
OFAT, 118
orthogonal arrays, 155
orthogonal Latin hypercube, 59
orthogonal-maximin Latin hypercube, 59
Plackett-Burman, 149
prediction-based, 39
regular, 145
response surface, 157
robust, 175
screening, 111
sequential, 127
simplex centroid designs, 159
simplex lattice designs, 158
single array, 158
sliced Latin hypercube, 67
Sobol', 120
space-filling, 47
sparse grids, 13
stacking, 74
support points, 97
three-level, 154
two-level, 143
uniform, 89
weighted space-filling, 75
Discrepancy
centered L_2, 93
kernel, 93
maximum mean, 103
measures, 93
model, 177, 182
star L_2, 90, 93
wrap-around, 93
See also model bias
Divide-and-conquer, 190

Empirical distribution, 27, 89, 90, 98, 99, 103, 185, 189
Energy distance, 98–100, 103, 157, 158, 186, 189, 191

Factors
branching, 70
continuous, 67
control, 46
discrete-numeric, 67
nested, 70
noise, 46
qualitative, 66
quantitative, 67
shared, 70
Fill distance, 50
See also Covering radius
FIRST, 199
`first`, 200
`pyfirst`, 200
Function
completely monotone, 33
positive definite, 14
positive semi-definite, 14
radial, 14

Gaussian process, 7, 19, 82, 92, 109, 116, 128, 130, 150, 172, 177, 187, 200
`rkriging`, 23, 32, 65, 116, 130, 134, 135, 152, 187, 200
Brownian motion, 99, 121
composite, 192
deep, 74
empirical Bayes, 21
functional, 173
global, 201
graphical, 74
heteroskedastic, 157
hierarchical Bayes, 22
local, 201
orthogonal, 35, 178
treed, 192
TwinGP, 201
Vecchia's approximation, 201
See also Kriging
Goodness-of-fit, 97, 98
Cramér-von Mises, 90, 97
Kolmogorov-Smirnov, 89, 97
See also Energy distance

Hadamard product, 16, 45
Half-normal plot, 146, 152
Hypersphere, 77, 109
`uniformly`, 109

Identifiability, 167, 178

structural, 167, 179
Importance support points, 106, 170, 179, 191
Information theory, 7
entropy, 43, 130, 170
expected information gain, 170
Fisher information, 168, 171, 180
Interval score, 36, 204
Inverse distance weighting predictor, 35
Inverse probability transform, 27, 83, 96, 97, 101, 108, 159

Kennedy-O'Hagan model, 177
Kernel, 20, 29, 93, 203
Brownian motion, 82
Brownian sheet, 93
density, 103, 104
discrepancy, 93
distance-based, 102
Euclidean, 102
Gaussian, 30, 102
herding, 102
inverse quadratic, 53
positive definite, 102
rational quadratic, 61
ridge regression, 28, 30, 32
See also Correlation function
Kernel herding, 102
Koksma-Hlawka inequality, 91, 99
Kriging, 16
`rkriging`, 23, 32, 65, 116, 130, 134, 135, 152, 187, 200
intrinsic, 99
ordinary, 18
rational, 34, 35
universal, 33
See also Gaussian process
Kronecker product, 151, 173

Leave-out-covariate, 198
Leverage, 28
Linear predictor, 17, 28, 33

Matrix
`Matrix`, 167
`rankMatrix`, 167
identities, 15, 26, 152, 173
Maximin, 52, 75, 77
`SFDesign`, 54
`maximin.augment`, 55
`maximin`, 75
sequential, 55
sphere packing, 55
Maximin LHD, 58, 108
`SFDesign`, 58
`SLHD`, 58, 67
MaxPro, 60, 77, 101, 130, 156
`MaxProAugment`, 64, 85
`MaxProQQ`, 69, 84
`MaxPro`, 62
`SFDesign`, 62
`maxpro.remove`, 74
criterion, 60, 69, 102, 156
LHD, 62, 108, 122, 136, 137
sequential, 64, 179
Mean reversion, 187
Minimax, 49
`LatticeDesign`, 52
`minimaxdesign`, 52
covering design, 51
Minimum energy designs, 75, 76, 78, 83, 105
`mined`, 79
constrained, 80
sequential, 77
Mixture experiments, 158
`mixexp`, 159
mixture-of-mixtures, 160
MOFAT design, 120, 122, 123, 131
`MOFAT`, 123
Monte Carlo, 5, 48, 93, 94, 99–101, 103, 113, 116, 119, 121, 129, 171, 198, 199
high-dimensional integration, 78, 91, 113, 170
MCMC, 22, 104
PMC, 106
Morris
elementary effects, 118
screening, 118
MSE-rep points, 103
See also Principal points
Multifidelity, 85
fidelity parameter, 81
finite element analysis, 71, 81
mesh density, 82
model, 72, 82, 85
nested designs, 71
Multiplet, 195
`multiplet`, 194
See also Twinning

Noise factors, 46
external, 46

internal, 46
Nugget, 32, 134
Numerical integration, 41, 80, 91
See also Monte-Carlo

Optimization, 132
`optim`, 42
Bayesian, 132, 175
convex-concave procedure, 100
derivative-free, 167
difference-of-convex, 100, 186
discrete, 186
expected improvement, 133
Fedorov exchange algorithm, 26, 144
genetic algorithm, 58
global, 54, 132
gradient-based, 54, 61, 153
greedy, 64
LBFGS, 42
local, 132
majorization-minimization, 100
mixed integer programming, 169
particle swarm, 52
robust parameter design, 109
sequential, 73
simulated annealing, 58
stochastic, 187

Polynomial
Chebyshev, 11, 12
interpolation, 10–14
Lagrange interpolation, 10
orthogonal coding, 154
regression, 24, 30
Principle
effect heredity, 147
effect hierarchy, 146, 153

Quantization, 48, 107
See also Clustering
Quasi-Monte Carlo, 93, 113, 129
`spacefillr`, 93
PQMC, 104
Sobol' sequence, 93

Radial basis functions, 13, 14, 28
Regression, 23
Bayesian, 145
GP, 7, 8, 30–32, 143, 201
kernel ridge, 28, 30, 31
lasso, 145
linear, 7, 8, 23, 25, 27, 42, 112, 143, 144, 167, 168, 185, 192
logistic, 180
nonlinear, 4, 166, 167, 169
nonparametric, 8, 25, 192, 197, 198
polynomial, 24
probit, 180
ridge, 28, 30
Reproducing kernel Hilbert space, 29, 103
Resolution, 148, 157
maximum resolution designs, 148
Robust design, 9, 175, 178
nested, 182
Robust parameter design, 46, 109, 157, 158
cross array, 46
single array, 158
Runge's phenomenon, 11, 12

Scoring method, 68
Sensitivity
analysis, 111
derivative-based, 116
first-order index, 112
matrix, 166
total index, 112
variance-based, 111
Separation distance, 53–55
See also packing radius
Sequential
`OSFD`, 136
designs, 127
herding, 102
maximin, 55
MaxPro, 64
minimum energy, 77
See also Active learning
Shapley values, 198
Simplex, 159, 160
Sobol'
design, 120
first-order index, 112
sequence, 93
total index, 112, 196
Sphere packing, 55, 76
See also Maximin
Subsample, 185, 188, 191, 192, 199, 201
`SPlit`, 186
`subsample`, 186
`twinning`, 190
support points-based, 186

Support points, 97, 103, 104, 160, 176, 186, 188, 189
`support`, 101, 104, 160
importance, 106, 170
projected, 102
scenario reduction, 187
stochastic programming, 187
See also Twinning
Survey sampling, 190
`BalancedSampling`, 191
`SPlit`, 191
`twinning`, 191
spatial balance, 191

Twinning, 189
`twinning`, 190, 191, 201, 202

Uncertainty
aleatoric, 178
epistemic, 178
Uncertainty propagation, 5, 6, 103, 104
Uniform design, 89, 129, 161
`SFDesign`, 93

Validation, 64, 65, 194
See also Cross-validation
Variable selection, 149, 196
backward elimination, 199
forward, 153, 199
Gauss-Dantzig selector, 149
LARS, 149
lasso, 149
nonnegative garrote, 153
See also Factor screening
Voronoi regions, 47–50, 192, 193

Wordlength pattern, 148, 153
generalized, 155